EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)

Volume 41 (2022), Number 2

A-ULMPM: An Adaptively Updated Lagrangian Material Point Method for Efficient Physics Simulation without Numerical Fracture

Haozhe Su† Tao Xue† Chengguizi Han Mridul Aanjaneya

Department of Computer Science, Rutgers University

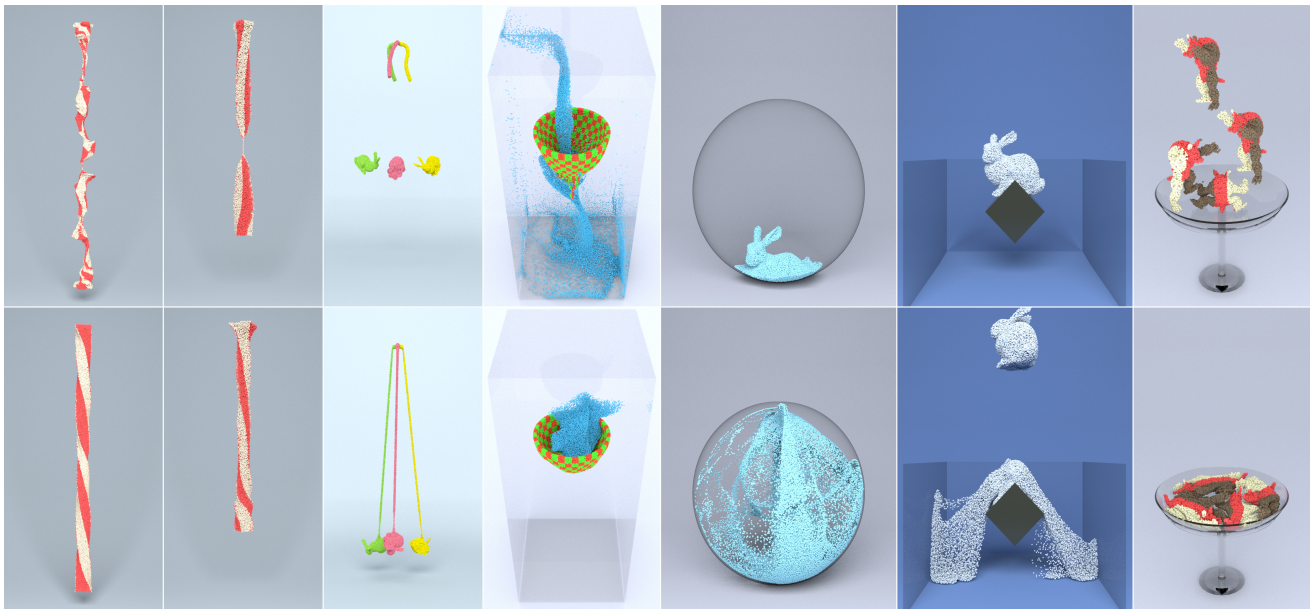


Figure 1: *Our A-ULMPM framework (bottom) avoids numerical fracture that plagues Eulerian approaches to MPM (top) and allows for large deformations in solid and fluid simulations while requiring* $4.27\times - 31.52\times$ *less computational overhead for configuration updates.*

Abstract

We present an adaptively updated Lagrangian Material Point Method (A-ULMPM) to alleviate non-physical artifacts, such as the cell-crossing instability and numerical fracture, that plague state-of-the-art Eulerian formulations of MPM, while still allowing for large deformations that arise in fluid simulations. A-ULMPM spans MPM discretizations from total Lagrangian formulations to Eulerian formulations. We design an easy-to-implement physics-based criterion that allows A-ULMPM to update the reference configuration adaptively *for measuring physical states, including stress, strain, interpolation kernels and their derivatives. For better efficiency and conservation of angular momentum, we further integrate the APIC [JSS*15] and MLS-MPM [HFG*18] formulations in A-ULMPM by augmenting the accuracy of velocity rasterization using both the local velocity and its first-order derivatives. Our theoretical derivations use a nodal discretized Lagrangian, instead of the weak form discretization in MLS-MPM [HFG*18], and naturally lead to a "modified" MLS-MPM in A-ULMPM, which can recover MLS-MPM using a completely Eulerian formulation. A-ULMPM does not require significant changes to traditional Eulerian formulations of MPM, and is computationally more efficient since it only updates interpolation kernels and their derivatives during large topology changes. We present end-to-end 3D simulations of stretching and twisting hyperelastic solids, viscous flows, splashing liquids, and multi-material interactions with large deformations to demonstrate the efficacy of our new method.*

CCS Concepts

• ***Computing methodologies*** → *Computer graphics; Physical simulation;*

† Joint first authors

1. Introduction

The Material Point Method (MPM) family of discretizations [SCS94], such as Fluid Implicit Particle (FLIP) [BKR88] and Particle-in-Cell (PIC) [SZS95], emerged as an effective choice for simulating various materials and gained popularity in visual effects (VFX) for providing high-fidelity physics simulations of snow [SSC*13], sand [KGP*16, DBD16], phase change [SSJ*14, GWW*18], viscoelasticity [RGJ*15, YSB*15, SH*21], viscoplasticity [FLGJ19], elastoplasticity [GTJS17], fluid structure interactions [FQL*20], fracture [WFL*19, HJST13], fluid-sediment mixtures [TGK*17, GPH*18], baking and cooking [DHW*19], and diffusion-driven phenomena [XSH*20]. In contrast to Lagrangian mesh-based methods, such as the Finite Element Method (FEM) [ZTNZ77, SB12], and pure particle-based methods, such as Smoothed Particle Hydrodynamics (SPH) [DG96, LLZ08], MPM merges the advantages of both Lagrangian and Eulerian approaches and automatically supports dynamic topology changes such as material splitting and merging. It uses Lagrangian particles to carry material states, while the background grid acts as an Eulerian "scratch pad" for computing the divergence of stress and performing spatial/temporal numerical integration. The use of a background grid allows for regular numerical stencils, benefiting from cache-locality, while the use of particles avoids the numerical dissipation issues characteristics of Eulerian grid-based schemes.

Conventional MPM discretization employs an *Eulerian* formulation, which measures stress and strain and computes derivatives and integrals with respect to the Eulerian coordinates (i.e., the "current" configuration). Eulerian MPM (EMPM) has been acknowledged as a powerful tool for physics-based simulations [JST*16], particularly if the system involves large deformations, such as fluid-like motion. However, it suffers from a number of shortcomings such as the *cell-crossing instability* that is caused when a material point crosses between cells of the background grid because of the discontinuous shape function gradient at the grid boundary [HBG16]. Many studies have shown that when cell-crossing occurs, the MPM solutions can either be non-convergent or reduce the convergence rate when refining grid, with the spatial convergence rate varying between first and second order [WG07] (see Figure 14). The deficiency of cell-cross instability has been reduced in the latest MPM formulations, such as the Affine Particle-In-Cell (APIC) method [JST17], the generalized interpolation MPM (GIMP) [BK04, GTJS17], and the Convected Particles Domain Interpolation (CPDI) [SBB11]. Among them, the APIC approach has been widely adopted in computer graphics. The central idea behind APIC is to retain the filtering property of PIC, but reduce dissipation by interpolating more information, such as the velocity and its *derivatives*, aiming to conserve linear and angular momentum. An improved APIC, namely PolyPIC, was proposed in [FGG*17] that allows for locally *high-order approximations*, rather than approximations to the grid velocity field. Later on, a *moving least squares* MPM formulation (MLS-MPM) [HFG*18] was developed by introducing the MLS technique to elevate the accuracy of the internal force evaluation and velocity derivatives.

Although EMPM discretizations have been improved to a certain degree via the aforementioned MLS techniques, they still suffer from *numerical fracture* that occurs when particles are displaced apart by a distance greater than one cell width. Such non-physical fracture depends only on the background grid resolution and is not related to any other issue that would ultimately limit the accuracy of the EMPM to model actual physical fracture of materials under large deformations (see Figure 2), particularly in solid simulations.

As a counterpart to Eulerian formulations, *total Lagrangian formulations* offer a promising alternative for avoiding the cell-crossing instability and numerical fracture in MPM [dVNH20, dVN21]. Unlike EMPM, total Lagrangian MPM (TLMPM) measures stress and strain and computes derivatives and integrals with respect to the *original* configuration at time t^0 (similar to traditional FEM [SB12, ZTNZ77]). By doing this, no matter what the deformation, the reference configuration being always the same, there is neither any cell-crossing instability nor any numerical fracture. Despite its high efficacy in solid simulations, traditional TLMPM fails to model extremely large deformations that arise in fluid simulations. We show a 2D droplet example in Figure 3. TLMPM is not able to capture the dynamics of splashes because the interpolation kernel and its derivatives in TLMPM only reflect the fixed topology at time t^0 and do not support extreme topologically changing dynamics. To alleviate the cell-crossing instability and numerical fracture while allowing for large deformations, we present an adaptively updated Lagrangian discretization of MPM (*A*-ULMPM) that spans from total Lagrangian formulations to Eulerian formulations. Unlike EMPM and TLMPM, *A*-ULMPM allows the configuration to be updated *adaptively* for measuring physical states including stress, strain, interpolation kernels and their derivatives.

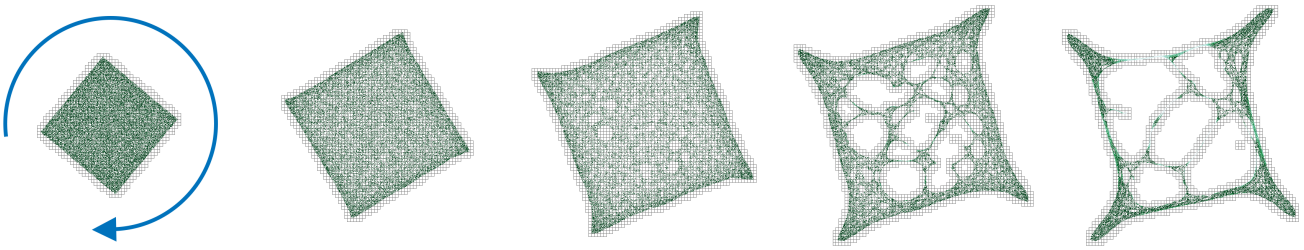
Our theoretical derivations focus on a nodal discretized Lagrangian (see equation (13)), instead of the weak form discretization in MLS-MPM [HFG*18], and naturally lead to a modified MLS-MPM in *A*-ULMPM, which can recover MLS-MPM by using a completely Eulerian formulation.

A-ULMPM does not require significant changes to traditional EMPM, and is more computationally efficient since it only updates interpolation kernels and their derivatives during large topology changes. To summarize, our main contributions are as follows:

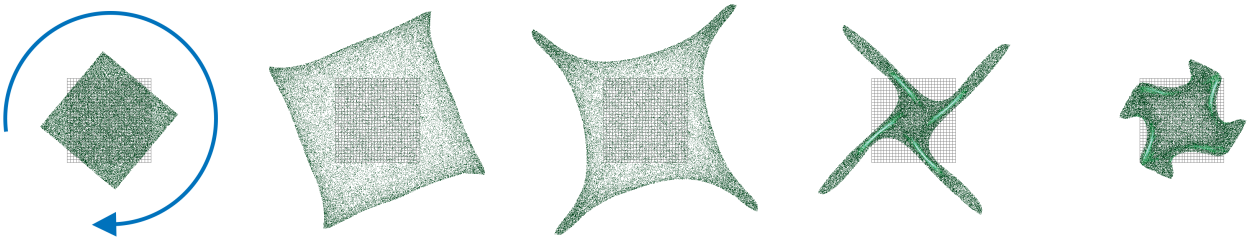
1. An adaptively updated Lagrangian MPM (*A*-ULMPM) that spans discretizations from TLMPM to EMPM and avoids the cell-crossing instability and numerical fracture;
2. An easy-to-implement criterion that automatically updates the reference topology to enable fluid-like simulations;
3. Integration of APIC and MLS-MPM in *A*-ULMPM to allow angular momentum conservation and efficiency by reconstructing the interpolation kernel only during topology updates;
4. End-to-end 3D simulations of stretching and twisting hyperelastic solids, splashing liquids, and multi-material interactions to highlight the benefits of our *A*-ULMPM framework.

2. RELATED WORK

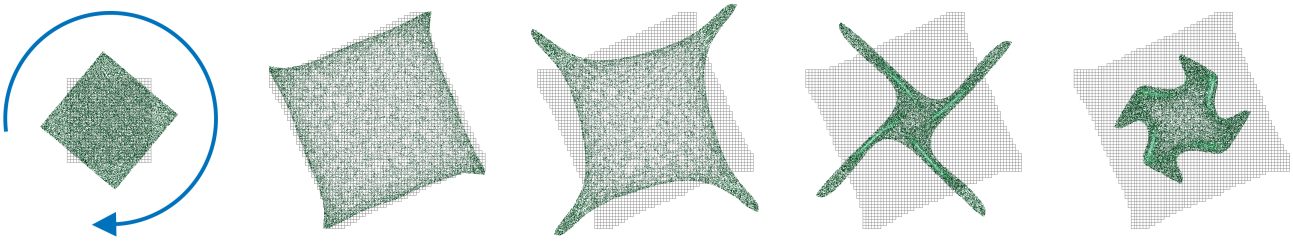
In this work, we only review prior work related to MPM [JST*16] since our focus is on MPM. However, we note that there are several established methods for particle-based simulations, including SPH [DG96], position-based dynamics [MKN*04, MHHR07, MMCK14], linear complementarity formulations [Erl13], and geometric computing techniques [dGWH*15, SBH09].



