A Moving Least Square Reproducing Kernel Particle Method for Unified Multiphase Continuum Simulation

ANONYMOUS AUTHOR(S)
SUBMISSION ID: 233

In physically based-based animation, pure particle methods are popular due to their simple data structure, easy implementation, and convenient parallelization. As a pure particle-based method and using Galerkin discretization, the Moving Least Square Reproducing Kernel Method (MLSRK) was developed in engineering computation as a general numerical tool for solving PDEs. The basic idea of Moving Least Square (MLS) has also been used in computer graphics to estimate deformation gradient for deformable solids. Based on these previous studies, we propose a multiphase MLSRK framework that animates complex and coupled fluids and solids in a unified manner. Specifically, we use the Cauchy momentum equation and phase field model to uniformly capture the momentum balance and phase evolution/interaction in a multiphase system, and systematically formulate the MLSRK discretization to support general multiphase constitutive models. A series of animation examples are presented to demonstrate the performance of our new multiphase MLSRK framework, including hyperelastic, elastoplastic, viscous, fracturing and multiphase coupling behaviours etc.

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

Additional Key Words and Phrases: reproducing kernel particle method, physically-based animation, modelling and simulation, fluids and solids

1 INTRODUCTION

Multiphase materials with multi-physics interactions are common in our daily life, e.g. frying egg, making coffee and dipping bread, but these trivial phenomena present nontrivial technical challenges for computer simulation. A simulation framework that can conveniently work with multiple media and multi-physics coupling remains a major target in the research of physically-based animation. Thanks to the simplicity in concept, data structure and implementation, particle-based methods are widely used in computer graphics to animate various continua. With easy implementation, particles can be used alone in a simulation, such as the popular SPH (Smoothed Particle Hydrodynamics) method. Phase-field model used with SPH [Yang et al. 2017] can simulate various phenomenon like diffusion, reaction, dissolving, melting etc. Particles can also be combined with a background mesh/grid to improve interpolation accuracy, such as the PIC/MPM (Particle-in-Cell / Material Point Method) which has attracted growing attention in recent years. The MPM is well known for simulating snow [Stomakhin et al. 2013], and successful simulations of various materials and phase-change phenomena have also been achieved. The versatility of MPM comes from its capability of using nonlinear continuum models. But the inclusion of a background grid makes the system more complex, and this gives us the motivation to incorporate general continuum models into a pure particle-based approach.

Continuum constitutive models are widely used in MPM to animate various complex materials, but due to the higher accuracy requirement on derivative calculation it is not straightforward to support them in a pure particle-based simulation. Smoothed Particle Hydrodynamics, as its name indicates, was mainly used for fluid simulation historically. In the attempt of extending SPH to solids, many problems of the original SPH have been exposed, e.g. interpolation inconsistency and tensile instability. In the field of engineering computation, extensive efforts have been made to improve the original SPH scheme for better accuracy and stability, and some of these improvements have also been adopted and further developed by the graphics community. For example, linear consistent gradient estimation [Bonet and Lok 1999] was used for elastic and elastoplastic solid simulation [Gissler et al. 2020; Peer et al. 2018]. Using consistent interpolation and Galerkin discretization, the RKPM [Liu et al. 1995] is another well-known SPH improvement, which is widely used in engineering for modelling various physical processes

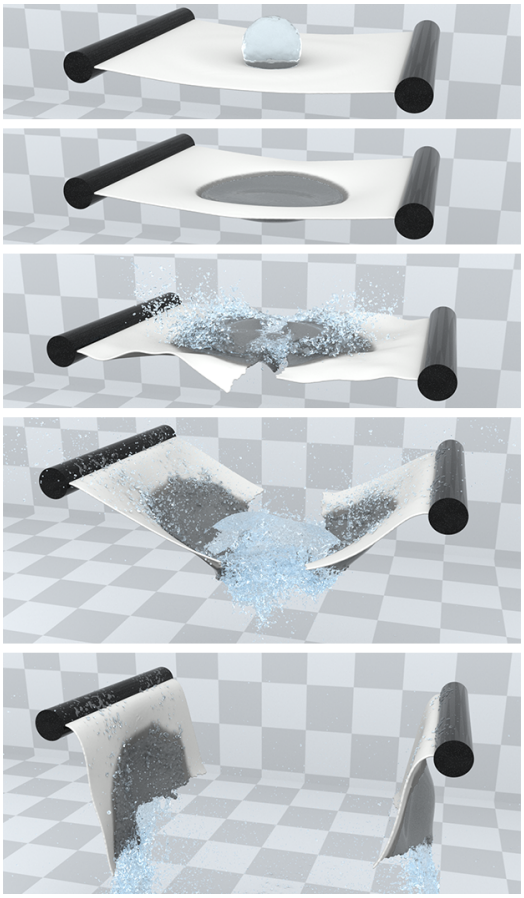

Fig. 1. **Tearing Wet Paper** A paper scroll is hit by a water ball, wetted and then teared apart into two pieces. Paper, water and their interaction are uniformly handled in our MLSRK framework. The fracture begins at the wet part as it is softened by water.

of materials. In the wider context, the RKPM can be viewed as a variation of the MLS (Moving Least Square) approach, and thus it is sometimes referred to as the MLSRK [Liu et al. 1997]. The general idea of point-based MLS has been used in computer graphics for solid simulation [Gerszewski et al. 2009; Jones et al. 2014; Müller et al. 2004], where it is mainly used to enhance the estimation of deformation gradient in deformable solids. However, compared with the successful applications in engineering fields, the full potential of MLS in terms of improving discretization accuracy for differentiation and integration appears to be less explored. The uniform discretization for partial differential equations (PDEs) in MLSRK makes it particularly suitable for unified simulation of multiphase materials with multi-physics interactions.

Based on these existing studies, we propose a quasi-linear MLSRK framework for unified simulation of multiphase materials and phase-change phenomena, which is accurate, robust and purely based on particles. The proposed framework handles both solids and fluids and supports such complex material behaviors as elastoplasticity, fracture and multiphase phenomena like diffusion, dissolution. Using only particle data structure, the new multiphase MLSRK framework is also conceptually simpler than hybrid methods for implementation. Our main technical contributions include:

- A pure particle-based approach is established for unified simulation of multiphase continua simulation. Different kinds of fluids and solids and their interactions are uniformly handled, including hyperelasticity, viscosity, elatoplasticity etc.
- Phase-field model is integrated into our MLSRK framework by discretizing Cahn-Hilliard equation, which enables simulation of multiphase phenomenon like diffusion, dissolving, reaction etc. Material volume change in phase evolution is handled by adding a phase related factor in deformation decomposition.

2 RELATED WORK

As a pure particle-based method, SPH was first created to solve astrophysics problems [Gingold and Monaghan 1977], and later used for simulate hydrodynamics problems. In computer graphics, SPH was used to simulate water with splashes [Müller et al. 2003], and it quickly became popular for fluid animation due to its simplicity in surface tracking and splashes handling. Becker and Teschner [2007] used the weakly compressible state equation to approximate incompressibility of fluid. Adams et al. [2007] used adaptive sampling to capture more details with less computational cost. To improve performance for animating incompressible flow, pressure solving techniques were developed. Solenthaler and Pajarola [2009] and Ihmsen et al. [2013] solved the pressure field to make the estimated density a constant. The Position-based Fluid (PBF) method was proposed in [Macklin and Müller 2013] for incompressible flow by using SPH density estimation with Position-based Dynamics (PBD), and it was later combined with PBD for a unified simulation framework in [Macklin et al. 2014]. Bender and Koschier [2015] solved the incompressibility condition by combining a constant density solver and a divergence free solver, which allowed the use of larger time steps. More recently, Reinhardt et al. [2019] used a modified Shepard interpolation to reduce noise in SPH fluid simulation. The SPH method can work for other types of flow. Alduán and Otaduy [2011] used SPH for simulating granular flow. Simulation of highly viscous fluids in SPH is also achieved in [Peer et al. 2015], which was later improved to correctly handle rotational motion using vorticity diffusion by Peer and Teschner [2016]. Weiler et al. [2018] used a Galilean invariant formulation of Laplacian operator to obtain physically consistent results of viscous fluids.

Linear inconsistency of the standard SPH interpolation can cause artifacts in rotational motion. This drawback becomes fatal in solid simulation thus correction is required to obtain good simulation result. There are already some successful solutions used in computer graphics. Becker et al. [2009] generalized shape matching [Müller et al. 2005] to fit rotation on each particle, and corotational formulation was used to be rotational invariant. Peer et al. [2018] used corrected SPH gradient formulation and coupled the simulator with SPH liquids. Based on this gradient correction, Gissler et al. [2020] further improved the SPH scheme to support elastoplastic materials.

Researchers in engineering fields have been developing more general meshfree methods that can be used to solve a wide range of PDE systems. The Reproducing Kernel Particle Method (RKPM) developed by Liu et al. [1995] is a successful meshfree method, which has been widely tested in engineering problems for modeling various mechanical processes of practical materials, including non-linear deforming solids [Chen et al. 1996], fragment-impact problems [Guan et al. 2009], air-blast–structure interaction [Bazilevs et al. 2017; Moutsanidis et al. 2018]. The RKPM can be viewed as a variation of the MLS approach, whose idea has been used in graphics Müller et al. [2004] for estimating deformation gradient of elastoplastic and melting solids. Pauly et al. [2005] modified weight of MLS near the fracture surface to handle discontinuity of fracturing solids, and it was later combined with SPH fluids [Keiser et al. 2005]. Gerszewski et al. [2009] allowed dynamically changing topology by updating deformation gradient per step, instead of fitting deformation gradient from initial configuration. Jones et al. [2014] handles both elastic and plastic solids by globally fitting plastic deformation into a embedded space. Martin et al. [2010] used Generalized MLS (GMLS) to construct shape function and avoid singularity problem of moment matrix in degenerated configuration like sheet and rod, and used "elaston" as a general tool to perform quadrature for volumes, sheets and rods, achieving unified simulation of elastic solids of different dimension.

Hybrid methods use both moving particles and a fixed background mesh/grid for numerical interpolation, and they have good accuracy and stability for the regularity of grid. The PIC/FLIP was introduced into graphics by Zhu and Bridson [2005] for fluid and sand animation. The use of MPM in computer animation has been increasing steadily since Stomakhin et al. [2013] used it to simulate snow. The APIC transfer [Jiang et al. 2015] was proposed to preserve affine motion, removing unwanted damping in PIC. It was later extended to Poly-PIC [Fu et al. 2017] and finally unified as MLS-MPM [Hu et al. 2018]. Various materials were successfully handled by MPM, including snow [Stomakhin et al. 2013], sand [Klár et al. 2016], cloth [Jiang et al. 2017] and thin shell [Guo et al. 2018]. There have been some studies focusing on the performance improvement of MPM. Gao et al. [2017] developed an adaptive sampling technique which can give more detailed result. GPU optimized MPM simulator was developed

in [Gao et al. 2018b]. Hu et al. [2019] proposed a domain specific programming language to efficiently implement and execute MPM algorithms.

In order to animate complex scenes involving multiphase materials and their interactions, some multiphase simulation techniques have been developed. Multiphase mixture fluids were modeled by integrating the mixture model with SPH [Ren et al. 2014]. Yang et al. [2015] used Cahn-Hilliard equation to handle chemical reaction in multiphase phenomena. Yan et al. [2016] extended this framework for solids. Yang et al. [2017] extended the phase field model using Alan-Cahn equation to simulate phase-change processes related to temperature. The mixture model was also ported to MPM [Yan et al. 2018]. The MPM was augmented to handle phase-change in [Stomakhin et al. 2014]. Baking and cooking scenes were animated using MPM in [Ding et al. 2019]. Although the simulation of single type of materials is well studied, coupling of multiple materials is still a challenging task. As different simulation methods need to work together, the interaction between them must be explicitly handled. The coupling between rigid body and liquid was handled in [Akinci et al. 2012], and later improved in [Gissler et al. 2019] with better performance in strongly coupled scenes. Sand and water mixtures were simulated in [Tampubolon et al. 2017] with two species solved on two grids coupled by a linear drag force. Coupling between rigid body and viscous liquid was studied in [Takahashi and Lin 2019]. Gao et al. [2018a] simulated particle-laden flow. The coupling between hair and liquid was modelled in [Fei et al. 2017], cloth and liquid in [Fei et al. 2018], strands and shear-dependent fluid in [Fei et al. 2019].

