Implicit Bonded Discrete Element Method with Manifold Optimization

JIA-MING LU, Tsinghua University, Beijing, China
GENG-CHEN CAO, Tsinghua University, Beijing, China
CHENFENG LI, College of Engineering, Swansea University, Swansea, United Kingdom
SHI-MIN HU, Tsinghua University, Beijing, China

Fig. 1. (a) A ceramic plate dropped on a hard floor and broken into pieces. Our simulation captures a great level of details using only 219K particles, with a 2.4x speedup over the state-of-the-art method. (b) A piece of knitted fabric being torn apart, demonstrating our algorithm's ability to handle extreme deformations and material failure in textiles, with a 3.3x speedup over the state-of-the-art method.

This paper proposes a novel simulation approach that combines implicit integration with the Bonded Discrete Element Method (BDEM) to achieve faster, more stable and more accurate fracture simulation. The new method leverages the efficiency of implicit schemes in dynamic simulation and the versatility of BDEM in fracture modelling. Specifically, an optimization-based integrator for BDEM is introduced and combined with a manifold optimization approach to accelerate the solution process of the quaternion-constrained system. Our comparative experiments indicate that our method offers better scale consistency and more realistic collision effects than FEM and MPM fragmentation approaches. Additionally, our method achieves a computational speedup of 2.1 ~ 9.8 times over explicit BDEM methods.

Authors' Contact Information: Jia-Ming Lu, Tsinghua University, Beijing, Beijing, China; e-mail: jaimeyzzz@outlook.com; Geng-Chen Cao, Tsinghua University, Beijing, Beijing, China; e-mail: cgc21@mails.tsinghua.edu.cn; Chenfeng Li, College of Engineering, Swansea University, Swansea, United Kingdom; e-mail: c.f.li@swansea.ac.uk; Shi-min Hu, Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: shimin@tsinghua.edu.cn.

ACM 1557-7368/2025/1-ART
https://doi.org/10.1145/3711852

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

Additional Key Words and Phrases: Discrete Element Method, Fracture Simulation, Optimization-based Integrator, Manifold Optimization

1 Introduction

Fracture and damage occur to all materials, and they represent a fundamental type of interaction in virtual environments. As an important topic in physics-based simulation, fracture modelling is a challenging task due to the complex underlying physical processes. The ubiquity of fractures in real life sets a high standard for plausible results to satisfy human perception. To date, most simulation methods for materials are based on the continuum theory, which have limited capacity in capturing fracture phenomena. The BDEM can better simulate fractures [Lu et al. 2021]. One of the most significant challenges in fracture simulation is keeping the error of internal forces under control whilst allowing an acceptable error rate in fracture propagation. For simulation methods based on the continuum theory, the accuracy in stresses is typically much poorer than the accuracy in displacements, making it hard if not impossible to manage the simulation error during fracture propagation. The BDEM does not have this limitation, as its stress and displacement are not hardwired through differential operators.

However, solving BDEM systems is complicated by the stiff nature of the unit length constraint in quaternion systems, and it becomes more challenging for such materials as rope and cloth due to the added complexity of shear stiffness. The collision stiffness in discrete element simulations is another major challenge, as it can significantly increase the number of Newton iterations required. These issues are prevalent and well recognized in BDEM simulations [André et al. 2012; Nguyen et al. 2021]. Implicit integrators are widely used in computer graphics to accelerate the efficiency of dynamic simulations, where the backward Euler scheme is the most prevalent method [Baraff and Witkin 1998; Hirota et al. 2001; Liu et al. 2013; Martin et al. 2011; Volino and Magnenat-Thalmann 2001]. Other integrators, such as the implicit-explicit method [Eberhardt et al. 2000; Stern and Grinspun 2009] and exponential integrators [Chen et al. 2017; Michels et al. 2014], have also received significant attention. Recent work used a variational integrator to model the inter-granule contact [De Klerk et al. 2022]. However, none of these well-established implicit schemes is effective for BDEM.

To address this challenge, we introduce an optimization-based integrator to reformulate BDEM and combine it with a manifold optimization approach to deliver faster, more stable and more accurate fracture simulations. Our approach leverages the strength of BDEM in fracture modelling, whilst achieving superior computational efficiency with the resulting implicit solution scheme. Our experiments show that without loss of accuracy, the proposed approach is more efficient than explicit BDEM methods. The main contributions include:

- A stable optimization-based integrator suitable for implicit BDEM is formulated, and it allows large time steps.
- A manifold optimization approach for BDEM is introduced to speed up the solution of quaternion-constrained systems.

The rest of the paper is organized as follows. §2 briefly recaps related works in fracture simulation and discrete element methods. The main body of work is presented in §3, §4, which present respectively the optimization-based integration for BDEM and the manifold optimization approach to accelerate the solution process. More implementation details are provided in §5, followed by results and discussions in §6. Finally, concluding remarks are made in §7.

2 Related Works

Fracture Simulation. Fracture simulation is an important topic in computer graphics, owing to the ubiquity of fracture phenomena in real life. A number of non-physical methods have been proposed to produce crack patterns [Bao et al. 2007; Hellrung et al. 2009; Müller et al. 2013; Raghavachary 2002; Schvartzman and Otaduy

2014; Su et al. 2009; Zheng and James 2010], whilst physics-based methods are more accurate and have become the preferred approach for fracture simulation. Till recently, the Finite Element Method (FEM) is the most widely used physics-based technique for fracture simulation [Koschier et al. 2014; Müller et al. 2001; O'Brien and Hodgins 1999; Pfaff et al. 2014; Wicke et al. 2010]. Other physics-based methods include the Extended Finite Element Method (XFEM) [Chitalu et al. 2020; Kaufmann et al. 2009], the Boundary Element Method (BEM) [Hahn and Wojtan 2015, 2016; Zhu et al. 2015], the mesh-less method [Pauly et al. 2005], peridynamics [Chen et al. 2018], and the Material Point Method (MPM) [Fan et al. 2022; Wolper et al. 2020, 2019]. Each of these methods has its own pros and cons and is suitable for different fracture scenarios. Besides addressing underlying physics, accurately capturing fracture surfaces is crucial for realistic visual effects in fracture propagation. For FEM fracture simulation, methods like adaptive remeshing [Chen et al. 2014; Koschier et al. 2014; Pfaff et al. 2014; Wicke et al. 2010], duplicate elements [Molino et al. 2004], and sub-elements [Nesme et al. 2009; Wojtan and Turk 2008] are employed to capture and prevent poor-quality elements. Explicit surface methods [Koschier et al. 2017], post-processing techniques [Chitalu et al. 2020; Kaufmann et al. 2009; Wang et al. 2019], level-set methods [Hahn and Wojtan 2015], and mesh-based methods [Chitalu et al. 2020; Sifakis et al. 2007; Zhu et al. 2015] offer alternative approaches for accurately representing fracture surfaces.

Discrete Element Methods. Introduced by Cundall and Strack for solving problems in rock mechanics [Cundall 1971; Cundall and Strack 1979], the Discrete Element Method (DEM) is widely used in engineering for modelling granular materials. The DEM has also been adopted in computer graphics for visual effects involving granules and particles [Alduán et al. 2009; Bell et al. 2005; Rungjiratananon et al. 2008; Yue et al. 2018]. To simulate continuum media, Potyondy and Cundall extended the DEM to form the BDEM [Potyondy and Cundall 2004], and since then it has seen growing adoption in various engineering applications. Examples include [André et al. 2012], where a cohesive bond model was used to represent continuum objects, and [Nguyen et al. 2021], where a plastic cohesive bond was used to simulate ductile fractures. The BDEM was recently introduced by [Lu et al. 2021] to the computer graphics community, and it was proven to be robust and flexible for producing realistic visual effects involving fractures. As far as plausible visual effects are concerned, the main limitation of BDEM fracture simulation is its computational efficiency, due to the need for small time steps in the simulation.

Simulation with Rotation. Different from FEM, where the degrees of freedom of motion are defined on a set of points / nodes and include only translations, the degrees of freedom for DEM are defined on rigid spheres, which include both translational and rotational degrees of freedom. The explicit inclusion of rotation as part of element states is a major advantage for DEMs to represent the deformation and motion of real world objects. There are other methods that also explicitly account for the rotational motion. A prominent and well-established category within this domain is the simulation of rigid body motion, which has been subjected to extensive exploration for several decades [Baraff 1989; Bender et al. 2014; Hahn 1988; Stewart 2000]. Recent endeavors in rigid body simulation include the employment of the Extended Position-Based Dynamics method for solving the motion of rigid bodies [Müller et al. 2020], differentiable rigid contact models [Geilinger et al. 2020], and the modeling of rigid body motion and collision resolution through the formulation of incremental potential [Ferguson et al. 2021]. Certainly, in addition to rigid body simulations, the incorporation of rotation as a degree of freedom is also utilized in various other methods. One such method uses particles with orientation [Müller and Chentanez 2011], but this geometric approach does not capture basic physical concepts such as momentum and stress, which can limit its ability in producing realistic visual effects. Another example is rod elements for modelling slender objects. The Discrete Elastic Rod (DER) approach [Bergou et al. 2008] was used to compute rod curvature, whilst Cosserat rods were used in [Kugelstadt and Schömer 2016; Soler et al. 2018] to model complex bending and torsion effects. The rod elements approach is effective for slender and flexible structures (e.g. ropes and fibers), but they are less effective for simulating generic objects.

