Computational Mechanics
https://doi.org/10.1007/s00466-024-02524-x

Unstructured moving least squares material point methods: a stable kernel approach with continuous gradient reconstruction on general unstructured tessellations

Yadi Cao[1] · Yidong Zhao[1,2] · Minchen Li[1,3] · Yin Yang[4] · Jinhyun Choo[2] · Demetri Terzopoulos[1] · Chenfanfu Jiang[1]

Received: 18 December 2023 / Accepted: 1 July 2024

Abstract
The material point method (MPM) is a hybrid Eulerian Lagrangian simulation technique for solid mechanics with significant deformation. Structured background grids are commonly employed in the standard MPM, but they may give rise to several accuracy problems in handling complex geometries. When using (2D) unstructured triangular or (3D) tetrahedral background elements, however, significant challenges arise (e.g., cell-crossing error). Substantial numerical errors develop due to the inherent $\mathcal{C}^0$ continuity property of the interpolation function, which causes discontinuous gradients across element boundaries. Prior efforts in constructing $\mathcal{C}^1$ continuous interpolation functions have either not been adapted for unstructured grids or have only been applied to 2D triangular meshes. In this study, an unstructured moving least squares MPM (UMLS-MPM) is introduced to accommodate 2D and 3D simplex tessellation. The central idea is to incorporate a diminishing function into the sample weights of the MLS kernel, ensuring an analytically continuous velocity gradient estimation. Numerical analyses confirm the method's capability in mitigating cell crossing inaccuracies and realizing expected convergence.

Keywords Material point method · Moving least square method · Cross cell instability · Unstructure mesh

1 Introduction

The material point method (MPM) [1] was introduced to solid mechanics as an extension of both the Fluid-Implicit Particle (FLIP) method [2] and the Particle-in-Cell (PIC) method [3]. The MPM is a hybrid Eulerian-Lagrangian method, often referred to as a particle-grid method that retains and monitors all physical attributes on a collection of particles. A background grid serves in solving the governing equations. Both Eulerian and Lagrangian descriptions are incorporated in the MPM to overcome the numerical challenges stemming from nonlinear convective terms inherent in a strictly Eulerian approach, while avoiding significant grid distortions typically found in purely Lagrangian methods. The efficacy of the method has been demonstrated in problems concerning extreme deformation of solid materials, such as biological soft tissues [4, 5], explosive materials [6, 7], sand [8–10], and snow [11–13].

Based on the specific Lagrangian formulations, the MPM is categorized into total Lagrangian [14–17] and updated Lagrangian [18] variants, in which equations are formulated in different reference configurations. In the total Lagrangian MPM, numerical dissipation errors or artificial fractures are not observed; however, challenges arise due to mesh distortions as the connectivity is preserved in a manner similar to the Finite Element Method (FEM). Conversely, the updated Lagrangian MPM has been found to exhibit greater robustness, particularly in dealing with demanding scenarios such as impacts and shocks [8, 19–21], failures and cracks in both single-phase and multi-phase materials [10, 22, 23], and contact mechanics [15, 24–30].

Despite its numerous successes, the updated Lagrangian MPM mainly adopts a uniformly-structured background grid that aligns with the axes of the global Cartesian coordinate system, using (2D) quadrilaterals or (3D) hexahedra for spatial discretization. When boundaries involve complex geometry, however, the aforementioned approach may

✉ Yadi Cao
yadicao95@gmail.com

1 University of California, Los Angeles, Los Angeles, USA

2 Korea Advanced Institute of Science & Technology, Daejeon, South Korea

3 Carnegie Mellon University, Pittsburgh, USA

4 University of Utah, Salt Lake City, USA

Published online: 23 July 2024

introduce significant challenges in conformally discretizing the space. Remarkably, many engineering problems, such as those in mechanical and geotechnical engineering [31], involve complex boundary geometry. Hence, some researchers [32–35] have proposed using unstructured (2D) triangles or (3D) tetrahedra for discretization, which provides substantial flexibility in the presence of geometrically complex boundaries.

Unfortunately, most of the existing approaches using unstructured triangular or tetrahedral elements adopt a piecewise linear (C^0) basis function [32–35] whose gradient is discontinuous along element boundaries. In this case, when particles move from one element to another (i.e., crossing element boundaries), a significant error arises—the so-called cell-crossing error [36]. Because the function gradient becomes discontinuous along element boundaries, the cell-crossing error leads to severe stress oscillations, causing significant numerical errors.

Several approaches have been proposed for circumventing the cell-crossing error, including the generalized interpolation material point (GIMP) method [36, 37], the dual domain MPM (DDMPM) [38], the use of high-order basis functions such as B-splines [39, 40], and approaches based on moving least squares (MLS) basis functions [41, 42]. Unfortunately, they are either limited to structured quadrilaterals/hexahedra or are only applicable to 2D cases using triangles [43]. This leaves the cell-crossing error as an unsolved challenge when using unstructured tessellations in both the 2D and 3D MPM.

The objective of this study is to address the aforementioned cell-crossing challenge for general unstructured meshes in both 2D and 3D. The proposed approach is built upon a new MLS reconstruction process that is suitable for general unstructured discretization. By incorporating a diminishing function into the sample weights of the MLS kernel, an analytically continuous function gradient is achieved, which efficiently eliminates the cell-crossing error. A new MLS kernel function is derived that can be straightforwardly implemented into an existing MPM framework.

The remainder of this paper is structured as follows: Sect. 2.1 introduces the general governing equations of the MPM and the details of a typical explicit MPM process. The Moving Least Squares (MLS) approximation and the MLS-MPM method are discussed in Sect. 2.3.1. A seemingly straightforward yet inherently flawed extension of the MLS-MPM to unstructured meshes, along with the associated cell-crossing challenge, is presented in Sect. 2.3.2. Section 2.3.4 develops a solution to this challenge, accompanied by an in-depth analysis and kernel reconstruction for representative unstructured meshes. Numerical results affirming the efficacy of the proposed method are reported in Sect. 3. The paper concludes in Sect. 4 with reflections and recommendations for future work.

2 Methodology

2.1 Governing equations

Following standard continuum mechanics [44], consider the mapping $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}, t)$, which maps points from the (reference) material configuration, represented by $\boldsymbol{X}$, to their corresponding locations in the (current) spatial configuration, represented by $\boldsymbol{x}$. In this framework, velocity is defined in two different but equivalent manners. On the one hand, $\boldsymbol{V}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{x}}{\partial t}(\boldsymbol{X}, t)$ defines the Lagrangian velocity in the material configuration. On the other hand, the Eulerian velocity in the spatial configuration, is denoted by $\boldsymbol{v}(\boldsymbol{x}, t) = \boldsymbol{V}(\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t), t)$. Furthermore, the deformation experienced by the material points is quantified using the deformation gradient, given by $\boldsymbol{F}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}(\boldsymbol{X}, t)$. The determinant of this gradient, represented by J, is also crucial as it provides insights into volumetric changes associated with the deformation process.

Given these definitions, the conservation equations for mass and momentum (neglecting external forces) are [44, 45]

$$\begin{aligned} \rho J &= \rho_0, \\ \rho \frac{D\boldsymbol{v}}{Dt} &= \nabla \cdot \boldsymbol{\sigma}, \end{aligned} \tag{1}$$

where ρ represents the density, D/Dt is the material derivative, and

$$\boldsymbol{\sigma} = \frac{1}{J} \boldsymbol{P} \boldsymbol{F}^T. \tag{2}$$

is the Cauchy stress tensor, which is related to the first Piola-Kirchhoff stress $\boldsymbol{P} = \frac{\partial \Psi}{\partial \boldsymbol{F}}$, where Ψ denotes the strain energy density. The evolution of the deformation gradient is given by

$$\dot{\boldsymbol{F}} = (\nabla \boldsymbol{v}) \boldsymbol{F}. \tag{3}$$

Consider a domain represented by Ω. Boundaries on which the displacement is known, represented as $\partial \Omega_u$, are governed by the Dirichlet boundary condition

$$x_k(\boldsymbol{x}, t) = \bar{x}_k(\boldsymbol{x}, t), \quad \forall \boldsymbol{x} \in \partial \Omega_u, \tag{4}$$

where $\bar{x}_k$ denotes the predetermined displacement for component k. Boundaries on which the tractions (forces per unit

area) are predefined, represented as $\partial\Omega_\tau$, adhere to the Neumann boundary condition

$$\sigma_{kl}(\boldsymbol{x},t)n_l = \bar{\tau}_k(\boldsymbol{x},t), \quad \forall \boldsymbol{x} \in \partial\Omega_\tau, \tag{5}$$

where $\bar{\tau}_k$ is the prescribed traction for component k, and $\sigma_{kl}(\boldsymbol{x},t)n_l$ represents the traction inferred from the stress tensor σ_{kl} acting in the direction of the outward unit normal vector n_l. For ease of reference and notational clarity in our framework, the subscripts k and l refer to components k and l of any given vector or tensor.

To solve the conservation equations for mass and momentum within the MPM framework, one often turns to the weak form. Specifically, a continuous test function ϕ, which vanishes on $\partial\Omega_u$, is employed. Then, both sides of the equation are multiplied by ϕ and integrated over the domain Ω:

$$\int_\Omega \phi\rho\ddot{x}_k d\Omega = \int_{\partial\Omega_\tau} \phi\tau_k dA - \int_\Omega \frac{\partial\phi}{\partial \boldsymbol{x}_l}\sigma_{kl} d\Omega. \tag{6}$$

At this juncture, integration by parts and the Gauss integration theorem are utilized, nullifying the contributions on $\partial\Omega_u$ due to the vanishing of the test function on this boundary subset.

For clarity, in the remainder of this paper, the terms "grids" or "grid nodes" will exclusively refer to regular background grid nodes. In contrast, "mesh nodes" will denote nodes in general, unstructured meshes. For simplicity, we do not change the common abbreviations such as Particle-To-Grid (P2G) and Grid-To-Particle (G2P).

In the standard implementation of the MPM, physical quantities such as mass and velocity are retained at material points and then projected onto background grid nodes for further computation. (6) is discretized on these nodes by the Finite Element Method (FEM) and then solved using either implicit or explicit time integration schemes. This article focuses on the explicit symplectic Euler time integration method. While the extension to implicit methods is possible and straightforward, it would be orthogonal to the contribution of the article.

2.2 Explicit MPM pipeline

The explicit MPM pipeline in each time step has four main stages: (1) the transfer of material point quantities to the background nodes, known as Particle-To-Grid (P2G), (2) the computation of the system's evolution on these background nodes, (3) the back-transfer of the evolved quantities to the material points, known as Grid-To-Particle (G2P), and (4) the execution of necessary post-processings, such as elastoplasticity return mapping and material hardening. Algorithm 1 presents an overview of the MPM pipeline, and the main stages are elaborated below.

Algorithm 1 Explicit MPM

1: Determine material point-node connectivity, calculate kernel functions $w_{p,i}$
2: **P2G:**
Nodal mass: $m_i = \sum_p \rho_p V_p w_{p,i}$
Nodal momentum: $\boldsymbol{p}_i = \sum_p \boldsymbol{v}_p \rho_p V_p w_{p,i}$
Nodal velocity: $\boldsymbol{v}_i = \boldsymbol{p}_i / m_i$
3: Internal force: $\boldsymbol{f}_i^{\text{int}} = -\sum_p V_p \boldsymbol{\sigma}_p \nabla w_{p,i}$
4: Gravity: $\boldsymbol{f}_i^{\text{ext}} = \sum_p w_{p,i} m_p \mathbf{g}_p$
5: Nodal force: $\boldsymbol{f}_i = \boldsymbol{f}_i^{\text{ext}} + \boldsymbol{f}_i^{\text{int}}$
6: **Deformation of background nodes:**
Updated nodal accelerations: $\ddot{\boldsymbol{x}}_i = \boldsymbol{f}_i / m_i$
Update nodal velocities: $\tilde{\boldsymbol{v}}_i = \boldsymbol{v}_i + \Delta t \ddot{\boldsymbol{x}}_i$
Enforce Dirichlet conditions: $\ddot{\boldsymbol{x}}_i = 0$ and $\boldsymbol{f}_i = 0$
7: **G2P:**
Update point velocities: $\boldsymbol{v}_p^{\Delta t} = \boldsymbol{v}_p + \Delta t \sum_i w_{p,i} \ddot{\boldsymbol{x}}_i$
Update point positions: $\boldsymbol{x}_p^{\Delta t} = \sum_i w_{p,i} \tilde{\boldsymbol{x}}_i$
8: Update deformation gradient: $\boldsymbol{F}_p^{\Delta t} = \left(\boldsymbol{I} + \sum_i (\tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i)(\nabla w_{p,i})^T\right) \boldsymbol{F}_p$
9: Update point volume: $V_p^{\Delta t} = \det(\boldsymbol{F}_p^{\Delta t}) V_p^0$
10: Update point stresses: $\boldsymbol{\sigma}_p = C(\boldsymbol{F}_p)$
11: **Enforce plasticity, reset background deformation, advance to next timestep**

Stage 1: P2G In the MPM, material points are the Lagrangian particles that track the location of the continuum along with physical attributes such as mass, position, and velocity. To evolve the dynamics on the background grid or mesh nodes, an interpolation function—also known as a transfer kernel or simply a kernel—needs to be determined to relate the information from particles to their nearby active nodes. Generally, for a particle located at $\boldsymbol{x}_p$ and all surrounding nodes at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the stacked kernel values associating the two sides are:

$$\boldsymbol{w}_p = [w_{p,1}, \ldots, w_{p,N}]^T = \boldsymbol{w}(z; \boldsymbol{x}_p, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)\Big|_{z=\boldsymbol{x}_p} \tag{7}$$

Specifically, we include $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ in this definition because constraints, such as the partition of unity, can only be determined while considering all active neighbors. Nonetheless, the neighbors are implicitly detected by $\boldsymbol{x}_p$; for conciseness, we omit this implicit condition in the remaining part of this paper. The kernel supports transferring information between the nodes and any location $\boldsymbol{z}$ near $\boldsymbol{x}_p$; in most MPM works, the kernel is evaluated at the lagged $\boldsymbol{x}_p$ before the completion of P2G, i.e., we set $\boldsymbol{z} = \boldsymbol{x}_p$ and keep the kernel unchanged for both P2G and G2P.

Following this, the stacked gradients of the kernels are obtained with respect to the spatial variable $\boldsymbol{z}$, then evaluated at $\boldsymbol{x}_p$:

$$\boldsymbol{G}_p = [\boldsymbol{g}_{p,1}, \ldots, \boldsymbol{g}_{p,N}]^T = \left[\nabla_z \boldsymbol{w}(z; \boldsymbol{x}_p)\right]\Big|_{z=\boldsymbol{x}_p}. \tag{8}$$

In the explicit MPM framework, the lumped mass at each background node is defined as $m_i = \sum_p \rho_p V_p w_{p,i}$, where ρ_p represents the density and V_p the volume of each nearby particle. This definition facilitates the calculation of the background node momentum, expressed as

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{f}_i^{\text{int}} + \boldsymbol{f}_i^{\text{ext}}, \tag{9}$$

where $\ddot{\boldsymbol{x}}_i$ is the acceleration of node i, and $\boldsymbol{f}_i^{\text{int}}$ and $\boldsymbol{f}_i^{\text{ext}}$ represent the internal forces and the external forces acting on the it, respectively:

$$\boldsymbol{f}_i^{\text{int}} = -\sum_p V_p \boldsymbol{\sigma}_p \nabla w_{p,i}, \tag{10}$$

$$\boldsymbol{f}_i^{\text{ext}} = \sum_p m_p w_{p,i} \boldsymbol{b}_p + \sum_p m_p w_{p,i} \boldsymbol{g}_p. \tag{11}$$

The stress tensor $\boldsymbol{\sigma}$ is determined by the deformation gradient $\boldsymbol{F}$ through some constitutive relation, indicating how material deformation influences internal forces:

$$\boldsymbol{\sigma} = C(\boldsymbol{F}). \tag{12}$$

Stage 2: Evolution on the Background Nodes Using the accelerations obtained from (9), we integrate the velocities and positions of the background nodes using a symplectic Euler time integrator employed throughout this work:

$$\tilde{\boldsymbol{v}}_i = \boldsymbol{v}_i + \Delta t \ddot{\boldsymbol{x}}_i \quad \text{(Velocity Update)}, \tag{13}$$

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \Delta t \tilde{\boldsymbol{v}}_i \quad \text{(Position Update)}, \tag{14}$$

where the time step size Δt is chosen based on the CFL condition [46].

