

# Scene-Level Heterogeneous Physics Simulation with 3D Gaussian Splats

## Supplementary Material

### Overview

In this supplementary material, we provide additional technical details and results. Section [S1](#) presents more simulation cases; Section [S2](#) details the particle initialization process; Section [S3](#) derives the physical consistency of Gaussian covariance recoupling; Section [S4](#) provides the performance analysis; Section [S5](#) describes our physics solvers, and Section [S6](#) explains the coupling mechanisms.

### Contents

<a href="#">S1.</a>	More Simulation Cases .....
<a href="#">S2.</a>	Particles Initialization .....
<a href="#">S3.</a>	Physical Consistency of Gaussian Covariance Recoupling .....
<a href="#">S4.</a>	Time Complexity and Performance Analysis .....
<a href="#">S5.</a>	Physics Simulation Solvers .....
<a href="#">S6.</a>	The Coupling Mechanism of Unified Simulation Kernel .....

## S1. More Simulation Cases

Due to the dynamic nature of physics-based simulation, static images cannot fully convey the temporal coherence and stability of our results. We provide a self-contained offline webpage in the supplementary material for better visualization. Please refer to `index.html` in the attached zip file to view comprehensive video demonstrations, including heterogeneous material assignments, granular-soft body interactions, and comparisons with baseline methods.

## S2. Particles Initialization

### S2.1. Particles Filling for Gaussian Splatting

To enable the physical simulation of 3D Gaussian Splatting (3DGS) assets, which are inherently surface-focused representations, we employ a robust internal particle filling mechanism to generate the necessary volumetric data. Following prior methodologies, we first define the continuous opacity field  $d(x)$  derived from the static GS representation  $\mathcal{G}_{static}$ :

$$d(x) = \sum_{g \in \mathcal{G}_{static}} \sigma_g \exp \left( -\frac{1}{2} (x - \mathbf{k}_g)^T \Sigma_g^{-1} (x - \mathbf{k}_g) \right)$$

This continuous field is discretized onto a dense Eulerian grid with a cell size of  $\Delta x$ . We identify the interior region of the object using a user-defined opacity threshold  $\sigma_{th}$ . Specifically, a grid cell  $i$  is considered internal if its center opacity  $d(\mathbf{x}_i) > \sigma_{th}$ .

Once the internal volume is identified, we generate the physical particle set  $\mathcal{X}_{GS}$  by randomly sampling positions  $\mathbf{x}_j$  within the designated internal grid cells. We employ a Poisson Disk Sampling strategy to ensure a quasi-uniform distribution of particles, preventing clustering and enhancing simulation stability. The minimum sampling distance is set to  $r_{min} = 0.5\Delta x$ .

Crucially, to address potential density disparities during multi-physics coupling, we implement a density homogenization step. If the native particle resolution of the 3DGS asset is significantly lower than that of interacting external solvers (e.g., high-resolution fluids), numerical instability or boundary leakage may occur. In such cases, we inject auxiliary ghost particles into the internal volume to match the target simulation density. These ghost particles participate fully in the physical time integration to enforce robust contact constraints but are explicitly excluded from the rendering pipeline.

### S2.2. Physics Property Allocation

Each sampled physical particle  $p_j \in \mathcal{X}_{GS}$ , including the ghost particles generated for stability, must be assigned kinematic and constitutive properties for the Material Point Method (MPM) simulation.

- **Initial Position and Velocity:** The initial position  $\mathbf{x}_j^0$  is the sampled position. The initial velocity  $\mathbf{v}_j^0$  is set to zero or inherited from a kinematic input.
- **Mass and Volume:** We assume the object has a constant, user-specified material density  $\rho_{mat}$ . The initial volume  $V_j^0$  of each particle is determined by the total volume of the object's occupied space  $V_{total}$  divided by the total number of particles  $N_{total}$ :  $V_j^0 = V_{total}/N_{total}$ . The mass is then calculated as  $m_j = \rho_{mat} V_j^0$ .  $V_{total}$  is approximated by summing the volume of all internal grid cells identified in Section S2.
- **Initial Deformation Gradient:** The initial elastic deformation gradient is set to the identity matrix:  $\mathbf{F}_j^{E,0} = \mathbf{I}$ .
- **Visual Entity Recoupling:** For subsequent visual reconstruction, we store the particle's corresponding rest-state Gaussian covariance  $\Sigma_{rest,j}$  in the Visual Asset Cache.  $\Sigma_{rest,j}$  is either initialized from the nearest existing GS kernel or set as an isotropic sphere  $\text{diag}(r_p^2, r_p^2, r_p^2)$  where  $r_p$  is the particle radius inferred from  $V_j^0$ . For ghost particles, this step is skipped to ensure they remain invisible.