Optimization-based Integrator. In physics-based simulations, the resolution of complex nonlinear equations is often required, which can lead to inefficient and unstable solutions due to their inherent complexity. An integrator based on energy optimization is a method designed to enhance the efficiency and stability of solving nonlinear equations. This approach has been applied to various types of computational problems, including fluid dynamics [Weiler et al. 2016], crowd simulation [Karamouzas et al. 2017], and elastic bodies [Fratarcangeli et al. 2016; Gast et al. 2015; Martin et al. 2011; Wang and Yang 2016] and nonlinear materials [Kee et al. 2023; Li et al. 2019; Löschner et al. 2020]. Projective dynamics represent a class of optimization-based methods for solving constrained systems, capable of rapidly resolving a diverse array of materials [Bouaziz et al. 2014; Liu et al. 2013, 2016; Narain et al. 2016; Weiler et al. 2016]. Incremental potential is also a class of methods based on energy optimization, which has recently been employed to handle collisions between various objects [Ferguson et al. 2021; Lan et al. 2021; Li et al. 2020a,b]. By introducing the interior point method to resolve impacts, it consistently achieves interpenetration-free outcomes. In our work, akin to the approach presented in [Ferguson et al. 2021], we have derived an integrator with rotational degrees of freedom in an energy-optimized form. In contrast to the complex formulation originally targeted at rigid body simulation, our integrator has been specifically tailored for the BDEM system, resulting in a significantly streamlined solution that achieves efficient computation.

Fig. 2. showcases the experimental results of our method on a chocolate drop impact simulation. The figure illustrates the moment when a chocolate drop hits the ground and cracks into several pieces and small fragments. The simulation shows the effectiveness of our approach in accurately modeling high-speed object collisions and fracture phenomena.

3 Implicit Bonded Discrete Element Method

A significant difference between BDEM and other approaches is the direct consideration of rotational Degrees of Freedom (DoFs) as system variables. Previous methods solve BDEM systems using explicit formulations that store rotation status as quaternions, Euler angles or axis angles, and compute torques and angular velocities explicitly to alter the rotation status of discrete elements. In implicit formulations, position and rotation states cannot be explicitly updated, and forces or torques are often implicit or invisible during the solution process. To form an implicit BDEM scheme, it requires a different formulation.

Solving generic nonlinear equations are unstable, a promising strategy to tackle this challenge is transforming the nonlinear simulation problem into a nonlinear optimization problem that is easier to solve [Gast et al. 2015]. Optimization-based integrators are effective in improving the stability and convergence speed of nonlinear solution processes. Considering BDEM system, a direct idea is that we formulate the solution of motion equations with rotational degrees of freedom into an energy minimization form to achieve a fast and stable integrator. In conventional rigid body solutions, due to the introduction of rotational degrees of freedom, the mass matrix M in the extended motion equations is varies with the motion state [Geilinger et al. 2020; Liu and Jain 2012]. Therefore, we cannot directly write the extended motion equations in the form of energy minimization. In pursuit of an improved solution form, akin to the approach detailed in [Kane et al. 2000], we have re-derived the motion

equation for BDEM based on quaternions. We have discovered that, leveraging the inherent characteristics of BDEM, we can ultimately arrive at an exceedingly succinct formulation for optimization-based integrator.

3.1 Optimization-based Integrator

Building upon Hamiltonian mechanics for quaternion-based rigid body dynamics, we present a simplified computational framework for BDEM simulations of spherical particles. Our key contribution is the derivation of a constant mass matrix formulation that maintains solution convergence while eliminating the need for rotation state-dependent mass matrices. The complete mathematical foundation is presented in Appendix C. Our formulation yields a block-diagonal extended mass matrix M_s:

$$M_s = \begin{pmatrix} mI_3 & \\ & \frac{8}{5}mr^2 I_4 \end{pmatrix}. \tag{1}$$

Where m is the particle mass, r is the sphere radius, and I_3 and I_4 are identity matrices of dimensions 3 and 4, respectively.

This constant mass matrix enables us to express the dynamics as a discrete variational principle, leading to the following optimization formulation:

$$\Phi(\boldsymbol{x}_{i+1}) = \frac{1}{2\Delta t^2}(x_{i+1} - \hat{z})^T M_s (x_{i+1} - \hat{z}) + V(\frac{x_{i+1} + x_i}{2}). \tag{2}$$

where Δt is the time step size, x_i represents the system state at time step i (including both position and quaternion components), and V denotes the potential energy. The auxiliary variables $\hat{x}$ and $\hat{z}$ are defined as:

$$\begin{aligned} \hat{x} &= 2x_i - x_{i-1} \\ \hat{z} &= \hat{x} - \frac{1}{2}M_s^{-1}\Delta t^2 \nabla V(\frac{x_{i-1} + x_i}{2}) \end{aligned} \tag{3}$$

These variables are computed from known states at previous time steps. The system evolution is then determined by solving the constrained optimization problem:

$$x_{i+1} = \arg\min_x \Phi(x) \quad \text{s.t.} \quad C(x) = 0, \tag{4}$$

where $C(x)$ represents the quaternion unit length constraint (see Appendix A). This variational integrator, a symplectic integrator with second-order accuracy, is now ready for our simulation and compatible with our following manifold optimization approach. However, to further reduce computational costs, we introduce an approximation in our experiments to tradeoff the computational speed and simulation accuracy. We approximate both $\nabla V(\frac{x_{i+1}+x_i}{2})$ and $\nabla V(\frac{x_{i-1}+x_i}{2})$ with $\nabla V(x_{i+1})$. This approximation, while reducing the integrator to first-order accuracy, maintains compatibility with our manifold optimization approach. Empirical comparisons reveal that the primary drawback is slightly increased damping, a trade-off for improved computational efficiency. The modified form, similar to the implicit Euler method, is:

$$\frac{1}{\Delta t^2}M_s(x_{i+1} - \hat{x}) + \nabla V(x_{i+1}) = 0. \tag{5}$$

with the corresponding objective function:

$$\Phi(\boldsymbol{x}_{i+1}) = \frac{1}{2\Delta t^2}(\boldsymbol{x}_{i+1} - \hat{\boldsymbol{x}})^T M_s (\boldsymbol{x}_{i+1} - \hat{\boldsymbol{x}}) + V(\boldsymbol{x}_{i+1}). \tag{6}$$

Note that this formulation bears a striking resemblance to its counterpart without rotational degrees of freedom. This similarity might raise questions about the treatment of rotation-related terms. To address this potential source of confusion and provide a comprehensive understanding, we present an alternative derivation with a detailed exposition of the rotational dynamics in Appendix B.

Fig. 3. A ceramic plate dropped on a hard floor and broken into pieces. Our simulation captures highly detailed fractures using only 219K particles, with a 2.4x speedup over the state-of-the-art method.

3.2 Potential Energy in BDEM

Now we require only the specification of a potential function to solve the BDEM system. In a setup where two vertices x_i=$\{p_i, q_i\}$ and x_j=$\{p_j, q_j\}$ are bonded together, the deformation of the bond can be decomposed into four forms: stretch, shear, bend, and twist, akin to the form presented in [Lu et al. 2021]. It is straightforward to derive the bond energy with these four deformation types:

$$V = V_{\text{stretch}} + V_{\text{shear}} + V_{\text{bendtwist}}. \tag{7}$$

Stretch. The stretch energy of a bond can be modeled using a simple spring model and expressed as follows:

$$V_{\text{stretch}} = \frac{1}{2} k_n (\|p_i - p_j\| - l_0)^2, \tag{8}$$

where $k_n = \frac{ES}{l_0}$, l_0 is the original bond length, E is Young's modulus, and S is the bond's cross-sectional area, with $S = \pi(\frac{r_i + r_j}{2})^2$, using the radii r_i and r_j of two bonded elements.

Shear. The rotation of two elements can be decomposed into a common rotational component and a differential rotational component. As elucidated in the article [Lu et al. 2021], the shear energy is modeled by the variation of the common rotational component relative to the rest state of the bond. To extract the common rotation, the equation $q_c = (q_i + q_j)/\|q_i + q_j\|$ is used, where $d = q_c \odot d_0$ represents the current direction. The corresponding energy can be expressed as

$$V_{\text{shear}} = \frac{1}{2} k_t \theta^2, \tag{9}$$

where $\theta = \langle d_0, d_q \rangle$ denotes the angle between the bond's initial direction d_0 and current direction d. The shear stiffness is $k_t = GS/l_0$, where G represents the material's shear modulus.