Stage 3: G2P The FLIP scheme [2] is utilized for all experiments discussed in Sect. 3. In FLIP, the particle positions and velocities are updated as follows:

$$x_p^{\Delta t} = \sum_i w_{p,i} \tilde{\boldsymbol{x}}_i, \tag{15}$$

$$v_p^{\Delta t} = v_p + \Delta t \sum_i w_{p,i} \ddot{\boldsymbol{x}}_i. \tag{16}$$

Subsequently, the evolution of the deformation gradient $\boldsymbol{F}$ in (3) is conducted as follows:

$$\boldsymbol{F}_p^{\Delta t} = \left(\boldsymbol{I} + \sum_i (\tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i)(\nabla w_{p,i})^T \right) \boldsymbol{F}_p. \tag{17}$$

Given the initial deformation gradient $\boldsymbol{F}^0 = \boldsymbol{I}$ and initial volume V_p^0, particle volumes are updated as:

$$V_p^{\Delta t} = \det(\boldsymbol{F}_p^{\Delta t}) V_p^0. \tag{18}$$

Stage 4: Post-Processing and Resetting the Background Nodes This stage encompasses all post-processing tasks such as plasticity return mapping and material hardening [47]. In the updated Lagrangian MPM, the grid is reset to a non-deformed state at the end of each timestep. This is achieved by keeping the grid or mesh constant while zeroing all information such as velocity and acceleration.

2.3 Transfer kernel

In the MPM, the transfer kernel is crucial for relaying particle information to adjacent background nodes. Techniques such as the B-spline MPM [39] and GIMP [36] use a specific compact support function to smoothly influence nearby grid nodes, whereas methods like Moving Least Squares MPM (MLS-MPM) [41] determine the kernel implicitly, based on the proximity of nodes. However, both strategies follow a similar workflow, which involves for every particle: (1) identifying the set of nearby nodes, and (2) calculating the transfer kernel and gradient for every particle-node pair.

This section first introduces the general MLS reconstruction process and the application of MLS-MPM with a comprehensive linear polynomial basis. It is followed by a discussion on a naive extension of MLS-MPM to unstructured meshes, highlighting the steps of identifying nearby nodes and computing the transfer weights. We then delve into the desirable properties of the kernel, emphasizing why the naive extension fails to yield continuous gradient reconstructions when particles cross cell boundaries. Finally, we propose a solution addressing the issue of discontinuous gradient reconstructions and introduce UMLS-MPM.

2.3.1 Introduction to general MLS and MLS-MPM

Given the kernel definition in (7), the Moving Least Squares (MLS) method aims to use a polynomial-based kernel to reconstruct $\hat{u}$ for some function u at any location $\boldsymbol{z}$ near a given particle location $\boldsymbol{x}_p$. It is defined as follows:

$$\hat{u}(\boldsymbol{z}; \boldsymbol{x}_p) = \boldsymbol{p}^T(\boldsymbol{z} - \boldsymbol{x}_p)\boldsymbol{c}(\boldsymbol{x}_p), \tag{19}$$

where $\boldsymbol{p}(\boldsymbol{z} - \boldsymbol{x}_p) = [p_0(\boldsymbol{z} - \boldsymbol{x}_p), p_1(\boldsymbol{z} - \boldsymbol{x}_p), \ldots, p_l(\boldsymbol{z} - \boldsymbol{x}_p)]^T$ represents the polynomial basis, $\boldsymbol{c}(\boldsymbol{x}_p) = [c_0(\boldsymbol{x}_p), c_1(\boldsymbol{x}_p), \ldots, c_l(\boldsymbol{x}_p)]^T$ are the corresponding coefficients, and l indicates the total order of the basis. The coefficients $\boldsymbol{c}(\boldsymbol{x}_p)$ are determined by minimizing the sum of weighted square errors between the sampled function values u_i and the reconstructed values $\hat{u}_i$ at nearby node positions $\boldsymbol{x}_i$:

$$\boldsymbol{c}(\boldsymbol{x}_p) = \text{argmin} \sum_{i \in \mathcal{B}_{\boldsymbol{x}_p}} d(\boldsymbol{x}_i - \boldsymbol{x}_p) \\ ||u_i - \boldsymbol{p}^T(\boldsymbol{x}_i - \boldsymbol{x}_p)\boldsymbol{c}(\boldsymbol{x}_p)||^2, \tag{20}$$

where d is a weighting function that takes proximity as input, and $\mathcal{B}_{\boldsymbol{x}_p}$ is the set of sample points in the local region around $\boldsymbol{x}_p$ where the weighting function is non-zero.

This minimization leads to the following solution for $\boldsymbol{c}(\boldsymbol{x}_p)$:

$$\boldsymbol{c}(\boldsymbol{x}_p) = \boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{B}(\boldsymbol{x}_p)\boldsymbol{u}, \tag{21}$$

where

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{x}_p) &= \sum_{i\in\mathcal{B}_{\boldsymbol{x}_p}} d(\boldsymbol{x}_i-\boldsymbol{x}_p)\boldsymbol{p}(\boldsymbol{x}_i-\boldsymbol{x}_p)\boldsymbol{p}^T(\boldsymbol{x}_i-\boldsymbol{x}_p)\\ &= \boldsymbol{P}(\boldsymbol{x}_p)\boldsymbol{D}(\boldsymbol{x}_p)\boldsymbol{P}(\boldsymbol{x}_p)^T,\end{aligned} \tag{22}$$

and

$$\boldsymbol{B}(\boldsymbol{x}_p) = \boldsymbol{P}(\boldsymbol{x}_p)\boldsymbol{D}(\boldsymbol{x}_p). \tag{23}$$

Here we use the stacked notations: $\boldsymbol{u} = [u_1, \ldots, u_N]^T$ is the stacked sample values, $\boldsymbol{P}(\boldsymbol{x}_p) = [\boldsymbol{p}(\boldsymbol{x}_1-\boldsymbol{x}_p), \ldots, \boldsymbol{p}(\boldsymbol{x}_N-\boldsymbol{x}_p)]$ is the stacked basis, and $\boldsymbol{D}(\boldsymbol{x}_p)$ is the diagonal sample weighting matrix with $\boldsymbol{D}_{i,i}(\boldsymbol{x}_p) = d(\boldsymbol{x}_i-\boldsymbol{x}_p)$.

Substituting (21) into (19), we obtain the reconstruction:

$$\begin{aligned}\hat{u}(z;\boldsymbol{x}_p) &= \boldsymbol{p}^T(z-\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{B}(\boldsymbol{x}_p)\boldsymbol{u}\\ &= \boldsymbol{w}(z;\boldsymbol{x}_p)^T\boldsymbol{u},\end{aligned} \tag{24}$$

where the last derivation is obtained by defining the kernel for MLS, and note that $\boldsymbol{M}^{-1}(\boldsymbol{x}_p)$ is symmetric:

$$\boldsymbol{w}(z;\boldsymbol{x}_p) = \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{p}(z-\boldsymbol{x}_p). \tag{25}$$

Similar to (7) and (8), in the context of MPM, we again set the spatial variable z to be the particle positions $\boldsymbol{x}_p$ before completing P2G and obtain the kernel value:

$$\boldsymbol{w}_p = \boldsymbol{w}(z;\boldsymbol{x}_p)\Big|_{z=\boldsymbol{x}_p} = \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{p}(0), \tag{26}$$

as well as the kernel gradient:

$$\boldsymbol{G}_p = \left[\nabla_z \boldsymbol{w}(z;\boldsymbol{x}_p)\right]\Big|_{z=\boldsymbol{x}_p} = \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)(\nabla_z\boldsymbol{p})(0). \tag{27}$$

The Linear Polynomial Basis Case A special case involves using a complete linear polynomial basis, as in MLS-MPM

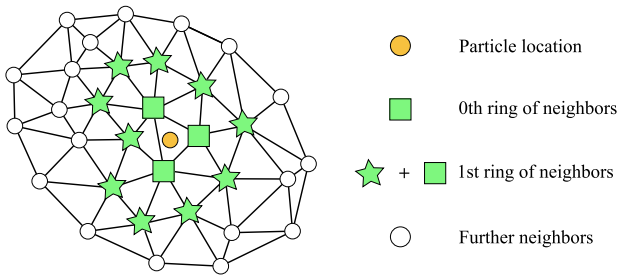


Fig. 1 Schematic plot of the zeroth and first ring of neighbors

[41], where $\boldsymbol{p}(z-\boldsymbol{x}_p) = [1, (z-\boldsymbol{x}_p)^T]^T$. Setting $z = \boldsymbol{x}_p$ in this basis, we have:

$$\begin{aligned}&\boldsymbol{p}(0) = \begin{bmatrix}1\\ \boldsymbol{0}_{\text{dim}}\end{bmatrix},\\ &(\nabla_z\boldsymbol{p})(0) = \begin{bmatrix}\boldsymbol{0}_{\text{dim}}^T\\ \boldsymbol{I}_{\text{dim,dim}}\end{bmatrix},\\ &[\boldsymbol{p}(0), (\nabla_z\boldsymbol{p})(0)] = \boldsymbol{I}_{\text{dim+1,dim+1}};\end{aligned} \tag{28}$$

Substituting (28) into (26) and (27) and stacking $\boldsymbol{w}_p$ and $\boldsymbol{G}_p$ in a column, we obtain a compact formula for both the kernel and the gradient:

$$\begin{aligned}[\boldsymbol{w}_p, \boldsymbol{G}_p] &= \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)[\boldsymbol{p}(0), (\nabla_z\boldsymbol{p})(0)]\\ &= \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{I}_{\text{dim+1,dim+1}}\\ &= \boldsymbol{B}^T(\boldsymbol{x}_p)\boldsymbol{M}^{-1}(\boldsymbol{x}_p).\end{aligned} \tag{29}$$

Applying (29) to the stacked function sample values at nodes $\boldsymbol{u}$, we obtain a compact formula for both the reconstructed function value $\hat{u}_p$ and the gradient $\nabla_z\hat{u}_p$ of u:

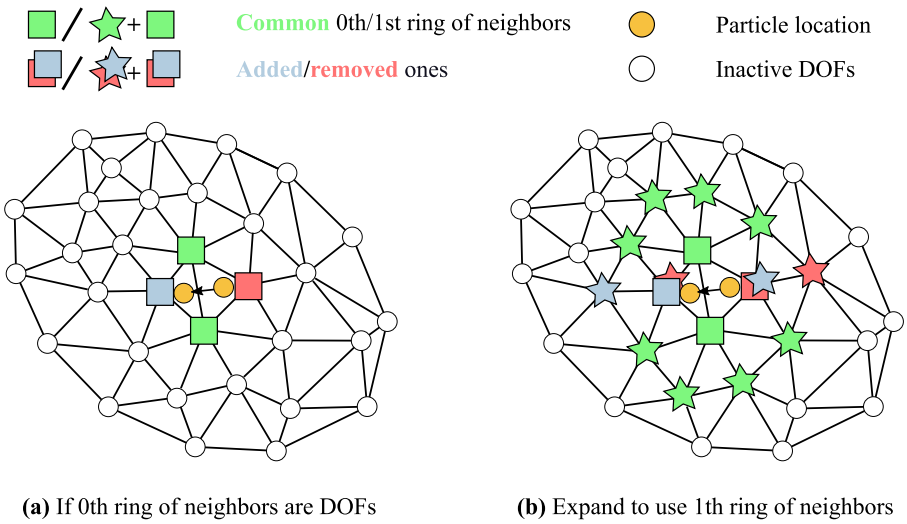
$$\begin{bmatrix}\hat{u}_p\\ \nabla_z\hat{u}_p\end{bmatrix} = [\boldsymbol{w}_p, \boldsymbol{G}_p]^T\boldsymbol{u} = \boldsymbol{M}^{-1}(\boldsymbol{x}_p)\boldsymbol{B}(\boldsymbol{x}_p)\boldsymbol{u}. \tag{30}$$

We adopt the linear basis throughout this work.

2.3.2 Extending MLS-MPM onto unstructured meshes

We select MLS-MPM as our foundation because of its inherent versatility, allowing it to be applied to adjacent nodes without reliance on specific topological or positional constraints. Our implementation and experiments are based on triangular and tetrahedral cells. Nonetheless, it is worth noting that our method can easily be extended to any tessellation by designing a smooth and locally diminishing function η_v compatible with the tessellation, such as the one in (32) for simplex cells.

Identifying Nearby Nodes Around a Particle To determine the nearby nodes for a given particle p, we first locate the cell



(a) If 0th ring of neighbors are DOFs

(b) Expand to use 1th ring of neighbors

Fig. 2 **a** When $\mathcal{N}_p^0$ alone is selected as the active nearby nodes, as a particle crosses the cell edge, the nodes indicated by the blue and red boxes are added or removed, respectively. Consequently, the weights there must approach zero to ensure $\mathcal{C}^0$ continuity, resulting in kernel degeneration along the edge. **b** Advancing to $\mathcal{C}^1$ addresses this issue by incorporating a sufficient number of surrounding nodes to fully encompass the particle

that encompasses p and refer to its nodes as $\mathcal{N}_p^0$, representing the 0-ring neighbors of p. Then, we define $\mathcal{N}_p^1$ as the 1-ring neighbors, which comprise all nodes connected to $\mathcal{N}_p^0$. Note that $\mathcal{N}_p^0 \subset \mathcal{N}_p^1$. Similarly, we can define $\mathcal{N}_p^2, \ldots$ in an analogous manner, as illustrated in Fig. 1.

To quickly search for the cell that contains p, we pre-store the adjacency relationship between the spatial hashing grid and the mesh cells. When given a $\boldsymbol{x}_p$, the spatial hash grid is queried, and we then only check the cells adjacent to this hash grid. The detailed pipeline can be found in Appendix A.

Ring Level Selection for Nearby Nodes When a specific level of ring neighbors is chosen as the active set of nodes, a natural question arises:

What is the minimum number of rings required to satisfy the desired properties of the MPM kernel?

Assume $\mathcal{N}_p^0$ is selected, and the particle only affects the nodes $i \in \mathcal{N}_p^0$; since at least $\mathcal{C}^0$ continuity is required for the kernel, when the particle passes one interface of the cell, the node not on the interface is removed from the active set and the kernel for it must be zero. This leads to the kernel degenerating, i.e., the kernel affecting merely the interface when the particle crosses it, as depicted in Fig. 2a. Conversely, opting for 1-ring neighbors, $\mathcal{N}_p^1$, effectively circumvents this issue, ensuring a non-degenerate kernel interaction as illustrated in Fig. 2b.

Computing the Weights For conciseness, we replace the function input with the subscript, for example replacing $(\boldsymbol{x}_p)$ with p in all related equations, the reconstruction reads:

$$\begin{bmatrix} \hat{u}_p \\ \nabla_z \hat{u}_p \end{bmatrix} = \boldsymbol{M}_p^{-1} \boldsymbol{B}_p \boldsymbol{u}. \tag{31}$$

2.3.3 Required properties for the transfer kernel

Consider the essential desirable properties for an MPM kernel:

1. The kernel must be a non-negative partition of unity. This means that the sum of the kernel weights for all nearby vertices of a particle should equal 1; i.e., $\sum_{v \in \mathcal{N}_p^1} w_v = 1$, with each individual weight $w_v \geq 0, \forall v \in \mathcal{N}_p^1$.
2. There should be a continuous reconstruction of both the function value and gradient as the particle crosses the cell boundary.

With MLS-MPM, the partition of unity is inherently assured by the characteristics of MLS [48], and non-negativity is assured by the uniform sampling of grid nodes (i.e., no degenerate samples). Lastly, with uniform grid nodes, MLS-MPM ensures continuous reconstruction by utilizing a B-spline for sample weighting and provides $\mathcal{C}^1$ continuity.