MLS-EMPM: $t = \{0.00833, 0.01667, 0.02500, 0.03333, 0.04167\}$s



MLS-TLMPM: $t = \{0.00833, 0.02500, 0.04167, 0.06250, 0.07917\}$s



MLS-*A*-ULMPM: $t = \{0.00833, 0.02500, 0.04167, 0.06250, 0.07917\}$s

Figure 2: ***Eliminating numerical fracture.*** *Our A-ULMPM (bottom row) and TLMPM (middle row) for solid simulation capture the appealing hyperelastic rotation, preserve the angular momentum for long simulation periods, and completely eliminate numerical fracture. Traditional EMPM (top row) suffers from severe numerical fracture in solid simulations when large deformations occur, while TLMPM (middle row) does not. EMPM updates configurations simultaneously with respect to particles, while TLMPM does not update configurations at all. A-ULMPM only updates configurations whenever the shape changes reach to our predefined criterion (see equation (33)). The background grid represents the configuration linking to the present particle dynamics.*

2.1. MPM in Graphics

The seminal work of Zhu and Bridson [ZB05] first introduced the FLIP method for sand simulation. Subsequent works further explored its strength in simulating a broader spectrum of material behaviors including snow [SSC*13], granular materials [DBD16, KGP*16, TGK*17, GPH*18], foam [RGJ*15, YSB*15], complex fluids [FLGJ19, GTJS17], cloth, hair and fiber collisions [JGT17, FMB*17, FBGZ18], fracture [WFL*19, WCL*20] and phase change [SSJ*14, GWW*18, SH*21]. We also note the related works of [MSW*09] for hair simulation, [SMT08] for cloth simulation, [NGL10] for sand simulation and [PAKF13] for bubble simulation, which bear similarities to MPM due to their hybrid nature. Various works have improved or modified aspects of the standard MPM techniques commonly used in graphics [FQL*20, YSC*18, XSH*20, DHW*19]. Among them, notably, Jiang et al. [JSS*15, JST17] proposed an Affine Particle-In-Cell (APIC) approach that conserves angular momentum and prevents visual artifacts such as noise, instability, clumping and volume loss/gain existing in both FLIP and PIC methods. Furthermore, APIC was enhanced in [FGG*17, HFG*18] to improve the kinetic energy conservation in particle/grid transfers.

2.2. MPM in Engineering

In the engineering community, MPM was first introduced in [SCS94] as an extension of the FLIP method [BKR88], and substantial improvements and variants have been proposed thereafter, including experimental validation for studying dynamic anticrack propagation in snow avalanches [GGT*18] and the use of MPM for designing differentiable physics engines for robotics applications [HLS*19]. Different strategies for updating the stress were compared to investigate the energy conservation error in MPM in [Bar02]. The quadrature error and cell-crossing error of MPM was investigated in [SKB08]. A generalized interpolation material point method (GIMPM) [BK04] was proposed to obtain a smoother field representation by combining the shape functions of the grid with the particle characteristic function. The cell-crossing error in the MPM discretization was alleviated by introducing the local *tangent* affine deformation of particles in the convected particle domain method (CPDI) [SBB11]. However, CPDI does not com-

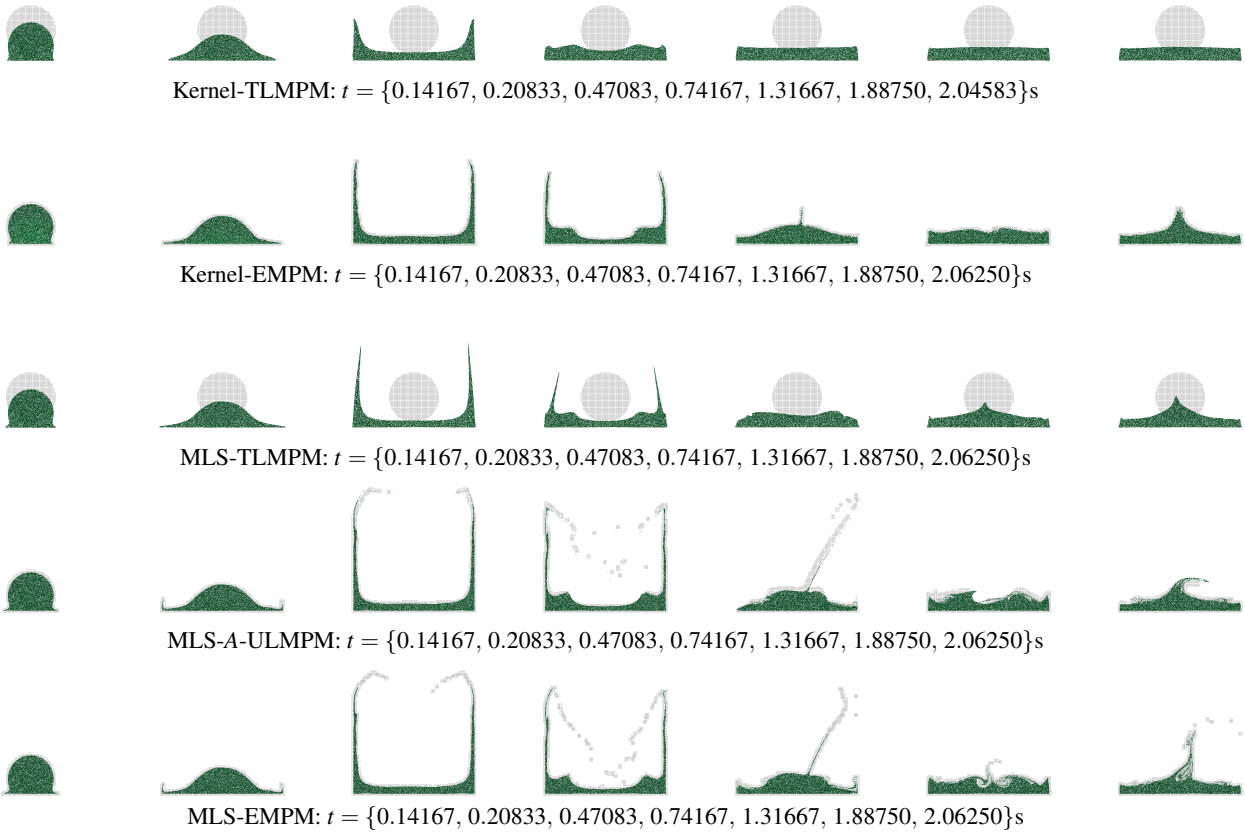


Figure 3: ***2D Droplet.*** *We integrate A-ULMPM with traditional Kernel-MPM [SSC*13] (see Appendix B) and MLS-MPM (see Section 5). Although TLMPM can eliminate numerical fracture in solid simulations as shown in Figure 2, it fails to capture very large deformations, such as fluid-like motion (see rows 1 and 3) in both Kernel-EMPM and MLS-EMPM. Our proposed A-ULMPM automatically updates configurations to produce similarly detailed dynamics as those with EMPM for fluid simulation (see rows 4 and 5). Background grid represents the configuration linking to the present particle dynamics.*

pletely remove the numerical fracture issue due to gaps between particles and grid cells. Recently, the standard MPM was reformulated with respect to the initial topology, which provides the total Lagrangian formulation for MPM (TLMPM) [dVNH20], which has been proven to completely eliminate the numerical fracture issue and cell-crossing instability and has been further extended in [dVN21] to support multi-body contacts. The central idea behind TLMPM is very similar to the Lagrange force model described by [JST17] (see also [SSIF07, WDG*19]).

3. Overview of Different Formulations for Continuum Mechanics

In this section, we briefly revisit the Lagrangian framework from continuum mechanics for describing the governing equations of motion and summarize the different formulations that can be derived depending on the choice of the reference configuration.

3.1. Lagrangian Framework

The Lagrangian $\mathcal{L}$ for holonomic systems is defined as:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{K}(\dot{\mathbf{q}}) - \mathcal{U}(\mathbf{q}) \tag{1}$$

where $\mathbf{q}$ is the generalized displacement, $\dot{\mathbf{q}}$ is the generalized velocity, $\mathcal{K}(\dot{\mathbf{q}})$ is the kinetic energy and $\mathcal{U}(\mathbf{q})$ is the potential energy. By omitting the energy due to external body forces and traction for simplicity, the kinetic and potential energies can be defined as:

$$\mathcal{K}(\dot{\mathbf{q}}) = \frac{1}{2}\int_{\mathcal{B}} \rho_0 \dot{\mathbf{q}}^T \dot{\mathbf{q}} dV, \quad \mathcal{U}(\mathbf{q}) = \int_{\mathcal{B}} \rho_0 \Psi(\mathbf{F}) dV \tag{2}$$

where ρ_0 is the material density at time t_0, $\Psi(\mathbf{F})$ denotes the Helmholtz free energy per unit mass in homogeneous materials, and $\mathbf{F}$ is the deformation gradient tensor. Consequently, the Lagrangian density function can be defined as follows:

$$\bar{\mathcal{L}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{F}) = \frac{1}{2}\rho_0 \dot{\mathbf{q}}^T \dot{\mathbf{q}} - \Psi(\mathbf{F}) \tag{3}$$

Based on the Lagrangian framework, the governing equations of motion at time t_n can be described as:

$$\frac{d}{dt}\left(\frac{\partial \bar{\mathcal{L}}}{\partial \dot{\mathbf{q}}_n}\right) - \frac{\partial \bar{\mathcal{L}}}{\partial \mathbf{q}_n} = \mathbf{0} \tag{4}$$

where

$$\frac{d}{dt}\left(\frac{\partial \bar{\mathcal{L}}}{\partial \dot{\mathbf{q}}_n}\right) = \rho_0 \ddot{\mathbf{q}}_n \tag{5}$$

MLS-EMPM: Time instants from left to right $t = \{0, 0.0005, 0.0009, 0.0016, 0.002, 0.0027\}$s

MLS-*A*-ULMPM: Time instants from left to right $t = \{0, 0.0005, 0.0009, 0.0016, 0.002, 0.0027\}$s

Figure 4: ***3D Twisting column.*** *The two ends of a rectangular beam are kinematically separated while twisting the beam. Standard MLS-EMPM (top row) suffers from severe numerical fracture. In contrast, our MLS-A-ULMPM (bottom row) nicely captures the rich twisting surface and preserves column shape.*

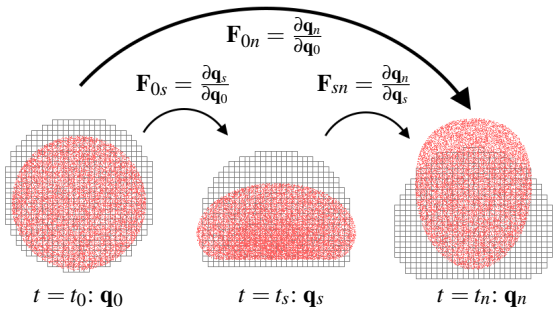


Figure 5: ***Elastic ball.*** *The deformation gradient can be described with respect to the initial configuration (total Lagrangian formulation) at time t_0 as $\mathbf{F}_{0s}$ and $\mathbf{F}_{0n}$, or with respect to an intermediate configuration at time t_s (updated Lagrangian formulation) as $\mathbf{F}_{sn}$, where $\mathbf{q}$ is the displacement.*

and

$$\frac{\partial \bar{\mathcal{L}}}{\partial \mathbf{q}_n} = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} \frac{\partial \mathbf{F}}{\partial \mathbf{q}_n}.$$

Note that the term $\partial\Psi(\mathbf{F})/\partial\mathbf{F}$ represents a stress tensor that is determined by the material constitutive model, and the term $\partial\mathbf{F}/\partial\mathbf{q}$ can be further expressed in terms of a divergence operator. These two terms together define the internal force as the material deforms.

3.2. Total Lagrangian Formulation

In this formulation, the stress and strain in the material are measured relative to the original configuration at time t_0 (see Figure 5), such that the deformation gradient tensor $\mathbf{F}$ is the displacement derivative of $\mathbf{q}_n$ with respect to $\mathbf{q}_0$. Substituting $\mathbf{F}_{0n}$ to $\partial\bar{\mathcal{L}}/\partial\mathbf{q}_n$ gives:

$$\frac{\partial \bar{\mathcal{L}}}{\partial \mathbf{q}_n} = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}_{0n}} \frac{\partial}{\partial \mathbf{q}_n} \left(\frac{\partial \mathbf{q}_n}{\partial \mathbf{q}_0} \right) = \nabla_0 \cdot \mathbb{P}$$

and have the following governing equation of motion:

$$\rho_0 \ddot{\mathbf{q}}_n = \nabla_0 \cdot \mathbb{P}_0 \tag{6}$$

where the divergence operator ∇_0 is also evaluated with respect to the original configuration at time t_0. In the total Lagrangian formulation, the stress tensor $\mathbb{P}_0$ is the *first Piola–Kirchhoff stress*.