Bend and Twist. The bend and twist energy of discrete elements can be defined as the difference in their rotational component. It is important to note that the stiffness of bend and twist have distinct relationships with material properties, thus requiring separate decomposition of the two types of rotations. The difference of quaternion, expressed in the form of $G_l(q_j)q_i$, can be transformed to the local frame via the rotation matrix $R_0 = G_l(q_0)G_r(q_0)$, where q_0 is the rotation between d_0 and $\{1, 0, 0\}$. The resulting vector can then be represented as $t = R_0 G_l(q_j)q_i$. With the aid of the stretch and shear stiffness matrix denoted as K, the energy form can be expressed using

$$V_{\text{bendtwist}} = \frac{1}{2} t^T K t. \tag{10}$$

The stiffness matrix is defined by

$$K = \frac{1}{l_0}\begin{pmatrix} GJ & & \\ & EI & \\ & & EI \end{pmatrix}, \tag{11}$$

here, $I = \pi r_0^4/4$ and $J = \pi r_0^4/2$ are the second moments of area for bending and twisting, respectively.

3.3 Fracture Criteria

In the study by [Lu et al. 2021], the tensile and shear stresses within the bond in BDEM are calculated as:

$$\sigma = \frac{\|\boldsymbol{F}_n\|}{S} + \frac{\|\boldsymbol{M}_b\| r_0}{I} \tag{12}$$

$$\tau = \frac{\|\boldsymbol{F}_s\|}{S} + \frac{\|\boldsymbol{M}_t\| r_0}{J}, \tag{13}$$

where $\boldsymbol{F}_n$ and $\boldsymbol{F}_s$ represent the normal and shear forces acting on the bond, respectively. The quantities $|\boldsymbol{M}_b|$ and $|\boldsymbol{M}_t|$ indicate the magnitudes of the bending moments about the bond's normal and transverse axes, respectively, and S, I, and J denote the cross-sectional area, second moment of area, and polar moment of area of the bond's cross-section, respectively. If the stress within the bond exceeds the respective strengths, i.e. $\sigma > \sigma_c$ or $\tau > \tau_c$ with σ_c and τ_c representing the tensile and shear strengths, the bond breaks.

In the explicit method, forces and torques are calculated directly, thus allowing the aforementioned fracture criteria to be readily computed. For our implicit method, where forces and torques are not computed directly, in order to employ the same fracture criteria, it is necessary to calculate the forces within the implicit setting. To achieve this, we have utilized the following equations:

$$F_n = \nabla_p V_{\text{stretch}} \tag{14}$$

$$F_s = \nabla_p V_{\text{shear}} \tag{15}$$

$$\|M_b\| = \|\langle Kt, e_1\rangle e_1 + \langle Kt, e_2\rangle e_2\| \tag{16}$$

$$\|M_t\| = \|\langle Kt, e_0\rangle e_0\|, \tag{17}$$

where K and t represent the stiffness matrix and rotation vector in the bending and twisting potential equation given by Eq. (10). In our experiments, we have found that this approach works effectively, yielding results that are nearly identical to those produced by the explicit method.

3.4 Element Packing

In the domain of discrete element simulations, the concept of packing is analogous to the discretization process in continuum methods. Common packing techniques employed in discrete element simulations include random close packing and hexagonal close packing. For engineering applications that aim to accurately simulate material properties, random close packing combined with calibration is a frequently utilized approach. For the sake of simplicity in our application, we have employed hexagonal close packing. Our experiments have revealed that even with this most basic packing method, it is possible to effectively capture rich fracture details without introducing visually discernible artifacts that might result from overly regular patterns.

4 Manifold Optimization

The above optimization approach generally improves the solution stability, but to solve it efficiently remains a challenge due to the quaternion constraint in Eq. (54). Our experiments have revealed that traditional methods such as the penalty method and the Lagrange multiplier method perform poorly. In this work, we present a manifold optimization scheme, where the quaternion constraint is represented as a hyper sphere enabling more

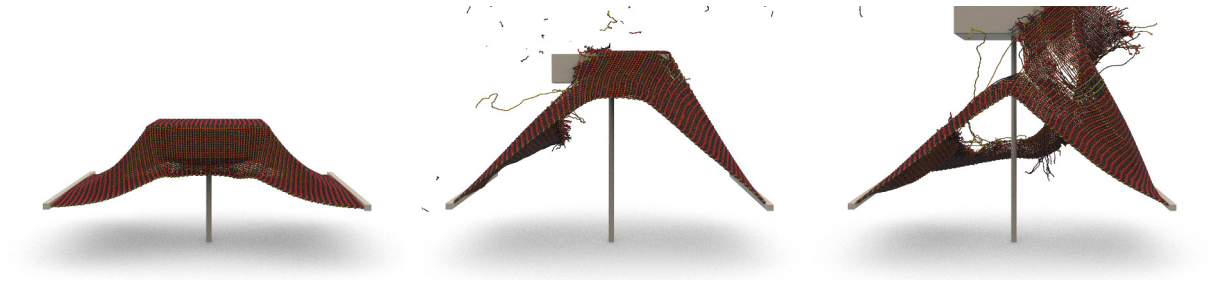

Fig. 4. This scene illustrates the simulation of a woven fabric undergoing progressive stretching and eventual rupture using our method. With only 165K particles, our approach effectively captures the realistic behavior of yarn-level cloth and the phenomenon of fabric tearing, demonstrating a convincing rupture process and rich details at the tear propagation. Compared to current explicit methods, our technique achieves a threefold acceleration.

effective optimization search on a manifold. Experimental results demonstrate that our new approach significantly improves the efficiency and accuracy of the BDEM solution compared to previous methods.

The manifold optimization approach [Boumal 2022] replaces the constrained optimization problem on $\mathbb{R}^n$ with an unconstrained optimization problem on a specific manifold $\mathcal{M}$. In our approach, we utilize the second-order manifold optimization and employ a nullspace operator to further reduce the number of unknowns in the system. To better explain this solution process, we also introduce the first-order manifold optimization and compare its performance with our more advanced approach. To facilitate subsequent discussions, we establish specific nomenclature to differentiate functions defined on the full space from those on the manifold. The objective function is represented as $\bar{f}$ in the original Euclidean space and as f in the new manifold $\mathcal{M}$, such that $\bar{f}|\mathcal{M} = f$. In our scenario, the manifold is defined as $\mathcal{M} = \prod_{i=1}^{n} \mathbb{R}^3 \otimes \mathbb{S}^3$, where n denotes the number of particles. Additionally, grad f and Hess f refer to the gradient and Hessian in the manifold space, while grad $\bar{f}$ and Hess $\bar{f}$ pertain to the gradient and Hessian in the full space. The subscript p denotes the $\mathbb{R}^3$ space, and the subscript q denotes the $\mathbb{S}^3$ space.

4.1 First-Order Manifold Optimization

Algorithm 1 First-Order Manifold Optimization

Require: Initial guess x_0, termination threshold ϵ

1: **while** true **do**
2: Compute the grad $f(x_k)$ on the manifold $\mathcal{M}$.
3: **if** $\|\text{grad } f(x_k)\| < \epsilon$ **then**
4: **return** $x_k \in \mathcal{M}$
5: **end if**
6: Use descent direction $-\text{grad } f(x_k)$ with manifold line search step α to update $x_{k+1} \leftarrow x_k - \alpha \text{ grad } f(x_k)$ on the manifold $\mathcal{M}$.
7: Project x_{k+1} to the manifold $\mathcal{M}$.
8: $k \leftarrow k + 1$
9: **end while**

The first-order manifold optimization process is outlined in Algorithm 1. It closely parallels standard gradient-based optimization methods but incorporates specific adjustments to align with the manifold.

In Line 2, the gradient computation in manifold optimization involves a manifold tangent projection in comparison to standard gradient computation. Specifically, for our quaternion constraint, the gradient grad f is calculated as:

$$\operatorname{grad}_p f = \operatorname{grad}_p \bar{f} \tag{18}$$

$$\operatorname{grad}_q f = (I - qq^T) \operatorname{grad}_q \bar{f}, \tag{19}$$

where the subscript p refers to the $\mathbb{R}^3$ space and the subscript q refers to the $\mathbb{S}^3$ space.

Line 6 is analogous to the line search and stepping procedures within standard gradient based optimization. In standard gradient based optimization, we employ direct addition to perform line searches and steps, which may lead to the system's state leaving the manifold. In contrast, manifold optimization requires that the corresponding line search and stepping be conducted on the manifold, ensuring that every step within the search and step progression adheres to the constraints of the manifold. Particularly, for our quaternion constraints, once the descent direction is known, we can ensure that the search and stepping remain on the manifold by employing a line search along a geodesic on $\mathbb{S}^3$. For element i with position p^i and rotation q^i, with a fast descent direction $\Delta p^i = -\operatorname{grad}^i_p f$, $\Delta q^i = -\operatorname{grad}^i_q f$ and step α, the stepping on quaternion manifold is computed by

$$p^i_{\text{new}} = p^i + \alpha \Delta p^i \tag{20}$$

$$q^i_{\text{new}} = q^i \cos \|\alpha \Delta q^i\| + \frac{\alpha \Delta q^i}{\|\alpha \Delta q^i\|} \sin \|\alpha \Delta q^i\|. \tag{21}$$

In the line search procedure, this approach is adopted for the incremental search of the system state. This same method is also applied during the final iterative update step.

In Line 7, the q vector is guaranteed to lie on the manifold through a normalization step. This step can be regarded as a manifold projection, and is given by:

$$q^i_k \leftarrow \frac{q^i_k}{\|q^i_k\|}. \tag{22}$$

It ensures that q satisfies the manifold constraint, allowing the optimization process to proceed effectively.