However, this property holds only under uniform grid nodes with spacing properly aligned with the support of the B-spline weighting function. The key is that the B-spline function approaches zero for grid nodes that are about to be added or removed. Consequently, the influence of the discrete change in the set of active nodes on the assembly of $\boldsymbol{M}_p$ and $\boldsymbol{B}_p$ in (30) is infinitesimal, ensuring no abrupt change during the reconstruction.

Due to the varying spacing of the unstructured meshes, the weighting function is not guaranteed to approach zero for the added or removed nodes during cell crossings, leading to discontinuous reconstruction. This issue will be addressed in the next section.

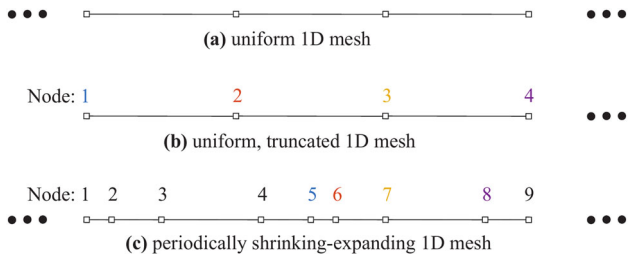


Fig. 3 1D meshes: **a** uniform. **b** Uniform but truncated. **c** Periodically shrinking/expanding

Still, we can borrow the key insight from MLS-MPM that "the weighting function for added or removed nodes should approach zero" and design a scheme to enforce this property. The solution will be discussed and presented in the next section.

2.3.4 Remedying discontinuous reconstruction across the cell boundary

The jump change originates from the discrete change of the active set $\mathcal{N}_p^1$ during particle cell crossing if their influence on the MLS assembly is nonzero. Hence, an intuitive solution is to artificially diminish their influence on the MLS assembly. To achieve this, we multiply any initial sample weighting function $d_{p,i}$, such as B-spline, by a smooth diminishing function $\eta_{p,i}$; i.e., $d'_{p,i} \leftarrow \eta_{p,i} d_{p,i}$. Here, $\eta_{p,i} \to 0$ for nodes that are added or removed from the active set $\mathcal{N}_p^1$ during the cell crossing. A detailed proof of the efficacy of this approach is provided in Appendix B.

For simplex elements, we design the following $\eta_{p,i}$:

$$\eta_{p,i} = \sum_{n \in \mathcal{N}_p^0} B_{p,n} \boldsymbol{A}_{i,n}, \tag{32}$$

where $\boldsymbol{A}$ denotes the mesh's adjacency matrix; an adjacency matrix is a binary matrix representing the connectivity of a graph, where each $\boldsymbol{A}_{i,j} = 1$ indicates the presence of an edge between nodes i and j. A mesh can naturally be viewed as a graph by connecting an edge between every pair of adjacent nodes in a cell. $B_{p,n}$ represents the barycentric coordinate for particle p with respect to a specific node $n \in \mathcal{N}_p^0$. Taking a 2D simplex, the triangular cell, as an example, the barycentric coordinates of a location $\boldsymbol{x}_p$ are triplets of numbers b_1, b_2, b_3, subject to $b_1 + b_2 + b_3 = 1$. If these values are considered as masses placed at the nodes of the triangle, the centroid of these masses will be at $\boldsymbol{x}_p$. Generally, barycentric coordinates can be calculated as follows:

$$B_{p,n_i} = \frac{\det\left([\boldsymbol{x}_{n_1}, \ldots, \boldsymbol{x}_{n_{i-1}}, \boldsymbol{x}_p, \boldsymbol{x}_{n_{i+1}}, \ldots, \boldsymbol{x}_{n_{\dim+1}}]\right)}{\det\left([\boldsymbol{x}_{n_1}, \ldots, \boldsymbol{x}_{n_{\dim+1}}]\right)}. \tag{33}$$

Combining (33) with the adjacency matrix definition provides a more geometric interpretation of the design (32): for $i \in \mathcal{N}_p^1$, $\eta_{p,i}$ is the sum of the barycentric weights for all $n \in \mathcal{N}_p^0$ that are connected to i. Note that $\eta_{p,i} = 1,\ \forall v \in \mathcal{N}_p^0$. Appendix B proves the claimed diminishing property for this design in the simplex cell, while Fig. 25 provides a simple visual illustration of the proposed $\eta_{p,i}$.

2.3.5 Verification of the proposed kernel

To verify that the proposed method can produce continuous reconstruction, analytical and numerical solutions of some examples are produced in 1D and 2D test cases, respectively.

For the 1D case, the first basic verification is conducted on a uniform mesh, as shown in Fig. 3a. Figure 4a shows the correct kernel reconstruction with the diminishing function η, while Fig. 4b, as an ablation, shows that the reconstruction is discontinuous even for the simplest uniform mesh, proving the necessity of η. The detailed setup for this analytical solution is provided in Appendix C.

Note that when a particle is in a boundary cell, such as Node 3 in Fig. 5, negative weight values may be obtained for some interior nodes. This is caused by kernel degeneration due to the absence of a first ring of neighbors on the boundary side during MLS sampling. To remedy this problem, which can cause numerical instabilities [49], an extra layer of cells beyond the real boundary is included in our experiments.

The next verification is on a periodically shrinking and expanding 1D mesh (Fig. 3c). The mesh contains cyclic cell sizes of $[\ldots, 1, R, R^2, R, 1, \ldots]$ designed to mimic the transition between varying mesh resolutions. The size transition ratios tested range from 1.1 to 1.5 to correspond with typi-

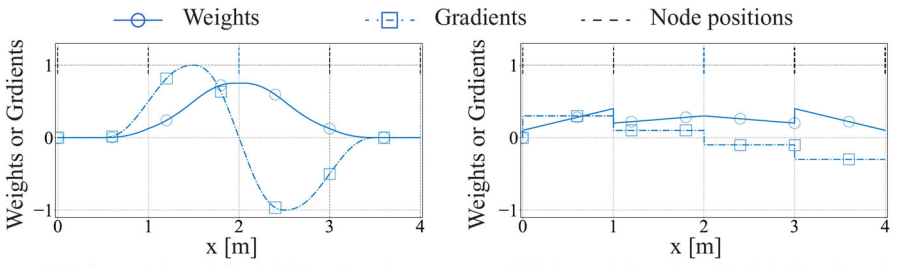


Fig. 4 Comparison of kernel values and gradient estimations on a uniform 1D mesh **a** with and **b** without applying the diminishing function

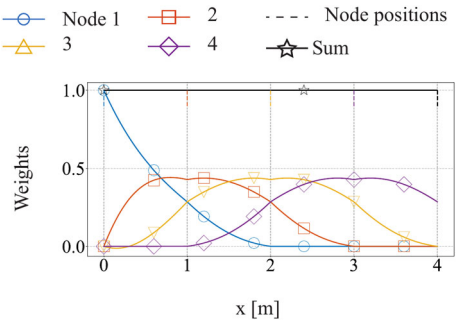


Fig. 5 The negative weight for Node 3 (yellow) when the particle is in the boundary cell and there is no extra layer

cal transition ratios in FEM analysis. Kernel reconstructions are conducted on Nodes 5, 6, 7, and 8 as a full cycle. As shown in Fig. 6, both the kernel and the gradient estimations are piece-wise $\mathcal{C}^1$.

We note that kinks can be seen in the function value plots in Fig. 6, leading to the potential confusion that the gradient is discontinuous. However, the plots of kernel values are $\boldsymbol{w}(z;\boldsymbol{x}_p)|_{z=\boldsymbol{x}_p} = \boldsymbol{w}(\boldsymbol{x}_p;\boldsymbol{x}_p)$ versus $\boldsymbol{x}_p$, as in (7); for the misleading statement to hold true, the gradient should have been taken with respect to the plot axis $\boldsymbol{x}_p$, i.e., $\nabla_{\boldsymbol{x}_p}\boldsymbol{w}(\boldsymbol{x}_p;\boldsymbol{x}_p)$, which is not the case according to (8).

The ablation tests are also performed on a 2D unstructured mesh featuring a "&" shape. The comparison between scenarios with and without the use of η, as shown in Fig. 7a, b respectively, validates the importance of η and the effectiveness of the proposed method in managing unstructured meshes.

Finally, we experimentally show that UMLS-MPM can seamlessly be combined with other schemes, such as the Affine Particle in Cell (APIC) scheme [50, 51] to help conserve the total angular momentum of the system. For details, see Appendix D.

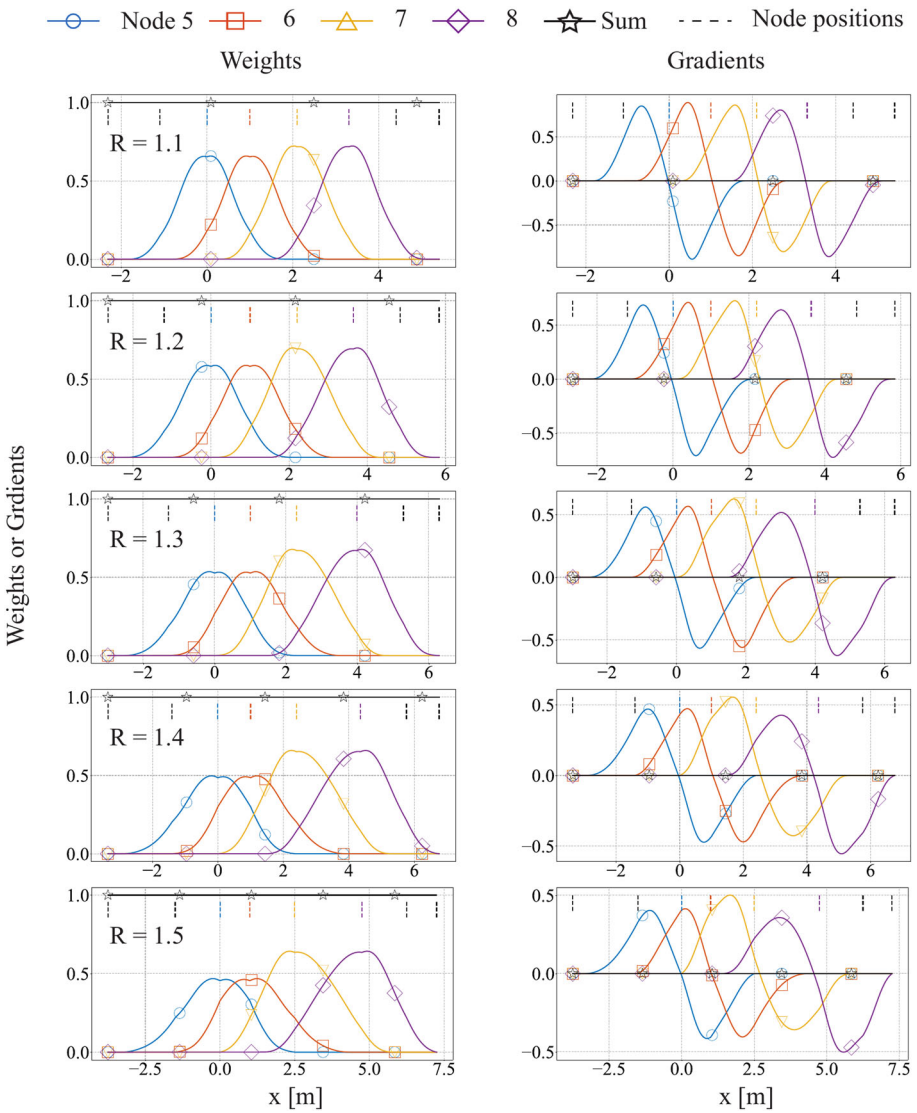
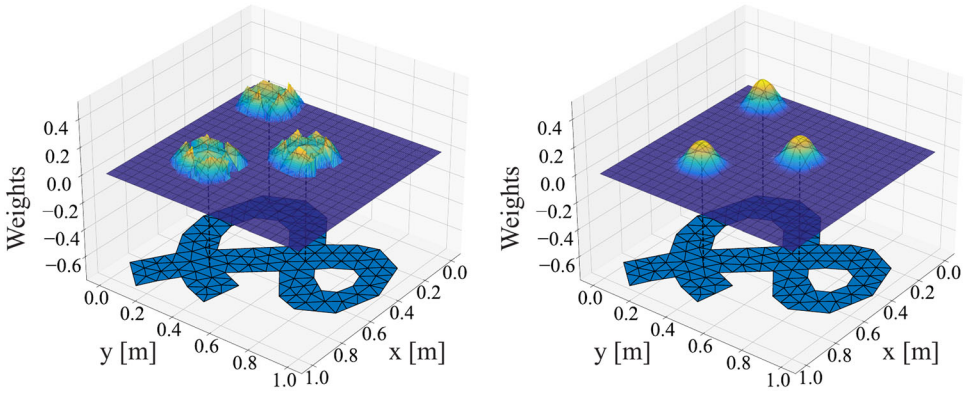
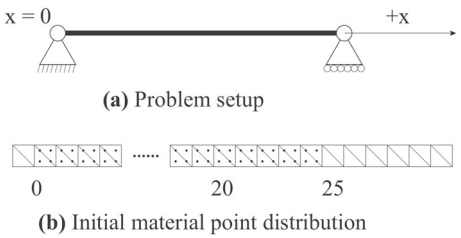


Fig. 6 Kernel values (left column) and gradient estimations (right column) on a periodically shrinking/expanding 1D mesh with varying size transition rate R



(a) The weight w/out diminishing function **(b)** The weight w/ diminishing function

Fig. 7 Comparison of the kernel on an unstructured mesh **a** without and **b** with the application of the diminishing function



(a) Problem setup

(b) Initial material point distribution

Fig. 8 Setup of the 1D bar vibration test

3 Experiments and results

To demonstrate and assess the effectiveness of our approach, particularly its reduced cross-cell error owing to the continuous gradient reconstruction, we have chosen representative test cases from prior related studies. Our benchmarking relies on analytical solutions when feasible; alternatively, we use the standard MPM with B-spline or GIMP basis functions at a sufficiently high resolution. All experiments were carried out on a single PC equipped with an Intel® Core™ i9-10920X CPU.

3.1 1D vibrating bar

Consider the 1D vibration bar problem shown in Fig. 8a [52]. The left end of the bar is fixed and the right has a sliding condition in the x direction. The physical properties of the bar are: $E = 100\,\text{Pa}$, $\nu = 0$, $L = 25\,\text{m}$, and $\rho = 1\,\text{kg/m}^3$. The initial velocity conditions are $\dot{u}(x, t = 0) = v_0 \sin(\beta_1 x)$ with $\beta_1 = \frac{\pi}{2L}$.

The analytical expression of the center of mass in this problem is

$$x(t)_{CM} = \frac{L}{2} + \frac{v_0}{\beta_1 L \omega_1} \sin(\omega_1 t)\,, \tag{34}$$

and

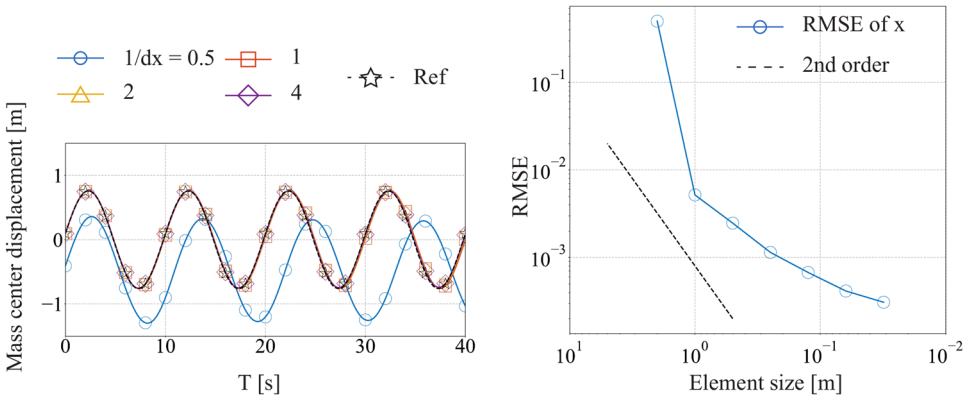
$$\dot{u}(t)_{CM} = \frac{v_0}{\beta_1 L} \cos(\omega_1 t)\,, \tag{35}$$

with $\omega_1 = \beta_1 \sqrt{E/\rho}$.