3.3. Eulerian Formulation

In this formulation, the stress and strain in the material are measured relative to the current configuration at time t_n (see Figure 5). Thus, the expression for $\partial\bar{\mathcal{L}}/\partial\mathbf{q}$ can be expanded as follows:

$$\frac{\partial\bar{\mathcal{L}}}{\partial\mathbf{q}_n} = \frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}_{0n}}\frac{\partial\mathbf{F}_{0n}}{\partial\mathbf{q}_n} = \frac{\partial}{\partial\mathbf{q}_n}\left(\frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}_{0n}}\frac{\partial\mathbf{q}_n}{\partial\mathbf{q}_0}\right) = \nabla_n\cdot\left(\mathbb{P}_0\mathbf{F}_{0n}^T\right)$$

Besides, ρ_0 can be mapped to ρ_n using the relation $\rho_0 = J_{0n}\rho_n$, where ρ_n is the density at time t_n and $J_{0n} = \det(\mathbf{F}_{0n})$. Consequently, the Eulerian formulation gives the following equation of motion:

$$\rho_n\ddot{\mathbf{q}} = \nabla_n\cdot\mathbb{P}_n \tag{7}$$

where $\mathbb{P}_n = \mathbb{P}_0\mathbf{F}_{0n}^T/J_{0n}$ provides the definition of the *Cauchy stress*.

3.4. Adaptively Updated Lagrangian Formulation

A general formulation for measuring stress and strain with respect to an arbitrary reference configuration at time t_s has been derived in [ZTNZ77, Sha18]. In this formulation, the expression for $\partial\bar{\mathcal{L}}/\partial\mathbf{q}$ can be expanded as follows:

$$\frac{\partial\bar{\mathcal{L}}}{\partial\mathbf{q}_n} = \frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}_{0n}}\frac{\partial\mathbf{F}_{0n}}{\partial\mathbf{q}_n} = \frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}_{0n}}\frac{\partial\mathbf{F}_{0n}}{\partial\mathbf{F}_{sn}}\frac{\partial\mathbf{F}_{sn}}{\partial\mathbf{q}_n} = \nabla_s\cdot\left(\mathbb{P}_0\mathbf{F}_{0s}^T\right) \tag{8}$$

Similar to the Eulerian formulation, we map ρ_0 to ρ_s using the relation $\rho_s = J_{0s}\rho_s$, where ρ_s represents the density at time t_s and $J_{0s} = \det(\mathbf{F}_{0s})$. Consequently, the adaptively updated Lagrangian formulation gives the following governing equation of motion:

$$\rho_s\ddot{\mathbf{q}} = \nabla_s\cdot\mathbb{P}_s \tag{9}$$

where $\mathbb{P}_s = \mathbb{P}_0\mathbf{F}_{0s}^T/J_{0s}$ defines a stress measured at time t_s. By setting $s = 0$ and using the defining properties of the initial configuration $J_{00} = 1$ and $\mathbf{F}_{00} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, equation (9) recovers the total Lagrangian formulation in equation (6). Likewise, setting $s = n$ yields the Eulerian formulation in equation (7).

4. Method Overview

Our method introduces a nodal discretized Lagrangian into hybrid grid-to-particle discretization for MPM and takes advantage of the zero cell-crossing error and zero numerical fracture properties of TLMPM. In contrast to prior MPM formulations (see equations (6) and (7)), our method adopts adaptive reference configurations (see grids at time t_s in Figure 5) to measure stress, strain, interpolation kernels and their derivatives. Data from particles is first transferred to grids (P2G) in their latest configuration. Next, the forces are evaluated and the velocities are updated on these grids. Subsequently, the updated grid velocities are interpolated back to the particles (G2P) and used to update the particle positions following the APIC method [JSS*15]. The velocity gradients and deformation gradients are then updated on the particles leveraging information on the grids. Our method uses a novel criterion (see equation (33)) for automatically determining when to update the grid configuration. This prevents data transfers between the particles and grids from accumulating errors as the interpolation functions are only updated when necessary. Figure 6 provides a high-level overview of our method and the essential algorithmic steps are summarized below:

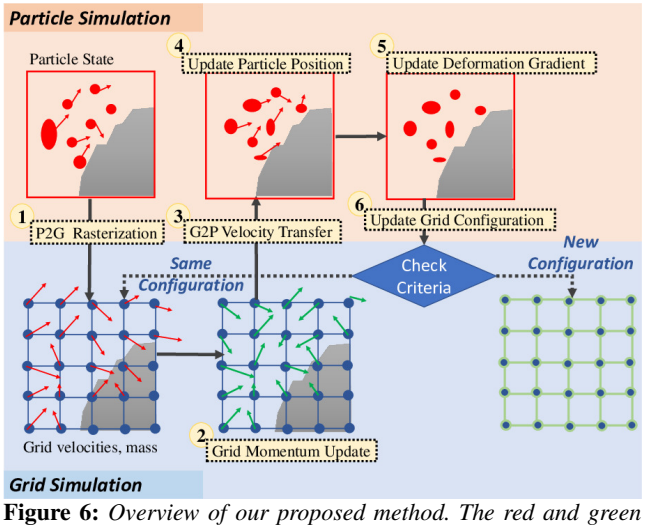


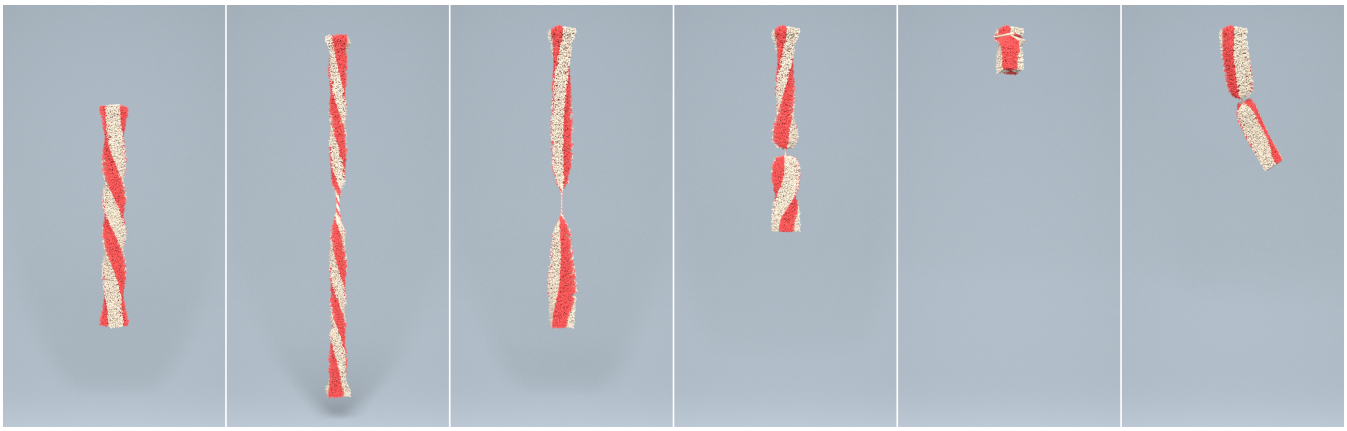
Figure 6: *Overview of our proposed method. The red and green arrows represent the velocity and force vectors at the nodes of the background grid. The blue grids represent the previous configuration while the green grids denote the new configuration according to our criterion for updating the state.*

1. **P2G Rasterization:** Velocities on particles are reconstructed by first-order Taylor expansion, and mass and momentum from particles at time t_n are transferred to grids at time t_s.
2. **Grid Momentum Update:** Grid momentum is updated using explicit or implicit schemes. Note that forces are evaluated with respect to the latest configuration at time t_s.
3. **G2P Velocity Transfer:** Use APIC [JSS*15] to transfer velocities from the grid to particles.
4. **Update Particle Positions:** Particle positions are updated with their new velocities.
5. **Update Particle Deformation Gradients:** Use the MLS gradient operator to update the particle deformation gradient and account for plasticity, if it occurs.
6. **Update Grid Configuration:** Check if the deformation is extreme according to the update criterion in equation (33). In case of a large deformation, update weights between particles and the grid and the $\mathbb{K}$ matrices on particles.

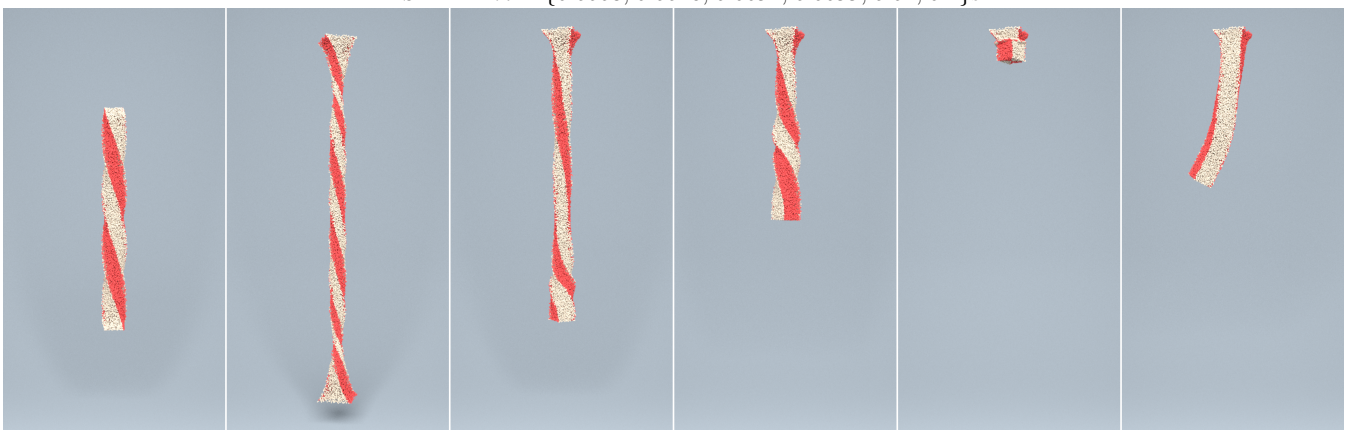
4.1. Terminology

We show integration of our adaptively updated Lagrangian formulation with Kernel-MPM [SSC*13] in Appendix B and MLS-MPM [HFG*18] in Section 5. In Kernel-MPM, the interpolation is performed using shape functions and stress divergence is evaluated by derivatives of the shape function, while MLS-MPM utilizes APIC particle-to-grid velocity rasterization and uses MLS shape functions to derive the internal force term. Using these definitions, our terminology for different methods is provided as follows:

1. **Kernel-MPM**: In kernel-MPM, we have kernel-EMPM and kernel-TLMPM represent the kernel-MPM in Eulerian and total Lagrangian frameworks, respectively. Kernel-EMPM was presented in [SSC*13] and kernel-TLMPM was present in [dVNH20]. These two methods can be recovered in our



MLS-EMPM: $t = \{0.0008, 0.0026, 0.0032, 0.0035, 0.04, 0.1\}$s



MLS-*A*-ULMPM: $t = \{0.0008, 0.0026, 0.0032, 0.0035, 0.04, 0.1\}$s

Figure 7: *The kinematic constraint on one end of the beam is released after a certain time. MLS-EMPM (row 1) fails to recover from the twisting deformation, while MLS-A-ULMPM (row 2) produces realistic elastic response and recovers from the elastic deformation induced by the pulling and twisting motion.*

Table 1: *Physical quantities stored on particles and grid nodes.*

Particle	Description	Grid
$\mathbf{q}_p$	position	$\mathbf{q}_i$
$\mathbf{v}_p$	velocity	$\mathbf{v}_i$
$\mathbf{F}_p^{0n}$	deformation gradient at t_n with respect to configuration at t_0	$--$
$\mathbf{F}_p^{0s}$	deformation gradient at t_s with respect to configuration at t_0	$--$
$\mathbf{F}_p^{sn}$	deformation gradient at t_n with respect to configuration at t_s	$--$
$\mathbb{P}_p^s$	Stress tensor	$--$
J_p^{sn}	volume change at t_n with respect to configuration at t_s	$--$
$--$	force	$\mathbf{f}_i$
V_p	volume	$--$
m_p	mass	m_i

kernel-*A*-ULMPM framework by setting $s = n$ for EMPM and $s = 0$ for TLMPM.

2. **MLS-MPM**: In MLS-MPM, we have MLS-EMPM and MLS-TLMPM represent MLS-MPM in Eulerian and total Lagrangian frameworks, respectively. MLS-EMPM was presented in MLS-MPM [HFG*18]. Besides, MLS-EMPM and MLS-TLMPM can be recovered in MLS-*A*-ULMPM by setting $s = n$ for MLS-EMPM and $s = 0$ for MLS-TLMPM.

5. *A*-ULMPM Formulation

Starting from a nodal Lagrangian formulation, we derive an alternate expression for equation (6) in the hybrid particle-grid framework of MPM. Interestingly, our formulation leads to a new MLS-MPM method (MLS-*A*-ULMPM) whose variant recovers MLS-MPM [HFG*18] in the Eulerian setting. We use subscript i to denote quantities on grid nodes, subscript p to denote quantities on particles, and subscript s to denote the intermediate configuration map. Table 1 summarizes the notation used in this section.

5.1. Grid Setting

We use a global grid that covers the entire computational domain. Each object has its own *configuration map* ϕ that maps points $\mathbf{x}$ within the object to specific locations $\phi(\mathbf{x})$ in the global grid. To reduce the associated memory overhead, we implement the global grid using sparsity aware data structures [SABS14].