3 MOVING LEAST SQUARE REPRODUCING KERNEL INTERPOLATION

The basic idea of MLS has already been used in graphics [Gerszewski et al. 2009; Jones et al. 2014; Müller et al. 2004] to improve the accuracy of deformation gradient estimation, but the general application in numerical interpolation remains less explored. Moving least square interpolation can guarantee interpolation consistency to any selected order. In this section, the formulation of MLSRK is briefly recapped in § 3.1 and § 3.2 following the derivation of [Liu et al. 1997].

3.1 MLSRK Interpolation in Continuous Space

Reproducing kernel interpolation can be obtained by applying the idea of MLS to continuous space. In a d-dimensional space, to approximate a function $u : R^d \rightarrow R$ in a neighborhood of a point $\mathbf{x}_0 \in R^d$, one can use a linear combination of a set of basis functions $h_\alpha : R^d \rightarrow R\,(\alpha = 1, \ldots, N_b)$ as follows:

$$\tilde{u}(\mathbf{x}; \mathbf{x}_0) = \sum_{\alpha=1}^{N_b} c_\alpha(\mathbf{x}_0) h_\alpha(\mathbf{x}) \tag{1}$$

where c_α denotes the combination coefficient. The basis functions h_α is often formed by polynomials. This can be written in matrix form as $\tilde{u}(\mathbf{x}; \mathbf{x}_0) = \mathbf{c}(\mathbf{x}_0)^T \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{c}(\mathbf{x}_0)$. Over the domain of interest Ω, the weighted square error functional is:

$$E(\tilde{u}; \mathbf{x}_0) = \int_\Omega \left[\tilde{u}(\mathbf{x}; \mathbf{x}_0) - u(\mathbf{x})\right]^2 \Phi(\mathbf{x} - \mathbf{x}_0) \mathrm{d}\mathbf{x} \tag{2}$$

where the weight function Φ is usually set similar to Gaussian kernel with a finite support radius.

Minimizing the error functional E with respect to the coefficient $\mathbf{c}$ gives the optimal approximation $\tilde{u}$ for the original function u. This leads to a quadratic optimization problem that can be solved by setting $\partial E / \partial \mathbf{c} = \mathbf{0}$. The resulting linear equation is

$$\mathbf{M}(\mathbf{x}_0)\mathbf{c}(\mathbf{x}_0) = \mathbf{r}(\mathbf{x}_0) \tag{3}$$

where $\mathbf{M}(\mathbf{x}_0) = \int_\Omega \mathbf{h}(\mathbf{x})\mathbf{h}(\mathbf{x})^T \Phi(\mathbf{x} - \mathbf{x}_0)\mathrm{d}\mathbf{x}$ is called moment matrix and $\mathbf{r}(\mathbf{x}_0) = \int_\Omega \mathbf{h}(\mathbf{x})\Phi(\mathbf{x} - \mathbf{x}_0)u(\mathbf{x})\mathbf{d}\mathbf{x}$. The linear problem in Eqn. (3) has the same dimension as the number of basis functions. As linear and quadratic basis are often used in 2D and 3D space, the dimension of the linear system is usually not more than 10, thus it can be easily solved. If the moment matrix is non-singular, a unique solution $\mathbf{c}$ can be obtained.

Using the optimal coefficients $\mathbf{c}(\mathbf{x}_0) = \mathbf{M}^{-1}(\mathbf{x}_0)\mathbf{r}(\mathbf{x}_0)$ to evaluate $\tilde{u}$ with $\mathbf{x}_0 = \mathbf{x}$ yields the reproducing kernel interpolated function $u^h(\mathbf{x}) = \tilde{u}(\mathbf{x}; \mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{c}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{M}^{-1}(\mathbf{x})\mathbf{r}(\mathbf{x})$. Substituting $\mathbf{r}$ into the expression, the reproducing kernel interpolation can be rewritten as:

$$\begin{aligned} u^h(\mathbf{x}) &= \int_\Omega \left[\mathbf{h}(\mathbf{x})^T \mathbf{M}^{-1}(\mathbf{x})\right] \mathbf{h}(\mathbf{x}')\Phi(\mathbf{x}' - \mathbf{x})u(\mathbf{x}')\mathrm{d}\mathbf{x}' \\ &= \int_\Omega \mathbf{b}(\mathbf{x})^T \mathbf{h}(\mathbf{x}')\Phi(\mathbf{x}' - \mathbf{x})u(\mathbf{x}')\mathrm{d}\mathbf{x}' \\ &= \int_\Omega C(\mathbf{x}', \mathbf{x})\Phi(\mathbf{x}' - \mathbf{x})u(\mathbf{x}')\mathrm{d}\mathbf{x}' \end{aligned} \tag{4}$$

where $\mathbf{b}(\mathbf{x})^T = \mathbf{h}(\mathbf{x})^T \mathbf{M}^{-1}(\mathbf{x})$ and $C(\mathbf{x}'; \mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{h}(\mathbf{x}')$. Comparing Eqn. (4) with the standard SPH interpolation $u^h(\mathbf{x}) = \int_\Omega \Phi(\mathbf{x}' - \mathbf{x})u(\mathbf{x})\mathrm{d}\mathbf{x}$, the only difference is the factor $C(\mathbf{x}', \mathbf{x})$, which corrects the inconsistency of standard SPH interpolation. In this sense, MLSRK can be viewed as an improvement to standard SPH.

If the original function happens to be a combination of basis functions, i.e. $u(\mathbf{x}) = \mathbf{c}_0^T \mathbf{h}(\mathbf{x})$, where $\mathbf{c}_0$ is the vector of the combination coefficients, the unique optimal solution of coefficient $\mathbf{c}$ is $\mathbf{c} = \mathbf{c}_0$, which makes $\tilde{u} = u$ and E is minimized to be 0. Consequently, $u^h = u$, i.e. the interpolated function is exactly the same as the original function, or the function is reproduced. This property guaranties completeness of the interpolation.

3.2 Interpolation with Particles and Polynomial Basis

The reproducing kernel interpolation described above can be readily applied to a particle system. Consider a domain sampled by a set of particles, where the i-th particle has its position $\mathbf{x}_i$, volume V_i and some other general quantity u_i. Using these particles as the quadrature points to approximate the integrations in Eqns. (3) and (4), the quantity u can be interpolated as

$$u^h(\mathbf{x}) = \sum_i \mathbf{b}(\mathbf{x})^T \mathbf{h}(\mathbf{x}_i)\Phi(\mathbf{x}_i - \mathbf{x})V_i u_i \tag{5}$$

where $\mathbf{b}(\mathbf{x})^T = \mathbf{h}(\mathbf{x})^T \mathbf{M}^{-1}(\mathbf{x})$ and $\mathbf{M}(\mathbf{x}) = \sum_i \mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_i)^T \Phi(\mathbf{x}_i - \mathbf{x})V_i$. Note that, the summation is for all particles theoretically, but due to the finite support of kernel function Φ, the actual computation may include only neighborhood particles. This rule applies to all other summations in this paper. The above interpolation can be

reorganized as:

$$u^h(\mathbf{x}) = \sum_i N_i(\mathbf{x}) u_i \tag{6}$$

where $N_i(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{h}(\mathbf{x}_i)\Phi(\mathbf{x}_i - \mathbf{x})V_i$ is the shape function corresponding to particle i.

Basis functions are usually selected to be polynomials, and some commonly used options are listed in Table 1. For the kernel Φ, cubic spline kernel is widely adopted for its smoothness:

$$\text{cubicspline}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & |x| < 1, \\ \frac{1}{6}(2 - |x|)^3 & 1 \le |x| < 2, \\ 0 & \text{else} \end{cases} \tag{7}$$

The kernel can be formed as Cartesian product of each coordinate, i.e. $\Phi(x, y) = \text{cubicspline}(x)\ \text{cubicspline}(y)$ for 2D, and $\Phi(x, y, z) = \text{cubicspline}(x)\ \text{cubicspline}(y)\ \text{cubicspline}(z)$ for 3D. Radial basis cubic spline kernel is also applicable, i.e. $\Phi(\mathbf{x}) = \text{cubicspline}(||\mathbf{x}||)$. Note that the normalization factor is not necessary in MLSRK, as a positive constant factor will not affect the result of minimizing E in Eqn. (2).

Table 1. Commonly used basis functions in MLSRK

Consistency	Dimension	Basis functions
Constant	any	$\mathbf{h} = [1]^T$
Linear	1D	$\mathbf{h} = [1\ x]^T$
	2D	$\mathbf{h} = [1\ x\ y]^T$
	3D	$\mathbf{h} = [1\ x\ y\ z]^T$
Quadratic	1D	$\mathbf{h} = [1\ x\ x^2]^T$
	2D	$\mathbf{h} = [1\ x\ y\ x^2\ y^2\ xy]^T$
	3D	$\mathbf{h} = [1\ x\ y\ z\ x^2\ y^2\ z^2\ xy\ yz\ zx]^T$

In practice, polynomial basis functions should be shifted to point $\mathbf{x}$ and scaled according to the kernel radius scale a for numerical stability. The shifted and scaled interpolation functions are:

$$\begin{aligned} N_i(\mathbf{x}) =& \mathbf{b}(\mathbf{x})^T \mathbf{h}(\frac{\mathbf{x}_i - \mathbf{x}}{a})\Phi(\frac{\mathbf{x}_i - \mathbf{x}}{a})V_i, \\ \mathbf{b}(\mathbf{x})^T =& \mathbf{h}(\mathbf{0})^T \mathbf{M}^{-1}(\mathbf{x}), \\ \mathbf{M}(\mathbf{x}) =& \sum_i \mathbf{h}(\frac{\mathbf{x}_i - \mathbf{x}}{a})\mathbf{h}(\frac{\mathbf{x}_i - \mathbf{x}}{a})^T \Phi(\frac{\mathbf{x}_i - \mathbf{x}}{a})V_i. \end{aligned} \tag{8}$$

Comparing with the MLS directly applied on points, an additional volume factor V_i present in the MLSRK formulation. The derivatives of above interpolation functions are required in later discussions and for convenience, they are summarized in Appendix A. The radius of kernel is selected to be proportional to the diameter of particles, and it should be set to include a sufficient number of neighborhood particles. In most examples, the radius of kernel support is selected to be around $2 \sim 3$ times the diameter of particles.

The consistency of MLSRK interpolation depends on the selection of basis functions. For constant basis, constant is reproduced, i.e. $\sum_i N_i(\mathbf{x}) = 1$, which leads to $\sum_i \nabla N_i(\mathbf{x}) = 0$. For linear basis, we additionally have $\sum_i N_i(\mathbf{x})(\mathbf{x}_i - \mathbf{x}) = 0$ and $\sum_i \nabla N_i(\mathbf{x}) \otimes (\mathbf{x}_i - \mathbf{x}) = \mathbf{I}$. Note that, if constant basis is used, the interpolation is exactly the same as Shepard interpolation. Generally, using higher order basis functions gives better accuracy in interpolation, but the computation cost also increases. We found that linear basis is accurate enough for animation purpose and is used in all our examples.