The original experiments in [52] included two velocity settings: $v_0 = 0.1\,\text{m/s}$ and $v_0 = 0.75\,\text{m/s}$. The lower velocity setting, $v_0 = 0.1\,\text{m/s}$, was utilized solely for validation against the linear kernel MPM, as it does not involve cell crossings. Here, we focus on the higher-velocity setting to assess the effectiveness of UMLS-MPM in addressing cell-crossing errors.

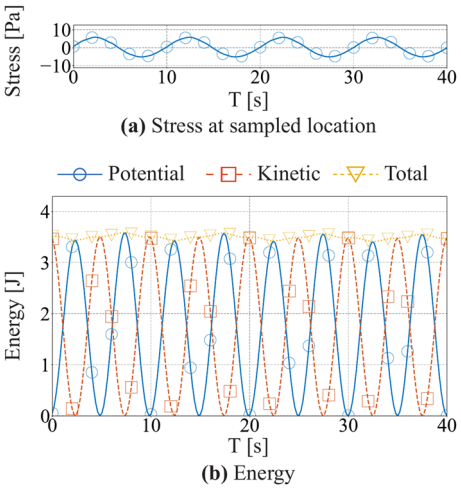
Figure 9 presents the convergence rate of UMLS-MPM with grid refinement. Specifically, Fig. 9a shows that, with the exception of the coarsest resolution $dx = 2\,\text{m}$, UMLS-MPM consistently achieves high accuracy, with a maximum root mean square error (RMSE) of 0.554% in particle displacements. Figure 9b indicates that the convergence rate is approximately second order on coarser grids, but it starts to level off on finer grids due to mounting temporal errors, aligning with established MPM theory [53].

Figure 10a displays the stress profile for a particle located at $x_0 = 12.75\,\text{m}$, which undergoes the most frequent cell crossings during its vibrational motion. The outcomes achieved with UMLS-MPM showcase a remarkable level of smoothness and precision. Figure 10b illustrates the energy dynamics for the entire system, revealing that the system's energy is largely conserved throughout the simulation, with only slight fluctuations. We believe the fluctuations in the energy plot are due to the symplectic integration schemes or the combined effect of the FLIP scheme. Similar phenomena have been observed in previous works [54] and [55], respectively. These findings collectively underscore the robustness and precision of UMLS-MPM in managing intense cell crossings by particles.



(a) Mass center displacement **(b)** Convergence rate plot

Fig. 9 Plots of **a** the center of mass displacement of the bar and **b** convergence rate of the RMSE of particle displacements



(a) Stress at sampled location **(b)** Energy

Fig. 10 Plots of **a** the stress at the sampled particle closest to [17.5, 0.5] and **b** the system energy

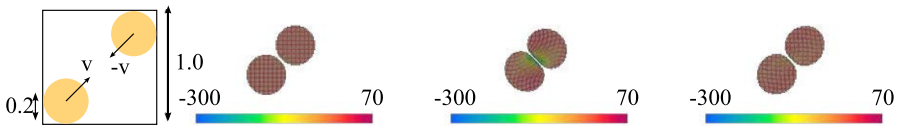
3.2 2D collision disks

Next, we considered the problem of two colliding elastic disks shown in Fig. 11a [52]. The physical properties of the disks are: $E = 1000\,\text{Pa}$, $\nu = 0.3$, $\rho = 1000\,\text{kg/m}^3$, and $\boldsymbol{v} = \pm(0.1, 0.1)\,\text{m/s}$ for the left and right disks, respectively. Each disk was discretized with 462 material points using the triangle mesh of a disk. The background mesh was generated using Delaunay triangulation with a target element size of 0.025 m. We plot key snapshots of the simulation in Fig. 11b–d, with the impact at 1.5 s, total retardation right before 2.0 s, and rebounding separation right before 2.5 s.

Quantitative results for the collision disks are presented in Fig. 12. In Fig. 12a, a comparison of momentum recovery during collision between UMLS-MPM and the B-spline MPM with sufficiently high resolution is shown. While a perfect momentum recovery, such as that in the rigid collision (dashed gray line in Fig. 12a), is not expected, UMLS-MPM approaches this limit effectively. Similarly, Fig. 12b displays the kinetic energy recovery during the collision. The results indicate that UMLS-MPM effectively preserves the system energy. Figure 12c illustrates the stress log at the center particle of the left disk. The results align perfectly with the reference, but only for negligible fluctuations, showing that UMLS-MPM does not generate spurious stress oscillations either from the collision or cell crossings.

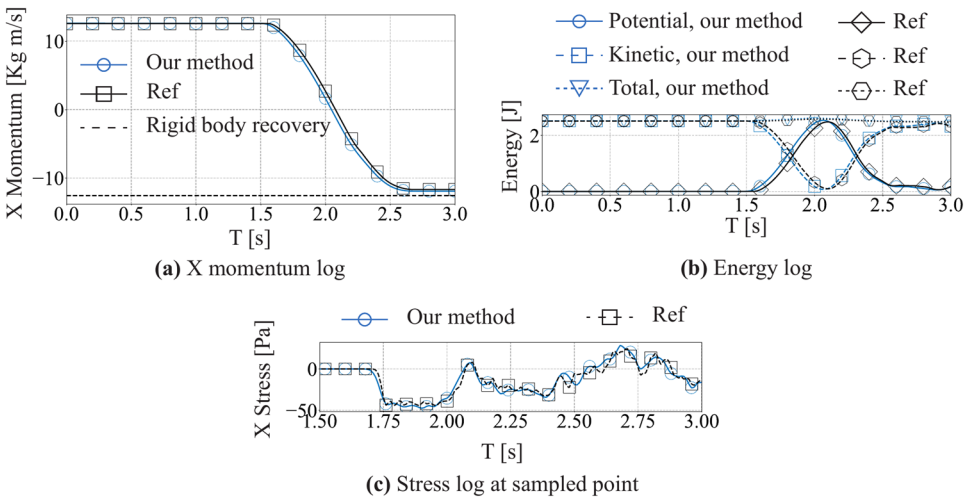
3.3 2D cantilever with rotations

Although an unstructured mesh offers the adaptability to match any boundary shape, the cell orientation, or a different tessellation, can potentially affect accuracy. To illustrate the precision of our method under various rotation angles, we examined the case of a cantilever under its own weight, as shown in Fig. 13a [52]. The cantilever's physical characteristics are as follows: length $l = 10\,\text{m}$, height $h = 2\,\text{m}$, gravitational acceleration $g = 9.81\,\text{m/s}^2$, Young's modulus $E = 100000\,\text{Pa}$, Poisson's ratio $\nu = 0.29$, and density $\rho = 2\,\text{kg/m}^3$. The cantilever was discretized with uniformly spaced particles in both directions. We created the background mesh using Delaunay triangulation, aiming for an element size of 0.5 m. Additionally, we rotated the mesh of the cantilever by angles of 15°, 30°, and 45° to showcase the



(a) setup **(b)** t = 1.5s; before contact **(c)** t = 2.0s; total retardation **(d)** t = 2.5s; rebounding

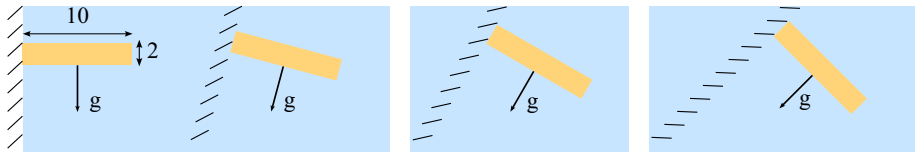
Fig. 11 2D collision disks: **a** problem setup. **b–d** Snapshots of the simulation at 1.5 s, 2.0 s, and 2.5 s



(a) X momentum log **(b)** Energy log

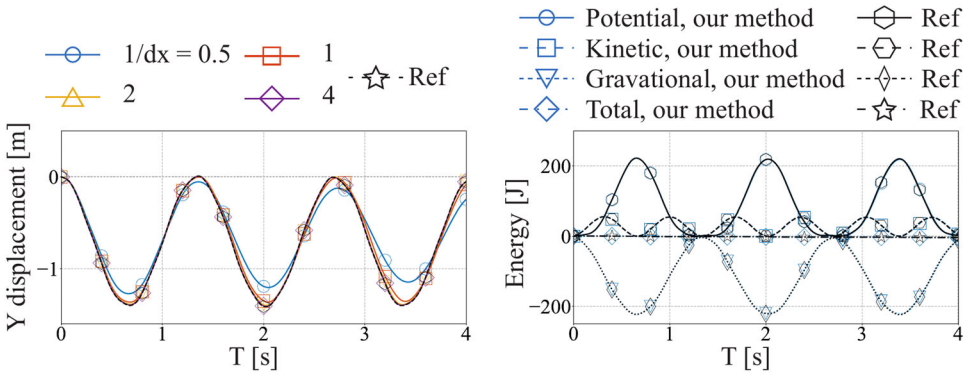
(c) Stress log at sampled point

Fig. 12 Plots of **a** the momentum in the x-direction of the left disk, **b** the energies of the system, and **c** the stress at the sampled particle closest to the center of the left disk



(a) zero rotation **(b)** rotated 15^{o} **(c)** rotated 30^{o} **(d)** rotated 45^{o}

Fig. 13 2D cantilever problem under different rotation angles: **a** 0°, **b** 15°, **c** 30°, and **d** 45°



(a) Y displacement at the right tip **(b)** Energy

Fig. 14 Plots of **a** the displacement in the y-direction at the right tip of the cantilever and **b** the energies of the system

resilience of our method to rotation, as depicted in Fig. 13b–d.

Figure 14a illustrates the spatial convergence of the y-displacement at the right tip of the cantilever beam under grid refinement. Notably, except for the coarse resolutions of $dx = 2\,\text{m}$ and $dx = 1\,\text{m}$, errors for all finer resolutions are negligible. Therefore, a resolution of $dx = 0.5\,\text{m}$ was employed to ensure sufficient accuracy for all subsequent plots in this experiment. Figure 14b demonstrates that UMLS-MPM effectively conserves energy, aligning with the reference B-spline MPM.

Figure 15a shows snapshots of the cantilever with different initial mesh rotation angles. The results indicate that UMLS-MPM is robust under mesh rotation with only minor visible errors. Figure 15b quantitatively compares the y-displacement at the right tip. The results align well overall with both zero rotation and the reference, with errors of 1.27%, 2.18%, and 4.72% for 15°, 30°, and 45° rotation, respectively.

The convergence rate of UMLS-MPM is demonstrated in Fig. 16. The results indicate that for cases with zero rotation, the convergence rate is second order. While the RMSE increases slightly for cases with mesh rotation, it still remains in the magnitude of $1E{-}2$, and the convergence rate remains near second order. These combined results demonstrate the robustness and accuracy of UMLS-MPM under mesh rotation.

3.4 2D ball in a wavy channel

We highlight the proposed method's ability to conform to irregular geometric boundaries. To this end, we consider the

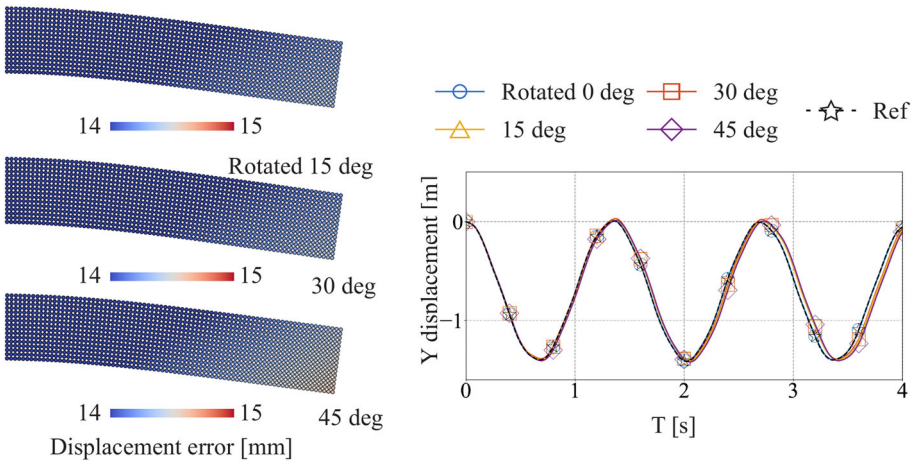


Fig. 15 **a** Snapshots of the cantilever with different initial rotating angles. **b** Comparison of the displacement in the y-direction at the right tip of the cantilever

(a) Error comparison

(b) Y displacement comparison

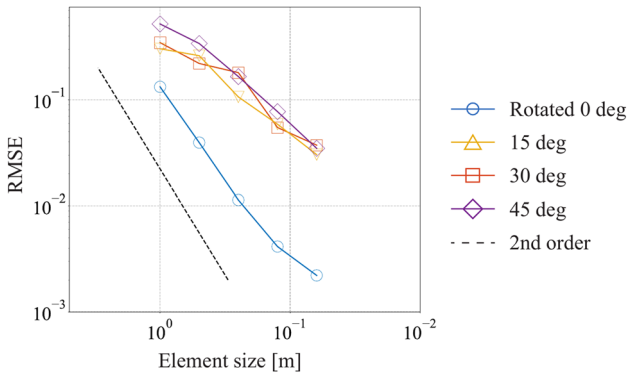


Fig. 16 Convergence plot of the RMSE of particle displacements

case of a ball freely falling but confined in a wavy channel, as shown in the leftmost subfigure of Fig. 17. The physical properties of the ball are: radius $r = 1.0$ m, Young's modulus $E = 100$ kPa, Poisson's ratio $\nu = 0.29$, and density $\rho = 400$ kg/m^3. The ball was discretized with 4735 randomly sampled material points. The background wavy channel has a sinusoidal shape.

The left wall of the channel has an analytical expression of:

$$x_l = \begin{cases} A\sin(\omega_1 y)\sin(\omega_2 y), & 0 < y \leq 20.0 \text{ m} \\ 0, & \text{Otherwise,} \end{cases} \tag{36}$$

where $A = 1.0$ m, $\omega_1 = \frac{\pi}{5}$ rad/m, and $\omega_2 = \frac{\pi}{20}$ rad/m. The right wall is created by shifting the left wall by 2.0 m, i.e., $x_r = x_l + 2.0$ m. The background mesh was generated using Delaunay triangulation with a target element size of 0.05 m, resulting in 43,360 cells.

The snapshots of the simulation at 1.4 s, 3.0 s, and 6.5 s are shown in Fig. 17, while the zoomed-in views of the ball's deformation and hydrostatic stress are shown in Fig. 18. UMLS-MPM captures both the bouncing into the wavy channel (at 1.1 s, 1.9 s, 3.7 s) and the squeezing through the narrow part of the channel (at 1.4 s, 2.4 s, 3.0 s) with no rasterization artifacts, proving the robustness of the proposed method in simulating under general mesh tessellation and handling irregular geometry boundaries.

3.5 3D slope failure

Next, the performance of the proposed approach was investigated when dealing with material behavior involving plasticity. To this end, we simulated failure of a 3D slope comprosed of sensitive clay. The problem geometry was adopted from [56] and is illustrated in Fig. 19. Here, the bottom boundary of the slope is fixed and the three lateral sides are supported with rollers. To model the elastoplastic behavior of the sensitive clay in an undrained condition, a combination of Hencky elasticity and J2 plasticity with softening was used. The softening behavior is governed by the following exponential form: $\kappa = (\kappa_p - \kappa_r)e^{-\eta\varepsilon_q^p} + \kappa_r$, where κ, κ_p, and κ_r denote the yield strength, the peak strength, and the residual strength, respectively, ε_q^p denotes the equivalent plastic strain, and η is a softening parameter. The specific parameters were adopted from [56]. They are a Young's modulus of $E = 25$ MPa, a Poisson's ratio of $\nu = 0.499$, a peak strength of $\kappa_p = 40.82$ kPa, a residual strength of $\kappa_r = 2.45$ kPa, and a softening parameter of $\eta = 5$. The assigned soil density is $\rho = 2.15$ t/m^3.