## S3. Physical Consistency of Gaussian Covariance Recoupling

The visual recoupling of the 3D Gaussian covariance matrix  $\Sigma$  is derived from the principles of continuum mechanics, specifically how a local affine transformation affects a Gaussian function.

A 3D Gaussian kernel  $G(\mathbf{X})$  in the material space  $\Omega^0$  is defined by its center  $\mathbf{X}_p$  and covariance matrix  $A_p$  (denoted as  $\Sigma_{rest}$  in our notation):

$$G(\mathbf{X}) = e^{-\frac{1}{2}(\mathbf{X}-\mathbf{X}_p)^T A_p^{-1}(\mathbf{X}-\mathbf{X}_p)}$$

Under a continuous deformation map  $\mathbf{x} = \phi(\mathbf{X}, t)$ , the deformed kernel  $G(\mathbf{x}, t)$  in the world space  $\Omega^t$  is:

$$G(\mathbf{x}, t) = e^{-\frac{1}{2}(\phi^{-1}(\mathbf{x}, t)-\mathbf{X}_p)^T A_p^{-1}(\phi^{-1}(\mathbf{x}, t)-\mathbf{X}_p)}$$

Following PhysGaussian, we assume the material point undergoes a local affine transformation characterized by the first-order Taylor expansion around  $\mathbf{X}_p$ :

$$\tilde{\phi}(\mathbf{X}, t) \approx \mathbf{x}_p + \mathbf{F}_p(\mathbf{X} - \mathbf{X}_p)$$

where  $\mathbf{F}_p = \nabla_{\mathbf{X}}\phi(\mathbf{X}, t)$  is the deformation gradient at point  $\mathbf{X}_p$ . By inverting this affine map and substituting it back into the exponent, the deformed kernel  $\tilde{G}(\mathbf{x}, t)$  remains a Gaussian distribution in the world space  $\Omega^t$  with a new center  $\mathbf{x}_p$  and a new covariance matrix  $\mathbf{a}_p$  (denoted as  $\Sigma'$  in our notation):

$$\tilde{G}(\mathbf{x}, t) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_p)^T (\mathbf{F}_p A_p \mathbf{F}_p^T)^{-1}(\mathbf{x}-\mathbf{x}_p)}$$

This derivation provides the physically grounded rule for updating the covariance matrix:

$$\Sigma' = \mathbf{F}\Sigma_{rest}\mathbf{F}^T$$

This ensures that the shape and orientation of the 3D Gaussian ellipsoids evolve consistently with the underlying continuum mechanics, preserving the non-rigid deformation (stretch and shear) induced by the simulation.

## S4. Time Complexity and Performance Analysis

**Rendering Workflow and Comparison.** A core contribution of our framework is the seamless integration of physically augmented 3D Gaussian Splatting (3DGS) assets into industrial-grade rendering ecosystems, specifically Unreal Engine 5 (UE5). Unlike prior methods that rely on standalone, Python-based rasterizers for visualization, our pipeline involves exporting the deformed states to UE5 to leverage advanced features such as global illumination and complex shadowing. Since this workflow includes manual operations (e.g., asset importation, scene composition, and sequence rendering in UE5), a direct runtime comparison with end-to-end, code-only rendering methods is neither feasible nor meaningful. Our focus is on achieving cinematic-quality offline rendering rather than real-time frame rates.

**Simulation Costs.** The computational cost of the physical simulation stage is inherently variable, governed largely by the user-defined simulation precision (grid resolution) and the complexity of the multi-physics coupling (e.g., particle density required for stability). However, the simulation remains efficient for offline content creation. For all five main demonstration scenarios presented in the paper, the physical simulation process was completed in under 10 minutes per sequence on a standard workstation.