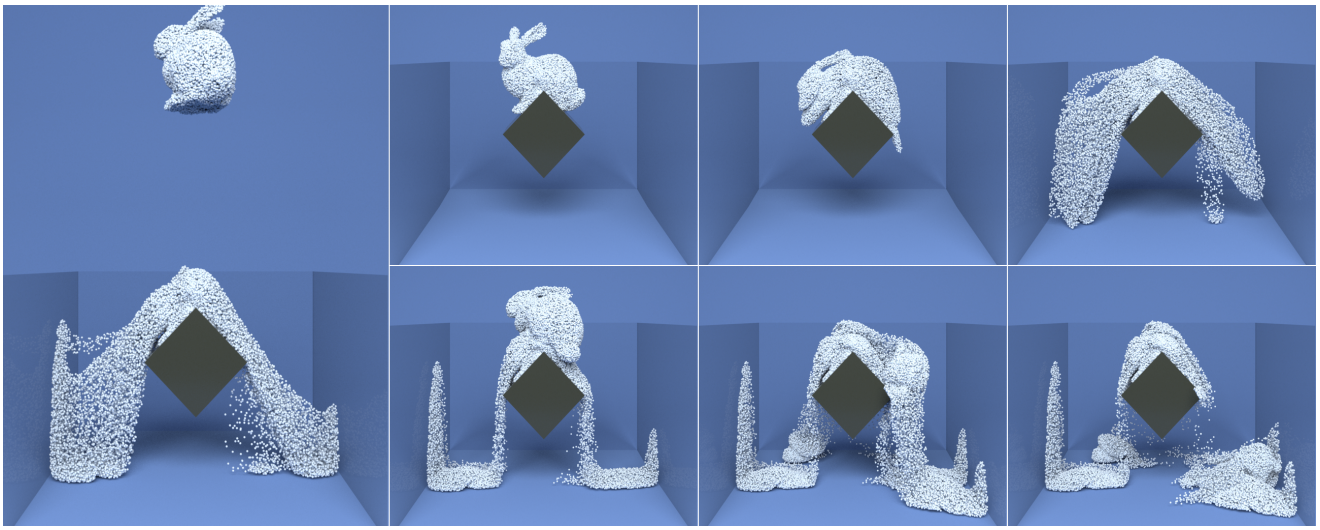


Figure 8: ***Snow bunnies break over a wedge.*** *Our A-ULMPM framework captures rich interactions of several snow bunnies smashing and scattering after falling on a solid wedge, demonstrating the extreme deformations that our method can capture, similar to its Eulerian counterparts proposed in prior works.*

5.2. P2G Rasterization

In contrast to EMPM, the velocity gradients $\nabla_s \mathbf{v}_p$ and weights W_{ip}^s are all evaluated with respect to the configuration map ϕ at time t_s. We use a first order Taylor expansion to reconstruct particle velocities and rasterize particle momentum to the grid as follows:

$$m_i^s \mathbf{v}_i^n = \sum_p m_p \left(\mathbf{v}_p^n + \nabla_s \mathbf{v}_p \mathbf{r}_{ip}^s\right) W_{ip}^s \tag{10}$$

where $\mathbf{r}_{ip}^s = \left(\mathbf{q}_p^s - \mathbf{q}_i^s\right)$ and $\nabla_s \mathbf{v}_p^n = \frac{\partial \mathbf{v}_p^n}{\partial \mathbf{q}_s}$, which is further expanded out in equation (31). The mass/velocity at grid nodes are given by:

$$m_i^s = \sum_p m_p W_{ip}^s, \quad \mathbf{v}_i^n = \frac{m_i^s \mathbf{v}_i^n}{\sum_p m_p W_{ip}^s} \tag{11}$$

We also rasterize particle positions to the grid to simplify the theoretical derivations for the deformation gradient in equation (29) and internal forces in equation (19) as shown below:

$$\mathbf{q}_i^n = \frac{\sum_p \mathbf{q}_p^n W_{ip}^s}{\sum_p W_{ip}^s} \tag{12}$$

The idea of introducing a first-order Taylor expansion to enhance velocity rasterization is not new and can also be found in APIC [JSS*15] and MLS-MPM [HFG*18].

5.3. Grid Momentum Update

5.3.1. Nodal Lagrangian

We interpolate the Lagrangian $\mathcal{L}_i^{n+1}$ at grid node i using values associated with its nearby particles p as follows:

$$\mathcal{L}_i^{n+1} = \sum_p \left[\frac{1}{2}\rho_p^0 \left(\dot{\mathbf{q}}_p^{n+1}\right)^T \dot{\mathbf{q}}_p^{n+1} - \Psi\left(\mathbf{F}_p^{0n+1}\right)\right] W_{ip}^s V_p^s \tag{13}$$

Substituting the expression for $\mathcal{L}_i^{n+1}$ to equations (4) and (8) gives:

1. Kinetic term:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial \mathcal{L}_i^{n+1}}{\partial \dot{\mathbf{q}}_i^{n+1}}\right) &= \sum_p \rho_p^0 \dot{\mathbf{q}}_p^{n+1} W_{ip}^s V_p^s = \sum_p \rho_p^0 W_{ip}^s V_p^s \ddot{\mathbf{q}}_p^{n+1} \\ &= m_i^s \ddot{\mathbf{q}}_p^{n+1}\end{aligned} \tag{14}$$

2. Deformation term:

$$\begin{aligned}\frac{\partial \mathcal{L}_i^{n+1}}{\partial \mathbf{q}_i^n} &= -\sum_p \frac{\partial \Psi(\mathbf{F}_p^{0n+1})}{\partial \mathbf{q}_i^n} W_{ip}^s V_p^s \\ &= -\sum_p \mathbb{P}_p^s \frac{\partial (\mathbf{F}_p^{s(n+1)})^T}{\partial \mathbf{q}_i^s} W_{ip}^s V_p^s\end{aligned} \tag{15}$$

where $\mathbb{P}_j^s = \mathbb{P}_j^0 (\mathbf{F}_j^{0s})^T / J_j^{0s}$.

Thus, the equation of motion for grid node i at time t_n is given by:

$$m_i^s \ddot{\mathbf{q}}_i^{n+1} + \sum_j \mathbb{P}_p^s (\mathbf{F}_p^{0s})^T \frac{\partial (\mathbf{F}_p^{s(n+1)})^T}{\partial \mathbf{q}_i^s} W_{ip}^s V_j^s = \mathbf{0} \tag{16}$$

The reader may have noticed that an explicit expression of equation (16) relies on a concrete formulation of $\mathbf{F}_p^{s(n+1)}$ and its derivative $\partial(\mathbf{F}_p^{s(n+1)})/\partial \mathbf{q}_i^s$.

5.3.2. MLS-based gradient operator

We use the gradient operator based on *moving least squares* (MLS) that was proposed in [XZT19] (see Appendix A) that locally minimizes the error of a certain position over its neighborhood. Following the same notation in equation (16), the deformation gradient of particle p at time t_n relative to an arbitrary configuration map at time t_s is given as:

$$\mathbf{F}_p^{s(n+1)} = \left[\sum_k (\mathbf{q}_k^{n+1} - \mathbf{q}_p^{n+1}) \otimes \mathbf{r}_{pk}^s W_{pk}^s\right] \left[\sum_k \mathbf{r}_{pk}^s \otimes \mathbf{r}_{pk}^s W_{pk}^s\right]^{-1} \tag{17}$$

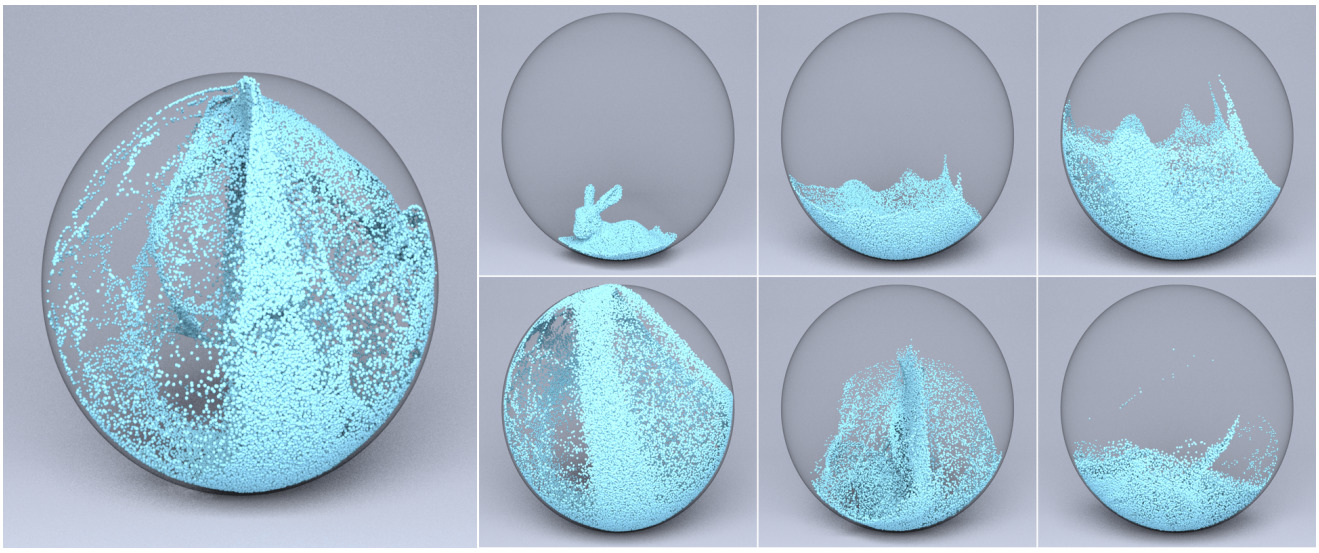


Figure 9: ***Liquid bunny splash inside a ball.*** *A liquid bunny is dropped inside a spherical container and undergoes vibrant and dynamic splashes, demonstrating that our proposed A-ULMPM framework can capture the rich motion of incompressible fluids similar to existing Eulerian approaches proposed in prior works.*

where $\mathbf{r}^s_{pk} = \mathbf{q}^s_k - \mathbf{q}^s_p$.

5.3.3. Evaluation of $\partial \mathbf{F}_p^{s(n+1)} / \partial \mathbf{q}_i^s$

Based on equation (42) in Appendix A, we evaluate the derivative of $\mathbf{F}_p^{sn}$ with respect to $\mathbf{q}_i^s$ as:

$$\frac{\partial \mathbf{F}_p^{s(n+1)}}{\partial \mathbf{q}_i^s} = \sum_k \left[\left(\delta_{ik}^{n+1} - \delta_{ip}^{n+1} \right) \otimes \mathbf{r}^s_{pk} W^s_{pk} \right] \left[\sum_k \mathbf{r}^s_{pk} \otimes \mathbf{r}^s_{pk} W^s_{pk} V^s_k \right]^{-1} \quad (18)$$

5.3.4. MPM Equation of Motion with Adaptively Updated Lagrangian

Substituting equation (18) to equation (16) yields the following equation of motion for grid node i at time t_n:

$$m_i^s \ddot{\mathbf{q}}_i^{n+1} + \sum_p \left[\mathbb{P}_p^s \mathbb{K}_p^s + \mathbb{P}_i^s \mathbb{K}_i^s \right] \mathbf{r}^s_{ip} W^s_{ip} V^s_p = \mathbf{0} \quad (19)$$

where $\mathbb{P}_p^s = \mathbb{P}_p^0 (\mathbf{F}_p^{s(n+1)})^T$ and $\mathbb{P}_i^s = \mathbb{P}_i^0 (\mathbf{F}_i^{s(n+1)})^T$. $\mathbb{K}_p^s$ and $\mathbb{K}_i^s$ are given by:

$$\mathbb{K}_p^s = \left(\sum_i \mathbf{r}^s_{pi} \otimes \mathbf{r}^s_{pi} W^s_{pi} V^s_i \right)^{-1}, \quad \mathbb{K}_i^s = \left(\sum_p \mathbf{r}^s_{ip} \otimes \mathbf{r}^s_{ip} W^s_{ip} V^s_p \right)^{-1}.$$

We take advantage of equation (12) to simplify the term $\mathbb{P}_i^0 (\mathbf{F}_i^{0s})^T \mathbb{K}_i^s$ in equation (19) as follows:

$$\sum_p \mathbb{P}_i^s \mathbb{K}_i^s \mathbf{r}^s_{ip} W^s_{ip} V^s_p = \mathbb{P}_i^s \mathbb{K}_i^s \sum_p \mathbf{r}^s_{ip} W^s_{ip} V^s_p = 0$$

Setting $V_p^s = J_p^s V_p^0$, equation (19) can be rewritten as follows:

$$m_i^s \ddot{\mathbf{q}}_i^{n+1} + \sum_p \mathbb{P}_p^0 (\mathbf{F}_p^{0s})^T \mathbb{K}_p^s \mathbf{r}^s_{ip} W^s_{ip} V^0_p = \mathbf{0} \quad (20)$$

Equation (20) is a general MPM formulation that allows the use of arbitrary intermediate configurations. It spans from total Lagrangian formulations to updated Lagrangian formulations. Specifically:

1. Setting $s = 0$ yields total Lagrangian MPM:

$$m_i^0 \ddot{\mathbf{q}}_i^{n+1} + \sum_p \mathbb{P}_p^0 \mathbb{K}_p^0 \mathbf{r}^0_{ip} W^0_{ip} V^0_p = \mathbf{0} \quad (21)$$

2. Setting $s = n+1$ yields Eulerian MPM:

$$m_i^n \ddot{\mathbf{q}}_i^{n+1} + \sum_p \mathbb{P}_p^0 (\mathbf{F}_p^{0(n+1)})^T \mathbb{K}_p^{n+1} \mathbf{r}^{n+1}_{ip} W^{n+1}_{ip} V^0_p = \mathbf{0} \quad (22)$$

In the total Lagrangian formulation, there is no need to update W_{ip}, $\mathbf{r}_{ip}$, m_i, and $\mathbb{K}_p$ matrices in equation (21), while they require an update at every time step in the Eulerian setting (see equation (22)).