The space was discretized using Delaunay triangulation with the shortest edge length of 0.2 m. The material points were initialized with a spacing of 0.1 m in each direction, amounting to 311,250 material points in the initial slope region. Note that the spatial discretization aligns with the one used in [56] in terms of both the shortest edge length of the background element and the number of material points. Also, the $\bar{F}$ approach proposed in [56] was utilized to circumvent volumetric locking that UMLS-MPM solutions encounter

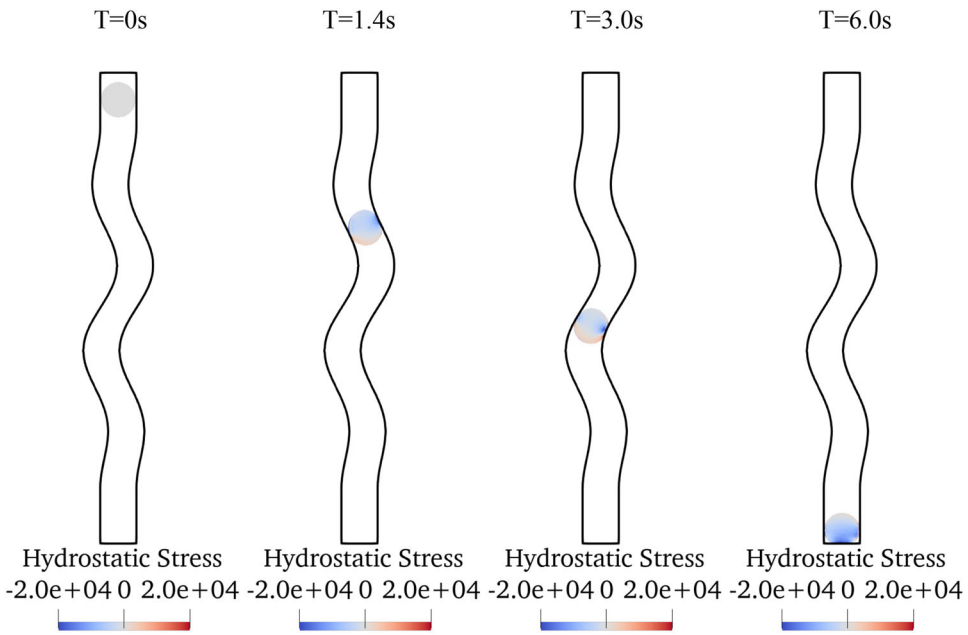


Fig. 17 2D ball in a wavy channel: snapshots of the simulation at 0 s, 1.4 s, 3.0 s, and 6.5 s

when simulating a large number of particles of incompressible materials. As a reference to verify the correctness of the proposed formulation, the $\bar{\boldsymbol{F}}$ solution in [56] was used.

Figures 20 and 21 show the snapshots of the slope simulated by the standard and UMLS-MPM, where particles are colored by the equivalent plastic strain and mean normal stress, respectively. We can see that UMLS-MPM effectively captures the retrogressive failure pattern of slopes made of sensitive clay. Also, in terms of equivalent plastic strain fields and mean normal stress fields, we observe a strong similarity between the UMLS-MPM solution and the reference solution from [56].

For a further quantitative comparison, Fig. 22 presents the time evolutions of the run-out distance—a measure of the farthest movement of the sliding mass. Observe that the distances in the standard and UMLS-MPM solutions are remarkably similar. Taken together, these findings confirm that the proposed method performs similarly to the standard MPM.

3.6 3D elastic object expansion in a spherical container

Finally, we examined the performance of UMLS-MPM in problems involving complex boundary geometry. In this problem, the standard MPM with a structured grid may be challenged to impose conforming boundary conditions. Hence, a collision between an elastic body with a spherical container was considered and simulated.

The geometry of the problem, as demonstrated in Fig. 23, involves an elastic object in the shape of a Metatron, which is located at the center of a spherical container (with a radius of 0.5 m). The object is initially compressed isotropically (with an initial deformation gradient of $\boldsymbol{F} = 0.75\boldsymbol{I}$), storing non-zero elastic potential energy. At the onset of the simulation, the stored elastic energy is released, causing the object to expand and collide with the spherical container's boundary. To capture the elastic behavior of the object, a Neo-hooken elasticity was adopted with a Young's modulus of 3.3 MPa and a Poisson's ratio of $\nu = 0.49$. The elastic object was discretized using a significant number of material points (2,392,177) for high-fidelity simulation. Also, the spherical container was discretized using 2,178,129 tetrahedral elements, each with an average edge length of $h = 0.025$ m. Note that to avoid negative kernel values at boundary nodes, an extra layer of elements was added outside the original boundary, as discussed in Sect. 2.3.5.

To consider the frictional collision between the elastic object and the container boundary, a barrier approach [57] was adopted, ensuring that the elastic object does not penetrate the boundary. Contact forces are applied when the distance between a material point and the boundary is below a specific value $\hat{d}$, which was chosen to be a quarter of h for sufficient accuracy. Also, a friction coefficient of $\mu = 0.5$ was introduced to stop the sliding of the elastic object in the later stages. The simulation ran with a time increment of $\Delta t = 6.16 \times 10^{-5}$ s until $t = 1.5$ s.

Figure 24 presents six snapshots simulated with UMLS-MPM, where the particles are colored based on the magnitude of the contact force. The dynamic behavior of the object at various stages is well captured, including the initial expansion stage (a), the first collision stage (b–d), the rebounding stage (e), and the final static stage (f). Overall, the UMLS-MPM effectively handles complex geometry with

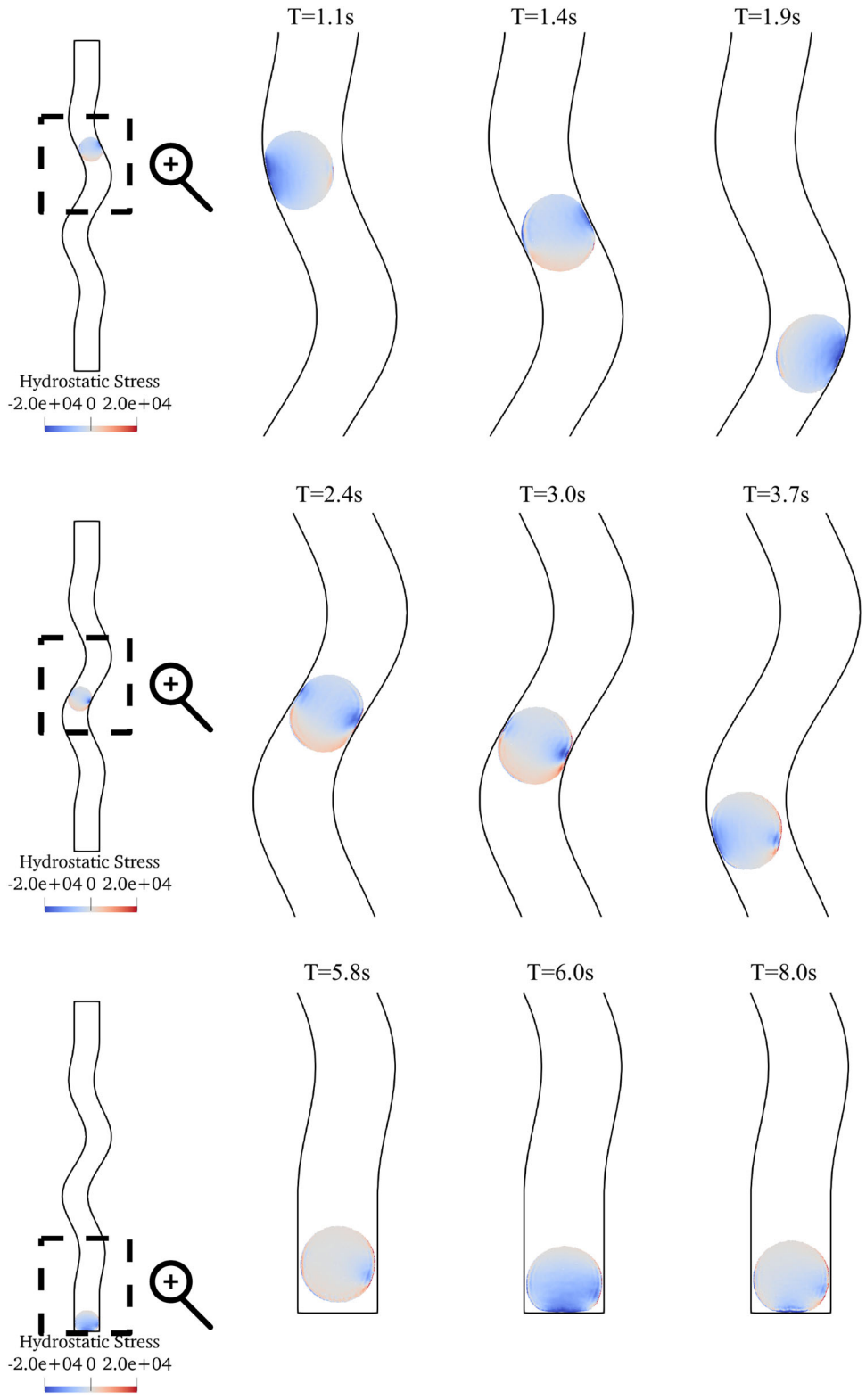


Fig. 18 Zoomed-in view of the ball's deformation in the wavy channel at key timestamps. From top to bottom, timestamps around 1.4 s, 3.0 s, and 6.0 s

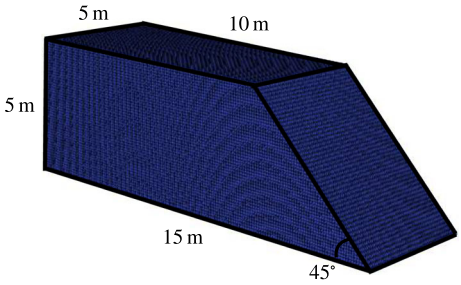


Fig. 19 Problem geometry of the 3D slope failure (adapted from [56])

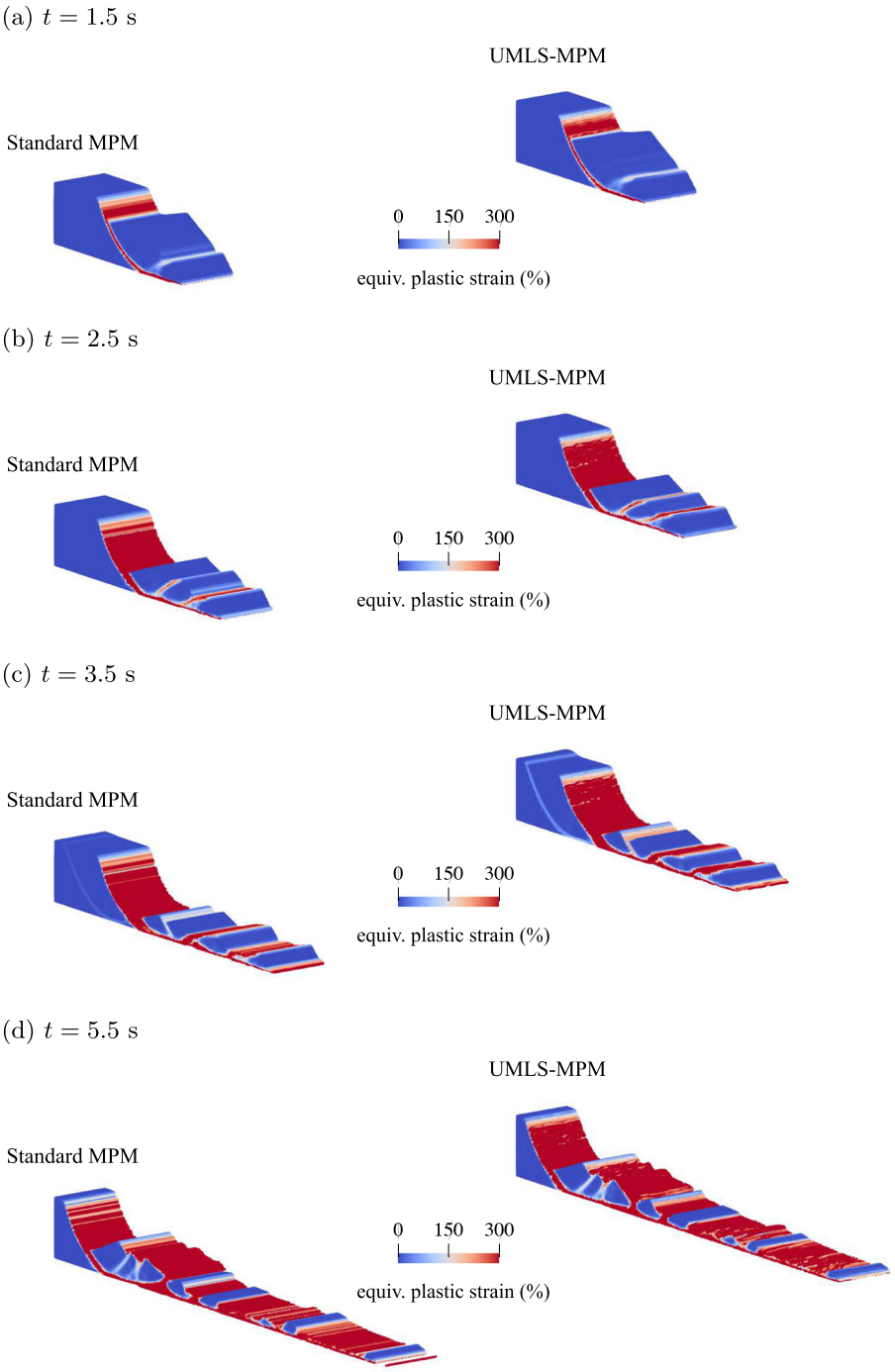


Fig. 20 Snapshots of the solutions from the UMLS-MPM and the standard MPM with GIMP basis functions in Zhao et al.[56]. Particles are colored based on the equivalent plastic strain

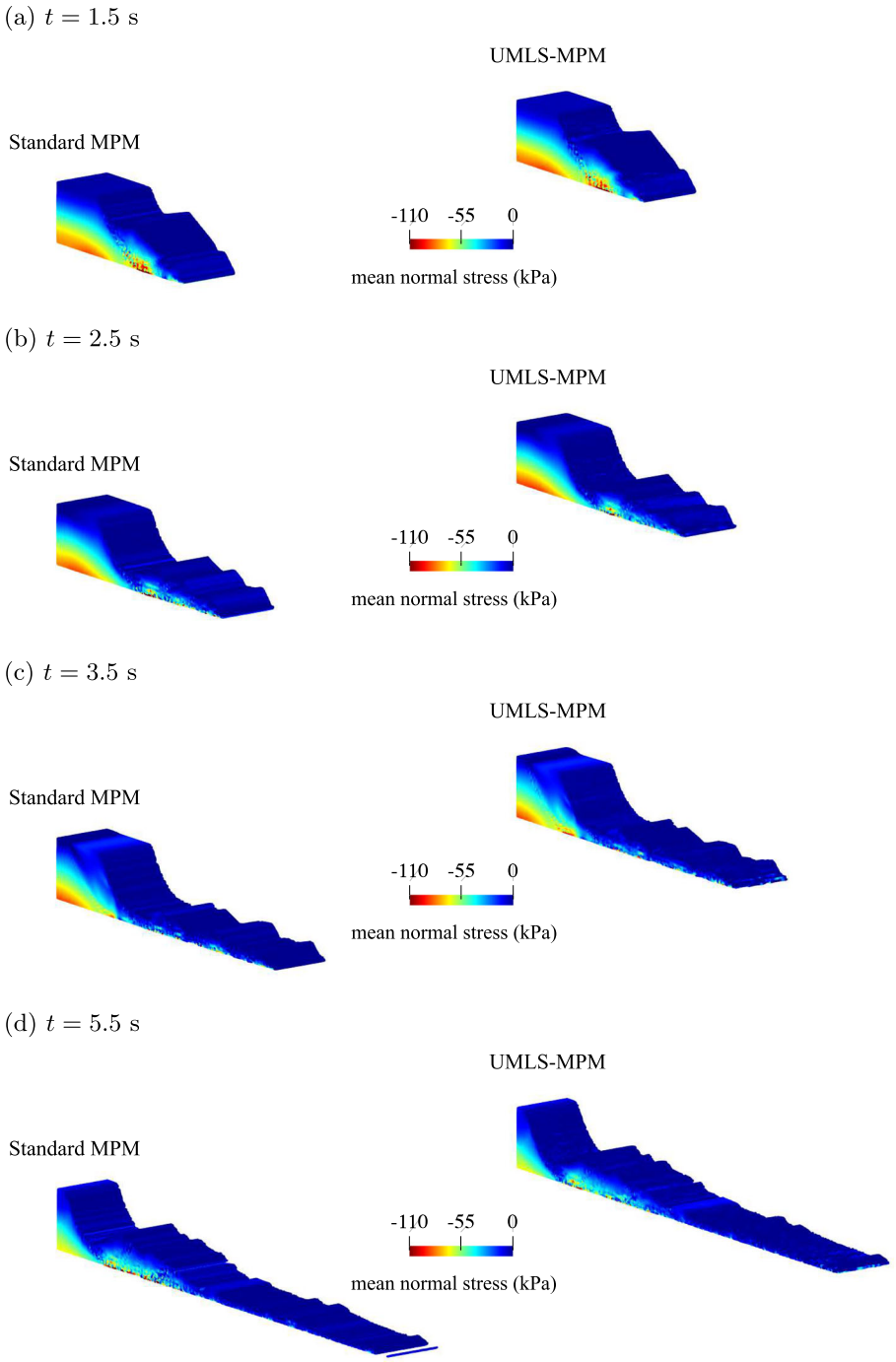


Fig. 21 Snapshots of the solutions from the UMLS-MPM and the standard MPM with GIMP basis functions in Zhao et al.[56]. Particles are colored based on the mean normal stress

a conformal discretization, which is critical for simulating a wide range of interactions between deformable objects and complex boundaries.