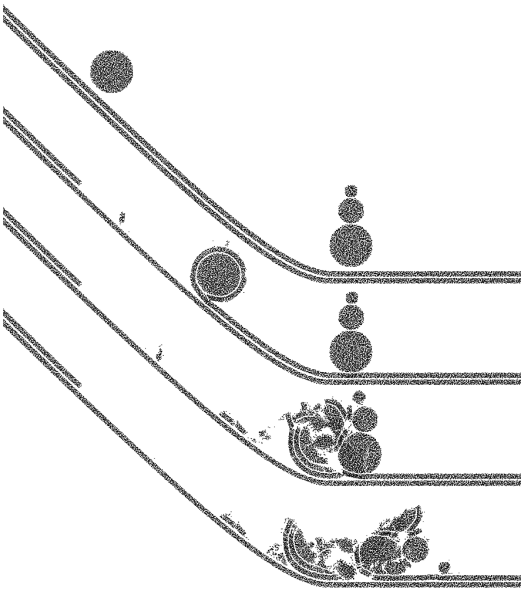

Fig. 2. **Rolling Snowball** A snow ball rolls down a slope covered with snow, and smashes into a stack of snow balls [Gissler et al. 2020; Stomakhin et al. 2013]. Snow on the slope can stick to the rolling ball, making it grow, and fractures occur when the falling ball hits the stacking balls at the foot of slope.

3.3 The relationship of SPH, MPM and MLSRK

SPH, MPM and MLSRK can all handle complex continuum models in the current state of the art. A demo snowball simulation using MLSRK is illustrated in Fig. 2, where similar visual effects as previous methods [Gissler et al. 2020; Stomakhin et al. 2013] are produced.

MLSRK can be viewed as an improved SPH scheme, which introduces a correction factor to remove the interpolation error caused by poor particle distribution. There are also other SPH improvements, and some of them have been explored by the graphics community. For example, corotated formulation [Becker et al. 2009] and linear consistent gradient formulation [Gissler et al. 2020; Peer et al. 2018] have been used to simulate elastic and elastoplastic solids. These SPH improvements share similar data and program structures, and information only flows among neighborhood particles in each iteration step. In addition, they all require solving a small matrix for each particle to improve the approximation accuracy. Nevertheless, MLSRK has some technical advantages that make it a more suitable foundation for establishing a unified simulation framework for multiphase materials. Specifically, by using Galerkin discretization, MLSRK provides a systematic way to discretize a wide range of PDEs. Compared to the collocation discretization used in other SPH schemes, it is much easier to uniformly discretize the various PDEs encountered in complex multiphase systems following the Galerkin approach (see § 4).

SPH and MLSRK use only particles for discretization, while MPM requires an additional background grid as a scratch pad for computation. MPM interpolation has a simpler form due to the regularity of background grid, but the additional grid data structure increases the complexity of theoretical analysis and implementation. Pure particle-based methods can easily obtain unbounded region adaptivity and as the particles only have information transfer within neighborhood particles, the program structure is simply two nested loops with a cached neighborhood list (Algorithm 1). To simulate unbounded domain, MPM usually requires the algorithm to be tightly coupled with sophisticated sparse grid structures like OpenVDB [Museth 2013] or SPGrid [Setaluri et al. 2014]. Implementing P2G and G2P transfer for MPM is not an easy job, especially when spatial data structure is used. To simplify the implementation of MPM, the Taichi programming language [Hu et al. 2019] with built-in support for sparse data structure is designed recently for programming with spatial sparsity.

Compared with pure particle-based methods, the information transfer between MPM particles and MPM grids can cause information loss. The standard PIC P2G velocity transfer is linear inconsistent, and causes error especially near boundaries of sampled domain with unwanted damping effects. This problem is addressed in APIC [Jiang et al. 2015] and Poly-PIC [Fu et al. 2017] by tracking higher order neighborhood information on particles, and these methods are later unified as MLS-MPM [Hu et al. 2018]. In particular, Hu et al. [2018] used the same shape functions as RKPM/MLSRK. Benefiting from the regular background grid, the moment matrix is constant in MLS-MPM, which saves some computation effort. However, the information loss in P2G transfer is still not fully avoided by these improvements. Because particles usually have at least two times resolution of the grid, the quantities tracked on the grid are practically "blurred" compared to the information captured by particles. Consequently, the sub-grid particle motion cannot be captured in MPM simulation. Pure particle-based methods do not suffer from this issue, and therefore produces much better results in simulating multiphase separating processes (Fig. 3). Refining resolution does not help with this issue. This advantage makes pure particle-based methods more suitable than MPM for multiphase simulation.

As MLS interpolation and Galerkin discretization are used by both MLSRK and MLS-MPM, they share some similarity. Benefiting from the regularity of background grid, MLS-MPM always has constant moment matrix, while MLSRK has varying moment matrix to be computed. For MPM, the grid tracks degrees of freedom (DoFs), and particles serve as quadrature points. For MLSRK, particles are used to track DoFs and are also used as quadrature points. Knowing this difference, comparison can be made with the same number of particles or DoFs. For a fair comparison, we implemented an MLSRK simulator with Taichi programming language [Hu et al. 2019] and compared the performance with Taichi's MLS-MPM implementation. Using the same number of particles, MPM is faster than MLSRK but delivers less details, because the grid has significantly less DoFs. Using the same number of DoFs, MPM produces similar splashes like MLSRK, but runs slower than MLSRK. If more particles are used, MPM can provide better results in surface tracking.

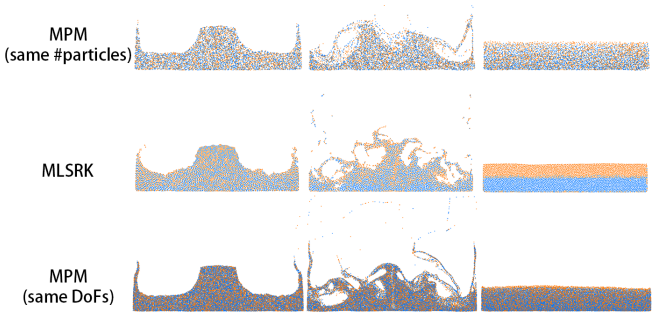

Fig. 3. **Mixture Dam-break** Dam-break of a two-phase liquid mixture with density of 10^4 and 10^3 is simulated with MLSRK and MPM. As MPM has less DoFs on the grid, it cannot capture sub-grid flow details, and fails to simulate the unmixing process. MLSRK successfully captures the phase separation behavior without the information bottleneck of grids. Using same DoFs as MLSRK, MPM can produce similar motion as MLSRK, but this does not help for simulating the unmixing process, because DoFs on grid are always significantly less than the particles.

4 MLSRK FORMULATION FOR MULTIPHASE CONTINUA

In this section, we describe how to discretize multiphase systems using MLSRK shape functions and Galerkin scheme, thereby animating multiple coupled media with a unified simulation framework that is based on particles only and easy to implement. In continuum-based simulation, governing equations are formulated as partial differential equations, which can be discretized using numerical schemes and then solved by computers to obtain an approximate solution. In SPH, PDEs are satisfied only at the particle location, and such discretization approach is known as the collocation method, with SPH particles serving as the collocation points. In FEM (Finite Element Method), PDEs are converted to equivalent weak form formulations, and then projected onto the approximate solution space using shape functions as trial functions, which is known as the Galerkin method. The Galerkin method is preferred for more general numerical discretization, as a wide range of PDEs can be easily discretized in a similar manner.

We follow the Galerkin approach to discretize the governing equations of multiphase systems with MLSRK interpolation. First, the PDE governing equations for multiphase continua including momentum conservation and phase evolution are described in § 4.1 and § 4.2, respectively. Next, the general constitutive models for multiphase continua are explained in § 4.3 together with the corresponding MLSRK discretization. Finally, § 4.4 describes a new regularization technique to cope with ill particle distribution and the overall MLSRK algorithm framework for multiphase continuum simulation.

4.1 Momentum Conservation of Multiphase Continua

Momentum conservation have different versions in fluid dynamics, e.g. Euler equation and Navier-Stokes equation. A more general momentum conservation equation for a continuum in Eulerian

viewpoint is Cauchy momentum equation:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \sigma + \rho \mathbf{g} \tag{9}$$

where ρ denotes mass density, $\mathbf{u}$ velocity, σ Cauchy stress and $\mathbf{g}$ body force per unit mass (gravitational acceleration in most cases). Using the Cauchy momentum equation, both fluids and solids can be handled in a unified manner. Its equivalent weak form formulation is:

$$\int_\Omega \left[\rho(D\mathbf{u}/Dt - \mathbf{g})v + \nabla v \cdot \sigma\right] \mathrm{d}V - \oint_{\partial\Omega} n \cdot \sigma v \mathrm{d}S = 0 \tag{10}$$

where v denotes the trial function. For simplicity, only free boundary is considered, thus the second term is always zero. Using the previously defined shape function N_i as trial function, i.e. $v = N_i$, and performing nodal integration on particles yields

$$\sum_j \left[\rho_j(\dot{\mathbf{u}}_j^h - \mathbf{g})N_i(\mathbf{x}_j) + \nabla N_i(\mathbf{x}_j) \cdot \sigma_j\right] V_j = 0,$$

where $\dot{\mathbf{u}}_j^h = \sum_k \dot{\mathbf{u}}_k N_k(\mathbf{x}_j)$. This can be rewritten in matrix form $\mathbf{M}\dot{\mathbf{u}} = \mathbf{f}$:

$$\sum_j \left(\sum_k N_i(\mathbf{x}_k)N_j(\mathbf{x}_k)\rho_k V_k\right)(\dot{\mathbf{u}}_j - \mathbf{g}) = -\sum_j \nabla N_i(\mathbf{x}_j) \cdot \sigma_j V_j.$$

Using mass lumping, we can avoid solving the linear problem of mass matrix. While mass matrix is $\mathbf{M}_{ij} = \sum_k N_i(\mathbf{x}_k)N_j(\mathbf{x}_k)\rho_k V_k$, the lumped mass matrix is then $\tilde{\mathbf{M}}_{ij} = \delta_{ij}\sum_k \mathbf{M}_{ik} = \delta_{ij}\sum_l N_i(\mathbf{x}_l)\rho_l V_l$. The final discretized formulation is:

$$\dot{\mathbf{u}}_i = -\frac{\sum_j \nabla N_i(\mathbf{x}_j) \cdot \sigma_j V_j}{\sum_j N_i(\mathbf{x}_j)\rho_j V_j} + \mathbf{g}. \tag{11}$$

Direct nodal integration is inaccurate, and causes noise and instability in the motion. Voronoi diagrams were generated in [Chen et al. 2001] to improve the integration accuracy. However, such approach is not suitable for our case which involves frequently changing topology, as dynamically maintaining Voronoi diagrams in every time step will be very complicated. To address the instability issue, we borrow ideas from PIC/FLIP [Zhu and Bridson 2005] and XSPH [Monaghan 1992], the velocity of each particle is blended with the interpolated velocity:

$$\tilde{\mathbf{u}}_i = \alpha_b \mathbf{u}_i + (1 - \alpha_b)\mathbf{u}^h(\mathbf{x}_i) \tag{12}$$

where $\mathbf{u}^h(\mathbf{x}_i) = \sum_j N_j(\mathbf{x}_i)\mathbf{u}_j$ is the interpolated velocity at $\mathbf{x}_i$. This gives a similar effect as the artificial viscosity and helps stabilize the simulation. We tested a set of α_b values from 0 to 1 (see Fig. 4), and found $\alpha_b = 0.8 \sim 0.95$ delivers good stabilization effect in all our examples without influencing the simulation result too much. We expect to replace this with more sophisticated methods in the future, like APIC replacing FLIP [Jiang et al. 2015].