4.2 Second-Order Manifold Optimization

The second-order manifold optimization takes into account the Hessian and gradient of the objective function $f(x)$ on the manifold $\mathcal{M}$, and the algorithm steps are summarized in Algorithm 2.

The computation of Hess f in Line 6 is performed using Proposition 5.8 in [Boumal 2022], and the procedure is summarized below:

$$\operatorname{Hess}_{p_i,p_j} f = \operatorname{Hess}_{p_i,p_j} \bar{f} \tag{23}$$

$$\operatorname{Hess}_{p_i,q_j} f = \operatorname{Hess}_{p_i,q_j} \bar{f}(I - q_j q_j^T) \tag{24}$$

$$\operatorname{Hess}_{q_i,p_j} f = (I - q_i q_i^T) \operatorname{Hess}_{q_i,p_j} \bar{f} \tag{25}$$

$$\operatorname{Hess}_{q_i,q_j} f = (I - q_i q_i^T) \operatorname{Hess}_{q_i,q_j} \bar{f}(I - q_j q_j^T) \tag{26}$$

$$+ \begin{cases} 0 & i \neq j \\ -q_i^T (\operatorname{grad}_{q_i} \bar{f})(I - q_i q_i^T) & i = j \end{cases}. \tag{27}$$

The symmetric nature of Hess f allows us to solve the large sparse system using the commonly used preconditioned conjugate gradient method. With manifold optimization, the constrained optimization problem is transformed into an unconstrained problem. Even so, the unknowns of the system are still more than the true

Algorithm 2 Second-Order Manifold Optimization

Require: Initial guess x_0, termination threshold ϵ
while true **do**
  Compute the grad $f(x_k)$ on the manifold $\mathcal{M}$.
  if $\|\text{grad}\, f(x_k)\| < \epsilon$ **then**
    return $x_k \in \mathcal{M}$
  end if
  Compute Hess $f(x_k)$ on the manifold $\mathcal{M}$.
  Solve $(\text{Hess}\, f(x_k))\Delta x = \text{grad}\, f(x_k)$ in tangent space $T\mathcal{M}$.
  Use descent direction Δx with manifold line search step α to update $x_{k+1} \leftarrow x_k + \alpha\Delta x$ on the manifold $\mathcal{M}$.
  Project x_{k+1} to the manifold $\mathcal{M}$.
  $k \leftarrow k + 1$
end while

degrees of freedom of the physical system. To overcome this issue, we use the nullspace operator. Note that both the gradient project operator $P(q) = I - qq^T$ and the nullspace operator $G(q)$ have the similar property $P(q)q = 0$ and $G(q)q = 0$ (detailed introductions to these two operators are provided in Appendix A). Here we use $H = \text{Hess}\, f$ and $b = \text{grad}\, f$ as shorthand notation. The original second-order solution formulation for rotation $(I - qq^T)H(I - qq^T)\Delta x = (I - qq^T)b$ is reformulated as:

$$G(q)^T G(q) H (G(q)^T G(q))\Delta x = G(q)^T G(q) b. \tag{28}$$

By defining $y = G(q)\Delta x$ and $z = G(q)b$, the original solution is transformed to:

$$G(q) H G(q)^T y = z. \tag{29}$$

The solution can then be expressed as $\Delta x = G(q)^T y$. This approach reduces the number of unknowns in the system to the DoFs of the BDEM system and improves the solution efficiency.

Table 1. The simulation statistics for our experiments are listed below, with the slow-motion of 1x corresponding to one frame per 1/60th of a second, and 0.125x corresponding to one frame per 1/480th of a second. The time step and time consumption for the proposed approach are compared with explicit and implicit methods at different element numbers and stiffness settings.

Demo	N	E	Δt (explicit)	Δt (implicit)	Slow-Motion Scale	s/frame(explicit)	s/frame(implicit)	speedup
Beam Stretch	19K	1E7	2.6E-5	0.016	1x	34.6	3.52	9.8
Beam Twist	19K	1E7	2.6E-5	0.0015	1x	34.6	14.76	2.3
Beam Drape	64K	1E9	1.7E-6	0.008	1x	2230.9	711.7	3.1
Beam Drape	124K	1E6	4.4E-5	0.0024	1x	255.2	121.9	2.1
Chocolate	58K	1E9	4.4E-7	4.4e-5	0.125x	857.8	343.8	2.5
Plate	219K	1E9	3.5E-7	3.5e-5	0.125x	4564.6	1911.8	2.4
Cloth	165K	1E9	3.5E-7	1.74e-5	0.125x	1299.6	394.2	3.3

5 Implementation Details

The utilization of optimization-based integrator formulation and manifold optimization helps to develop an unconditionally stable integrator suited for implicit BDEM simulation with large time steps. The simulation workflow is summarized in Algorithm 3, and the implementation details are explained in the following subsections.

The proposed methodology significantly improves the stability and efficiency of BDEM simulations, enabling them to tackle complex and computationally demanding scenes.

Algorithm 3 Stable Integrator

Require: timestep Δt, termination threshold ϵ

$k \leftarrow 0$
Initial guess $x_0 \leftarrow x_n + v_n \Delta t$
while true **do**
  Compute the grad $f(x_k)$ on the manifold $\mathcal{M}$
  if $\| \text{grad}\, f(x_k) \| < \epsilon$ **then**
    $x_{k+1} \leftarrow x_k$
    break
  end if
  Compute Hess $f(x_k)$ on the manifold $\mathcal{M}$
  Project Hess $f(x_k)$ to semi-positive definite Hess$'$ $f(x_k)$
  Compute relative error ϵ_r
  Solve $(\text{Hess}' f(x_k))\Delta x = -\text{grad}\, f(x_k)$ in tangent space $T\mathcal{M}$ with preconditioned conjugate gradient method with relative error ϵ_r
  Use descent direction Δx with manifold line search step α to update $x_{k+1} \leftarrow x_k + \alpha \Delta x$ on the manifold $\mathcal{M}$.
  Project x_{k+1} to the manifold $\mathcal{M}$.
  $k \leftarrow k + 1$
end while
$x_{n+1} \leftarrow x_{k+1}$.
$v_{n+1} \leftarrow \frac{1}{\Delta t}(x_{n+1} - x_n)$
Project v_{n+1} to friction cone
Update bond state with x_{n+1}

5.1 Timestep Selection

In our research, the selection of time steps for both implicit and explicit simulations requires careful consideration. For explicit simulations, we must choose a time step that is sufficiently large to permit a reasonable comparison of efficiency. Regarding implicit simulations, the time step should be sufficiently increased to ensure significant performance acceleration without excessively compromising the realism of the fragmentation effects. Taking into account these factors, we have followed the approach established in explicit BDEM for estimating the system's period to inform our choice of an appropriate time step. In [Lu et al. 2021], the appropriate time step is ascertained by analyzing the system's period T, which correlates with the system's largest eigenvalue λ as delineated by the equation $T = 2/\sqrt{\lambda}$. The largest eigenvalue λ of the system is calculated using the expression

$$\lambda = \frac{k}{m} N, \tag{30}$$

where m denotes the effective mass, k signifies the stiffness coefficient, and N corresponds to the maximum eigenvalue of the Laplacian matrix representing the inter-bond connections. For a hexagonal arrangement, $N \approx 16$. Additionally, the stiffness k is given by $k = k_n \approx \frac{\pi}{2} Er$, with E representing the Young's modulus and r being the average radius of the elements.

For explicit simulation, our experimental findings indicate that a time step of $0.2T$ is the threshold that avoids numerical instability and oscillations. This time step is consistently applied across all explicit simulation

comparisons with implicit methods presented in the text. Regarding implicit simulations, our optimization-based method supports stable computations with large time steps, potentially allowing for one time step per frame. However, increasing the time step can lead to more iterations in the Newton's solver, decreasing search efficiency. Beyond a certain point, the computational savings from increasing the time step are not substantial. Additionally, an overly large time step may result in infrequent fracture detection, which can impair the simulation's accuracy. In our experiments, we have found that a time step ranging from 20 to 100 times the period T is reasonable.

5.2 Semi-Positive Definite Projection

The Hessian matrix on a manifold may not be semi-positive definite, which poses a significant challenge as it may lead to a wrong search direction. To guarantee that the search direction is a correct descent direction, a semi-positive definite projection onto the manifold is applied to our Hessian matrix. Similar regularization treatments are also found in other methods [Li et al. 2020a].

5.3 Preconditioned Conjugate Gradient

The preconditioned conjugate gradient method is employed to efficiently solve large sparse symmetric systems of the form Hess $f(x_k)$. Specifically, we employ an algebraic multigrid (AMG) precondition to facilitate fast and accurate solutions. Similar to [Gast et al. 2015], the absolute error is not explicitly managed in the beginning steps, and instead the relative error is taken as a termination criterion for the conjugate gradient method. This relative error is calculated as:

$$\epsilon_r = \min(0.5, \sqrt{|\operatorname{grad} f(x_k)|}). \tag{31}$$

The AMG preconditioning in conjunction with the relative error termination criterion lead to efficient and accurate solutions for large sparse symmetric systems.