4 Conclusion, limitations, and future work

This study has extended the Moving Least Square Material Point Method to encompass general tessellations within both 2D and 3D meshes. This advance has been achieved through the multiplication of a diminishing function to the MLS sample weights. Analytically proved, the approach ensures continuous kernel reconstruction and provides a sound foundation for MPM on any unstructured mesh types.. Several numerical experiments in both 2D and 3D domains have demonstrated the method's effectiveness in achieving high-order convergence and eliminating cell-crossing errors. However, the proposed method still has limitations.

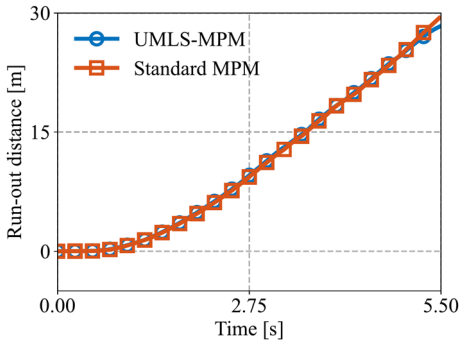


Fig. 22 Time evolutions of the run-out distance from UMLS-MPM and the standard MPM with GIMP basis functions

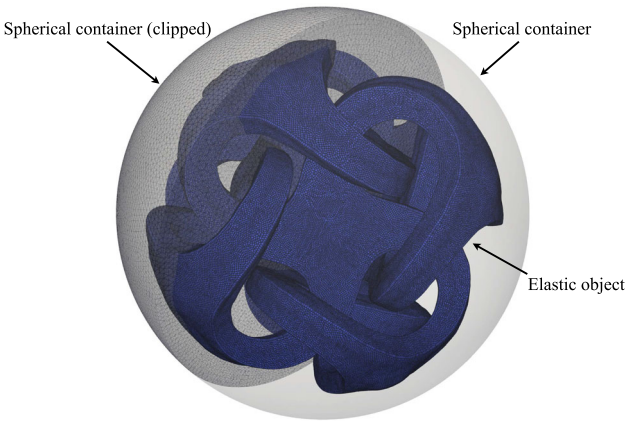


Fig. 23 Problem setting of the 3D elastic object expansion in a spherical container

To ensure the MLS solution does not degenerate, there are quality requirements for the surrounding nodes of the particle, which pose challenges, especially in meshes with sharp resolution transitions or poor quality. Additionally, related to varying mesh resolution is the sample weights: we used a B-spline function with a fixed support radius equaling the maximum edge size to ensure that the first ring of neighbors is reserved in the coarsest area. However, this strategy leads to nearly uniform sample weights in finer regions, blurring the kernel. A potential solution could be to incorporate a sizing field within the mesh to dynamically adjust the sample weight function.

Although UMLS-MPM has advantages in conforming to irregular geometry, this also brings potential issues and leaves room for further improvements. For example, the lack of surrounding samples on one side when the particle is inside the boundary cell can lead to kernel degeneration. While this issue was mitigated in our experiments by drawing an extra layer of cells, an automatic and algorithmic approach is more appealing. Combining UMLS-MPM with regular MPM is also promising, as it leverages the advantage of UMLS-MPM in conforming to boundary shapes, as well as the robustness and efficiency of regular MPM for interior domains. For the transition between interior nodes and boundary nodes, methods similar to immersed FEM [58–60] could be effective, as they project information from regular background grids to irregular meshes.

From a practical standpoint, we need not a higher-order polynomial basis or higher rings of neighbors as long as the current schemes, which have the lowest cost as discussed in Sect. 2, provides sufficient accuracy. However, from a theoretical perspective, these two prososals are still interesting. Several challenges can be noted: 1. Changing to higher rings of neighbors will require a different solution for the diminishing function (32) to accommodate the new discrete change of active sets. 2. The higher-order polynomial basis will result in a different MLS solution compared to the linear solution (31). Additionally, since the propositions in Appendix B are based on the linear solution, it remains an open question whether the higher-order solution has a continuous kernel and gradient, even with a diminishing function. 3. The higher-order polynomial basis will necessitate a more delicate selection of quadrature point locations. Unlike higher-order FEM simulations, where the quadrature points are usually well-designed and fixed [61], in MPM, these points may need to be adjusted with the particle locations, requiring extra design and attention.

From an application standpoint, exploring the integration between the material point method (MPM) and gas or fluid simulations via the Finite Volume Method (FVM) presents significant potential [21, 62]. For simulating granular materials in realistic and irregular container shapes, combining UMLS-MPM with the Discrete Element Method (DEM) [63] is also an interesting direction. Moreover, object

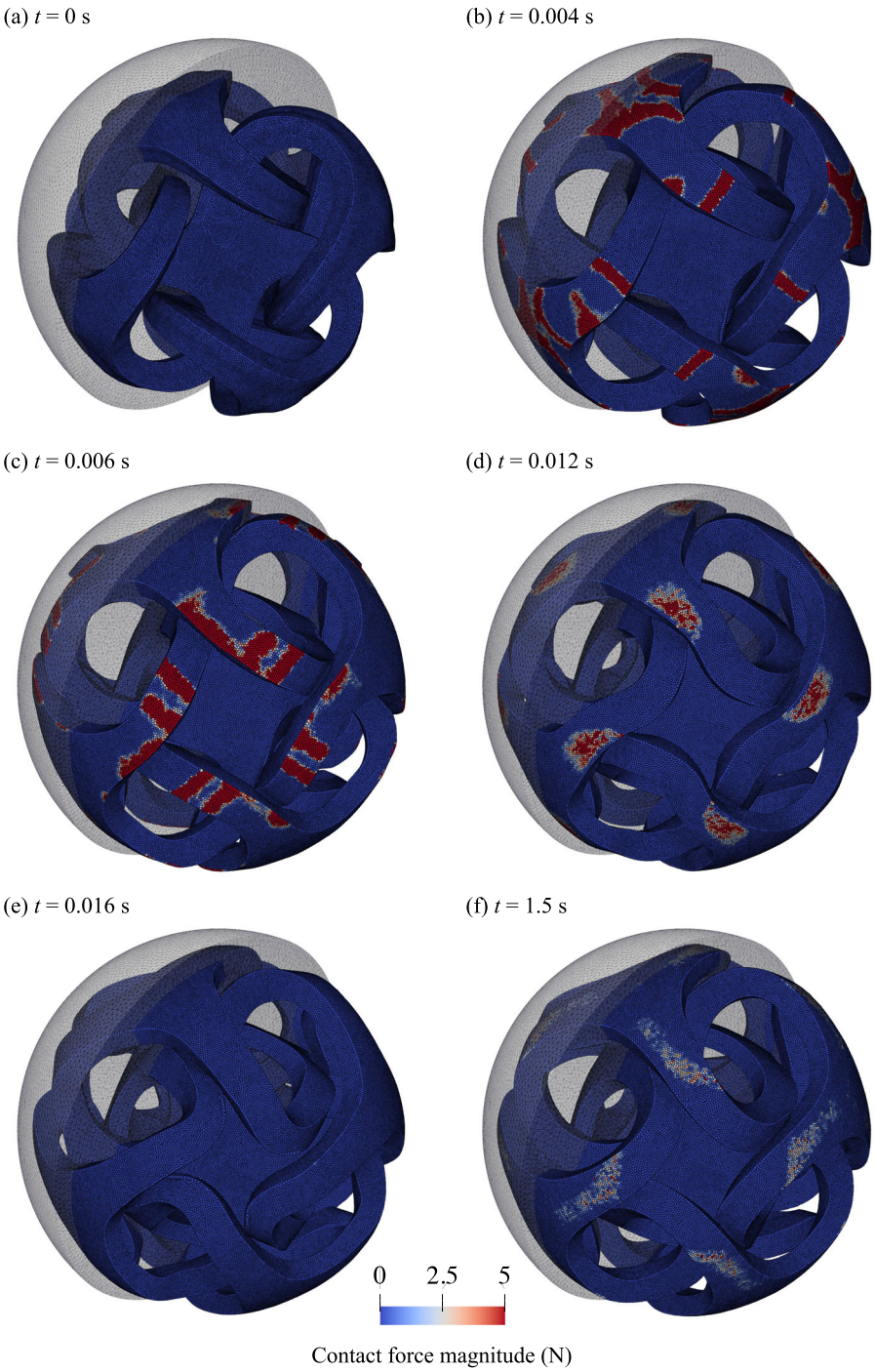


Fig. 24 Snapshots from the UMLS-MPM solutions in which particles are colored based on the magnitude of the contact force

contact detection and handling [64] is crucial to prevent artificial penetration and sticking commonly seen in MPM [53]. In the field of scientific machine learning, recent advances have demonstrated learning MPM or other mesh-based simulations using graph neural networks, accelerating the inferences [65–67]. Our kernel construction suggests a potentially novel learning paradigm for MPM on unstructured meshes, similar to embedding both kernel and mesh information into the network's channel [68, 69].

A Pipeline for rapid cell search

Fast determination of which cell contains a specific particle is crucial for the efficiency of UMLS-MPM. We propose a hash grid-based method to accelerate this process. The pipeline consists of two steps: (1) building the hash grid connectivity table, a dictionary mapping the hash grid as a key to all of its touching cell indices, as detailed in Algorithm 2, and (2)

Algorithm 2 Build Hash to Adjacent Cell

```
Input: Mesh node positions: pos, mesh cells: cell, hash grid size: dx
Output: Hash to adjacent cell connectivity: hash2cell
Calculate the bounding box of the whole mesh
        min_pos ← min(pos, axis=0)
        max_pos ← max(pos, axis=0)
Initialize hash2cell as an empty dictionary
        hash2cell ← {}
For each cell, find its bounding box, determine the spatial hash grids it touches,
  and append the cell to those hash grids
for cell_idx in range(len(cell)) do
          cell_min ← min(pos[cell[cell_idx]], axis=0)
          cell_max ← max(pos[cell[cell_idx]], axis=0)
          min_idx ← floor((cell_min - min_pos) / dx)
           max_idx ← ceil((cell_max - min_pos) / dx)
           range ← indices(max_idx - min_idx)
           candidates ← range.reshape(dim, -1).T + min_idx
    for c in candidates do
                    hash2cell[c].append(cell_idx)
    end for
end for
```

Algorithm 3 Find Containing Cell

```
Input: Mesh node positions: pos, mesh cells: cell, Mesh minimum position: min_pos,
  hash grid size: dx, Hash to adjacent cell connectivity: hash2cell, particle position: xp
Output: Containing cell index or None
Get the hash spatial index for the particle
        hash_idx ← floor((xp-min_pos) / dx)
Get all adjacent cells to this hash grid
        candidates ← hash2cell[hash_idx]
For each candidate cell, check if the particle is inside
for c in candidates do
   e ← cell[c]
   vs ← pos[e]
   # get the bounding box of the cell
    e_min ← min(vs, axis=0)
    e_max ← max(vs, axis=0)
    # quick filter using bounding box
    if all(xp >= e_min) and all(xp <= e_max) then
       # check barycentric coordinates
       bc ← barycentric_coord(vs, xp)
       if all(bc >= 0) then
          return c
       end if
    end if
end for
# If no candidate cells contain xp
return None
```

the online search for determining which cell contains a given particle, as described in Algorithm 3.

B Proofs for the continuous reconstructions

For conciseness, we drop the subscripts p in the following proofs. We start by assuming there exists a smooth, locally diminishing function η for the nodes added or removed from the set of nearby nodes $\mathcal{N}^1$ when a particle crosses the boundary of a cell. Under this assumption, we can prove that our kernel value and gradient estimation is continuous across the boundary. We present the proof in 2D when a particle crosses an edge; the extension to 3D and other crossing cases is straightforward. Finally, we prove that (32) satisfies the forementioned assumption.

Proposition *Our kernel value and gradient estimation is continuous across cell boundaries.*

Proof Let $\mathcal{N}^1_{o,n}$ be the sets of nearby nodes before/after the particle p crosses the common edge between the old/new cells $\mathcal{N}^0_{o,n}$. Here, the subscripts o, n denote the old/new cell, respectively, and the superscripts 0, 1 indicate the ring-0/1 neighbors of the cell, respectively. Let $\boldsymbol{x}^{o,n}$ be the position of particle p before/after the crossing and $||\boldsymbol{x}^n - \boldsymbol{x}^o|| = \mathcal{O}(\epsilon)$. Define the common node set $\mathcal{N}^1_c = \mathcal{N}^1_o \cap \mathcal{N}^1_n$, the added node set $\mathcal{N}^1_a = \mathcal{N}^1_n \backslash \mathcal{N}^1_c$, and the removed node set $\mathcal{N}^1_r = \mathcal{N}^1_o \backslash \mathcal{N}^1_c$. Since η is locally diminishing for $v \in \mathcal{N}^1_{a,r}$, we have a positive value K_1 such that $\eta = \mathcal{O}(K_1\epsilon) = \mathcal{O}(\epsilon)$. The pertubation for the assembled matrix $\boldsymbol{M}$ before/after the particle p crosses an edge is

$$\delta \boldsymbol{M} = \sum_{v \in \mathcal{N}^1_c} \delta(\eta d \boldsymbol{p}\boldsymbol{p}^T) + \sum_{v \in \mathcal{N}^1_a} \eta d \boldsymbol{p}\boldsymbol{p}^T - \sum_{v \in \mathcal{N}^1_r} \eta d \boldsymbol{p}\boldsymbol{p}^T, \tag{37}$$

where the first term is continuous by construction since every factor is smooth; i.e., $||\delta(\eta d \boldsymbol{p}\boldsymbol{p}^T)|| = \mathcal{O}(\epsilon)$. For the second and third terms, since $\eta = \mathcal{O}(\epsilon)$, we have

$$\begin{aligned} ||\delta \boldsymbol{M}|| &\le \sum_{v \in \mathcal{N}^1_c} ||\delta(\eta d \boldsymbol{p}\boldsymbol{p}^T)|| + \sum_{v \in \mathcal{N}^1_a} ||\eta d \boldsymbol{p}\boldsymbol{p}^T|| \\ &\quad + \sum_{v \in \mathcal{N}^1_r} ||\eta d \boldsymbol{p}\boldsymbol{p}^T|| \\ &\le \left[|\mathcal{N}^1_c| + \left(|\mathcal{N}^1_a| + |\mathcal{N}^1_r| \right) \max_{v \in \mathcal{N}^1_{a,r}} ||d \boldsymbol{p}\boldsymbol{p}^T|| \right] \mathcal{O}(\epsilon) \\ &= \mathcal{O}(|\mathcal{N}^1| h^2 \epsilon) \\ &= \mathcal{O}(\epsilon). \end{aligned} \tag{38}$$

Here, as long as the mesh has a reasonably good quality, $|\mathcal{N}^1|$ is finite and small; i.e., there is a finite and small amount of ring-1 neighbors. Also, h, a constant, is the support radius of the kernel, outside of which the weight is zero. In all, both $|\mathcal{N}^1|$ and h can be omitted in the analysis.