We assumed free boundary in the above derivation. But fixed boundary is also useful in animation, for which we adopt a velocity collision for all particles as described in [Stomakhin et al. 2013] to mimic the effect of external boundary force. Velocity fixed particles are also used for boundary of complex shape in some examples.

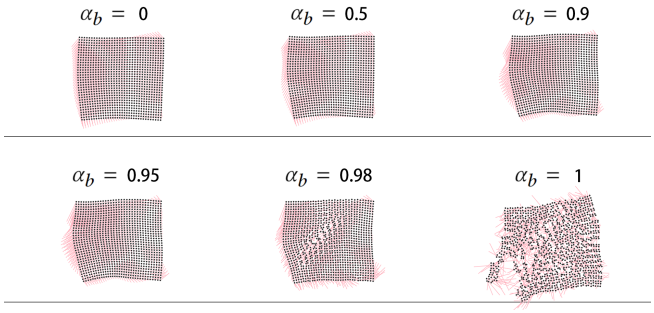

Fig. 4. **Stabilization** An elastic Jell-O drops onto the ground. To test the effect of velocity blending coefficient α_b in Eqn. (12), different α_b values are used to simulate the scenario. Without stabilization, the simulation is unstable. With a proper α_b, reasonable results are obtained. A larger α_b results in more flexible motion (red tails indicate velocity).

4.2 Phase Evolution

By carrying concentrations of each component on the particle system and evolving it during the simulation, multi-materials mixture can be simulated. In this system, each particle represents a certain amount of mass, its velocity presents the averaged motion of all different components. These different components can exchange among neighborhood particles, in which each particle keeps invariant mass. Cahn-Hilliard equation works well in simulating many phenomena dominated by phase-field evolution [Yang et al. 2017, 2015]. The PDE governing equation is

$$\frac{\partial \eta}{\partial t} = \nabla \cdot (L\nabla\mu_c), \tag{13}$$

where η denotes the volume density of a certain conservative order parameter, L degenerate mobility, and μ_c chemical potential. Using trial function v, the weak form of the above PDE reads:

$$\int_\Omega (\frac{\partial \eta}{\partial t}v + L\nabla\mu_c \cdot \nabla v)\mathrm{d}V - \oint_{\partial\Omega} vL\nabla\mu_c \cdot \mathbf{n}\mathrm{d}S = 0. \tag{14}$$

Assuming free boundary condition and following the Galerkin approach, we can obtain

$$\sum_j \left[\left(\frac{\partial \eta}{\partial t}\right)^h N_i(\mathbf{x}_k) + L_k\nabla\mu_c(\mathbf{x}_k) \cdot \nabla N_i(\mathbf{x}_k)\right] V_k = 0.$$

Applying a row-summing operation similar to mass lumping yields

$$\left(\frac{\partial \eta}{\partial t}\right)_i = -\frac{\sum_j \nabla N_i(\mathbf{x}_j) \cdot \nabla\mu_c(\mathbf{x}_j)L_j V_j}{\sum_j N_i(\mathbf{x}_j)V_j}.$$

Note that, Cahn-Hilliard equation is a conservation form for order parameter η. Thus η should be density of a conservative physics quantity. Yang et al. [2015] chose mass concentration c, i.e. $\eta = c$. As mainly focused on incompressible flow, the volume is conserved in their simulation, this choice worked without noticeable artifact. In reality, when two different material merge together, the total volume will be less than the sum of original volume. In such situation, volume varies, and using $\eta = c$ as in [Yang et al. 2015] does not conserve mass (see Fig. 5). In order to handle such volume varying phenomena, we choose η to be mass density of component materials

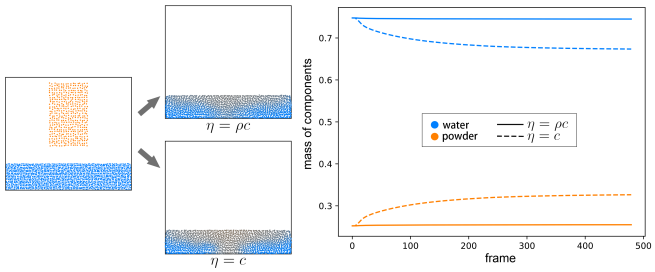

Fig. 5. **Mass conservation** Powder drops and dissolves in water. The formulation for Cahn-Hilliard equation used in [Yang et al. 2015] does not preserve mass. The total volume shrinks in this process, and mass conservation for each components is violated in the original formulation $\eta = c$ of [Yang et al. 2015], but it is met by our formulation by using $\eta = \rho c$.

which is related to mass concentration c and net mass density ρ, i.e. $\eta = c\rho$. The degenerate mobility is scaled by density $L = \tilde{L}\rho$ to eliminate the difference in diffusion rate caused by different density. The derivative of mass concentration is:

$$\left(\frac{\partial c}{\partial t}\right)_i = -\frac{1}{\rho_i}\frac{\sum_j \nabla N_i(\mathbf{x}_j)\cdot\nabla\mu_c(\mathbf{x}_j)\tilde{L}_j\rho_j V_j}{\sum_j N_i(\mathbf{x}_j)V_j}. \tag{15}$$

The mass conservation is met in our revised formulation as shown in Fig. 5.

The chemical potential μ_c is evaluated as:

$$\mu_c = \frac{\partial H}{\partial c} - \epsilon^2\nabla^2 c \tag{16}$$

where $H = H(c)$ is the Helmholtz free energy of the material and the second term is related to the free energy caused by interface effect. For simplicity, we consider only diffusion in our examples, for which the free energy function is set as $H(c) = (c-0.5)^2$. It is noted that other more sophisticated Helmholtz energy functions can also be used to handle chemical reaction. In our linear MLSRK implementation, the Laplacian is estimated by $(\nabla^2 c)_i = \frac{6}{a^2}\sum_j N_i(\mathbf{x}_j)(c_j - c_i)$ to avoid the calculation of the second derivative for shape functions, where a is the kernel radius scale used in Eqn. (8). Further explanation can be found in Appendix B. It is assumed ϵ is similar to a, which works well for simulating interface effects. After phase evolution, the mass concentration is then advected on particles.

4.3 Constitutive Models for Multiphase Continua

The uniform MLSRK Galerkin discretization enables the use of general constitutive models, whose effectiveness to animate complex material behaviors has been demonstrated in the literature using MPM. In this section, we describe how to work with general constitutive models in a unified framework based on the proposed quasi-linear MLSRK scheme. The unified continuum model is capable of handling various materials behaviors including hyperelastic, elastoplastic and viscous behaviors. Elastic and viscous stresses are added to get net stress inside the continuum. In our framework, fluids share the same continuum model as solids, and they are modelled by setting the friction angle to 0 in the Drucker-Prager yield condition. But other fluid constitutive model like WCSPH can also be used.

4.3.1 Deformation Gradient Update. Continuum constitutive models are often defined using the deformation gradient $\mathbf{F}$, which is commonly adopted in MPM. Likewise, they can be readily used with MLSRK. Specifically, the evolution of deformation gradient follows

$$\dot{\mathbf{F}} = (\nabla\mathbf{u})^T\mathbf{F}, \tag{17}$$

where the velocity gradient $\nabla\mathbf{u}$ can be estimated using MLSRK interpolation as

$$(\nabla\mathbf{u})_i = \sum_j \nabla N_j(\mathbf{x}_i)\otimes\mathbf{u}_j. \tag{18}$$

Different methods were used in previous MLS-related works to estimate deformation gradient. It can be obtained by fitting the initial position as in [Müller et al. 2004], or updated in each step as in [Gerszewski et al. 2009]. Our updating method is similar to [Gerszewski et al. 2009]. With initial position explicitly tracked on particles, the topology is fixed, which is suitable for elastic solids. If topology changes, as in plastic solids and fluids, the topology change cannot be handled in the algorithm. In the updated method, the initial configuration is not tracked, the topology is dynamically changing during the simulation, thus makes it more suitable for fluid simulation and also for solid merging and breaking behaviors. But during updating, the volume is estimated with errors caused by discretization, and the error accumulates as time advances, causing volume changing artifacts. MPM and FLIP fluid simulations also suffer from this problem. We propose a simple correction to address this issue in MLSRK. In SPH, the volume of a particle can be estimated by [Solenthaler and Pajarola 2008]:

$$V_i = \frac{1}{\sum_j W_{ij}}, \tag{19}$$

where W_{ij} is SPH kernel function. Although this estimation is affected by particle distribution, but the error does not accumulate as the time step advances. We use the above particle volume estimation to correct the deformation gradient, and the corrected deformation gradient is $\tilde{\mathbf{F}} = (\frac{V_i}{V_i^0 J})^{1/d}\mathbf{F}$, where $J = \det(\mathbf{F})$ and d is the dimension of space. This makes deformation volume change $\tilde{J} = \det(\tilde{\mathbf{F}})$ consistent with volume change V_i/V_i^0. This correction is used for liquid and granular material in our examples. Fig. 6 shows the effect of the correction.

4.3.2 Hyperelasticity. Given the deformation gradient, the Cauchy stress is determined for hyperelastic material by its property. We directly take the neo-Hookean solid model from [Jiang et al. 2016]. Specifically, the elastic energy density is

$$\Psi(\mathbf{F}) = \frac{\mu_e}{2}\mathrm{tr}(\mathbf{F}\mathbf{F}^T - \mathbf{I}) - \mu_e\log(J) + \frac{\lambda_e}{2}\log^2(J), \tag{20}$$

where $J = \det(F)$, and Lamé parameters μ_e and λ_e are related to Young's modulus E and Poisson ratio ν as:

$$\mu_e = \frac{E}{2(1+\nu)}, \quad \lambda_e = \frac{E\nu}{(1+\nu)(1-2\nu)}. \tag{21}$$

Consequently, given the deformation gradient, the Cauchy stress is:

$$\sigma_e = \frac{1}{J}(\mu_e(\mathbf{F}\mathbf{F}^T - I) + \lambda_e\log(J)\mathbf{I}). \tag{22}$$

The effect of elastic parameters are shown in Fig. 7.

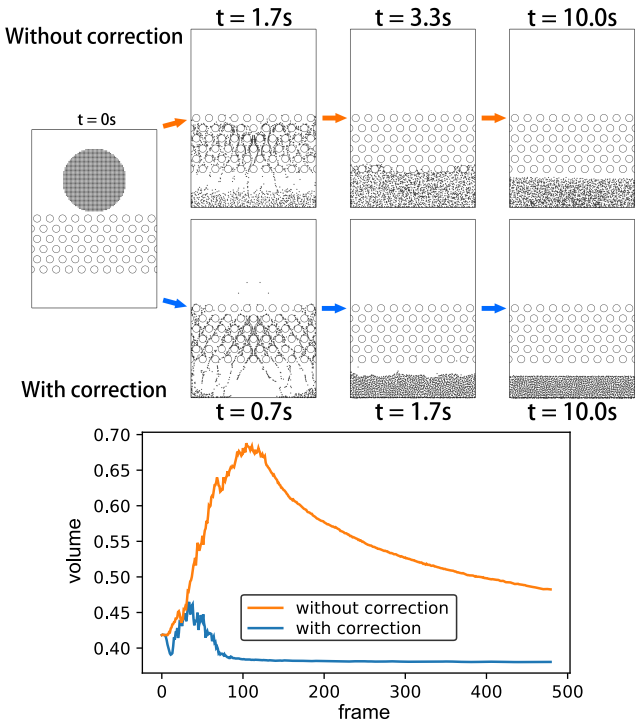

Fig. 6. **Galton Board** A water ball passes through a Galton board. With the proposed volume correction, the simulation is stable and delivers visually plausible results. Without volume correction, the simulation is unstable so that smaller time step and smaller blending coefficient α_b are used in Eqn. (12) to run the simulation, but still the results suffer from severe volume changing artifacts.