**Summary.** We explicitly clarify that our method does not target real-time interactivity. Instead, the primary design goal is to unlock high-fidelity, heterogeneous interactions between captured 3DGS scenes and diverse virtual assets (meshes, particles)—scenarios that require complex solver coupling and are best served by an offline, high-quality rendering workflow.

## S5. Physics Simulation Solvers

### S5.1. Rigid Body Dynamics

The Rigid Body Solver is designed to simulate articulated mechanisms and multi-body systems using the Reduced Coordinate formulation (also known as Generalized Coordinates). Unlike maximal coordinate methods that treat every link as a free body constrained by algebraic equations, our solver parameterizes the system using joint an-

gles  $\mathbf{q} \in \mathbb{R}^{n_{dof}}$  and joint velocities  $\dot{\mathbf{q}} \in \mathbb{R}^{n_{dof}}$ . This approach naturally satisfies joint constraints and significantly improves computational efficiency for kinematic chains.

#### S5.1.1. Equations of Motion

The dynamics of the articulated system are governed by the standard manipulator equation:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_{ctrl} + \boldsymbol{\tau}_{ext} + \boldsymbol{\tau}_c \quad (\text{S1})$$

where:

- $\mathbf{M}(\mathbf{q})$  is the symmetric, positive-definite joint-space inertia matrix (Mass Matrix).
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$  represents the Coriolis and centrifugal forces.
- $\mathbf{g}(\mathbf{q})$  is the gravity vector generalized to joint space.
- $\boldsymbol{\tau}_{ctrl}$ ,  $\boldsymbol{\tau}_{ext}$ , and  $\boldsymbol{\tau}_c$  are the torques/forces from actuation, external disturbances, and contact constraints, respectively.

In our implementation, the bias terms  $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}\dot{\mathbf{q}} + \mathbf{g}$  are computed efficiently using recursive kinematics.

#### S5.1.2. Composite Rigid Body Algorithm (CRBA)

To compute the mass matrix  $\mathbf{M}(\mathbf{q})$ , we employ the Composite Rigid Body Algorithm (CRBA). The algorithm operates in two passes. First, it computes the composite inertia  $I_i^c$  for each link  $i$ , which represents the inertia of the subtree rooted at  $i$  as if all joints in the subtree were locked:

$$I_i^c = I_i + \sum_{j \in \text{children}(i)} {}^i\mathbf{X}_j^* I_j^c {}^j\mathbf{X}_i \quad (\text{S2})$$

where  ${}^j\mathbf{X}_i$  is the spatial transform from link  $i$  to  $j$ . Second, the entries of  $\mathbf{M}$  are populated by projecting the composite inertias onto the joint motion subspaces. This typically achieves  $O(N^2)$  complexity but is highly optimized for parallel execution on GPUs.

#### S5.1.3. Implicit Integration and Linear Solver

To ensure stability under high stiffness (PD control) and damping, we implement a semi-implicit integration scheme (often referred to as "Implicit Fast"). We explicitly modify the mass matrix to include the contributions of damping  $D$  and stiffness  $K$  matrices:

$$\tilde{\mathbf{M}} = \mathbf{M} + D\Delta t + K\Delta t^2 \quad (\text{S3})$$

The discrete-time linear system  $\tilde{\mathbf{M}}\Delta\dot{\mathbf{q}} = \tilde{\boldsymbol{\tau}}\Delta t$  is then solved to find the change in velocity. We utilize a dense  $LDL^T$  Cholesky decomposition to factorize  $\tilde{\mathbf{M}}$ :

$$\tilde{\mathbf{M}} = \mathbf{L}D_{diag}\mathbf{L}^T \quad (\text{S4})$$

Forward and backward substitution steps are then performed to solve for accelerations  $\ddot{\mathbf{q}}$ . This method provides robust behavior for stiff robotic mechanisms without requiring excessively small time steps.

#### S5.1.4. Kinematics and Constraints

Forward kinematics are computed recursively from the root to leaves, updating the Cartesian position  $\mathbf{x}$  and orientation  $\mathbf{R}$  of each link. Constraint forces  $\boldsymbol{\tau}_c$  arising from collisions and joint limits are solved using an impulse-based approach or Projected Gauss-Seidel (PGS) solver, integrated tightly with the forward dynamics loop.