The perturbation of the inverse matrix is given by

$$\begin{aligned} ||\delta \boldsymbol{M}^{-1}|| &= ||(\boldsymbol{M} + \delta \boldsymbol{M})^{-1} - \boldsymbol{M}^{-1}|| \\ &= ||\boldsymbol{M}^{-1} - \boldsymbol{M}^{-1} \delta \boldsymbol{M} \boldsymbol{M}^{-1} \\ &\quad + \mathcal{O}(||\delta \boldsymbol{M}||^2) - \boldsymbol{M}^{-1}|| \\ &= ||\boldsymbol{M}^{-1} \delta \boldsymbol{M} \boldsymbol{M}^{-1} + \mathcal{O}(\epsilon^2)|| \\ &\le ||\boldsymbol{M}^{-1} \delta \boldsymbol{M} \boldsymbol{M}^{-1}|| + \mathcal{O}(\epsilon^2) \\ &\le ||\boldsymbol{M}^{-1}||^2 \cdot ||\delta \boldsymbol{M}|| + \mathcal{O}(\epsilon^2) \end{aligned}$$

$$
\begin{aligned}
&= \frac{||\delta \boldsymbol{M}||}{\sigma(\boldsymbol{M})^2_{\min}} + \mathcal{O}(\epsilon^2) \\
&= \mathcal{O}\left(\frac{\epsilon}{\sigma(\boldsymbol{M})^2_{\min}}\right) + \mathcal{O}(\epsilon^2) \\
&= \mathcal{O}\left(\frac{\epsilon}{\sigma(\boldsymbol{M})^2_{\min}}\right),
\end{aligned} \tag{39}
$$

where $\sigma(\boldsymbol{M})_{\min}$ is the minimum singular value of $\boldsymbol{M}$.

Similarly, for the perturbation in the assembled vector $\boldsymbol{Bu} = \boldsymbol{PDu}$ before/after the particle crossing is

$$
\begin{aligned}
&||\delta(\boldsymbol{PDu})|| \\
&= ||\sum_{v\in\mathcal{N}_c^1} \delta(\eta du \boldsymbol{p}) + \sum_{v\in\mathcal{N}_a^1} \eta du \boldsymbol{p} - \sum_{v\in\mathcal{N}_r^1} \eta du \boldsymbol{p}|| \\
&\le \sum_{v\in\mathcal{N}_c^1} ||\delta(\eta du \boldsymbol{p})|| + \sum_{v\in\mathcal{N}_a^1} ||\eta du \boldsymbol{p}|| + \sum_{v\in\mathcal{N}_r^1} ||\eta du \boldsymbol{p}|| \\
&\le \left[|\mathcal{N}_c^1| + \left(|\mathcal{N}_a^1| + |\mathcal{N}_r^1|\right) \max_{v\in\mathcal{N}_{a,r}^1} ||du\boldsymbol{p}||\right]\mathcal{O}(\epsilon) \\
&= \mathcal{O}(|\mathcal{N}^1|h\epsilon) \\
&= \mathcal{O}(\epsilon).
\end{aligned} \tag{40}
$$

Furthermore, we can establish the following bound for the assembled vector $\boldsymbol{PDu}$:

$$
\begin{aligned}
||\boldsymbol{PDu}|| &= ||\sum_{v\in\mathcal{N}^1} \eta du \boldsymbol{p}|| \\
&\le |\mathcal{N}^1| \cdot \max_{v\in\mathcal{N}^1} ||\eta du \boldsymbol{p}|| \\
&= \mathcal{O}(|\mathcal{N}^1|h) \\
&= \mathcal{O}(1).
\end{aligned} \tag{41}
$$

Finally, the perturbation for $\left[\hat{u}, \nabla\hat{u}^T\right]^T$ from (31) is

$$
\begin{aligned}
\left[\hat{u}, \nabla\hat{u}^T\right]^T &= ||\delta(\boldsymbol{M}^{-1}\boldsymbol{Bu})|| \\
&= ||\delta(\boldsymbol{M}^{-1}\boldsymbol{PDu})|| \\
&= ||\delta\boldsymbol{M}^{-1}\boldsymbol{PDu} + \boldsymbol{M}^{-1}\delta(\boldsymbol{PDu})|| \\
&\le ||\delta\boldsymbol{M}^{-1}\boldsymbol{PDu}|| + ||\boldsymbol{M}^{-1}\delta(\boldsymbol{PDu})|| \\
&\le ||\delta\boldsymbol{M}^{-1}|| \cdot ||\boldsymbol{PDu}|| + ||\boldsymbol{M}^{-1}|| \cdot ||\delta(\boldsymbol{PDu})|| \\
&= \mathcal{O}\left(\left(\frac{1}{\sigma(\boldsymbol{M})^2_{\min}} + \frac{1}{\sigma(\boldsymbol{M})_{\min}}\right)\epsilon\right).
\end{aligned} \tag{42}
$$

In the incomplete singular value decomposition of $\boldsymbol{M}$, the singular values will always be non-negative. And if the surrounding nodes are not degenerate, the minimum singular value $\sigma(\boldsymbol{M})_{\min}$ will always be positive and the condition number of $\boldsymbol{M}$ is bounded. Therefore, as long as the mesh is of reasonably good quality, both the function value and gradient estimation is $\mathcal{C}^0$ across the boundary. □

Proposition *The function η_i in (32) is locally diminishing for $\forall i \in \mathcal{N}_{a,r}^1$.*

Proof Formally, we need to prove that for any $i \in \mathcal{N}_{a,r}^1$, when $\boldsymbol{x}$ is crossing the edge of a triangle and $||\boldsymbol{x}^n - \boldsymbol{x}^o|| = \mathcal{O}(\epsilon)$, the smoothing function $\eta_i = \mathcal{O}(\epsilon)$.

Denote the edge that the particle is crossing as e and the portion of $||\boldsymbol{x}^n - \boldsymbol{x}^o||$ in the new/old cell as $L^{n,o}$. Trivially,

$$
\begin{aligned}
L^{n,o} &\le L^n + L^o \\
&= ||\boldsymbol{x}^n - \boldsymbol{x}^o|| \\
&= \mathcal{O}(\epsilon).
\end{aligned} \tag{43}
$$

Then, let the far-away node not on the edge but in the new/old cell be i_{far} (i.e., $i_{\text{far}}^{n,o} \notin e \wedge i_{\text{far}}^{n,o} \in \mathcal{N}_{o,n}^0$) and the height from a node i to an edge e be $H(i, e)$. Since the height is orthogonal to the edge, we have $H(\boldsymbol{x}^{n,o}, e) \le L^{n,o} = \mathcal{O}(\epsilon)$. Consider the barycentric coordinate contributed by the far-away node, in the new/old cell respectively, for $\boldsymbol{x}$:

$$
\begin{aligned}
B_{i_{\text{far}}}^{n,o} &= \frac{H(\boldsymbol{x}^{n,o}, e) \cdot ||e||}{H(i_{\text{far}}^{n,o}, e) \cdot ||e||} \\
&= \frac{H(\boldsymbol{x}^{n,o}, e)}{H(i_{\text{far}}^{n,o}, e)} \\
&= \mathcal{O}\left(\frac{\epsilon}{H(i_{\text{far}}^{n,o}, e)}\right) \\
&= \mathcal{O}(\epsilon).
\end{aligned} \tag{44}
$$

Finally, if a node is added/removed during the particle crossing (i.e., $i \in \mathcal{N}_{a,r}^1$), this means that i is only connected to the far-away nodes $i_{\text{far}}^{n,o}$ but not to the edge e; i.e., $A_{i,i_{\text{far}}^{n,o}} = 1, \forall i \in \mathcal{N}_{a,r}^1$, otherwise $A_{i,n} = 0, \forall n \in e \wedge \forall i \in \mathcal{N}_{a,r}^1$. Hence,

$$
\begin{aligned}
\eta &= \sum_{n\in\mathcal{N}^0} B_n A_{i,n} \\
&= B_{i_{\text{far}}}^{n,o} A_{i,i_{\text{far}}^{n,o}} + \sum_{n\in e} B_n A_{i,n} \\
&= B_{i_{\text{far}}}^{n,o} \cdot 1 + \sum_{n\in e} B_n \cdot 0 \\
&= B_{i_{\text{far}}}^{n,o} \\
&= \mathcal{O}(\epsilon), \quad \forall i \in \mathcal{N}_{a,r}^1.
\end{aligned} \tag{45}
$$

This concludes the proof. □

C Settings and analytical solutions for the verification experiments of the continuous reconstructions

This section presents the detailed setup and analytical solutions for the 1D verification experiments on a uniform 1D mesh in Sect. 2.3.5. The kernel value is denoted as f and the gradient estimation is denoted as g, respectively. For the uniform 1D mesh, each cell has a length of 1, and the unit support length for the B-spline used for the sample weights is also 1. The analytical solution for the uniform 1D mesh, obtained using `Mathematica 2023`, is as follows:

$$f = \begin{cases} \frac{0.25(0.5-x)^2 x}{x^5-6x^4+13.5x^3-13.75x^2+6.3125x-0.5625}, & 0.5 < x \le 1 \\ \frac{-x^6+5x^5-9.5x^4+8.5x^3-3.3125x^2+0.3125x+0.0625}{-x^5+4x^4-5.5x^3+3.25x^2-1.3125x+1.0625}, & 1 < x \le 1.5 \\ \frac{x^6-12x^5+58.5x^4-148.25x^3+205.563x^2-147.063x+42.25}{x^5-11x^4+47.5x^3-100.25x^2+103.313x-41.125}, & 1.5 < x \le 2 \\ \frac{x^6-12x^5+58.5x^4-147.75x^3+202.563x^2-141.438x+39}{-x^5+9x^4-31.5x^3+53.75x^2-45.3125x+16.125}, & 2 < x \le 2.5 \\ \frac{x^6-19x^5+149.5x^4-623.5x^3+1453.31x^2-1794.19x+915.687}{-x^5+16x^4-101.5x^3+318.75x^2-495.313x+304.188}, & 2.5 < x \le 3 \\ \frac{-0.25x^3+2.75x^2-10.0625x+12.25}{-x^5+14x^4-77.5x^3+212.25x^2-288.313x+156.688}, & 3 < x \le 3.5 \\ 0, & \text{Otherwise} \end{cases} \tag{46}$$

$$g = \begin{cases} -\frac{1(0.5-x)^2 x(x-2)}{x^5-6x^4+13.5x^3-13.75x^2+6.3125x-0.5625}, & 0.5 < x \le 1 \\ \frac{-x^5+5x^4-10.5x^3+11.5x^2-5.8125x+1.0625}{-x^5+4x^4-5.5x^3+3.25x^2-1.3125x+1.0625}, & 1 < x \le 1.5 \\ \frac{-x^5+9x^4-33.5x^3+65.75x^2-67.3125x+27.625}{-x^5+11x^4-47.5x^3+100.25x^2-103.313x+41.125}, & 1.5 < x \le 2 \\ \frac{-x^5+11x^4-49.5x^3+112.25x^2-125.313x+53.625}{x^5-9x^4+31.5x^3-53.75x^2+45.3125x-16.125}, & 2 < x \le 2.5 \\ \frac{-x^5+15x^4-90.5x^3+274.5x^2-417.813x+254.188}{x^5-16x^4+101.5x^3-318.75x^2+495.313x-304.188}, & 2.5 < x \le 3 \\ \frac{x^4-13x^3+62.25x^2-129.5x+98}{-x^5+14x^4-77.5x^3+212.25x^2-288.313x+156.688}, & 3 < x \le 3.5 \\ 0, & \text{Otherwise} \end{cases} \tag{47}$$

D Conservation of the linear and affine momentum when combined with affine particle-in-cell

Since UMLS-MPM by construction generates a kernel that is the partition of unity and conserves the linear basis [48], i.e.,

$$\sum_i w_{p,i}^n = 1,$$
$$\sum_i w_{p,i}^n \boldsymbol{x}_i^n = \boldsymbol{x}_p^n,$$
$$\sum_i w_{p,i}^n (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n) = 0,$$

then the system's total linear and angular momentum will be conserved when combined with APIC. A simple introduction to APIC is given here for the sake of completeness, while the detailed proof can found in the supplementary document of the original APIC paper [50].

In APIC, mass m_p, position $\boldsymbol{x}_p$, velocity $\boldsymbol{v}_p$, and an affine matrix $\boldsymbol{B}_p = \sum_i w_{p,i} v_i (\boldsymbol{x}_i - \boldsymbol{x}_p)^T$ are stored and tracked on particles. Then,

Definition 1 The total linear momentum on grids is

$$\boldsymbol{p}_i^{tot} = \sum_i m_i \boldsymbol{v}_i.$$

Definition 2 The total linear momentum on particles is

$$\boldsymbol{p}_p^{tot} = \sum_p m_p \boldsymbol{v}_p.$$

Definition 3 The total angular momentum on grids is

$$\boldsymbol{I}_i^{tot} = \sum_i \boldsymbol{x}_i \times m_i \boldsymbol{v}_i.$$

Definition 4 The total angular momentum on particles is

$$\boldsymbol{I}_p^{tot} = \sum_p \boldsymbol{x}_p \times m_p \boldsymbol{v}_p + \sum_p m_p (\boldsymbol{B}_p)^T : \epsilon,$$

where ϵ is the Levi-Civita permutation tensor, and for any matrix $\boldsymbol{A}$, the contraction $\boldsymbol{A} : \epsilon = \sum_{\alpha\beta} A_{\alpha\beta}\epsilon_{\alpha\beta\gamma}$, which is usually used to transition from a cross product into the tensor product $\boldsymbol{u} \times \boldsymbol{v} = (\boldsymbol{v}\boldsymbol{u}^T)^T : \epsilon$. Also note that for the total angular momentum of the particles: 1) the grid node

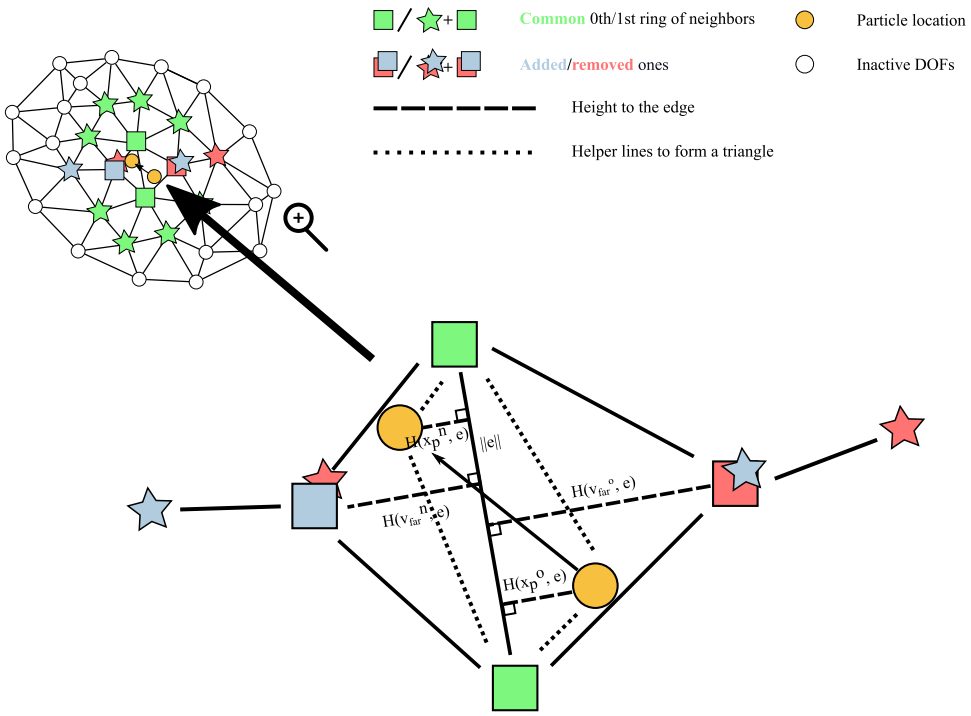


Fig. 25 A visual representation of the notations used to prove that η diminishes locally, as described in (44), for every vertex i within the first ring of neighbors, $\mathcal{N}^1_{a,r}$. The dashed line denotes the perpendicular height from a given position to the shared edge. Dotted lines are drawn to construct a triangle between the point $\boldsymbol{x}$ and the edge, facilitating the computation of the barycentric coordinates

locations can be perceived as the sample points of a rotating mass centered at the material particle location, and 2) the total angular momentum comprises both that of the center and that of the affine-rotation of the grids around the center.