4.3.3 Viscosity. The viscous stress of a Newtonian fluid depends only on the velocity gradient, and is given as:

$$\sigma_v = \mu_v \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T - \frac{2}{d}(\nabla \cdot \mathbf{u})\mathbf{I} \right] + \zeta_v (\nabla \cdot \mathbf{u})\mathbf{I}. \tag{23}$$

where μ_v is the shear viscosity coefficient, ζ_v is the volume viscosity coefficient. The effect of these viscous coefficients are shown in Fig. 7. In our experiments, this term is not very useful as large viscous coefficients limit time step of our explicit scheme, and reduce the performance and makes the simulation time impractical.

4.3.4 Elastoplasticity. To handle elastoplasticity, the deformation gradient $\mathbf{F}$ is decomposed into an elastic part $\mathbf{F}_e$ and a plastic part $\mathbf{F}_p$:

$$\mathbf{F} = \mathbf{F}_e \mathbf{F}_p. \tag{24}$$

Then return mapping is adopted with certain yield criterion to model plasticity. The elastic is first updated by Eqn. (17), then decomposed as $\mathbf{F}_e = \mathbf{U}\Sigma\mathbf{V}^T$ using singular value decomposition. The singular values are then projected to the region that satisfies the yield condition, obtaining $\tilde{\Sigma}$. Then new elastic deformation gradient is constructed as $\tilde{\mathbf{F}}_e = \mathbf{U}\tilde{\Sigma}\mathbf{V}^T$, while new plastic deformation gradient is $\tilde{\mathbf{F}}_p = \tilde{\mathbf{F}}_e^{-1}\mathbf{F}_e\mathbf{F}_p$ to maintain the deformation gradient unchanged $\mathbf{F} = \tilde{\mathbf{F}}_e\tilde{\mathbf{F}}_p = \mathbf{F}_e\mathbf{F}_p$.

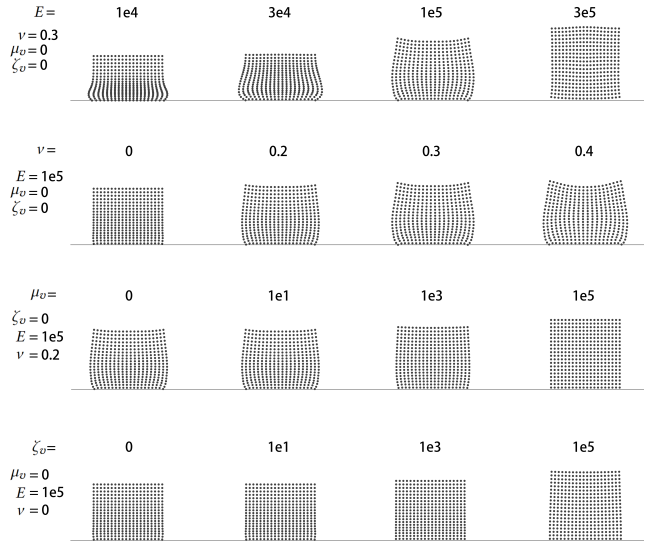

Fig. 7. **Viscoelastic Block** An viscoelastic cube drops onto the ground. Four groups of examples are tested to show the effect of different values of viscoelastic parameter E, ν, μ_v and ζ_v.

We can directly take return mapping used in MPM literatures. To simulate snow, we adopt the same return mapping as [Stomakhin et al. 2013], and the singular values $\sigma_i (i = 1, 2, 3)$ are clamped to range $[1 - \theta_c, 1 + \theta_s]$, where θ_c and θ_s is the criteria of compression and stretch. To better cope with fracture, we add a hardening factor to Lamé parameters like in [Stomakhin et al. 2013]:

$$\hat{\mu}_e = \mu_e e^{\xi(1-J_p)}, \ \hat{\lambda}_e = \lambda_e e^{\xi(1-J_p)}. \tag{25}$$

where ξ is the hardening coefficient and $J_p = \det(\mathbf{F}_p)$ is the determinant of plastic deformation gradient.

The Drucker-Prager yield condition is adopted for plasticity:

$$||\sigma_d||_F \leq a + b\sigma_m, \tag{26}$$

where $\sigma_m = \frac{\mathrm{tr}(\sigma)}{d}\mathbf{I}$ is mean stress, $\sigma_d = \sigma - \sigma_m$ is deviator stress, a represents cohesion, and b represents friction. Klár et al. [2016] used this model to animate sand. We adopt a similar return mapping augmented with cohesion handling like in [Tampubolon et al. 2017]. Before return mapping, the plastic volume change is first added back to the elastic part to avoid the volume gain like in [Tampubolon et al. 2017], $\tilde{\mathbf{F}}_e = (J_p)^{1/d}\mathbf{F}_e, \tilde{\mathbf{F}}_p = (J_p)^{-1/d}\mathbf{F}_p$. The singular values are first mapped to shifted log space

$$\epsilon = \log(\Sigma) - \epsilon_c \mathbf{I}, \tag{27}$$

where $\epsilon_c > 0$ determines the criteria deformation under tensile stress, and then projected to the cone which satisfies Drucker–Prager condition. Let

$$\tilde{\epsilon} = \epsilon - \frac{\mathrm{tr}(\epsilon)}{d}\mathbf{I}, \ \delta = ||\tilde{\epsilon}||_F + \frac{d\lambda_e + 2\mu_e}{2\mu_e}\mathrm{tr}(\epsilon)\alpha, \tag{28}$$

where α is determined from friction angle φ_f by

$$\alpha = \sqrt{\frac{2}{3}}\frac{2\sin\varphi_f}{3 - \sin\varphi_f}. \tag{29}$$

Then three difference cases are handled according to the relation to the cone. If $\mathrm{tr}(\tilde{\epsilon}) > 0$, $\mathbf{H} = \mathbf{0}$, else if $\delta > 0$, $\mathbf{H} = \epsilon - \delta\tilde{\epsilon}/||\tilde{\epsilon}||_F$, otherwise the Drucker–Prager condition is already satisfied $\mathbf{H} = \epsilon$. With $\mathbf{H}$ being the projected ϵ in log space, new singular values are reconstructed as $\tilde{\Sigma} = \exp(\mathbf{H} + \epsilon_c\mathbf{I})$. With cohesion added to Drucker–Prager yield condition, more material can be handled instead of being limited to sand. The capability of simulating fracture with the Drucker-Prager criterion is shown in Fig. 8, where the hardening term is also used for better handling of fracture.

4.3.5 Material Change in Phase Evolution. In multiphase simulation, the volume of a material phase may change with its concentration, and likewise the motion of a material may be affected by its phase evolution. For example, a wet sponge will shrink as it gets dried. To handle this mechanics and phase-field coupling phenomenon, we introduce an extra deformation gradient $\mathbf{F}_r$ to represent the deformation caused by concentration variation like in [Lubarda 2004], and incorporate this factor into multiplicative decomposition of deformation in Eqn. (24):

$$\mathbf{F} = \mathbf{F}_e\mathbf{F}_p\mathbf{F}_r \tag{30}$$

where $\mathbf{F}_r = (V_{rest}/V^0_{rest})^{\frac{1}{d}}\mathbf{I}$ represents the rest volume change caused by concentration variation. As each particle tracks a certain amount of mass, it can be calculated from rest density $\mathbf{F}_r = (\rho^0_{rest}/\rho_{rest})^{\frac{1}{d}}\mathbf{I}$. After updating phase concentration, ρ_{rest} may change according to the material property. When the concentration changes, the change of $\mathbf{F}_r$ is compensated by updating elastic deformation gradient to make $\mathbf{F}$ unchanged. Knowing the new $\tilde{\mathbf{F}}_r$ the updated elastic deformation gradient is

$$\tilde{\mathbf{F}}_e = \mathbf{F}_e\mathbf{F}_p\mathbf{F}_r\tilde{\mathbf{F}}_r^{-1}\mathbf{F}_p^{-1}. \tag{31}$$

where tilde mark denotes the value after phase concentration updation. This effect can be noticed in Fig. 1 and Fig. 5.

When materials get wet, the strength may vary. To describe the change of material behaviors caused by phase evolution, the parameters of a constitutive model vary with the phase concentration. For example a material tends to be softer and easier to tear when it is wet, we use a lower ϵ_c in Eqn. (27) in Fig. 1 and Fig. 13.

4.4 Regularization for Ill Particle Distribution and Algorithm Framework

The moment matrix $\mathbf{M}$ is assumed to be non-singular in MLSRK formulation. This is true in the continuous space, but not guaranteed when discretized with particles. The singularity of $\mathbf{M}$ depends on the distribution of neighborhood particles and the basis functions. For example, the moment matrix is always non-singular with constant basis if there is at least one neighborhood particle. With linear basis functions, the moment matrix is singular *iff* all neighborhood particles are co-planar. In dynamic scenes, e.g. water splashes and fracture fragments, individual particles often occur and cause singular moment matrices. To address this issue, adaptive basis can be set according to the distribution of neighborhood particles, but so doing introduces discontinuity in shape function. Generalized Moving Least Square (GMLS) was used to handle this problem for thin shell and rod simulation [Martin et al. 2010], but this requires more DOFs for each GMLS point and two set of points for interpolation and quadrature. Yreux and Chen [2017] tackled the problem by adding more quadrature points near the particles, which can violate linear consistency.

Considering the aforementioned solutions all complicate the implementation, we propose a simpler approach at no extra cost to avoid the singularity of $\mathbf{M}$. Specifically, a regularization term $E_r = a^d\sum_{\alpha=1}^{N_b}\frac{r_\alpha}{2}c_\alpha^2$ is introduced to the error functional in Eqn. (2), i.e. $\tilde{E} = E + E_r$. Here a is the kernel radius scale used in Eqn. (8) to make r_α dimensionless, so the value is independent of simulation scale. The correction coefficient is obtained by $\mathbf{c} = \mathrm{argmin}_{\mathbf{c}}(E+E_r)$. This is equivalent to adding a diagonal matrix $\mathbf{M}_r = a^d\mathbf{diag}(r_1, r_2, \ldots, r_n)$ to the moment matrix $\mathbf{M}$. Constant consistency is more important, without which momentum is not conserved (see Fig. 9). To ensure constant consistency, the coefficient for constant basis should not be penalized. We only penalize the linear coefficients, and consequently the linear consistency is a bit violated. The regularization matrix used in our quasi-linear MLSRK framework is:

$$\begin{aligned}\mathbf{M}_r &= a^d\begin{bmatrix}0&0&0\\0&r&0\\0&0&r\end{bmatrix}\text{ for 2D linear basis,}\\ \mathbf{M}_r &= a^d\begin{bmatrix}0&0&0&0\\0&r&0&0\\0&0&r&0\\0&0&0&r\end{bmatrix}\text{ for 3D linear basis.}\end{aligned} \tag{32}$$

With the proposed regularization term, the modified matrix is always non-singular if $r > 0$. If $r = 0$, the formulation is exactly the same as linear MLSRK. If $r = +\infty$, it is equivalent to constant MLSRK (also equivalent to Shepard interpolation), because linear coefficients are forced to be zero. In order to hold linear consistency better, r should be set a small value. The influence of different r values is demonstrated in Fig. 10. We also find a smaller r is necessary for simulating thin objects. The r in range $10^{-4} \sim 10^{-3}$ is used in our examples, which produces good results.

In our multiphase MLSRK framework, all physical quantities are tracked by a particle system. Each particle has its own position, velocity, mass, deformation gradient, velocity gradient, phase concentration, etc. Using explicit symplectic Euler integrator for time step advance, our MLSRK algorithm workflow is described in Algorithm 1. The algorithm mainly contains the following steps:

(1) Cache correction coefficient and its gradient for later use;
(2) Calculate stress from deformation gradient and velocity gradient;
(3) Calculate acceleration from stress and update velocity;
(4) Calculate chemical potential and update phase field parameter (only if multiphase simulation is involved);
(5) Update deformation gradient (perform return mapping if elastoplastic is involved), position.