#### S5.2. Material Point Method (MPM)

The Material Point Method (MPM) solver in our unified kernel serves as the primary engine for simulating continuum materials, including elastoplastic solids, granular media, and fluids. We implement the Moving Least Squares MPM (MLS-MPM) formulation combined with the Affine Particle-in-Cell (APIC) transfer scheme to ensure energy conservation and reduce numerical dissipation.

##### S5.2.1. Discretization and State Variables

The continuum body is discretized into a set of Lagrangian particles  $p$ , each carrying mass  $m_p$ , position  $\mathbf{x}_p$ , velocity  $\mathbf{v}_p$ , deformation gradient  $\mathbf{F}_p$ , and an affine velocity field matrix  $\mathbf{C}_p$ . A background Eulerian Cartesian grid with cell size  $\Delta x$  is employed as a scratchpad for computing spatial gradients and solving the equations of motion.

##### S5.2.2. Particle-to-Grid Transfer (P2G)

At the beginning of each time substep  $\Delta t$ , mass and momentum are transferred from particles to grid nodes  $i$  using quadratic B-spline shape functions  $N_i(\mathbf{x}_p)$ . Under the APIC formulation, the momentum transfer includes the affine velocity contribution to preserve angular momentum:

$$m_i = \sum_p N_i(\mathbf{x}_p) m_p \quad (\text{S5})$$

$$(m\mathbf{v})_i = \sum_p N_i(\mathbf{x}_p) m_p [\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_i - \mathbf{x}_p)] \quad (\text{S6})$$

In our MLS-MPM implementation, the internal forces derived from the material constitutive model are applied directly during this transfer via the affine term. Let  $\Psi(\mathbf{F})$  be the elastic energy density. The first Piola-Kirchhoff stress  $\mathbf{P}(\mathbf{F}) = \partial\Psi/\partial\mathbf{F}$  is computed and fused into the momentum update to avoid explicit grid force accumulation:

$$(\mathbf{v}_i)^{new} = \frac{(m\mathbf{v})_i}{m_i} + \Delta t \mathbf{g} - \frac{\Delta t}{m_i} \sum_p V_p \mathbf{P}(\mathbf{F}_p)^T \nabla N_i(\mathbf{x}_p) \quad (\text{S7})$$

where  $V_p$  is the particle volume and  $\mathbf{g}$  is the gravity vector.

##### S5.2.3. Constitutive Model and Plasticity

To handle finite deformations and plasticity, we adopt a Singular Value Decomposition (SVD) based return-mapping

algorithm. The trial deformation gradient is updated via the velocity gradient:

$$\mathbf{F}_p^{trial} = (\mathbf{I} + \Delta t \mathbf{C}_p) \mathbf{F}_p^n \quad (\text{S8})$$

We decompose  $\mathbf{F}_p^{trial} = \mathbf{U}_p \boldsymbol{\Sigma}_p \mathbf{V}_p^T$ . Plasticity is applied by clamping the singular values  $\boldsymbol{\Sigma}_p$  based on the specific material model (e.g., Drucker-Prager for sand, Von Mises for metal, or Fixed Corotated for elastic tissues), yielding the projected deformation gradient  $\mathbf{F}_p^{n+1}$ .

##### S5.2.4. Grid-to-Particle Transfer (G2P)

After solving the grid velocities, the updated states are transferred back to the particles. The particle velocity and the affine velocity field are updated as follows:

$$\mathbf{v}_p^{n+1} = \sum_i N_i(\mathbf{x}_p) \mathbf{v}_i^{new} \quad (\text{S9})$$

$$\mathbf{C}_p^{n+1} = \frac{4}{\Delta x^2} \sum_i N_i(\mathbf{x}_p) \mathbf{v}_i^{new} (\mathbf{x}_i - \mathbf{x}_p)^T \quad (\text{S10})$$

Finally, particle positions are advected:  $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}$ .

#### S5.3. Position-Based Dynamics (PBD)

The Position-Based Dynamics (PBD) solver is employed to simulate deformable objects such as cloth, soft bodies, and Lagrangian fluids. Our implementation adheres to the Extended Position-Based Dynamics (XPBD) formulation, which introduces a compliance parameter to decouple material stiffness from time step size and iteration counts.