5.4. Grid Velocity and Position Update

With explicit time stepping, the velocity is updated as $\mathbf{v}_i^{n+1} = \hat{\mathbf{v}}_i^{n+1}$, where $\hat{\mathbf{v}}_i^{n+1}$ is given by:

$$\hat{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n - \frac{\Delta t}{m_i} \sum_p \mathbb{P}_p^0 (\mathbf{F}_p^{0s})^T \mathbb{K}_p^s \mathbf{r}^s_{ip} W^s_{ip} V^0_j \quad (23)$$

For a semi-implicit update, we follow [SSC*13, HFG*18] and take an implicit step on the velocity update by utilizing the Hessian of $\mathcal{L}^{n+1}$ with respective to $\mathbf{q}_i^{n+1}$. The action of this Hessian on an arbitrary increment $\delta \mathbf{q}^s$ is given as follows:

$$-\delta \mathbf{f}_i = \sum_p V_p^0 \mathbb{A}_p^s (\mathbf{F}_p^{0s})^T \mathbb{K}_p^s \mathbf{r}^s_{ip} W^s_{ip} V^0_j \quad (24)$$

where $\mathbb{A}_p^s$ is given by:

$$\mathbb{A}_p^s = \frac{\partial^2 \mathcal{L}^{n+1}}{\partial \mathbf{F}_p^{0s} \partial \mathbf{F}_p^{0s}} : \sum_j \delta \mathbf{q}_j^s \mathbb{K}_p^s \mathbf{r}^s_{ip} W^s_{ip} V^0_j \mathbf{F}_p^{0s}. \quad (25)$$

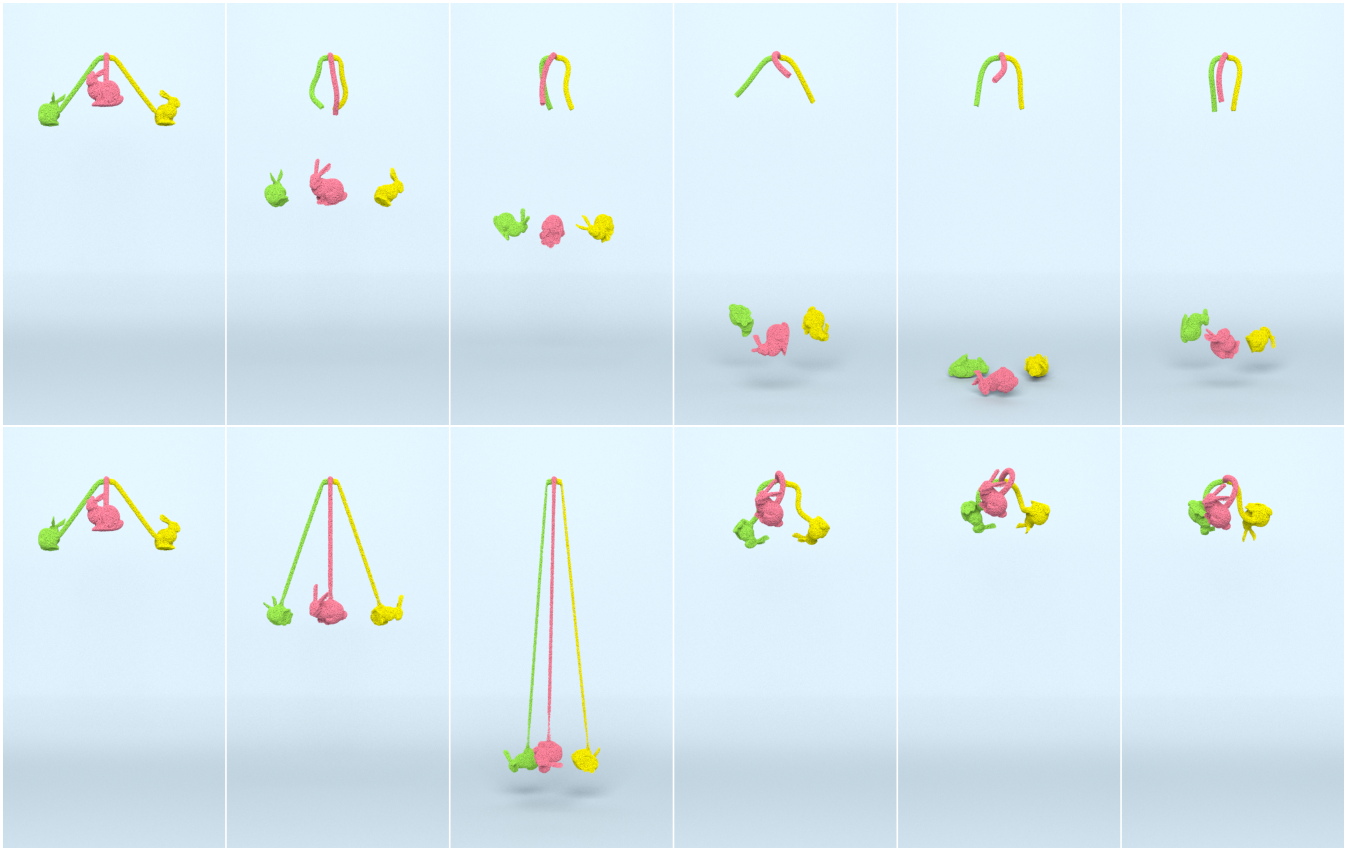


Figure 10: ***Hyperelastic bunny yo-yo.*** *Under the effects of gravity, a hyperelastic bunny yo-yo breaks mid-way due to numerical fracture, when simulated with MLS-EMPM (top), while MLS-A-ULMPM (bottom) robustly captures the stretching motion of the elastic cord to pull the bunnies back upwards.*

We linearize the implicit system with one step of Newton's method, which provides the following symmetric system for $\hat{\mathbf{v}}_i^{n+1}$:

$$\sum_j \left(\mathbf{I}\delta_{ij} + \frac{\Delta t^2}{m_i} \frac{\partial^2 \mathcal{L}^{n+1}}{\partial \mathbf{q}_i^s \partial \mathbf{q}_j^s} \right) \mathbf{v}_j^{n+1} = \hat{\mathbf{v}}_i^{n+1} \tag{26}$$

where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{v}}_i^{n+1}$ is given in equation (23). We update rasterized positions at grid nodes as shown below:

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n + \Delta t \mathbf{v}_i^{n+1} \tag{27}$$

5.5. G2P Velocity and Position Transfer

We project $\mathbf{q}_i^{n+1}$ to the global grid and process grid-based collisions [SSC*13] to compute $\mathbf{v}_i^{n+1}$, which is transferred back to particles using the APIC method [JSS*15]. The specific updates for $\mathbf{q}_p^{n+1}$ and $\mathbf{v}_p^{n+1}$ are given below:

$$\mathbf{v}_p^{n+1} = \sum_i \mathbf{v}_i^{n+1} W_{pi}^s, \quad \mathbf{q}_p^{n+1} = \mathbf{q}_p^n + \Delta t \mathbf{v}_p^{n+1} \tag{28}$$

5.6. Update Particle Deformation Gradients

We first update particle deformation gradients with respect to the latest configuration at time t_s and use the following MLS-based gradient operator (see equation (17)):

$$\mathbf{F}_p^{s(n+1)} = \frac{\partial \mathbf{q}_p^{n+1}}{\partial \mathbf{q}_p^s} = \left(\sum_i (\mathbf{q}_i^{n+1} - \mathbf{q}_p^{n+1}) \otimes \mathbf{r}_{ip}^s W_{pi}^s V_i^s \right) \mathbb{K}_p^s \tag{29}$$

Substituting $\mathbf{q}_i^{n+1} = \mathbf{q}_i^n + \Delta t \mathbf{v}_i^{n+1}$ and $\mathbf{q}_p^{n+1} = \mathbf{q}_i^n + \Delta t \mathbf{v}_p^{n+1}$ to equation (29) gives:

$$\begin{aligned} \mathbf{F}_p^{s(n+1)} &= \sum_i \left[\left(\mathbf{q}_i^n + \mathbf{v}_i^{n+1} \Delta t \right) - \left(\mathbf{q}_p^n + \Delta t \mathbf{v}_p^{n+1} \right) \right] \otimes \mathbf{r}_{pi}^s W_{pi}^s V_i^s \mathbb{K}_p^s \\ &= \mathbf{F}_p^{sn} + \Delta t \nabla_s \mathbf{v}_p^{n+1} \end{aligned} \tag{30}$$

where $\nabla_s \mathbf{v}_p^{n+1}$ is given by:

$$\nabla_s \mathbf{v}_p^{n+1} = \sum_i (\mathbf{v}_i^{n+1} - \mathbf{v}_p^{n+1}) \otimes \mathbf{r}_{pi}^s W_{pi}^s V_i^s \mathbb{K}_p^s \tag{31}$$

Using the chain rule, the update for $\mathbf{F}_p^{0(n+1)}$ is given by:

$$\mathbf{F}_p^{0(n+1)} = \frac{\partial \mathbf{q}_p^{n+1}}{\partial \mathbf{q}_p^s} \frac{\partial \mathbf{q}_p^s}{\partial \mathbf{q}_p^0} = \mathbf{F}_p^{s(n+1)} \mathbf{F}_p^{0s} \tag{32}$$

5.7. Update Grid Configuration Map

We designed the configuration update criterion based on an intuitive assumption that more severe deformation is accompanied with

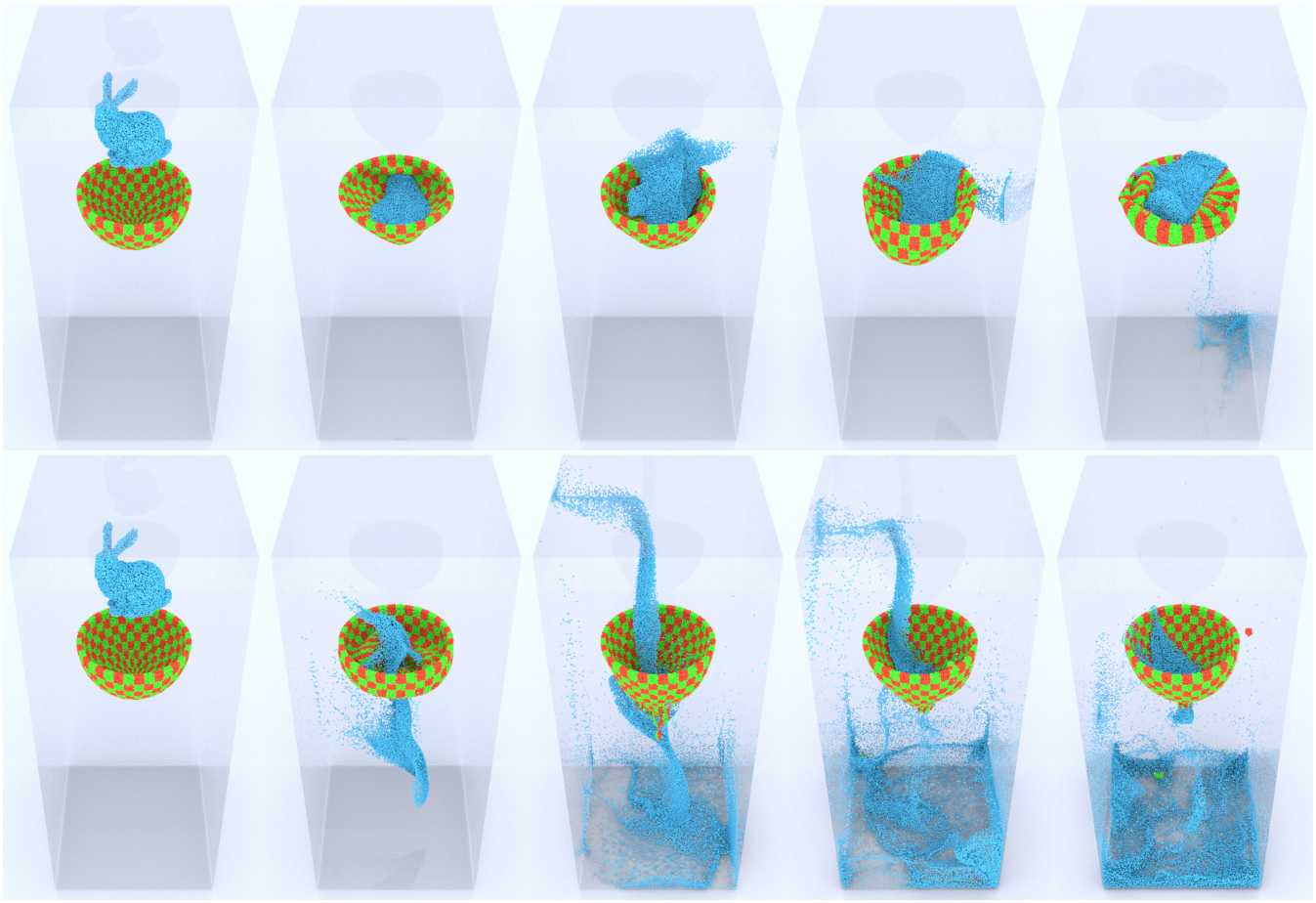


Figure 11: ***Liquid bunny falling on a hyperelastic bowl.*** *(Top) Our proposed A-ULMPM framework can robustly capture the vivid dynamic responses of fluid-solid interactions and preserves the bowl shape. (Bottom) In contrast, the bowl fractures and fails to hold the water when simulated with MLS-EMPM.*

large change of J in the next time step t^{n+1} with respect to the latest configuration t^s. Using this observation, our criterion is defined as a measure for the amount of particle deformation as follows:

$$\delta J_p = \|J_p^{s(n+1)} - J_p^{ss}\| \tag{33}$$

where $J_p^{sn+1} = \det(\mathbf{F}_p^{s(n+1)})$ and $J_p^{ss} = 1$. We mark particles with $\delta J_p \geq \varepsilon$ as indicators where large deformation is taking place and count the total amount of marked particles (n_{mp}). If $n_{mp}/n_p \geq \eta$, where n_p is the total number of particles, we update W_{ip}^s, $\mathbb{K}_p^s$, and $\mathbf{F}_p^{0s}$. Otherwise, these variables remain the same until a new update occurs. ε and η are user-defined parameters to adjust the update frequency. Intuitively, a greater ε or η implies that only when a larger percentage of particles are undergoing significant enough deformation will the configuration be updated, thus more computational time is needed for weight update. Theoretically, the run time will be bounded by EMPM as the the upper bound,and TLMPM as the lower bound respectively, as the EMPM updates the weight per time step while the TLMPM only needs the initial weight. Figure 12 visually demonstrates our update criterion, where large values of δJ_p represent large deformations.