A simple 2D MLSRK program implemented in less than 200 lines of readable C++ code is attached in the supplementary material of this paper, which implements the neo-Hookean model. This code only depends on standard library and OpenGL Mathematics (GLM) for vector operations, thus is handy for understanding the algorithm or using as an implementation reference.

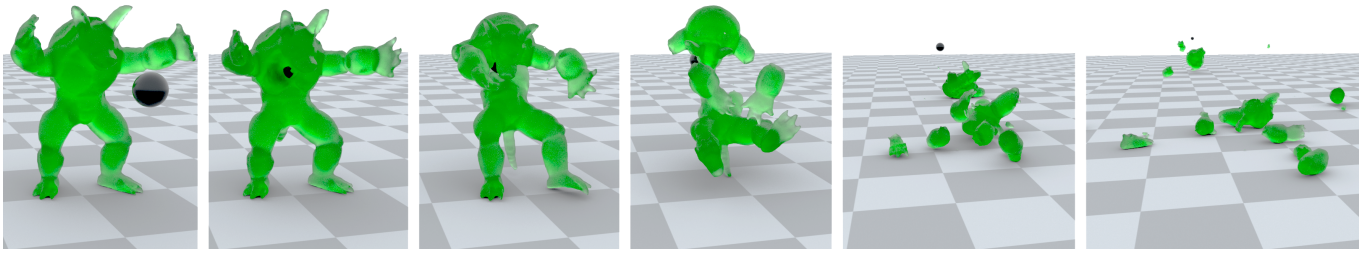

Fig. 8. **Shooting Armadillo** A Jell-O Armadillo is shot by a cannonball and smashed into pieces, demonstrating the capability to simulate fracture of complex materials.

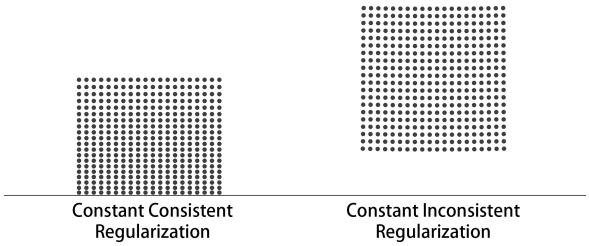

Fig. 9. **Falling Block** An elastic cube is dropped to the ground. Constant consistency plays an important role to ensure momentum conservation. With a regularization that guarantees constant consistency (left), the cube drops to the ground and behaves correctly. Without constant consistency (right), the cube floats unnaturally in the air due to erroneous 'drag force'.

5 RESULTS

A series of scenes with various materials are simulated to test the performance of the proposed multiphase MLSRK framework. In some examples, objects move at very fast speed, so we used higher framerate to generate a slow motion video. The program is coded with C++, using Intel TBB library for parallelization, except for the example in Fig 3 implemented in Taichi for performance comparison. The material density in most examples is set as 1.0×10^3. Discretization parameters and runtime performance data for all examples are listed in Table 2, and physical parameters are listed in Table 3. To save time, we run different examples on different machines. The computing hardware used for these simulations include an Intel Core i7-7700HQ platform with 8 threads, an Intel Xeon E5-2620 x2 platform with 24 threads, an Intel Xeon E5-2620 v3 x2 platform with 24 threads and an Intel Core i9-9980XE platform with 36 threads. These detailed information can be found in Table 2.

Rolling Snowball To demonstrate the capability of MLSRK for complex material, snow ball scene in [Gissler et al. 2020; Stomakhin et al. 2013] is revisited (see Fig. 2). At the top of the slope, the snowball is released and driven by gravity, it rolls down the slope, sticks snow on the slope and then smashes into the stack of balls. Our MLSRK simulation produces reliable results. The fracture path becomes regular and smooth if points are sampled on a regular grid, so point positions are jittered to get more natural fracture effect.

Mixture Dam-break The MPM cannot handle sub-grid particle-wise motion, because the grid discretization has less DoFs than the particles (see § 3.3). To demonstrate this, a scene of mixture dam-break is simulated with both methods as shown in Fig. 3. The densities of two liquid phases are set as 10^4 (blue) and 10^3 (orange), respectively. The volume correction described in § 4.3.1 is used in the MLSRK simulation. The WCSPH state equation is used like in [Tampubolon et al. 2017]. The phase separation process is successfully captured by the MLSRK simulation, but not by the MPM. Simply refining the resolution of MPM does not help, because grid always have less DoFs than the particles.

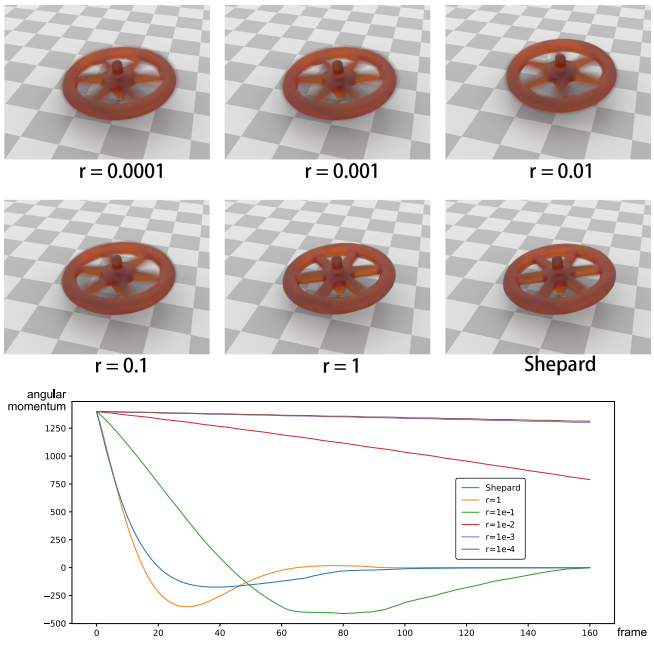

Fig. 10. **Spinning Top** A fast spinning elastic top is simulated in our MLSRK framework. To indicate the spinning speed, images are rendered with motion blur. Our method is robust in handling rotational motion with a proper regularization coefficient r. Different r values are tested: smaller values give more realistic results, while larger values prevent the rotational motion and result in incorrect behavior. The angular momentum along vertical axis is plotted for clarity.

Galton Board Water ball passes through a Galton board, which shows our MLSRK framework can handle the complex interaction

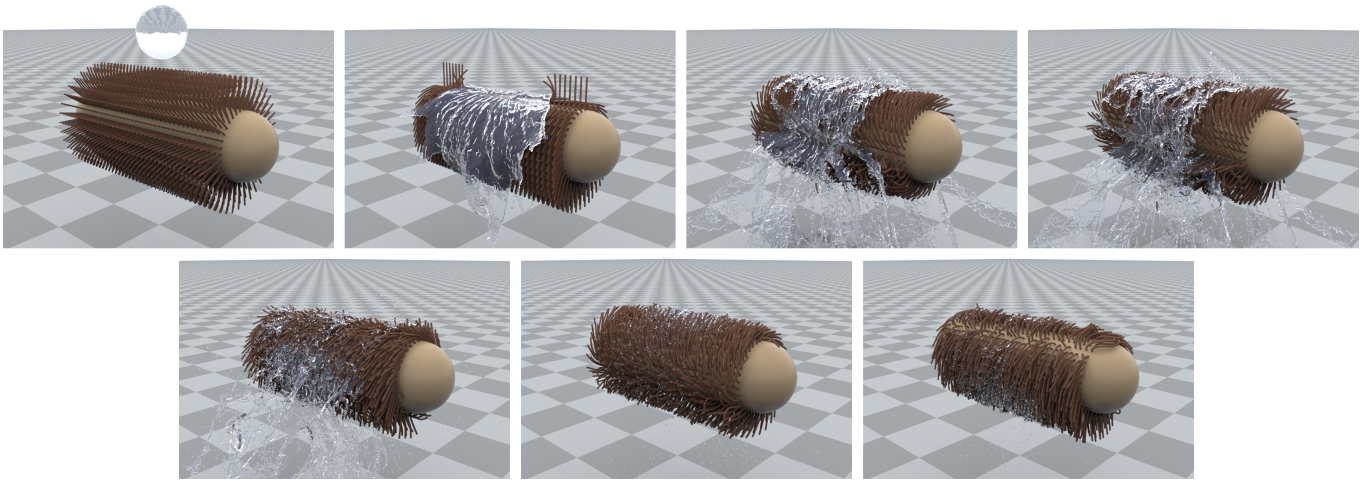

Fig. 11. **Pseudo-dog** A water ball hits a pseudo-dog, and it rotates body to shake off water.

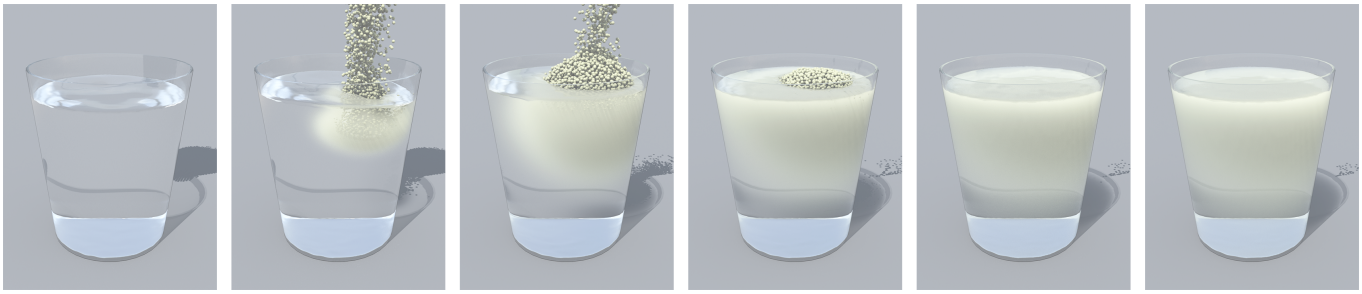

Fig. 12. **Starch Powder Dissolution** Starch powder is poured into a cup of water, which first floats on top the water accumulating into a pile and then slowly dissolves in water forming a sticky paste.

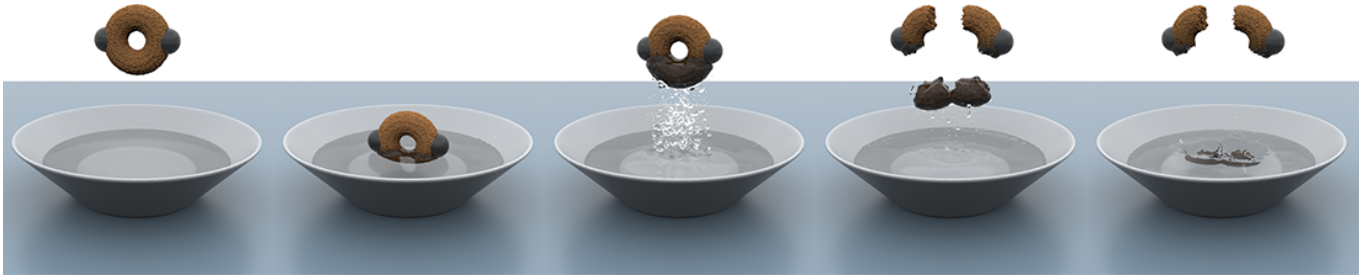

Fig. 13. **Biscuit** A ring-shape biscuit is first half dipped in a bowl of water and then pulled apart in the air, which causes the softer wet parts to drop into the bowl.

between fluid and obstacle (see Fig. 6). With the volume correction proposed in § 4.3.1, the simulation is stable. Without the volume correction, the simulation is less stable, so smaller time step and blending coefficient are used, but the resulted simulation still suffers from severe volume changing artifacts. In the plot, volume is estimated by measure the volume of the mesh produced by surface tracking.