##### S5.3.1. Simulation Loop

The solver follows a predictor-corrector scheme. For each particle  $i$ , a tentative position  $\mathbf{x}_i^*$  is first predicted using explicit Euler integration based on external forces  $\mathbf{f}_{ext}$  (e.g., gravity):

$$\mathbf{v}_i^* = \mathbf{v}_i^n + \Delta t \mathbf{M}^{-1} \mathbf{f}_{ext}, \quad \mathbf{x}_i^* = \mathbf{x}_i^n + \Delta t \mathbf{v}_i^* \quad (\text{S11})$$

Subsequently, the solver iteratively resolves a set of geometric constraints  $C(\mathbf{x}) = 0$ . The position correction  $\Delta \mathbf{x}$  for a constraint involving a set of particles is derived by minimizing the constraint potential energy:

$$\Delta \lambda = \frac{-C(\mathbf{x}^*) - \tilde{\alpha} \lambda}{\sum_j w_j |\nabla_j C(\mathbf{x}^*)|^2 + \tilde{\alpha}} \quad (\text{S12})$$

$$\Delta \mathbf{x}_i = w_i \Delta \lambda \nabla_i C(\mathbf{x}^*) \quad (\text{S13})$$

where  $w_i = 1/m_i$  is the inverse mass, and  $\tilde{\alpha} = \alpha/\Delta t^2$  is the discrete compliance derived from the physical compliance  $\alpha$ . Finally, velocities are updated:  $\mathbf{v}_i^{n+1} = (\mathbf{x}_i^{n+1} - \mathbf{x}_i^n)/\Delta t$ .

### S5.3.2. Structural Constraints

We implement specific constraints for different material types:

- **Distance Constraint (Cloth/Links):** To model stretching resistance, we enforce the distance between two particles  $i$  and  $j$ :

$$C_{dist}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| - l_0 = 0 \quad (S14)$$

- **Bending Constraint (Cloth):** To simulate bending resistance in triangular meshes, we utilize the isometric bending constraint based on the dihedral angle  $\phi$  between adjacent triangle faces:

$$C_{bend}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \pi = 0 \quad (S15)$$

- **Volume Constraint (Soft Bodies):** For tetrahedral meshes, volume conservation is enforced on each element to model material compressibility:

$$C_{vol}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \frac{1}{6}(\mathbf{x}_{21} \times \mathbf{x}_{31}) \cdot \mathbf{x}_{41} - V_0 = 0 \quad (S16)$$

### S5.3.3. Position-Based Fluids (PBF)

For liquid simulation, we adopt the PBF framework which enforces constant density. The density  $\rho_i$  at particle  $i$  is estimated using the Poly6 kernel  $W_{poly6}$ :

$$\rho_i = \sum_j m_j W_{poly6}(\|\mathbf{x}_i - \mathbf{x}_j\|, h) \quad (S17)$$

The density constraint requires  $C_{density}(\mathbf{x}_1, \dots, \mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1 = 0$ . Gradients are computed using the Spiky kernel  $\nabla W_{spiky}$  to avoid clustering instability.

Viscosity is handled via XSPH (Artificial Viscosity), which smoothes the velocity field before the position update:

$$\mathbf{v}_i^{new} = \mathbf{v}_i + c \sum_j \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) W_{poly6}(\|\mathbf{x}_i - \mathbf{x}_j\|, h) \quad (S18)$$

where  $c$  is a tunable viscosity coefficient.

## S5.4. Smoothed Particle Hydrodynamics (SPH)

The SPH solver provides a fully Lagrangian approach for simulating fluid dynamics, capable of handling free-surface flows and complex topological changes. Our implementation supports both Weakly Compressible SPH (WCSPH) for efficiency in dynamic scenarios and Divergence-Free SPH (DFSPH) for enforcing strict incompressibility with larger time steps.