APIC P2G is given by

$$\begin{aligned} m_i^n &= \sum_p w_{p,i}^n m_p \\ \boldsymbol{D}_p^n &= \sum_i w_{p,i}^n (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^T \\ m_i^n \boldsymbol{v}_i^n &= \sum_p w_{p,i}^n m_p (\boldsymbol{v}_p^n + \boldsymbol{B}_p^n (\boldsymbol{D}_p^n)^{-1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)) \end{aligned} \tag{48}$$

with G2P given by

$$\begin{aligned} \boldsymbol{v}_p^{n+1} &= \sum_i w_{p,i}^n \tilde{v}_i^{n+1} \\ \boldsymbol{B}_p^{n+1} &= \sum_i w_{p,i}^n \tilde{v}_i^{n+1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^T, \end{aligned} \tag{49}$$

where the superscript˜means the intermediate value after the update on grids but before the G2P process.

D.1 Numerical validation

A numerical validation as in [50] is also conducted here to verify these conservations. A square with a side length of $l = 0.2$ is discretized with 20×20 particles. The physical properties of the square are as follows: $E = 1 \times 10^4$ Pa, $\nu = 0.3$, and $\rho = 1.0\,\text{kg/m}^3$. Initially, the square is divided into two halves by a hypothetical vertical line through the middle. The left half is initialized with an upward velocity $\mathbf{v} = (1, 0)$ m/s, while the right half is initialized with a downward velocity $\mathbf{v} = (-1, 0)$ m/s. The experimental setup is illustrated in Fig. 26. The background mesh is generated using Delaunay triangulation with a target element size of 0.01 m in a $1 \times 1\,\text{m}^2$ box. The simulation is run for 1×10^6 time steps with a time step size of 1×10^{-5} s.

The proposed conservation is accurately illustrated in Fig. 27b–c, with only round-off errors on the order of 1×10^{-15} and 1×10^{-7} for the total linear and affine momentum of the system, respectively.

Funding this study was supported by Sony Corporation of America (Sony Faculty Innovation Award, 2023).

Fig. 26 Setup and snapshots of a rotating elastic square

(a) setup **(b)** step 2E4 **(c)** step 6E4 **(d)** step 2E5

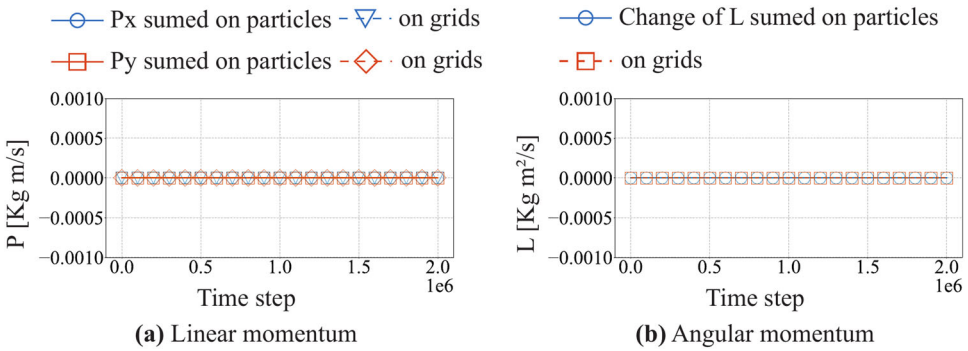


Fig. 27 Logs of **a** linear and **b** angular momentum of the rotating cube experiment after 10^6 time steps

References

1. Sulsky D, Zhou S, Schreyer HL (1995) Application of a particle-in-cell method to solid mechanics. Comput Phys Commun 87(1–2):236–252
2. Brackbill JU, Kothe DB, Ruppel HM (1988) FLIP: a low-dissipation, particle-in-cell method for fluid flow. Comput. Phys Commun 48(1):25–38
3. Harlow FH (1962) The particle-in-cell method for numerical solution of problems in fluid dynamics, Tech. rep., Los Alamos Scientific Lab., N. Mex
4. Ionescu I, Guilkey JE, Berzins M, Kirby RM, Weiss JA (2006) Simulation of soft tissue failure using the material point method. J Biomech Eng 128:917–924
5. Guilkey JE, Hoying JB, Weiss JA (2006) Computational modeling of multicellular constructs with the material point method. J Biomech 39(11):2074–2086
6. Guilkey J, Harman T, Banerjee B (2007) An Eulerian–Lagrangian approach for simulating explosions of energetic devices. Comput Struct 85(11–14):660–674
7. Ma S, Zhang X, Lian Y, Zhou X (2009) Simulation of high explosive explosion using adaptive material point method. Comput Model Eng Sci (CMES) 39(2):101
8. Homel MA, Brannon RM, Guilkey JE (2014) Simulation of shaped-charge jet penetration into drained and undrained sandstone using the material point method with new approaches for constitutive modeling. CIMNE, Barcelona, pp 676–687
9. Klár G, Gast T, Pradhana A, Fu C, Schroeder C, Jiang C, Teran J (2016) Drucker–Prager elastoplasticity for sand animation. ACM Trans Graph (TOG) 35(4):103
10. Tampubolon AP, Gast T, Klár G, Fu C, Teran J, Jiang C, Museth K (2017) Multi-species simulation of porous sand and water mixtures. ACM Trans Graph (TOG) 36(4):105
11. Stomakhin A, Schroeder C, Chai L, Teran J, Selle A (2013) A material point method for snow simulation. ACM Transa Graph (TOG) 32(4):102
12. Gaume J, Gast T, Teran J, Herwijnen A, Jiang C (2018) Dynamic anticrack propagation in snow. Nat Commun 9(1):1–10
13. Gaume J, Herwijnen A, Gast T, Teran J, Jiang C (2019) Investigating the release and flow of snow avalanches at the slope-scale using a unified model based on the material point method. Cold Reg Sci Technol 168:102847
14. Vaucorbeil A, Nguyen VP, Hutchinson CR (2020) A total-Lagrangian material point method for solid mechanics problems involving large deformations. Comput Methods Appl Mech Eng 360:112783
15. Vaucorbeil A, Nguyen VP (2021) Modelling contacts with a total Lagrangian material point method. Comput Methods Appl Mech Eng 373:113503
16. de Vaucorbeil A, Nguyen VP, Hutchinson CR, Barnett MR (2022) Total Lagrangian material point method simulation of the scratching of high purity coppers. Int J Solids Struct 239:111432
17. de Vaucorbeil A, Nguyen VP, Mandal TK (2022) Mesh objective simulations of large strain ductile fracture: a new nonlocal Johnson–Cook damage formulation for the total Lagrangian material point method. Comput Methods Appl Mech Eng 389:114388
18. Pretti G, Coombs WM, Augarde CE, Sims B, Puigvert MM, Gutiérrez JAR (2023) A conservation law consistent updated Lagrangian material point method for dynamic analysis. J Comput Phys 485:112075
19. Zhang X, Sze K, Ma S (2006) An explicit material point finite element method for hyper-velocity impact. Int J Numer Methods Eng 66(4):689–706
20. Huang P, Zhang X, Ma S, Huang X (2010) Contact algorithms for the material point method in impact and penetration simulation. Int J Numer Methods Eng 85(4):498–517
21. Cao Y, Chen Y, Li M, Yang Y, Zhang X, Aanjaneya M, Jiang C (2022) An efficient b-spline Lagrangian/Eulerian method for compressible flow, shock waves, and fracturing solids. ACM Trans Graph (TOG) 41(5):1–13
22. Chen Z, Hu W, Shen L, Xin X, Brannon R (2002) An evaluation of the MPM for simulating dynamic failure with damage diffusion. Eng Fract Mech 69(17):1873–1890
23. Zhang H, Wang K, Chen Z (2009) Material point method for dynamic analysis of saturated porous media under external contact/impact of solid bodies. Comput Methods Appl Mech Eng 198(17–20):1456–1472
24. Lian Y, Zhang X, Liu Y (2011) Coupling of finite element method with material point method by local multi-mesh contact method. Comput Methods Appl Mech Eng 200(47–48):3482–3494
25. Chen Z, Qiu X, Zhang X, Lian Y (2015) Improved coupling of finite element method with material point method based on a particle-to-surface contact algorithm. Comput Methods Appl Mech Eng 293:1–19
26. Homel MA, Herbold EB (2017) Field-gradient partitioning for fracture and frictional contact in the material point method. Int J Numer Methods Eng 109(7):1013–1044
27. Homel M, Herbold E (2018) Fracture and contact in the material point method: new approaches and applications. In: Advances in computational coupling and contact mechanics. World Scientific, pp 289–326
28. Cheon Y, Kim H (2018) An efficient contact algorithm for the interaction of material particles with finite elements. Comput Methods Appl Mech Eng 335:631–659
29. Guilkey J, Lander R, Bonnell L (2021) A hybrid penalty and grid based contact method for the material point method. Comput Methods Appl Mech Eng 379:113739

30. Nakamura K, Matsumura S, Mizutani T (2021) Particle-to-surface frictional contact algorithm for material point method using weighted least squares. Comput Geotech 134:104069
31. Fern J, Rohe A, Soga K, Alonso E (2019) The material point method for geotechnical engineering: a practical guide. CRC Press, Boca Raton
32. Więckowski Z (2004) The material point method in large strain engineering problems. Comput Methods Appl Mech Eng 193(39–41):4417–4438
33. Beuth L, Więckowski Z, Vermeer P (2011) Solution of quasi-static large-strain problems by the material point method. Int J Numer Analyt Methods Geomech 35(13):1451–1465
34. Jassim I, Stolle D, Vermeer P (2013) Two-phase dynamic analysis by material point method. Int J Numer Analyt Methods Geomech 37(15):2502–2522
35. Wang L, Coombs WM, Augarde CE, Cortis M, Brown MJ, Brennan AJ, Knappett JA, Davidson C, Richards D, White DJ et al (2021) An efficient and locking-free material point method for three-dimensional analysis with simplex elements. Int J Numer Methods Eng 122(15):3876–3899
36. Bardenhagen SG, Kober EM (2004) The generalized interpolation material point method. Comput Model Eng Sci 5(6):477–496
37. Charlton T, Coombs W, Augarde C (2017) igimp: an implicit generalised interpolation material point method for large deformations. Comput Struct 190:108–125
38. Zhang DZ, Ma X, Giguere PT (2011) Material point method enhanced by modified gradient of shape function. J Comput Phys 230(16):6379–6398
39. Steffen M, Kirby RM, Berzins M (2008) Analysis and reduction of quadrature errors in the Material Point Method (MPM). Int J Numer Methods Eng 76(6):922–948
40. Gan Y, Sun Z, Chen Z, Zhang X, Liu Y (2018) Enhancement of the material point method using b-spline basis functions. Int J Numer Methods Eng 113(3):411–431
41. Hu Y, Fang Y, Ge Z, Qu Z, Zhu Y, Pradhana A, Jiang C (2018) A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. ACM Trans Graph (TOG) 37(4):1–14
42. Tran Q, Wobbes E, Sołowski W T, Möller M, Vuik C (2019) Moving least squares reconstruction for b-spline material point method, In: International conference on the material point method for modelling soil–water-structure interaction, pp mpm2019–07
43. Koster P, Tielen R, Wobbes E, Möller M (2021) Extension of b-spline material point method for unstructured triangular grids using Powell–Sabin splines. Comput Particle Mech 8(2):273–288
44. Bonet J, Wood RD (2008) Nonlinear continuum mechanics for finite element analysis. Cambridge University Press
45. Zhang X, Chen Z, Liu Y (2016) The material point method: a continuum-based particle method for extreme loading cases. Academic Press
46. Ferziger JH, Perić M, Street RL (2019) Computational methods for fluid dynamics. Springer, Cham
47. Simo JC, Hughes TJ (2006) Computational inelasticity, vol 7. Springer Science & Business Media
48. Levin D (1998) The approximation power of moving least-squares. Math Comput 67(224):1517–1531
49. Andersen S, Andersen L (2010) Analysis of spatial interpolation in the material-point method. Comput Struct 88(7–8):506–518
50. Jiang C, Schroeder C, Selle A, Teran J, Stomakhin A (2015) The affine particle-in-cell method. ACM Trans Graph (TOG) 34(4):1–10
51. Jiang C, Schroeder C, Teran J (2017) An angular momentum conserving affine-particle-in-cell method. J Comput Phys 338:137–164
52. Wilson P, Wüchner R, Fernando D (2021) Distillation of the material point method cell crossing error leading to a novel quadrature-based c 0 remedy. Int J Numer Methods Eng 122(6):1513–1537
53. Jiang C, Schroeder C, Teran J, Stomakhin A, Selle A (2016) The material point method for simulating continuum materials. In: ACM SIGGRAPH 2016 courses. ACM, p 24
54. Donnelly D, Rogers E (2005) Symplectic integrators: an introduction. Am J Phys 73(10):938–945
55. Tran Q-A, Sołowski W (2019) Temporal and null-space filter for the material point method. Int J Numer Methods Eng 120(3):328–360
56. Zhao Y, Jiang C, Choo J (2023) Circumventing volumetric locking in explicit material point methods: a simple, efficient, and general approach. Int J Numer Methods Eng 124(23):5334–5355
57. Zhao Y, Choo J, Jiang Y, Li L (2023) Coupled material point and level set methods for simulating soils interacting with rigid objects with complex geometry. Comput Geotech 163:105708
58. Zhang L, Gerstenberger A, Wang X, Liu WK (2004) Immersed finite element method. Comput Methods Appl Mech Eng 193(21–22):2051–2067
59. Liu WK, Liu Y, Farrell D, Zhang L, Wang XS, Fukui Y, Patankar N, Zhang Y, Bajaj C, Lee J et al (2006) Immersed finite element method and its applications to biological systems. Comput Methods Appl Mech Eng 195(13–16):1722–1749
60. Li M-J, Lian Y, Zhang X (2022) An immersed finite element material point (IFEMP) method for free surface fluid-structure interaction problems. Comput Methods Appl Mech Eng 393:114809
61. Hughes TJ (2012) The finite element method: linear static and dynamic finite element analysis. Courier Corporation
62. Baumgarten AS, Couchman BL, Kamrin K (2021) A coupled finite volume and material point method for two-phase simulation of liquid-sediment and gas-sediment flows. Comput Methods Appl Mech Eng 384:113940
63. Cook BK, Jensen RP (2002) Discrete element methods: numerical modeling of discontinua. American Society of Civil Engineers
64. Li L, Lian Y, Li M-J, Gao R, Gan Y (2024) A contact method for b-spline material point method with application in impact and penetration problems. Comput Mech 73:1351–1369
65. Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW (2020) Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409
66. Li J, Gao Y, Dai J, Li S, Hao A, Qin H (2023) MPMNet: A data-driven MPM framework for dynamic fluid-solid interaction. IEEE Trans Vis Comput Graph 30(8)
67. Cao Y, Chai M, Li M, Jiang C (2023) Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In: International conference on machine learning, pp 3541–3558
68. Gao R, Deo IK, Jaiman RK (2022) A finite element-inspired hypergraph neural network: application to fluid dynamics simulations. Available at SSRN 4462715
69. Li T, Zhou S, Chang X, Zhang L, Deng X (2023) Finite volume graph network (FVGN): predicting unsteady incompressible fluid dynamics with finite volume informed neural network. arXiv preprint arXiv:2309.10050