Viscoelastic Block Four groups for parameters are used for simulating the same scene (see Fig. 7). Each groups varies in viscoelastic parameter E, ν, μ_v and ζ_v, showing the effect of these parameters.

Spinning Top To demonstrate the robustness of our MLSRK method in handling complex rotational motion, an elastic spinning top is simulated as shown in Fig. 10. The spinning top is made of a neo-Hookean material. Different values for the regularization coefficient r, defined in Eqn. (32), are tested. Using r in range 10^{-4} ~

ALGORITHM 1: Multiphase MLSRK Workflow

repeat
  for *each particle* **do**
    find neighborhood particles inside kernel support range;
    calculate MLSRK correction coefficient $\mathbf{b}$ and its gradient for the particle neighborhood from Eqn. (8);
  end
  for *each particle* **do**
    calculate stress tensor σ from deformation gradient $\mathbf{F}$ and velocity gradient $\nabla\mathbf{u}$ with Eqns. (22) and (23);
  end
  for *each particle* **do**
    calculate acceleration from stress σ using Eqn. (11), update velocity $\mathbf{u}_{n+1} = \mathbf{u}_n + \mathbf{a}_n\Delta t$;
    apply boundary condition to velocity;
  end
  for *each particle* **do** `// optional for phase evolution`
    calculate chemical potential μ_c from concentration with Eqn. (16);
  end
  for *each particle* **do** `// optional for phase evolution`
    update mass concentration c according to chemical potential with Eqn. (15);
  end
  for *each particle* **do**
    calculate velocity gradient $\nabla\mathbf{u}$ from Eqn. (18), update deformation gradient $\mathbf{F}^{n+1} = (\mathbf{I} + (\nabla\mathbf{u}^{n+1})^T\Delta t)\mathbf{F}^n$ following Eqn. (17);
    perform return mapping according to certain yield condition; `// optional for elastoplasticity`
  end
  for *each particle* **do**
    update position $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{u}^h_{n+1}\Delta t$ and perform velocity blending in Eqn. (12);
  end
until *simulation stops*;

10^{-3} results in realistic results, while larger values cause artifacts to different extent.

Shooting Armadillo To demonstrate the capability of fracture simulation, a Jell-O Armadillo is shot by a cannonball as shown in Fig. 8. The cannonball penetrates Armadillo and smashes it into pieces. The cannonball has a specified trajectory, and is used as boundary condition in simulation. The Drucker-Prager yield condition is used for handling elastoplasticity of Jell-O material. The hardening factor in Eqn. (25) is used to better handle fracture.

Pseudo-dog To demonstrate the capacity of animating coupled multiphase systems, a scene of water pouring onto a pseudo-dog is considered, as shown in Fig. 11. The pseudo-dog consists of a cylindrical body and a hemispherical head, both are rigid. Dog hairs are modelled as soft strings uniformly distributed around the cylindrical body. A water ball hits the pseudo-dog on the back, and it rotates body to shake off water. Dog hairs are treated as a neo-Hookean material. Each hair string is sampled with 2×2 particles on its cross section. The collision between water and hairs is automatically handled by the multiphase MLSRK framework. As shown in Fig. 11, plausible visual effects are obtained.

Starch Powder Dissolution To demonstrate the power of phase-field model, a scene of starch powder dissolving in water is simulated, as shown in Fig. 12. There is also a 2D version shown in Fig.5. The mass concentration of water c is recorded on each particle, with $c = 0$ indicating completely dry powder and $c = 1$ indicating pure water. We treat particles with $c > 0.3$ as water by setting the friction angle of Drucker–Prager condition to 0. A smoothly varied rest density function $\rho_{rest} = (0.45 + 1.15c - 0.6c^2) \times 10^3$ is used to determine the volume changing factor $\mathbf{F}_r$ (see Eqn. (30)) and volume correction is also used to solve volume changing artifacts, as described in § 4.3.1. After pouring into the water cup, starch powder first floats on top of the water and then slowly dissolves in water to form a sticky paste.

Biscuit In this example, as shown in Fig. 13, a biscuit is first half dipped in a bowl of water and then taken out and pulled apart in the air. Each particle has a tag attribute, which determines whether it is a biscuit particle or a water particle. The mass concentration of water c is also recorded on each particle. For biscuit particles, the cohesion criteria ϵ_c in Eqn. (27) decays as c increases, described by a smoothstep function, which causes the wet biscuit to be softer. After breaking the biscuit, the soft wet part cannot hold itself against gravity and drops into the bowl.

Tearing Wet Paper In this example, as shown in Fig. 1, a piece of paper is hit by a water ball and then the wet paper is teared into two pieces. The paper is sampled by two layers of particles, which is sufficient for stable MLSRK simulation of the paper sheet. The paper has a density map, which models the volume expansion of wet paper. The wet paper gets softened in the same way as Biscuit by changing ϵ_c according to c, which makes it break first when tore.

6 LIMITATIONS AND FUTURE WORK

Our MLSRK framework is not without limitation and some of the most demanding ones are highlighted here.

The first problem is performance. We used a conditionally stable explicit time scheme. When used with stiffer or more viscous material, smaller time step is required, which limits the performance and makes the simulation time impractical. It would be interesting to speed up the simulation by developing suitable implicit schemes for larger time step and using adaptive sampling techniques. For optimal performance, we are also interested in CPU acceleration with SIMD instructions and GPU computing. The improvement of performance will make the simulation of higher resolution and more details more affordable and allow larger stiffness and viscosity in Eqn. (22) and Eqn. (23).

In our deformation gradient updating scheme, each particle only influences its neighborhood within a fixed radius. When large deformation occurs, particles tend to be more sparse, they get out of neighborhood region of each other, artificial plasticity and artificial fracture will happen, for example a few hair get lost in the Pseudo-dog case (Fig. 11). A simple solution is to use larger radius, but doing so greatly increases the number of neighborhood particles and causes a loss in performance. Using varying neighborhood radius may help to address this issue in the future, and embedding Lagrangian particles like [Peer et al. 2018] in our framework may also help for extreme cases.

Table 2. Discretization parameters and runtime performance data

	Particle Number	Frame Rate (frames/s)	Simulation Speed (s/frame)	Particle Diameter (m)	Kernel Radius (m)	Time Step (s)	CPU
Rolling Snowball	6004 (snow) + 3000 (boundary)	24	0.81	0.005	0.015	4.0×10^{-4}	E5-2620 x2
Mixture Dam-break (MLSRK)	6.4k	24x2	1.05	0.005	0.0125	1.0×10^{-4}	i9-9980XE
Mixture Dam-break (MPM, same #particles)	6.4k		0.45	0.005	0.03	1.0×10^{-4}	
Mixture Dam-break (MPM, same DoFs)	25.6k		1.17	0.0025	0.015	5.0×10^{-5}	
Falling Block	250	24x2	0.031	0.05	0.15	1.0×10^{-3}	i7-7700HQ
Spinning Top	37.5k	24x2	20	0.02	0.06	4.0×10^{-5}	E5-2620 x2
Stabilization	1156	24x2	0.072	0.02	0.06	4.0×10^{-4}	E5-2620 x2
Mass Conservation	1887	24	0.65	0.01	0.03	2.0×10^{-4}	E5-2620 v3 x2
Galton Board (with correction)	1876	24x2	0.077	0.01	0.03	5.0×10^{-4}	i9-9980XE
Galton Board (without correction)			0.38			1.0×10^{-4}	i9-9980XE
Viscoelastic Block	250	24x2	0.037 ~ 0.7	0.05	0.15	$2.5 \times 10^{-5} \sim 1.0 \times 10^{-3}$	i7-7700HQ
Shooting Armadillo	238k	24x8	82	0.01	0.03	1.2×10^{-4}	E5-2620 x2
Pseudo-dog	259k (hair) + 262k (water)	24x8	1017 ~ 294	0.001	0.003	1.0×10^{-5}	E5-2620 v3 x2
Starch Powder Dissolution	175k (material) + 139k (boundary)	24x4	58	0.002	0.0054	1.0×10^{-4}	i9-9980XE
Biscuit	65k (biscuit) + 296k (water)	24x8	357	0.001	0.002	3.3×10^{-5}	E5-2620 x2
Tearing Wet Paper	400k (paper) + 493k (water)	24x16	217	0.002	0.006	2.0×10^{-5}	i9-9980XE

Table 3. Physical parameters for various materials in our examples

	Lamé parameters		Drucker-Prager		Snow		Hardening	Mobility
	μ_e	λ_e	φ_f	ϵ_c	θ_c	θ_s	ξ	$\tilde{L}$
Rolling Snowball	4.2×10^5	2.8×10^5	-	-	2.0×10^{-2}	7.5×10^{-3}	10	-
Mixture Dam-break	0	4.0×10^5	0	0	-	-	-	-
Spinning Top	1.0×10^7	1.0×10^7	-	-	-	-	-	-
Stabilization	5.0×10^4	5.0×10^4	-	-	-	-	-	-
Mass Conservation (powder)	1.0×10^5	1.0×10^5	57.3°	0	-	-	-	1.0×10^{-3}
Mass Conservation (water)			0	0	-	-	-	1.0×10^{-4}
Galton Board	1.0×10^5	1.0×10^5	0	0	-	-	-	-
Shooting Armadillo	5.0×10^5	1.0×10^6	45.8°	0.05	-	-	10	-
Pseudo-dog (hair)	5.0×10^4	5.0×10^4	-	-	-	-	-	-
Pseudo-dog (water)			0	0	-	-	-	-
Starch Powder Dissolution (powder)	1.0×10^5	1.0×10^5	17.2°	0	-	-	-	1.0×10^{-5}
Starch Powder Dissolution (water)			0	0	-	-	-	1.0×10^{-4}
Biscuit (biscuit)	1.0×10^5	1.0×10^5	28.7°	0.012(dry) ~ 0.002(wet)	-	-	-	2.0×10^{-5}
Biscuit (water)			0	0	-	-	-	
Tearing Wet Paper (paper)	4.0×10^6	4.0×10^6	57.3°	0.1(dry) ~ 0.02(wet)	-	-	-	5.0×10^{-4}
Tearing Wet Paper (water)			0	0	-	-	-	

The phase field model implemented in this work is relatively simple with limited capacity (e.g. it does not consider percolation), and it would be an interesting future work to examine other more advanced phase-field models in the MLSRK framework.

7 CONCLUSION

We propose an MLSRK approach for unified multiphase continuum simulation. This is achieved via two steps: (1) formulate accurate and stable MLSRK discretization of Cauchy momentum equation for general constitutive models in a systematic manner, to support hyperelastic, elastoplastic, viscous, fracturing with general continuum constitutive model to uniformly capture motion of all materials and their mechanical interaction; (2) describe phase evolution/interaction with phase field model to handle diffusing, dissolving material behaviors. The volume change caused by phase change is incorporated into constitutive model. Being a pure particle-based method with uniform discretization, our proposed MLSRK framework is as easy as SPH for implementation and as extensible as MPM for animating complex materials.

REFERENCES

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J Guibas. 2007. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (TOG)*, Vol. 26. Acm, 48.

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 62.

Iván Alduán and Miguel A. Otaduy. 2011. SPH granular flow with friction and cohesion. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '11)*. Association for Computing Machinery, Vancouver, British Columbia, Canada, 25–32. https://doi.org/10.1145/2019406.2019410

Y. Bazilevs, G. Moutsanidis, J. Bueno, K. Kamran, D. Kamensky, M. C. Hillman, H. Gomez, and J. S. Chen. 2017. A new formulation for air-blast fluid–structure interaction using an immersed approach: part II—coupling of IGA and meshfree discretizations. *Computational Mechanics* 60, 1 (July 2017), 101–116. https://doi.org/10.1007/s00466-017-1395-2

Markus Becker, Markus Ihmsen, and Matthias Teschner. 2009. Corotated SPH for deformable solids. In *Proceedings of the Fifth Eurographics conference on Natural Phenomena (NPH'09)*. Eurographics Association, Munich, Germany, 27–34.