### S5.4.1. Discretization

Fluid quantities at a particle  $i$  are interpolated from its neighbors  $j$  using a smoothing kernel  $W_{ij} = W(\|\mathbf{x}_i -$

$\mathbf{x}_j\|, h)$  with support radius  $h$ . The density  $\rho_i$  is evaluated via standard summation:

$$\rho_i = \sum_j m_j W_{ij} \quad (S19)$$

The momentum equation governing the fluid motion is given by:

$$\frac{d\mathbf{v}_i}{dt} = \frac{\mathbf{F}_i^{pressure}}{m_i} + \frac{\mathbf{F}_i^{viscosity}}{m_i} + \frac{\mathbf{F}_i^{surface}}{m_i} + \mathbf{g} \quad (S20)$$

### S5.4.2. Pressure Solvers

We provide two distinct formulations to resolve the pressure gradient term  $\mathbf{F}_i^{pressure}$ :

**Weakly Compressible SPH (WCSPH)** In this formulation, pressure is explicitly computed from density deviations using the Tait equation of state, allowing for slight compressibility (typically  $< 1\%$ ):

$$p_i = \frac{k\rho_0}{\gamma} \left( \left( \frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right) \quad (S21)$$

where  $k$  is the stiffness constant and  $\gamma = 7$ . The resulting pressure force is symmetric to ensure momentum conservation:

$$\mathbf{F}_i^{pressure} = -m_i \sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \quad (S22)$$

**Divergence-Free SPH (DFSPH)** For scenarios requiring stiff incompressibility, we implement DFSPH, which enforces two constraints using iterative pressure Poisson solvers. First, a *divergence-free solver* modifies the velocities to ensure  $\nabla \cdot \mathbf{v} = 0$  before position integration:

$$\frac{D\rho}{Dt} = -\rho(\nabla \cdot \mathbf{v}) = 0 \implies \sum_j m_j (\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla W_{ij} = 0 \quad (S23)$$

Second, a *constant-density solver* corrects position updates to maintain  $\rho = \rho_0$  (or equivalently, corrects predicted density error). This predictor-corrector scheme significantly improves stability under large time steps compared to WCSPH.

### S5.4.3. Viscosity and Surface Tension

To simulate viscous fluids, we apply a Laplacian-based viscosity force that dampens relative velocities between neighboring particles:

$$\mathbf{F}_i^{viscosity} = \sum_j \frac{m_j}{\rho_j} \mu \frac{(\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + \epsilon h^2} \nabla W_{ij} \quad (S24)$$

where  $\mu$  is the dynamic viscosity coefficient. Additionally, surface tension is modeled as a pairwise cohesive force proportional to a tension coefficient  $\gamma_{tension}$ :

$$\mathbf{F}_i^{surface} = -\gamma_{tension} \sum_j m_j (\mathbf{x}_i - \mathbf{x}_j) W(\|\mathbf{x}_i - \mathbf{x}_j\|) \quad (\text{S25})$$

This term mimics molecular attraction, promoting the formation of droplets and minimizing surface area.

## S6. The Coupling Mechanism of Unified Simulation Kernel

The Unified Simulation Kernel manages heterogeneous interactions by abstracting all assets into particles or meshes and utilizing specialized, coupled solvers. To ensure physical plausibility and conservation laws across different integration schemes, the coupling logic is categorized into three distinct mechanisms based on the solver types involved: (1) Signed Distance Field (SDF) based impulse response for Rigid-MPM/SPH interactions, (2) Position-based penetration correction for Rigid-PBD interactions, and (3) Grid-mediated momentum exchange for MPM-SPH/PBD interactions.

### S6.1. SDF-based Impulse Response (Rigid-MPM and Rigid-SPH)

For interactions between rigid bodies and Eulerian-Lagrangian hybrid solvers (MPM) or purely Lagrangian solvers (SPH), a velocity-level impulse method is employed. This approach utilizes the high-resolution SDF of rigid geometries to modulate collision softness and friction.

For a particle  $p$  (either an MPM grid node or an SPH particle) located at  $\mathbf{x}_p$  with velocity  $\mathbf{v}_p$ , we first compute the signed distance  $d(\mathbf{x}_p)$  and the surface normal  $\mathbf{n}$  from the rigid body's geometry. A blending weight  $w$ , determined by a softness parameter  $\epsilon$ , allows for smooth contact handling:

$$w(d) = \min \left( \exp \left( -\frac{d}{\max(\eta, \epsilon)} \right), 1 \right) \quad (\text{S26})$$

where  $\eta$  is a small numerical stabilizer.