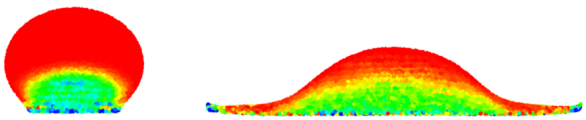


Figure 12: ***Falling droplet.*** *Our configuration update criterion can capture relatively severe deformation with respect to the last updated configuration. The values of δJ_p at particles are visualized with colors from red to blue for increasing values.*

Other configuration update criteria, such as those in [ZZL*17, QZG*19], can also be used to integrate with our *A*-ULMPM framework. For concreteness, we compare our configuration update criterion with that in [ZZL*17] that is described by the equation:

$$K(\mathbf{F}_p) = \frac{1}{\|\mathbf{F}_p^{-1}\|_p \|\mathbf{F}_p\|_p} \tag{34}$$

where $\|\cdot\|_F$ is the Frobenius norm. Larger values of $K(\mathbf{F}_p)$ denote better conditioned systems and vice-versa. The formulation in equation (34) uses the deformation gradient with respect to the original configuration, while our criterion uses the deformation gra-

dient with respect to the latest updated configuration. By doing this, our formulation is more intuitive and more reasonable. For example, in the 2D droplet case (see Figure 3) severe deformation of the drop continues after the droplet hits the wall. If we used equation (34) instead, the configuration would have to be updated every time step as the deformation is always large in comparison to the initial state. Our proposed configuration update criterion is developed with respect to the previous configuration that may have already experienced large deformation. Consequently, a slight change of relative deformation does not require the update. This is the major reason that our method is computationally efficient. Besides, the physical meaning of equation (34) is unclear as when the object experiences rigid motion $K(\mathbf{F}_p) = 0.5$, while our criterion $\delta J_p = 0$, which reflects a state of no deformation.

5.8. Similarities with MLS-EMPM and APIC

While our *A*-ULMPM framework is new to computer graphics, setting $s = n$ for the configuration map shares some similarities with MLS-EMPM [HFG*18] and APIC [JSS*15], including momentum preservation, volume change in simulations of incompressible materials. Instead of detailed comparison, we mathematically prove that the MLS-gradient of velocity is equivalent to the APIC-gradient of velocity by leveraging the properties of interpolation weights $\sum_i \mathbf{r}_{pi} W_{pi} = \mathbf{0}$ [JST17] as follows:

$$\begin{aligned}\nabla \mathbf{v}_p &= \sum_i (\mathbf{v}_i - \mathbf{v}_p) \otimes \mathbf{r}_{pi} W_{ip} V_i \mathbb{K}_p \\ &= \underbrace{\sum_i \mathbf{v}_i \otimes \mathbf{r}_{pi} W_{ip} V_i \mathbb{K}_p}_{\textbf{APIC:}\,\nabla \mathbf{v}_p} - \underbrace{\mathbf{v}_p \otimes \sum_i \mathbf{r}_{pi} W_{ip} V_i \mathbb{K}_p}_{\mathbf{0}}\end{aligned} \tag{35}$$

Our internal force in equation (23) has the same pattern as that in original MLS-EMPM [HFG*18], which is derived from the weak-form element-free Galerkin (EFG) framework. The $\mathbb{K}_p$ matrices are simplified by $\frac{4}{\Delta x^2}\mathbf{I}$ for quadratic W_{ip} and $\frac{3}{\Delta x^2}\mathbf{I}$ for cubic W_{ip}. This simplification comes from the properties of splines, and is not generally true for other interpolation functions.

6. Results

Accompanying this article, we open-source our code for running 3D examples with our proposed *A*-ULMPM framework (see Section 5). MLS-based MPM, including MLS-EMPM, MLS-TLMPM, and MLS-*A*-ULMPM, was applied to simulate all our 3D simulations. Besides the advantage of removing numerical fracture and the cell-crossing instability in solid and fluid simulations, *A*-ULMPM has the practical convenience of using the same numerical implementation to adaptively update the configuration map, without having to switch between TLMPM and EMPM. We use the example of a falling elastic ball (see Figure 5) to evaluate the computational cost of explicit (see equation (23)) and implicit (see equation (26)) Euler schemes. As shown in the first two rows of Table 2, the computational cost of these two schemes are comparable. Given the low stiffness in the materials, we utilize the explicit Euler scheme in all our 3D examples and did not experience any need for excessively small time steps. For our 3D solid simulations, we set $\varepsilon = 0.5$ and $\eta = 0.1$, and observed that no configuration update was required, showing that *A*-ULMPM inherits the advantage of TLMPM for solid simulation. Due to foreseeable large deformations that arise when simulating fluid-like materials, such as water splashes and snow scattering, we set $\varepsilon = 0.01$ and $\eta = 0.01$ to update the configuration maps more frequently. Even so, our *A*-ULMPM scheme reduces the configuration update overhead by $4.27\times - 31.52\times$ in fluid-like simulations (see Table 3), compared to standard EMPM. Table 2 summarizes the specific timings for all our examples.

Table 2: *All simulations were run on Machine 1: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz and Machine 2: Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz. Simulation time is measured in average seconds per time step.* ***Grid:*** *The number of occupied voxels in the background sparse grid.* ***Particle:*** *The total number of MPM particles in the simulation.*

Simulation	Time	Machine	Grid	Particle
2D Elastic ball (Fig.5)	0.0148	1	16.4K	20K
2D Elastic ball (implicit)	0.0165	1	16.4K	20K
2D Rotation (Fig.2(1st row))	0.0571	1	16.4K	42K
2D Rotation (Fig. 2(2nd row))	0.0571	1	16.4K	42K
2D Rotation (Fig. 2(3rd row))	0.0571	1	16.4K	42K
2D Droplet (Fig. 3(1st row))	0.0154	1	16.4K	5K
2D Droplet (Fig. 3(2nd row))	0.0108	1	16.4K	5K
2D Droplet (Fig. 3(3rd row))	0.0194	1	16.4K	5K
2D Droplet (Fig. 3(4th row))	0.0146	1	16.4K	5K
2D Droplet (Fig. 3(5th row))	0.0199	1	16.4K	5K
Twisting bar (Fig.4(1st row))	0.627	1	1.0M	193.3K
Twisting bar (Fig.4(2nd row))	0.624	1	1.0M	193.3K
Twisting bar (Fig.7(1st row))	0.125	2	131.1K	48.3K
Twisting bar (Fig.7(2nd row))	0.124	2	131.1K	48.3K
Stretchy yo-yo (Fig.10(1st row))	1.314	2	4.2M	179.3K
Stretchy yo-yo (Fig.10(2nd row))	1.310	2	4.2M	179.3K
Snow bunny (Fig.8)	0.965	2	524.3K	116.3K
Snow bunny (EMPM in video)	0.971	2	524.3K	116.3K
Water bunny (Fig.9)	0.341	2	2.1M	96.3K
Water bunny (EMPM in video)	0.381	2	2.1M	96.3K
FSI (Fig.11(1st row))	0.934	1	4.2M	169.1K
FSI (Fig.11(2nd row))	0.939	1	4.2M	169.1K

Table 3: *Configuration update cost in fluid-like simulations.* ε *and* η*: user-defined parameters to adjust the update frequency.* τ*: average update times per 104 time steps.* ***Cost****: average run-time.*

Simulation	ε	η	τ	Cost
Snow bunny (*A*-ULMPM in Fig.8)	0.01	0.01	24.33	0.112
Snow bunny (EMPM in video)	N/A	N/A	104	0.479
Water bunny (*A*-ULMPM in Fig.9)	0.01	0.01	3.31	0.021
Water bunny (EMPM in video)	N/A	N/A	104	0.662

6.1. 2D Simulations for Solids and Fluids

6.1.1. Numerical fracture and cell-crossing instability

We simulated a 2D rotating hyperelastic plate to demonstrate that EMPM suffers from the cell-crossing instability, which leads to severe numerical fractures, while TLMPM can completely eliminate these artifacts. Figure 2 shows that our *A*-ULMPM framework and TLMPM integrated with MLS-MPM captures the appealing hyperelastic rotation, preserving the angular momentum for long simulation periods while completely eliminating numerical fractures.

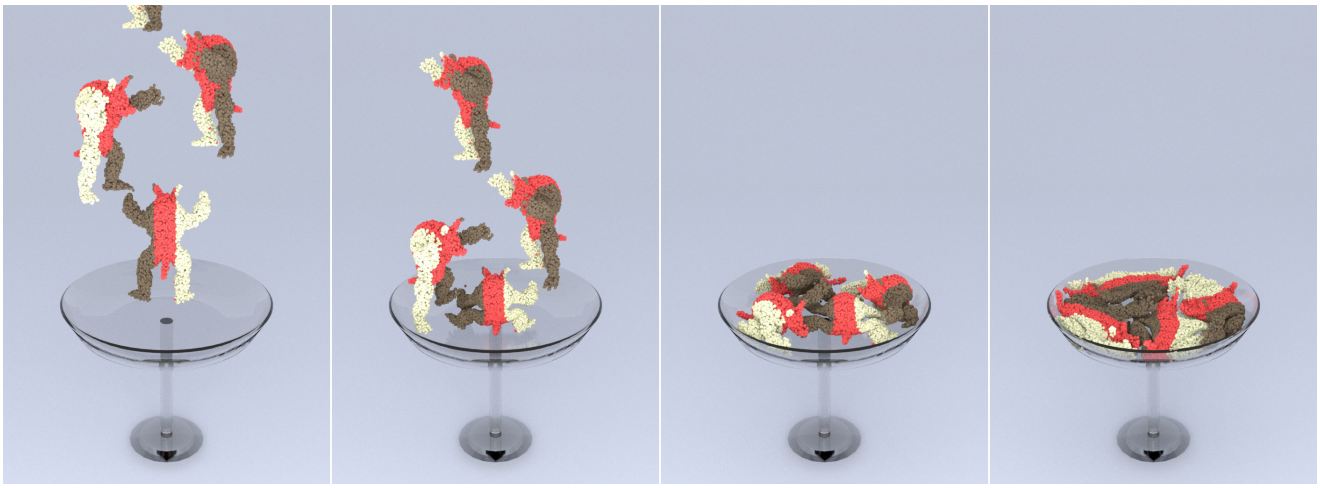


Figure 13: ***Simulation of falling viscous armadillos.*** *Our method captures rich interactions and surface details as four armadillos that exhibit Newtonian viscosity are dropped one by one on top of each other.*

However, traditional MLS-MPM [HFG*18] that uses an Eulerian approach suffers from severe numerical fractures when large deformations occur.

We quantitatively evaluate the spatial accuracy of our method on the rotating hyperelastic plate example by varying the grid resolution Δx. We utilized numerical results with resolution 256×256 as the benchmark solution (see Figure 2) and discretize sampling examples with spatial resolution $2^i \times 2^i$, $i \in \{2, \ldots, 7\}$ for the grid. We fixed the particle number $n_p = 41943$ for each case and ran simulations up to a total simulation time of $0.025s$ with a time step size of $10^{-4}s$. The error for numerical simulations is defined as:

$$E = \sqrt{\frac{|\phi_i - \phi_8|^2}{n_p}} \tag{36}$$

where ϕ_i is the variable evaluated by lower grid resolutions, while ϕ_8 is the value at resolution 256^2 (benchmark). Figure 14 shows the convergence plots for the average particle displacement and velocity for A-ULMPM integrated with MLS-MPM ($s = 0$), A-ULMPM integrated with MLS-MPM ($s \neq 0$), A-ULMPM integrated with MLS-MPM (black) ($s = n$), and A-ULMPM integrated with kernel-MPM ($s = 0$), where A-ULMPM ($s = 0$) recovers TLMPM and A-ULMPM ($s = n$) recovers EMPM, as described in Section 5. In general, TLMPM can produce more accurate simulations since the cell-crossing error is completely eliminated, while EMPM fails to converge when the cell-crossing error is significant. A-ULMPM with $s \neq n$ exhibits second-order accuracy while its integration with MLS-MPM has higher solution accuracy than that with kernel-MPM.