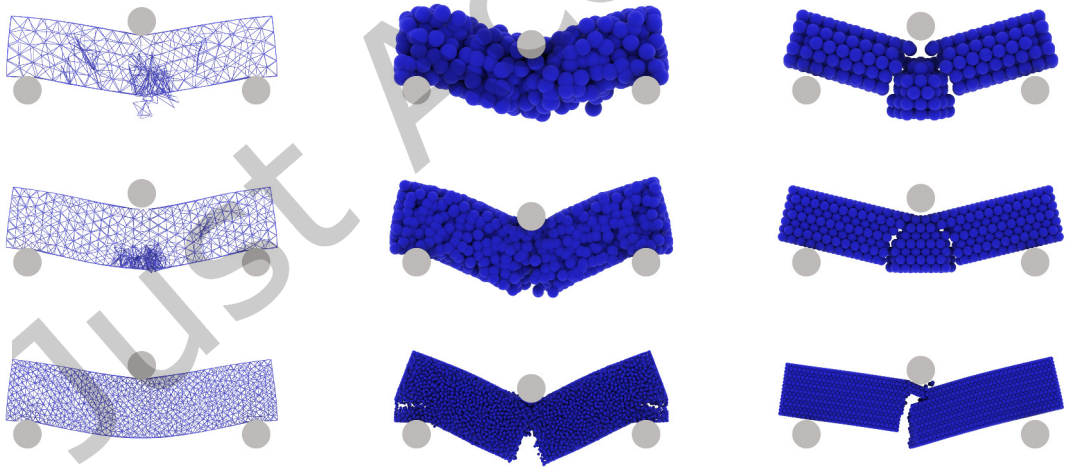

Fig. 5. The figure, from left to right, illustrates the performance of the FEM, MPM, and our approach across varying scales. The upper, middle, and lower rows represent the fracture behavior of the three methods under identical loads at different scales. It is observable that FEM exhibits completely distinct fracture occurrences across the three scales. MPM shows similar fracture timings at medium and high scales but deviates at lower scales. In contrast, our method consistently demonstrates fracture at the same load across all scales, indicating superior scale consistency.

5.4 Collision Handling

In accordance with our optimization formulation, a variety of collision handling methods can be seamlessly integrated into our approach, including simple penalty-based methods and recently utilized incremental potential contact [Li et al. 2020a]. However, this paper primarily focuses on fragmentation-related phenomena, where the impact of the collision module on the fragmentation performance is minimal and not the main emphasis here. Therefore, we have employed a very simple treatment method for visually acceptable simulation results.

Repulsion Term. A simple collision potential is introduced to model repulsion between discrete elements and external objects. Specifically, the potential includes two components: V_{self}, representing the repulsive forces between the discrete elements, and V_{external}, representing the repulsive forces between the elements and the external objects. The corresponding mathematical expressions are:

$$V_{\text{self}} = \frac{1}{2} k_{ec} (\|p_i - p_j\| - r_i - r_j)^2 \tag{32}$$

$$V_{\text{external}} = \frac{1}{2} k_c \|g(p_i) \nabla g(p_i)\|^2, \tag{33}$$

where k_{ec} denotes element collision stiffness, k_c external collision stiffness, p_i and p_j the positions of the discrete elements, and g the signed distance function of the external objects. In order to ensure that the collision term does not adversely affect the system's computational efficiency while maintaining a visually coherent outcome without noticeable interpenetration, we have set the collision stiffness and element collision stiffness to the order of $Er_{\max}$, where E is the max Young's Modulus of element and $r_{\max}$ is the max radius of elements. In our experiments, this selection has minimal impact on the system's convergence rate and yields visually acceptable simulation results.

Friction. Our approach for modifying the velocity and angular velocity of an element in the post-collision process relies on an explicit method. In the case of an external collision or self-collision, the relative velocity v_{ij} and the relative angular velocity w_{ij} are computed between element i and the external object j or another element j, respectively, using $v_{ij} = v_n + v_t$, where v_n and v_t denote the normal and tangential components of the relative velocity. To calculate the modification of velocity and angular velocity, we use the following equations:

$$\Delta v_i = -\min(1.0, \mu_s \frac{\|F_n\| \Delta t}{m_i \|v_t\|}) v_t \tag{34}$$

$$\Delta w_i = -\min(1.0, \mu_r \frac{\|F_n\| r_i^3 \Delta t}{I_i \|w_{ij}\|}) w_{ij}. \tag{35}$$

Here, m_i and I_i represent the mass and moment of inertia of element i, respectively, and μ_s and μ_r denote the coefficients of sliding and rotation friction. For external collisions, $F_n = k_c g(p_i) \nabla g(p_i)$, and for self-collisions, $F_n = k_{ec} |\|p_i - p_j\| - r_i - r_j| \frac{p_i - p_j}{\|p_i - p_j\|}$. Finally, we project the modifications onto the direction of v_i and w_i, and ensure that the magnitude of the modification does not exceed the modulus of v_i and w_i. Therefore, the velocity and angular velocity can be modified to zero at most. The explicit friction treatment method is evidently not sufficiently precise. In our experiments, we observed that different time step selections necessitated the use of varying friction coefficients to achieve similar visual outcomes. Smaller time steps required a reduction in the friction coefficient, while larger time steps demanded an increase. However, since the friction effect is not the focus of this paper, we intend to employ a more accurate model in future work to obviate the need for such parameter adjustments.

6 Results and Discussion

The performance of the proposed approach is evaluated through a series of experiments. The simulations were conducted using an Intel i7 13700K processor, and detailed experimental setups and time costs are presented in Table 1. In all experiments, we have employed the conventional particle fluid surface method for the reconstruction and rendering of fracture surfaces.

6.1 Comparison with Rotation Vector Representation

In the field of rigid body motion simulation, the rotation vector representation is commonly employed to compute the rotational degrees of freedom. Compared to our method, the rotation vector also upholds a computational size of six degrees of freedom. We have conducted experimental comparisons between our motion representation and the rotation vector-based motion formulation from ADD [Geilinger et al. 2020]. Detailed experimental results are presented in Figure 6. Utilizing the same BDEM Potential, we configured an identical three-point beam bending experiment and set an equivalent dimensionless convergence threshold. Our experiments revealed that the simulation outcomes were indistinguishable between the two methods; however, our algorithm achieved faster convergence than the ADD approach. Theoretically, the rotation vector representation introduces no constraints, suggesting that our manifold-based search algorithm should exhibit a similar convergence rate as the rotation vector. We preliminarily attribute the enhanced performance of our algorithm to two factors: firstly, the quaternion representation, which offers superior spatial interpolation properties favorable for search, and secondly, unlike our energy minimization-based method, the ADD approach requires the application of the Gauss-Newton method to solve the motion equations, which may impose limitations on the convergence efficiency to a certain extent.

6.2 Comparison with Other Optimization Methods

To examine the performance of our proposed optimization approach, several experiments were conducted to assess its outcomes against commonly used constrained optimization methods. The results of these experiments are presented in Figure 7, which shows our proposed approach outperforms commonly used constrained optimization methods.

Penalty Method. The penalty method uses an external potential to convert a constrained optimization problem to an unconstrained optimization problem:

$$\arg\min_{\boldsymbol{x}} \Phi(\boldsymbol{x}) + \frac{1}{2}\kappa\|C(\boldsymbol{x})\|^2. \tag{36}$$

This method is commonly used in constrained optimization due to its simplicity. However, its shortcoming is that indefinite stiffness is required to satisfy constraints, and the introduced external stiffness κ can make the entire system an ill-defined problem. In our experiments, we found that the penalty method failed to converge or converged very slowly in BDEM fracture simulations.

Lagrange Multiplier. The Lagrange multiplier adds external variables λ to the system to convert the constrained optimization problem to an unconstrained one:

$$\arg\min_{\boldsymbol{x},\lambda} \Phi(\boldsymbol{x}) + \lambda^T C(x). \tag{37}$$

One drawback of this method is the increase in the number of variables and computational cost. The actual DoFs of rotation is 3, but the use of quaternions increases the number of unknowns to 4. The addition of multipliers further increases the number of variables to 5. As the number of unknowns increases, the computational cost also