The relative velocity with respect to the rigid body velocity  $\mathbf{v}_r$  is defined as  $\mathbf{v}_{rel} = \mathbf{v}_p - \mathbf{v}_r$ . We decompose  $\mathbf{v}_{rel}$  into normal component  $v_n = \mathbf{v}_{rel} \cdot \mathbf{n}$  and tangential vector  $\mathbf{v}_t = \mathbf{v}_{rel} - v_n \mathbf{n}$ .

If  $v_n < 0$  (penetration velocity), the post-collision relative velocity is computed by applying restitution  $e$  and Coulomb friction  $\mu$ . The normal impulse response is  $\tilde{v}_n = -ev_n$ . The tangential response uses a friction cone constraint:

$$\tilde{\mathbf{v}}_t = \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|} \max(0, \|\mathbf{v}_t\| - \mu|v_n|) \quad (\text{S27})$$

The final updated velocity of the particle is a blend of the rigid body velocity and the new relative velocity, weighted

by  $w$ :

$$\mathbf{v}_p^{new} = \mathbf{v}_r + w(\tilde{\mathbf{v}}_t + \tilde{v}_n \mathbf{n}) + (1 - w)\mathbf{v}_{rel} \quad (\text{S28})$$

Momentum conservation is strictly enforced by applying the reverse impulse  $-\Delta \mathbf{P} / \Delta t$  as an external force to the rigid body solver.

### S6.2. Position-based Penetration Correction (Rigid-PBD)

Interactions between rigid bodies and Position-Based Dynamics (PBD) particles rely on geometric projection rather than velocity impulses. This method prioritizes resolving interpenetration immediately at the position level.

For a PBD particle with mass  $m_p$  and radius  $r$ , collision is detected when the signed distance  $d(\mathbf{x}_p) < r$ . The particle position is projected outward along the contact normal  $\mathbf{n}$  to resolve the overlap:

$$\mathbf{x}_p^{new} = \mathbf{x}_p^{old} + k(r - d(\mathbf{x}_p))\mathbf{n} \quad (\text{S29})$$

where  $k \in [0, 1]$  represents the stiffness coefficient. Unlike the impulse-based method, friction is explicitly neglected in this formulation to maintain stability in the position projection step.

The velocity is implicitly updated via the positional change:  $\mathbf{v}_p^{new} = (\mathbf{x}_p^{new} - \mathbf{x}_p^{old}) / \Delta t$ . The resulting momentum change  $\Delta \mathbf{P} = m_p(\mathbf{v}_p^{new} - \mathbf{v}_p^{old})$  is accumulated and applied as a reaction force on the coupled rigid body to satisfy Newton's third law.

### S6.3. Grid-mediated Momentum Exchange (MPM-SPH and MPM-PBD)

Coupling between the Material Point Method (MPM) and Lagrangian particles (SPH or PBD) is achieved via the background MPM Eulerian grid, which acts as a momentum exchange medium. This mechanism models a perfectly inelastic collision process within each time substep.

For a given MPM grid node  $g$ , we identify the set of neighboring external particles  $\mathcal{N}_g$  (SPH or PBD) within the local stencil. The algorithm forces a velocity synchronization where the grid velocity  $\mathbf{v}_g$  is overridden by the average velocity of the interacting particles:

$$\mathbf{v}_g^{sync} = \frac{1}{|\mathcal{N}_g|} \sum_{i \in \mathcal{N}_g} \mathbf{v}_{p_i} \quad (\text{S30})$$

This operation implies an instantaneous "sticking" condition. To conserve total momentum, the change in the grid's momentum  $\Delta \mathbf{P}_g = m_g(\mathbf{v}_g^{sync} - \mathbf{v}_g^{old})$  is calculated. This momentum difference is then distributed back to the external particles as a corrective feedback:

$$\mathbf{v}_{p_i}^{new} = \mathbf{v}_{p_i} - \frac{\Delta \mathbf{P}_g}{m_{p_i}} \quad (\text{S31})$$

This two-way transfer ensures that the MPM fluid effectively drags or is dragged by the immersed SPH/PBD particles, behaving similarly to a mixture model with high drag coefficients.