6.1.2. 2D droplet

We ran several simulations of falling droplets to compare our implementations in the A-ULMPM framework. As shown in Figure 3, although TLMPM has benefits over EMPM for solid simulations (see Figure 2), it fails to achieve detailed free surfaces in fluid simulations, as shown in Figure 3, since the topology changes significantly in fluid-like simulations. A-ULMPM automatically updates configuration maps to produce similar fluid dynamics as EMPM and captures rich interactions. Moreover, our integration with MLS-MPM (see Section 5) yields more energetic behavior compared to an integration with kernel-MPM (see Appendix B).

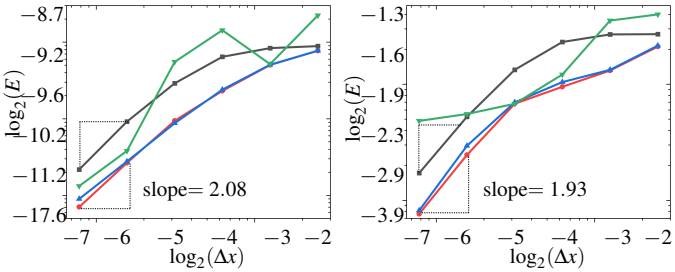


Figure 14: *Log-log plots of error (labeled E) vs. grid mesh size Δx. (Left) Error of the displacement, (right) error of the velocity; using A-ULMPM integrated with MLS-MPM ($s = 0$) (red), A-ULMPM integrated with MLS-MPM ($s \neq 0$) (blue), A-ULMPM integrated with MLS-MPM (black) ($s = n$), and A-ULMPM integrated with kernel-MPM (black) ($s = 0$). Our A-ULMPM framework recovers TLMPM ($s = 0$) and EMPM ($s = n$). Comparing the slopes and solution accuracy shows EMPM is non-convergent when grid-crossing occurs, while A-ULMPM with $s = 0$ and $s \neq n$ provides convergent simulations with second order accuracy for displacement and velocity with small Δx.*

6.2. Bar Twisting

We imposed torsion and stretch boundary conditions at the two ends of a hyperelastic beam to showcase that A-ULMPM allows large deformations in solids without non-physical fractures. As shown in Figure 4, traditional EMPM fails to preserve the shape of the beam during the twisting and pulling, while A-ULMPM can readily handle the challenging invertible elasticity when one end of the beam is released after twisting. Figure 7 shows that A-ULMPM is capable of robustly recovering the beam shape after extreme elastic deformations, while particles in EMPM cluster into one irreversible (or plastic) thin string blocking the "recovery" of elasticity.

6.3. Stretchy Yo-Yo

Next, we tossed a stretchy Stanford bunny yo-yo with gravity forces. Our *A*-ULMPM hyperelastic bunny demonstrates rich elastic responses and realistic bouncing dynamics, while EMPM breaks due to severe numerical fracture (see Figure 10). Our *A*-ULMPM framework can perfectly handle hyperelastic deformation under severe bending (see Figure 15 (left)) and stretching (see Figure 15 (right)).

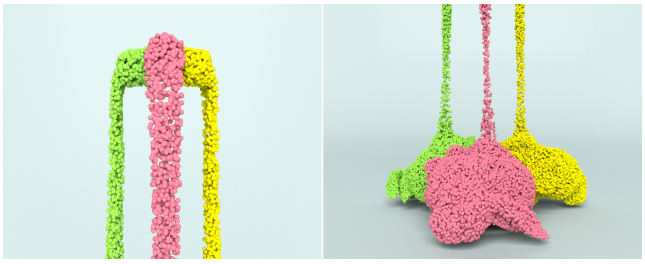


Figure 15: *A closer view of the hyperelastic bunny yo-yo from Figure 10.*

6.4. Fluid-like Simulations

We simulated the fluid-like behavior of different materials using our *A*-ULMPM framework, as described in Section 5, such as elastoplastic snow [SSC*13], weakly compressible water, and viscous armadillos. We dropped two copies of the snow Stanford bunny with different orientations to a solid wedge, as shown in Figure 8. *A*-ULMPM captures the vivid snow smashing and scattering after the bunnies fall on the wedge, similar to its Eulerian counterparts proposed in prior works [SSC*13] (see side-by-side comparisons in our video). We simulated a water Stanford bunny falling inside a spherical container (see Figure 9), showing the extreme large deformations and energetic splashes. Next, we show that our *A*-ULMPM method can be utilized to simulate viscous fluid flows. We couple our *A*-ULMPM solve in equation (26) with Newtonian viscosity [SXH*21], and dropped four copies of the viscous armadillos with random orientations into a cup, as shown in Figure 13. Both *A*-ULMPM explicit and implicit schemes do not update the configuration maps at each time step, in contrast to EMPM, so it is naturally more efficient in fluid-like simulations (see Table 3).

6.5. Fluid-Structure Interaction

Our *A*-ULMPM framework can also simulate realistic fluid-structure interaction problems, as shown in Figure 11. We dropped a water Stanford bunny to a hyperelastic bowl. *A*-ULMPM captures rich fluid-structure interactions, displaying advantages of both TLMPM and EMPM (see Figure 16 (right)), and highlighting that *A*-ULMPM can adaptively handle both fluid and solid simulations without having to switch between TLMPM and EMPM. In contrast, EMPM suffers from numerical fractures. As shown in Figure 11, the hyperelastic bowl fractures, causing the water to spill below.

6.6. Efficiency of Implicit Solve

Our *A*-ULMPM scheme supports both explicit and implicit solve. The semi-implicit update allows comparable time steps with respect to EMPM (see Table 4).

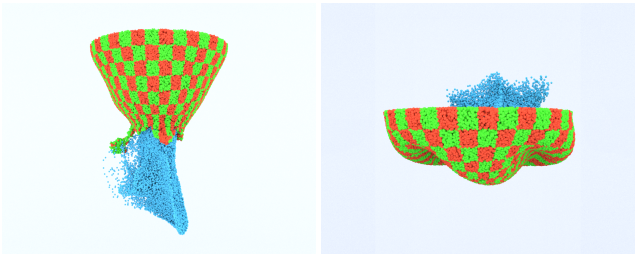


Figure 16: *A closer view of hyperelastic deformations in the fluid-structure interaction example from Figure 11. (Left) The bowl breaks in EMPM due to numerical fractures, while (right) A-ULMPM maintains a leakproof bowl.*

Table 4: ***Time step*** *is measured in average milliseconds ($\times 10^{-3}s$).* ***Grid:*** *Number of occupied voxels in the background sparse grid.* ***Particle:*** *Total number of MPM particles in the simulation.*

Simulation	Time Step	Grid	Particle
Twisting bar (Fig.4 (1st row))	0.178	1.0M	193.3K
Twisting bar (Fig.4 (2nd row))	0.125	1.0M	193.3K
Stretchy yo-yo (Fig.10 (1st row))	0.54	4.2M	179.3K
Stretchy yo-yo (Fig.10 (2nd row))	0.71	4.2M	179.3K
Snow bunny (Fig.8)	2.52	524.3K	116.3K
Snow bunny (EMPM in video)	2.31	524.3K	116.3K
Water bunny (Fig.9)	0.73	2.1M	96.3K
Water bunny (EMPM in video)	0.73	2.1M	96.3K
Viscous armadillo (Fig. 13)	4.31	524.3K	40K
Viscous armadillo (EMPM)	4.25	524.3K	40K

7. Discussion and Conclusion

7.1. Limitations and Future Work

Our model has generated a large number of compelling examples, but there remains much work to be done. Parameters to adjust the configuration update frequency were tuned by hand, and it would be interesting to calibrate them to measured models. Since each object has it own configuration map, we only briefly investigated contact by projecting particle positions to a global grid and processing grid-based collisions [SSC*13]. By doing this, we observed slight self-penetration in the 3D twisting beam example (see Figure 4). This could be addressed by further introducing contact algorithms, such as [JGT17, HGG*19]. While we did not experience a need for excessively small time steps given the low stiffness of the materials we considered, deforming materials with high wave speed, such as steel, could benefit from a fully implicit discretization.

The present work describes our initial attempt in designing an update criterion that applies to both solids and fluids while avoiding visual artifacts such as numerical fracture. We hope that it inspires future research in designing more accurate update criteria that have better energy conservation properties. Indeed, side-by-side comparisons with Eulerian MLS-MPM shows that while our *A*-ULMPM framework produces similar fluid-like dynamics as EMPM and captures rich interactions, it incurs a slight energy loss. It is also worth noting that our proposed configuration update cri-

terion is designed in a *global* manner. It would be interesting to develop a *local* update rule based on the current method.

Since each object has it own configuration map, we only briefly investigated contact by projecting particle positions to a global grid and processing grid-based collisions [SSC*13]. By doing this, we observed slight self-penetration in the 3D twisting beam example (see Figure 4). This could be addressed by further introducing contact algorithms, such as [JGT17, HGG*19]. Finally, while our focus was on the material responses of water, snow, and hyperelastic solids, it would be interesting to investigate other materials and multi-physics coupling problems with *A*-ULMPM.

7.2. Conclusion

We proposed an Adaptively Updated Lagrangian Material Point Method (*A*-ULMPM), which combines advantages of Total Lagrangian frameworks [dVN21, dVNH20] and Eulerian frameworks [SSC*13, HFG*18] in an adaptive fashion. *A*-ULMPM avoids the cell-crossing instability and numerical fracture in solid simulations, while still allowing for large deformations that arise in fluid-like simulations. It can be easily integrated with any existing MPM framework and builds a foundation for devising various MPM schemes, such as PolyPIC [FGG*17], for enhanced accuracy and visual vividness.

8. Acknowledgements

We thank the anonymous reviewers for their valuable feedback. This research was supported in part by the Rutgers University start-up grant and the National Science Foundation under awards CCF-2110861 and IIS-2132972. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[Bar02] BARDENHAGEN S.: Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics 180*, 1 (2002), 383–403. 3

[BK04] BARDENHAGEN S. G., KOBER E. M.: The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences 5*, 6 (2004), 477–496. 2, 3

[BKR88] BRACKBILL J. U., KOTHE D. B., RUPPEL H. M.: Flip: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications 48*, 1 (1988), 25–38. 2, 3

[DBD16] DAVIET G., BERTAILS-DESCOUBES F.: A semi-implicit material point method for the continuum simulation of granular materials. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–13. 2, 3

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96* (1996), pp. 61–76. 2

[dGWH*15] D. GOES F., WALLEZ C., HUANG J., PAVLOV D., DESBRUN M.: Power particles: An incompressible fluid solver based on power diagrams. *ACM Trans Graph 34*, 4 (2015). 2

[DHW*19] DING M., HAN X., WANG S., GAST T. F., TERAN J. M.: A thermomechanical material point method for baking and cooking. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–14. 2, 3

[dVN21] DE VAUCORBEIL A., NGUYEN V. P.: Modelling contacts with a total lagrangian material point method. *Computer Methods in Applied Mechanics and Engineering 373* (2021), 113503. 2, 4, 15

[dVNH20] DE VAUCORBEIL A., NGUYEN V. P., HUTCHINSON C. R.: A total-lagrangian material point method for solid mechanics problems involving large deformations. *Computer Methods in Applied Mechanics and Engineering 360* (2020), 112783. 2, 4, 6, 15

[Erl13] ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses* (2013), SIGGRAPH '13. 2

[FBGZ18] FEI Y., BATTY C., GRINSPUN E., ZHENG C.: A multi-scale model for simulating liquid-fabric interactions. *ACM Trans Graph 37*, 4 (2018), 51:1–51:16. 3

[FGG*17] FU C., GUO Q., GAST T., JIANG C., TERAN J.: A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–12. 2, 3, 15

[FLGJ19] FANG Y., LI M., GAO M., JIANG C.: Silly rubber: an implicit material point method for simulating non-equilibrated viscoelastic and elastoplastic solids. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–13. 2, 3

[FMB*17] FEI Y., MAIA H. T., BATTY C., ZHENG C., GRINSPUN E.: A multi-scale model for simulating liquid-hair interactions. *ACM Trans Graph 36*, 4 (2017). 3

[FQL*20] FANG Y., QU Z., LI M., ZHANG X., ZHU Y., AANJANEYA M., JIANG C.: Iq-mpm: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 51–1. 2, 3

[GGT*18] GAUME J., GAST T., TERAN J., V. HERWIJNEN A., JIANG C.: Dynamic anticrack propagation in snow. *Nature Comm 9*, 1 (2018), 3047. 3

[GPH*18] GAO M., PRADHANA A., HAN X., GUO Q., KOT G., SIFAKIS E., JIANG C.: Animating fluid sediment mixture in particle-laden flows. *ACM Trans Graph 37*, 4 (2018), 149. 2, 3

[GTJS17] GAO M., TAMPUBOLON A. P., JIANG C., SIFAKIS E.: An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–12. 2, 3

[GWW*18] GAO M., WANG X., WU K., PRADHANA A., SIFAKIS E., YUKSEL C., JIANG C.: Gpu optimization of material point methods. *ACM Trans Graph 37*, 6 (2018), 1–12. 2, 3

[HBG16] HOMEL M. A., BRANNON R. M., GUILKEY J.: Controlling the onset of numerical fracture in parallelized implementations of the material point method (mpm) with convective particle domain interpolation (cpdi) domain scaling. *International Journal for Numerical Methods in Engineering 107*, 1 (2016), 31–48. 2

[HFG*18] HU Y., FANG Y., GE Z., QU Z., ZHU Y., PRADHANA A., JIANG C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 1, 2, 3, 6, 7, 8, 9, 12, 13, 15