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 209–217.

Jan Bender and Dan Koschier. 2015. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM, 147–155.

J. Bonet and T. S. L. Lok. 1999. Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering* 180, 1 (Nov. 1999), 97–115. https://doi.org/10.1016/S0045-7825(99)00051-1

Jiun-Shyan Chen, Chunhui Pan, Cheng-Tang Wu, and Wing Kam Liu. 1996. Reproducing kernel particle methods for large deformation analysis of non-linear structures. *Computer methods in applied mechanics and engineering* 139, 1-4 (1996), 195–227.

Jiun-Shyan Chen, Cheng-Tang Wu, Sangpil Yoon, and Yang You. 2001. A stabilized conforming nodal integration for Galerkin mesh-free methods. *International journal for numerical methods in engineering* 50, 2 (2001), 435–466.

Mengyuan Ding, Xuchen Han, Stephanie Wang, Theodore F Gast, and Joseph M Teran. 2019. A thermomechanical material point method for baking and cooking. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 192.

Yun Raymond Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2018. A multi-scale model for simulating liquid-fabric interactions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 51.

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A Multi-Scale Model for Coupling Strands with Shear-Dependent Liquid. *ACM Trans. Graph.* 38, 6, Article Article 190 (Nov. 2019), 20 pages. https://doi.org/10.1145/3355089.3356532

Yun Raymond Fei, Henrique Teles Maia, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2017. A multi-scale model for simulating liquid-hair interactions. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 56.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 222.

Ming Gao, Andre Pradhana, Xuchen Han, Qi Guo, Grant Kot, Eftychios Sifakis, and Chenfanfu Jiang. 2018a. Animating fluid sediment mixture in particle-laden flows. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 149.

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 223.

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018b. Gpu optimization of material point methods. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 254.

Dan Gerszewski, Haimasree Bhattacharya, and Adam W. Bargteil. 2009. A point-based method for animating elastoplastic solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '09)*. Association for Computing Machinery, New Orleans, Louisiana, 133–138. https://doi.org/10.1145/1599470.1599488

R. A. Gingold and J. J. Monaghan. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 181, 3 (Dec. 1977), 375–389. https://doi.org/10.1093/mnras/181.3.375 Publisher: Oxford Academic.

Christoph Gissler, Andreas Henne, Stefan Band, Andreas Peer, and Matthias Teschner. 2020. An Implicit Compressible SPH Solver for Snow Simulation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. 2019. Interlinked SPH pressure solvers for strong fluid-rigid coupling. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 5.

Pai Chen Guan, Jiun-Shyan Chen, Y Wu, H Teng, J Gaidos, K Hofstetter, and M Alsaleh. 2009. Semi-Lagrangian reproducing kernel formulation and application to modeling earth moving operations. *Mechanics of Materials* 41, 6 (2009), 670–683.

Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran. 2018. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 147.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 150.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics* 38, 6 (Nov. 2019), 201:1–201:16. https://doi.org/10.1145/3355089.3356506

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2013), 426–435.

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 152.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 51.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses (SIGGRAPH '16)*. Association for Computing Machinery, Anaheim, California, 1–52. https://doi.org/10.1145/2897826.2927348

Ben Jones, Stephen Ward, Ashok Jallepalli, Joseph Perenia, and Adam W. Bargteil. 2014. Deformation embedding for point-based elastoplastic simulation. *ACM Transactions on Graphics* 33, 2 (April 2014), 21:1–21:9. https://doi.org/10.1145/2560795

R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutre, and M. Gross. 2005. A unified Lagrangian approach to solid-fluid animation. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* 125–148. https://doi.org/10.1109/PBG.2005.194073 ISSN: 1511-7813.

Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 103.

Wing K Liu, Shaofan Li, and Ted Belytschko. 1997. Moving least-square reproducing kernel methods (I) methodology and convergence. *Computer Methods in Applied Mechanics and Engineering* 143, 1-2 (1 1 1997), 113–154. https://doi.org/10.1016/S0045-7825(96)01132-2

Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. 1995. Reproducing kernel particle methods. *International journal for numerical methods in fluids* 20, 8-9 (1995), 1081–1106.

Vlado A Lubarda. 2004. Constitutive theories based on the multiplicative decomposition of deformation gradient: Thermoelasticity, elastoplasticity, and biomechanics. *Applied Mechanics Reviews* 57, 2 (2004), 95–108. https://doi.org/10.1115/1.1591000 _eprint: https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article-pdf/57/2/95/5440572/95_1.pdf.

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 104.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 153.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics* 29, 4 (July 2010), 39:1–39:10. https://doi.org/10.1145/1778765.1778776

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless deformations based on shape matching. *ACM Transactions on Graphics* 24, 3 (July 2005), 471–478. https://doi.org/10.1145/1073204.1073216

Joe J Monaghan. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics* 30, 1 (1992), 543–574.

Georgios Moutsanidis, David Kamensky, J. S. Chen, and Yuri Bazilevs. 2018. Hyperbolic phase field modeling of brittle fracture: Part II—immersed IGA–RKPM coupling for air-blast–structure interaction. *Journal of the Mechanics and Physics of Solids* 121 (Dec. 2018), 114–132. https://doi.org/10.1016/j.jmps.2018.07.008

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 154–159.

Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. 2004. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 141–151.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics* 32, 3 (July 2013), 27:1–27:22. https://doi.org/10.1145/2487228.2487235

Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. 2005. Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers*

(SIGGRAPH '05). Association for Computing Machinery, Los Angeles, California, 957–964. https://doi.org/10.1145/1186822.1073296

Andreas Peer, Christoph Gissler, Stefan Band, and Matthias Teschner. 2018. An implicit SPH formulation for incompressible linearly elastic solids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 135–148.

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 114.

Andreas Peer and Matthias Teschner. 2016. Prescribed velocity gradients for highly viscous SPH fluids with vorticity diffusion. *IEEE transactions on visualization and computer graphics* 23, 12 (2016), 2656–2662.

Stefan Reinhardt, Tim Krake, Bernhard Eberhardt, and Daniel Weiskopf. 2019. Consistent shepard interpolation for SPH-based fluid animation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 189.

Bo Ren, Chenfeng Li, Xiao Yan, Ming C Lin, Javier Bonet, and Shi-Min Hu. 2014. Multiple-fluid SPH simulation using a mixture model. *ACM Transactions on Graphics (TOG)* 33, 5 (2014), 171.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: a sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics* 33, 6 (Nov. 2014), 205:1–205:12. https://doi.org/10.1145/2661229.2661269

Barbara Solenthaler and Renato Pajarola. 2008. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 211–218.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM transactions on graphics (TOG)*, Vol. 28. ACM, 40.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 102.

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 138.

Tetsuya Takahashi and Ming C Lin. 2019. A Geometrically Consistent Viscous Fluid Solver with Two-Way Fluid-Solid Coupling. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 49–58.

Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 105.

Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. 2018. A physically consistent implicit viscosity solver for SPH fluids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 145–155.

Xiao Yan, Yun-Tao Jiang, Chen-Feng Li, Ralph R Martin, and Shi-Min Hu. 2016. Multiphase SPH simulation for interactive fluids and solids. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 79.

Xiao Yan, C-F Li, X-S Chen, and S-M Hu. 2018. MPM simulation of interacting fluids and solids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 183–193.

Tao Yang, Jian Chang, Ming C Lin, Ralph R Martin, Jian J Zhang, and Shi-Min Hu. 2017. A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 224.

Tao Yang, Jian Chang, Bo Ren, Ming C Lin, Jian Jun Zhang, and Shi-Min Hu. 2015. Fast multiple-fluid simulation using Helmholtz free energy. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 201.

Edouard Yreux and Jiun-Shyan Chen. 2017. A quasi-linear reproducing kernel particle method. *Internat. J. Numer. Methods Engrg.* 109, 7 (2017), 1045–1064.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

A DERIVATIVE OF MLSRK SHAPE FUNCTION

$$\begin{aligned}
\partial\mathbf{M}(\mathbf{x}) =& -\frac{1}{a}\sum_i [\partial\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})^T\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a}) \\
&+\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\partial\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})^T\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a}) \\
&+\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})^T\partial\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a})]V_i, \\
\partial\mathbf{M}^{-1}(\mathbf{x}) =& -\mathbf{M}^{-1}(\mathbf{x})\partial\mathbf{M}(\mathbf{x})\mathbf{M}^{-1}(\mathbf{x}), \\
\partial\mathbf{b}(\mathbf{x})^T =& \mathbf{h}(\mathbf{0})^T\partial\mathbf{M}^{-1}(\mathbf{x}), \\
\partial N_i(\mathbf{x}) =& [\partial\mathbf{b}(\mathbf{x})^T\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a}) \\
&-\frac{1}{a}\mathbf{b}(\mathbf{x})^T\partial\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a}) \\
&-\frac{1}{a}\mathbf{b}(\mathbf{x})^T\mathbf{h}(\frac{\mathbf{x}_i-\mathbf{x}}{a})\partial\Phi(\frac{\mathbf{x}_i-\mathbf{x}}{a})]V_i,
\end{aligned}$$

where ∂ can be partial derivative of coordinate x, y and z.

B LAPLACIAN ESTIMATION WITHOUT CALCULATING SECOND DERIVATIVE

In finite difference formulation, Laplacian is often calculated by a stencil. For example, Laplacian on a 2D grid is estimated as

$$\begin{aligned}
(\nabla^2 u)_{i,j} &= \frac{1}{\Delta x^2}(u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}-4u_{i,j}) \\
&= \frac{4}{\Delta x^2}(\frac{u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}}{4}-u_{i,j}),
\end{aligned}$$

where Δx is the size of grid cell. This indicates the possibility that $(u^h-u)/a^2$ can approximate Laplacian for MLSRK, where u^h is the interpolated value and a is the kernel scaling factor in Eqn. (8).

Consider the second order Tayler approximation of a function u at origin,

$$u(\mathbf{x}) = c + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x},$$

where $\mathbf{g}$ is the gradient, and $\mathbf{H}$ is the Hessian matrix. According to Eqn. (4), MLSRK interpolation of this function at origin is

$$u^h(\mathbf{0}) = \mathbf{h}(\mathbf{0})^T\mathbf{M}^{-1}(\mathbf{0})\int_\Omega \mathbf{h}(\mathbf{x})\Phi(\mathbf{x})u(\mathbf{x})\mathrm{d}\mathbf{x}.$$

If linear basis (Table 1) and Cartesian cubic spline kernel (Eqn. (7)) is used and Ω covers the support of Φ, the integration can be directly integrated analytically

$$u^h(\mathbf{0}) = c + \frac{\mathrm{tr}(\mathbf{H})}{6}.$$

Knowing $\mathrm{tr}(\mathbf{H}) = \nabla^2 u(\mathbf{0})$ and $c = u(0)$, it is equivalently

$$\nabla^2 u(\mathbf{0}) = 6(u^h(\mathbf{0}) - u(\mathbf{0})).$$

With the shifted and scaled formulation, an extra factor appears in the integration. We have

$$\nabla^2 u(\mathbf{x}) = \frac{6}{a^2}(u^h(\mathbf{x}) - u(\mathbf{x})).$$

Using particle discretization, we can rewrite this with shape function:

$$(\nabla^2 u)_i = \frac{6}{a^2}\sum_j N_j(x_i)(u_j - u_i).$$