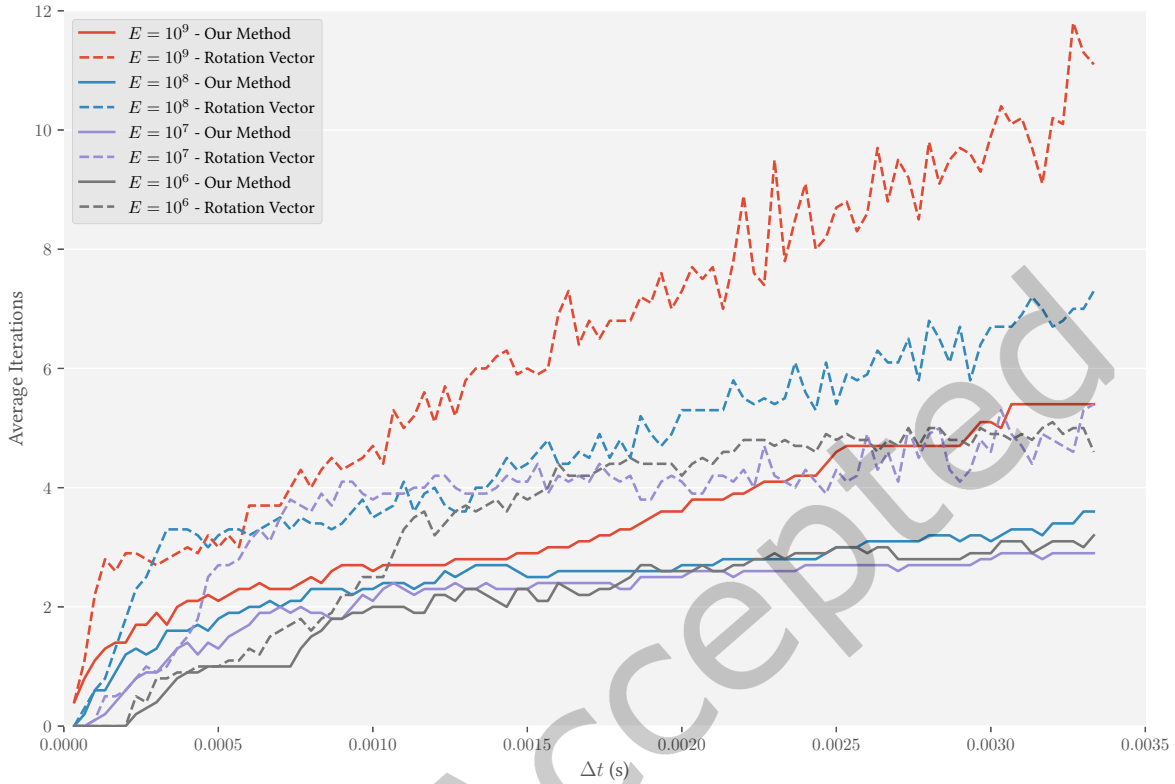

Fig. 6. We conducted a three-point beam bending experiment with identical settings for our method and rotation-vector formulation in [Geilinger et al. 2020]. The horizontal axis represents the time step size used in the simulation, with curves of different colors indicating various material Young's moduli. The vertical axis denotes the average number of iterations required during the simulation process. The dashed lines in the figure represent the rotation vector formulation, while the solid lines represent our approach. It is observable that, across different time steps and stiffness coefficients, our method needs fewer iterations to achieve the same convergence criteria.

increases. In our experiments, we found that the convergence rate was even slower than that of the first-order manifold optimizer.

Augmented Lagrangian. The augmented Lagrangian method can be seen as a combination of the penalty method and the Lagrange multiplier method, and it turns a constrained optimization problem into an unconstrained one.

$$\arg\min_{\boldsymbol{x},\lambda} \Phi(\boldsymbol{x}) + \lambda^T C(x) + \frac{1}{2}\kappa\|C(\boldsymbol{x})\|^2. \tag{38}$$

The external stiffness does not need to be infinite for the augmented Lagrangian method. However, in our experiments, the convergence rate was still lower than that of the manifold optimization method.

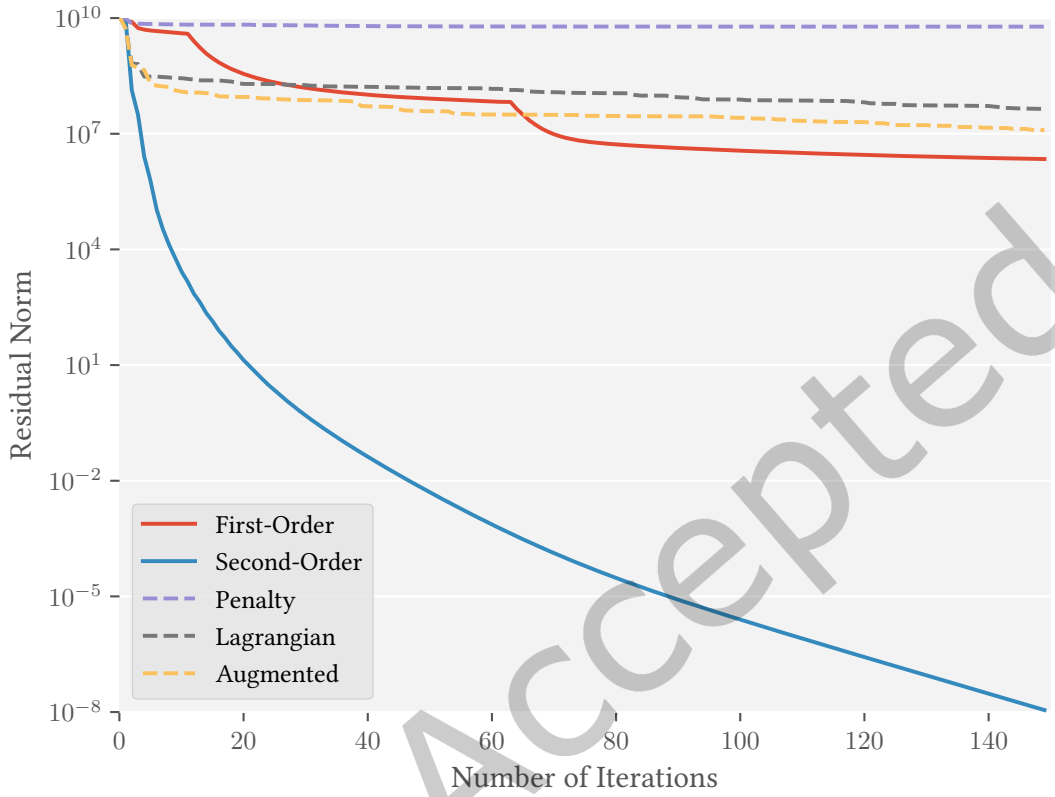

Fig. 7. Convergence rate comparison. The results show that the first-order manifold optimization and the second-order manifold optimization are both faster than the conventional constrained optimization methods, including the penalty method and the Lagrange multiplier method.

6.3 Comparison with FEM and MPM

FEM and MPM are two prevalent techniques for simulating fragmentation. FEM has been utilized in real-time simulations [O'Brien and Hodgins 1999; Parker and O'Brien 2009], while MPM can simulate fracture through the continuum damage model [Fan et al. 2022; Wolper et al. 2019]. We have implemented the methodologies presented in [Parker and O'Brien 2009] and [Wolper et al. 2019] as benchmarks for FEM and MPM, respectively, and conducted two experiments for a detailed comparison: the three-point bending test and the chocolate falling test. The three-point bending experiment highlights the differences in scale consistency among the three methods across various scales, and the chocolate falling test illustrates the disparities of fragmentation under high-velocity impacts.

To ensure a fair and comprehensive comparison, we provide implementation details of the baseline methods used in our experiments. For the MPM-based approach, we utilize the open-source phase field fracture model presented in [Wolper et al. 2019]. Our FEM implementation is based on the method described in [Parker and

Fig. 8. This figure, from left to right, demonstrates the fracture performance of FEM, MPM, and our method under the high-speed drop impact of a chocolate object. The FEM resulted in an excessive fragmentation into minute pieces, while the MPM failed to produce a physically plausible fragmentation. In contrast, BDEM accurately fragmented into a variety of piece sizes, effectively capturing the desired material failure behavior.

O'Brien 2009], with several modifications to enhance accuracy while maintaining computational efficiency. We reintroduce the residual propagation and sequential separation updating techniques from [O'Brien and Hodgins 1999] to achieve more precise fracture propagation. We remove the upper limit on propagation steps, allowing for a more faithful representation of the fracture process. Additionally, we eliminate the minimum fracture piece size constraint of three elements to fully capture the algorithm's behavior. To maintain physical plausibility, we implement an additional check to remove connections with floppy hinge or ball joints, as suggested in [Parker and O'Brien 2009]. This modified FEM implementation can be viewed as a simplified version of [O'Brien and Hodgins 1999], with the exclusion of the remeshing process for individual tetrahedra. This strategic simplification prevents an increase in tetrahedron count during simulation while still capturing key aspects of fracture behavior. The resulting baseline offers a balance between computational efficiency and physical fidelity.

Three Point Bending. Scale consistency is crucial for artists during the rapid prototyping phase of simulation, as it allows them to experiment with parameters at a smaller scale, thereby avoiding the time-consuming iteration associated with large-scale simulations. To analyze the scale consistency of our method in comparison to FEM and MPM, we conducted a three-point bending experiment. This test is a standard for analyzing material properties, where the relationship between beam fracture and the applied load reveals the intrinsic characteristics of the material. As depicted in the Figure 5, we selected discrete representations of the same material for the three methods across three different scales. FEM exhibits markedly different fracture events across scales. MPM demonstrates similar fracture behavior at medium and high scales but fails to accurately simulate fracture at smaller scales, with varying fracture initiation frames across resolutions (see Table 2). In contrast, our method maintains consistent fracture characteristics across all scales, demonstrating superior scale consistency. This consistency in our method arises from the discrete nature of fracture phenomena, clearly evident in the comparison experiment. Continuum approaches like FEM and MPM often struggle to accurately model small crack patterns due to insufficient local discretization, leading to resolution-dependent fracture initiation times. As fracture originates from small cracks, these continuum approaches may fail to capture the full complexity of the fracture process, resulting in resolution-dependent crack patterns. Our discrete method, however, inherently captures these small-scale interactions, yielding more consistent behavior across different scales.