[HGG*19] HAN X., GAST T. F., GUO Q., WANG S., JIANG C., TERAN J.: A hybrid material point method for frictional contact with diverse materials. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–24. 14, 15

[HJST13] HEGEMANN J., JIANG C., SCHROEDER C., TERAN J.: A level set method for ductile fracture. In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim* (2013), pp. 193–201. 2

[HLS*19] HU Y., LIU J., SPIELBERG A., TENENBAUM J. B., FREEMAN W. T., WU J., RUS D., MATUSIK W.: Chainqueen: A real-time differentiable physical simulator for soft robotics. *Proc of the Int Conf on Robotics and Automation (ICRA)* (2019), 6265–6271. 3

[JGT17] JIANG C., GAST T., TERAN J.: Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–14. 3, 14, 15

[JSS*15] JIANG C., SCHROEDER C., SELLE A., TERAN J., STOMAKHIN A.: The affine particle-in-cell method. *ACM Trans Graph 34*, 4 (2015), 51:1–51:10. 1, 3, 6, 8, 10, 12

[JST*16] JIANG C., SCHROEDER C., TERAN J., STOMAKHIN A., SELLE A.: The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*. 2016, pp. 1–52. 2

[JST17] JIANG C., SCHROEDER C., TERAN J.: An angular momentum conserving affine-particle-in-cell method. *Journal of Computational Physics 338* (2017), 137–164. 2, 3, 4, 12

[KGP*16] KLÁR G., GAST T., PRADHANA A., FU C., SCHROEDER C., JIANG C., TERAN J.: Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–12. 2, 3

[LLZ08] LIU M., LIU G., ZONG Z.: An overview on smoothed particle hydrodynamics. *International Journal of Computational Methods 5*, 01 (2008), 135–188. 2

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J Vis Comm Imag Repre 18*, 2 (2007), 109–118. 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 141–151. 2

[MMCK14] MACKLIN M., MULLER M., CHENTANEZ N., KIM T.: Unified particle physics for real-time applications. *ACM Trans Graph 33*, 4 (2014), 153:1–153:12. 2

[MSW*09] MCADAMS A., SELLE A., WARD K., SIFAKIS E., TERAN J.: Detail preserving continuum simulation of straight hair. *ACM Trans Graph 28*, 3 (2009), 62:1–62:6. 3

[NGL10] NARAIN R., GOLAS A., LIN M.: Free-flowing granular materials with two-way solid coupling. *ACM Trans Graph 29*, 6 (2010), 173:1–173:10. 3

[PAKF13] PATKAR S., AANJANEYA M., KARPMAN D., FEDKIW R.: A hybrid lagrangian-eulerian formulation for bubble generation and dynamics. In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim* (2013), pp. 105–114. 3

[QZG*19] QU Z., ZHANG X., GAO M., JIANG C., CHEN B.: Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–12. 11

[RGJ*15] RAM D., GAST T., JIANG C., SCHROEDER C., STOMAKHIN A., TERAN J., KAVEHPOUR P.: A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 157–163. 2, 3

[SABS14] SETALURI R., AANJANEYA M., BAUER S., SIFAKIS E.: Spgrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. Graph. 33*, 6 (2014), 1–12. 7

[SB12] SIFAKIS E., BARBIC J.: Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (2012). 2

[SBB11] SADEGHIRAD A., BRANNON R. M., BURGHARDT J.: 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. 2, 3

[SBH09] SIN F., BARGTEIL A. W., HODGINS J. K.: A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), SCA '09, p. 247–255. 2

[SCS94] SULSKY D., CHEN Z., SCHREYER H. L.: A particle method for history-dependent materials. *Computer methods in applied mechanics and engineering 118*, 1-2 (1994), 179–196. 2, 3

[SH*21] SU HAOZHE XUE T., , HAN C., JIANG C., AANJANEYA M.: A unified second-order accurate in time mpm formulation for simulating viscoelastic liquids with phase change. *ACM Trans. Graph. 40*, 4 (Aug. 2021). 2, 3

[Sha18] SHABANA A. A.: *Computational continuum mechanics*. John Wiley & Sons, 2018. 6

[SKB08] STEFFEN M., KIRBY R. M., BERZINS M.: Analysis and reduction of quadrature errors in the material point method (mpm). *International journal for numerical methods in engineering 76*, 6 (2008), 922–948. 3

[SMT08] SIFAKIS E., MARINO S., TERAN J.: Globally coupled collision handling using volume preserving impulses. In *Symp. Comp. Anim.* (2008), pp. 147–153. 3

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 2, 3, 4, 6, 9, 10, 14, 15, 17

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 81–90. 4

[SSJ*14] STOMAKHIN A., SCHROEDER C., JIANG C., CHAI L., TERAN J., SELLE A.: Augmented mpm for phase-change and varied materials. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–11. 2, 3

[SXH*21] SU H., XUE T., HAN C., JIANG C., AANJANEYA M.: A unified second-order accurate in time mpm formulation for simulating viscoelastic liquids with phase change. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–18. 14

[SZS95] SULSKY D., ZHOU S.-J., SCHREYER H. L.: Application of a particle-in-cell method to solid mechanics. *Computer physics communications 87*, 1-2 (1995), 236–252. 2

[TGK*17] TAMPUBOLON A. P., GAST T., KLÁR G., FU C., TERAN J., JIANG C., MUSETH K.: Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–11. 2, 3

[WCL*20] WOLPER J., CHEN Y., LI M., FANG Y., QU Z., LU J., CHENG M., JIANG C.: Anisompm: Animating anisotropic damage mechanics. *ACM Trans. Graph. 39*, 4 (2020). 3

[WDG*19] WANG S., DING M., GAST T. F., ZHU L., GAGNIERE S., JIANG C., TERAN J. M.: Simulation and visualization of ductile fracture with the material point method. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–20. 4

[WFL*19] WOLPER J., FANG Y., LI M., LU J., GAO M., JIANG C.: Cd-mpm: continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–15. 2, 3

[WG07] WALLSTEDT P., GUILKEY J.: Improved velocity projection for the material point method. *Computer Modeling in Engineering and Sciences 19*, 3 (2007), 223. 2

[XSH*20] XUE T., SU H., HAN C., JIANG C., AANJANEYA M.: A novel discretization and numerical solver for non-fourier diffusion. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–14. 2, 3

[XZT19] XUE T., ZHANG X., TAMMA K. K.: A non-local dissipative lagrangian modelling for generalized thermoelasticity in solids. *Applied Mathematical Modelling 73* (2019), 247–265. 8

[YSB*15] YUE Y., SMITH B., BATTY C., ZHENG C., GRINSPUN E.: Continuum foam: a material point method for shear-dependent flows. *ACM Trans Graph 34*, 5 (2015), 160:1–160:20. 2, 3

[YSC*18] YUE Y., SMITH B., CHEN P. Y., CHANTHARAYUKHONTHORN M., KAMRIN K., GRINSPUN E.: Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. *ACM Trans. Graph. 37*, 6 (2018). 3

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 965–972. 3

[ZTNZ77] ZIENKIEWICZ O. C., TAYLOR R. L., NITHIARASU P., ZHU J.: *The finite element method*, vol. 3. McGraw-hill London, 1977. 2, 6

[ZZL*17] ZHU F., ZHAO J., LI S., TANG Y., WANG G.: Dynamically enriched mpm for invertible elasticity. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 381–392. 11

Appendix A: MLS Gradient

Consider a domain (Ω_0) in Euclidean space $\mathbb{E}$. Let $\mathbf{q}_i \in \Omega_0 \subset \mathbb{E}$ denote the geometric position of a certain point in the domain and $\mathbf{q}_j \in \Omega_0 \subset \mathbb{E}$ denote another point which is close to the position $\mathbf{q}_i$. Therefore, any physical quantity at each point in space can be defined as $\varphi(\mathbf{q}_i)$ and following a first-order Taylor approximation gives:

$$\varphi(\mathbf{q}_j) \cong \varphi(\mathbf{q}_i) + \nabla\varphi_{ij}\mathbf{r}_{ij} \tag{37}$$

where $\mathbf{r}_{ij} = \mathbf{q}_j - \mathbf{q}_i$. A weighted error E_i of equation (37) is defined within its associated domain Ω_i as follows:

$$E_i = \int_{j\in\Omega_i} \|\varphi(\mathbf{q}_j) - \varphi(\mathbf{q}_i) - \nabla\varphi_{ij}\mathbf{r}_{ij}\|^2 W_{ij} dV \tag{38}$$

where Ω_i represents the neighboring domain of $\mathbf{q}_i$; W_{ij} represents the influence from j to i and is a function of $\|\mathbf{r}_{ij}\|$. To minimize the error E_i and get the best approximation, the Least Squares Technique is exploited by introducing $\frac{\partial E_i}{\partial \nabla\phi_i} = 0$:

$$\frac{\partial}{\partial\nabla\phi_i}\int_{j\in\Omega_i} \left(\varphi(\mathbf{q}_j) - \varphi(\mathbf{q}_i) - \nabla\varphi_{ij}\mathbf{r}_{ij}\right)^2 W_{ij} dV = 0 \tag{39}$$

Therefore, we have a MLS-gradient operator :

$$\nabla\varphi(\mathbf{q}_i) = \left(\int_{j\in\Omega_i} \varphi(\mathbf{q})_{ij} \otimes \mathbf{r}_{ij} W_{ij} dV\right)\left(\int_{j\in\Omega_i} \mathbf{r}_{ij}\otimes\mathbf{r}_{ij} W_{ij} dV\right)^{-1} \tag{40}$$

where $\varphi(\mathbf{q})_{ij} = \varphi(\mathbf{q})_j - \varphi(\mathbf{q})_i$ and its discrete form is given by:

$$\nabla\varphi(\mathbf{q}_i) = \left(\sum_{j\in\Omega_i} \varphi(\mathbf{q})_{ij} \otimes \mathbf{r}_{ij} W_{ij} V_j\right)\left(\sum_{j\in\Omega_i} \mathbf{r}_{ij}\otimes\mathbf{r}_{ij} W_{ij} V_j\right)^{-1} \tag{41}$$

where V_j denotes volume distributed at $\mathbf{q}_j$. Moreover, the derivation of $\nabla\varphi(\mathbf{q}_i)$ with respect to $\mathbf{q}_k$ is given as follows:

$$\begin{aligned}\frac{\partial\nabla\varphi(\mathbf{q}_i)}{\partial\varphi(\mathbf{q}_k)} &= \left(\sum_{j\in\Omega_i} \frac{\varphi(\mathbf{q})_{ij}}{\partial\varphi(\mathbf{q}_k)} \otimes \mathbf{r}_{ij} W_{ij} V_j\right)\left(\sum_{j\in\Omega_i} \mathbf{r}_{ij}\otimes\mathbf{r}_{ij} W_{ij} V_j\right)^{-1} \\ &= \left(\sum_{j\in\Omega_i} (\delta_{jk} - \delta_{ik}) \otimes \mathbf{r}_{ij} W_{ij} V_j\right)\left(\sum_{j\in\Omega_i} \mathbf{r}_{ij}\otimes\mathbf{r}_{ij} W_{ij} V_j\right)^{-1}\end{aligned} \tag{42}$$

where δ is the Kronecker delta function.

Appendix B: *A*-ULMPM Integration with kernel-MPM

We briefly describe the integration of our *A*-ULMPM with traditional MPM [SSC*13] as follows:

1. Transfer Particles to Grid.

$$m_i^s \mathbf{v}_i^n = \sum_p m_p \mathbf{v}_p^n W_{pi}^s, \quad m_i^s = \sum_p m_p W_{pi}^s, \quad \mathbf{v}_i^n = \frac{m_i^s \mathbf{v}_i^n}{m_i} \tag{43}$$

2. Update Grid Momentum.

$$\begin{aligned}\hat{\mathbf{v}}_i^{n+1} &= \mathbf{v}_i^n - \frac{\Delta t}{m_i}\sum_p \mathbb{P}_p^0 (\mathbf{F}_p^{0s})^T \nabla^s W_{ip}^s V_j^0 \\ \mathbf{q}_i^{n+1} &= \mathbf{q}_i^n + \Delta t \hat{\mathbf{v}}_i^{n+1}\end{aligned} \tag{44}$$

3. Transfer Grid Velocity to Particles. We project $\mathbf{q}_i^{n+1}$ for all objects to the global grid and consider grid-based collisions [SSC*13] to obtain $\mathbf{v}_i^{n+1}$. We transfer $\mathbf{v}_i^{n+1}$ to particles using a weighted combination of PIC and FLIP, as described in [SSC*13].
4. Update Particle Deformation Gradient.

$$\mathbf{F}_p^{s(n+1)} = \mathbf{F}_p^{sn} + \Delta t \nabla_s \mathbf{v}_p^{n+1} \tag{45}$$

$$\tag{46}$$

where $\nabla_s \mathbf{v}_p^{n+1}$ is given by:

$$\nabla_s \mathbf{v}_p^{n+1} = \sum_i \mathbf{v}_i^{n+1} \nabla^s W_{pi}^s \tag{47}$$

By differential chain rule, the update of $\mathbf{F}_p^{0(n+1)}$ is given by:

$$\mathbf{F}_p^{0(n+1)} = \frac{\partial \mathbf{q}_p^{n+1}}{\partial \mathbf{q}_p^s}\frac{\partial \mathbf{q}_p^s}{\partial \mathbf{q}_p^0} = \mathbf{F}_p^{s(n+1)}\mathbf{F}_p^{0s} \tag{48}$$