Chocolate Falling. The chocolate falling experiment was primarily designed to demonstrate the distinctions in simulation outcomes among the three methods under high-velocity impact conditions. In the experiment,

the chocolate was consistently dropped onto a floor surface at a velocity of $5m/s$. The fracture results of the three methods are presented in Figure 8. The baseline FEM exhibits severe "shattering" artifacts. Extensive parameter tuning yields either numerous tiny fragments or a few large pieces, failing to produce the desired intermediate fragmentation pattern with diverse fragment sizes. Figure 9 presents a detailed visualization of the damage phase field in MPM. While the MPM method captures the continuum damage of materials, it has failed to demonstrate a plausible fragmentation process. This limitation stems from the constraints of the phase field model, which necessitates the introduction of additional artificial rules and parameters for artistic control over cracking and fragmentation within this scenario [Fan et al. 2022; Wolper et al. 2019]. While it is conceivable that the introduction of further supplementary rules might enhance the simulation outcomes for this particular case, it is clear that such modifications are extrinsic to the material's fundamental properties. In contrast, our method necessitates only the specification of material parameters to simulate the fracture phenomenon, thus achieving a realistic and authentic fragmentation effect.

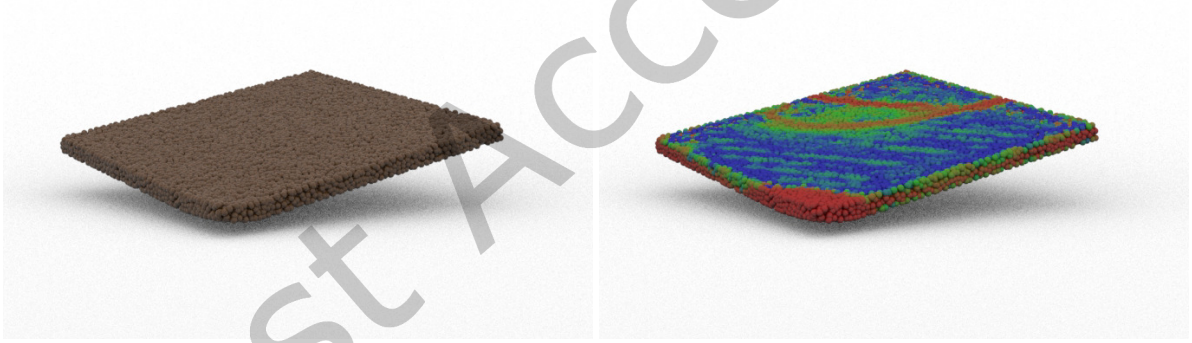

Fig. 9. This figure provides a visualization of the damage profile with MPM. It is evident that, with the correct material strength parameters applied, the chocolate material exhibits damage effects under high-velocity impact conditions. However, this setting does not precipitate fracture, necessitating the introduction of further rules and controls to direct the fragmentation process. This requirement highlights a limitation inherent to such continuum-based methods, where additional rules are needed to simulate realistic behavior.

Table 2 details the computational time costs and settings for both experiments. The data reveals that BDEM achieves more consistent fracture frames across resolutions compared to FEM and MPM, indicating fracture occurs at similar deformation states and demonstrating superior scale consistency.

Per-frame computational costs indicate FEM is the slowest method, requiring residual propagation for accurate crack patterns, resulting in more iterations per step. While MPM has lower per-frame computational costs than BDEM, its inferior scale consistency necessitates extensive parameter tuning of material strength and grid resolution to achieve satisfactory fracture results. In contrast, BDEM simplifies this process significantly: suitable

Table 2. The simulation statistics for our experiments are listed below, with the slow-motion of 1x corresponding to one frame per 1/60th of a second, and 0.125x corresponding to one frame per 1/480th of a second. The time step and time consumption for the proposed approach are compared with explicit and implicit methods at different element numbers and stiffness settings.

Scene	Method	# points	Δt(s)	E	ν	τ	time per frame (s)	fracture frame
Three Point Bending (Low)	FEM	578	4.2E-5	1E7	0.3	300	6.2	6
	MPM	500	4.2E-5	1E7	0.3	3E4	0.6	26
	Ours	532	2.0E-5	1E7	0.3	3E4	0.9	23
Three Point Bending (Middle)	FEM	1.7K	2.8E-5	1E7	0.3	300	57.6	13
	MPM	2K	2.8E-5	1E7	0.3	3E4	5.2	23
	Ours	1.8K	9.1E-6	1E7	0.3	3E4	7.3	25
Three Point Bending (High)	FEM	16.4K	2.0E-5	1E7	0.3	300	1240	29
	MPM	20K	2.1E-5	1E7	0.3	3E4	48.2	11
	Ours	18K	5.8E-5	1E7	0.3	3E4	137.1	25
Chocolate Falling (0.125x slow motion)	FEM	13.6K	5.2E-6	1E8	0.3	100	650.1	-
	MPM	20K	4.2E-6	1E8	0.3	5E4	24.5	-
	Ours	22K	4.4E-5	1E8	0.3	5E4	106.8	-

material strength parameters can be rapidly determined at low resolution and then applied directly to medium and high-resolution simulations with consistent results.

Analysis of fracture thresholds reveals an intriguing discrepancy. Despite all three methods employing thresholds with identical physical dimensions, the numerical value for the FEM threshold diverges significantly from the other two approaches. Notably, this smaller threshold value in FEM aligns with parameters in the original work by O'Brien and Hodgins [1999], while BDEM and MPM thresholds more closely approximate real-world material strengths.

6.4 Comparison with Explicit Method

Beam Stretch & Twist. In our experiments, we compared the simulation results of our implicit method with the explicit method proposed by [Lu et al. 2021]. Specifically, we evaluated their performance in beam stretching and twisting scenarios, and found that both methods produced identical results under the same settings.

Beam Drape. This experiment investigates and compares the simulation time of the implicit and explicit simulation methods using the beam drape scenario with different stiffness levels. Two curves are plotted in Figure 11 to examine the relationship between simulation time and stiffness, as well as simulation time and scale. Our results show that the implicit method outperforms the explicit method by 3 to 6 times in terms of run time.

Chocolate. In this experiment, the deformation of a chocolate drop with high stiffness is simulated to evaluate the computational efficiency between implicit and explicit simulation methods in rich contact scenarios. The outcomes indicate that the implicit method outperforms the explicit method by achieving a 2.5x speedup. The visualization of the simulation can be inspected in Figure 2.

Plate. This experiment compares the computational efficiency of implicit and explicit methods for simulating the fracture of a plate with high stiffness upon impact with the ground. Our method has demonstrated a 2.4x increase in computational speed relative to the explicit method under these conditions. The simulation results are illustrated in Figure 3.

Cloth. Here, we examine the computational efficiency of implicit versus explicit methods in the context of yarn-level cloth simulation, characterized by high tensile and low bending stiffness of the yarns. Our method

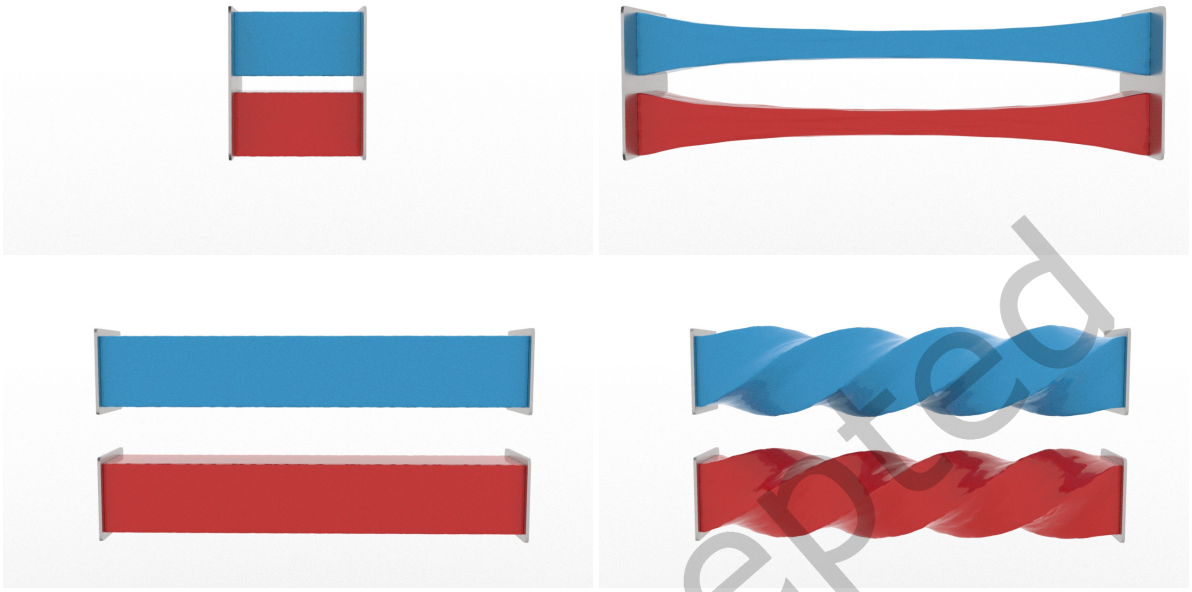

Fig. 10. Comparison of explicit (blue) and implicit (red) methods in simulating a stretched and twisted beam. Both approaches yield similar results.

provides a 3.3x computational speed advantage over the explicit approach under these circumstances. The dynamics of the cloth tearing, as simulated, are depicted in Figure 4.

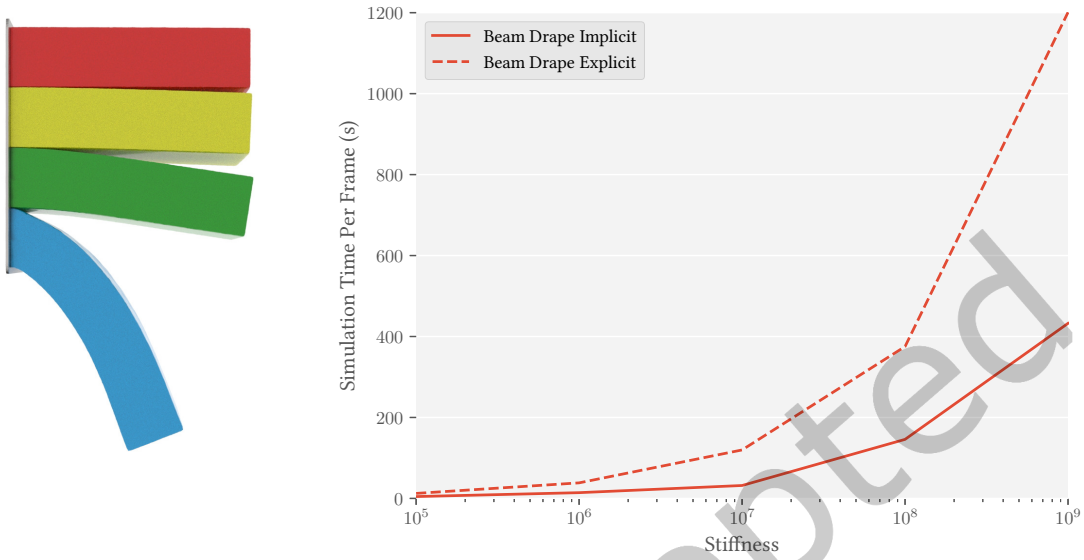

Fig. 11. This experiment compares explicit and implicit methods for simulating beam drape. The left images illustrate static drape scenarios with varying material stiffness: red (10^9), yellow (10^8), green (10^7), and blue (10^6). The curves in the right image plot the relationship between computational cost and material stiffness, demonstrating that implicit methods produce faster results for a range of stiffness values.

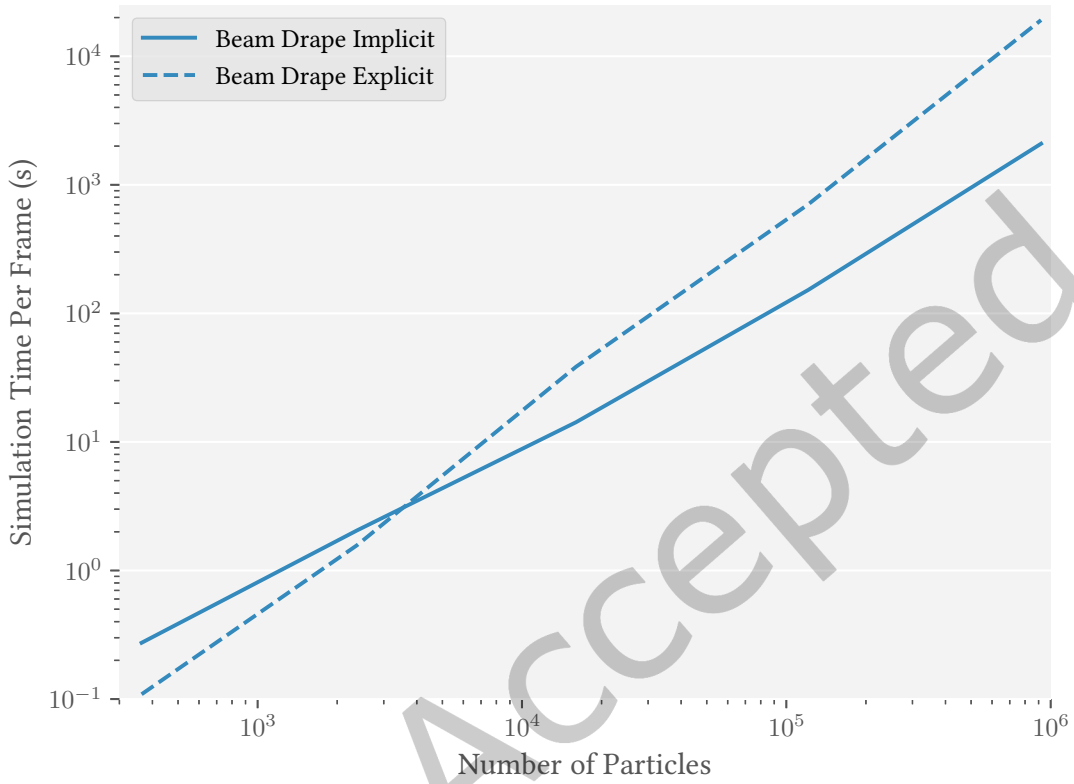

Fig. 12. This experiment compares explicit and implicit methods for simulating beam drape. The curves plot the relationship between the computational cost and simulated discrete elements, showing implicit methods significantly outperform explicit methods for finer simulation scales.

7 Conclusion

We present an optimization-based implicit integrator for BDEM system, and introduce a manifold optimization scheme to transform the nonlinear dynamic simulation into an unconstrained optimization problem on a spherical manifold, which significantly improves the solution stability and efficiency. In addition, a nullspace operator is proposed, which simplifies the optimization procedure and reduces the number of unknowns. Additionally, our new implicit BDEM approach uses the semi-positive definite projection method with preconditioned conjugate gradient and line search method on the manifold, which achieves a stable integrator. In our experiments, we have showcased that our method surpasses the explicit BDEM in terms of computational speed. Furthermore, when compared to FEM and MPM for fragmentation, our approach exhibits superior scale consistency, accurately simulating fracture effects at smaller scales. The simulation of fragmentation is more realistic and natural with our method, which necessitates only the provision of material parameters to get the authentic fracture process.

While our approach offers notable advancements, it is not without its limitations. Specifically, to mitigate complexities at the contribution points, we have utilized a rudimentary collision model for collision management. This simplification may lead to a diminished fidelity in the representation of collision effects. In future work, we plan to integrate more sophisticated collision models into our optimization framework to enhance the accuracy and realism of our simulations. Furthermore, the BDEM system we have developed is highly amenable to parallelization. Although our current exposition does not delve into the details of GPU acceleration for the sake of a clear computational comparison, we intend to explore a more detailed implementation of parallel acceleration in our methods moving forward. Additionally, while our approach successfully captures fine-grained collision details, the reconstruction and rendering of fracture surfaces also play a crucial role in the final visual outcomes. We aim to explore fracture surface reconstruction methods more tailored to the nuances of BDEM in our ongoing research.

References

Iván Alduán, Angel Tena, and Miguel A Otaduy. 2009. Simulation of High-Resolution Granular Media.. In *CEIG*. 11–18.

Damien André, Ivan Iordanoff, Jean-luc Charles, and Jérôme Néauport. 2012. Discrete element method to simulate continuous material by using the cohesive beam model. *Computer methods in applied mechanics and engineering* 213 (2012), 113–125.

Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. 2007. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (2007), 370–378.

David Baraff. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 223–232.

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH 1998*. 43–54.

Nathan Bell, Yizhou Yu, and Peter J Mucha. 2005. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 77–86.

Jan Bender, Kenny Erleben, and Jeff Trinkle. 2014. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 246–270.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*. 1–12.

Peter Betsch and Ralf Siebert. 2009. Rigid body dynamics in terms of quaternions: Hamiltonian formulation and conserving numerical integration. *International journal for numerical methods in engineering* 79, 4 (2009), 444–473.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics. *ACM Transactions on Graphics* 33, 4 (2014), 1–11.

Nicolas Boumal. 2022. An introduction to optimization on smooth manifolds. To appear with Cambridge University Press. https://www.nicolasboumal.net/book

Wei Chen, Fei Zhu, Jing Zhao, Sheng Li, and Guoping Wang. 2018. Peridynamics-Based Fracture Animation for Elastoplastic Solids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 112–124.

Yu Ju Chen, Uri M Ascher, and Dinesh K Pai. 2017. Exponential rosenbrock-euler integrators for elastodynamic simulation. *IEEE Transactions on Visualization and Computer Graphics* 24, 10 (2017), 2702–2713.

Zhili Chen, Miaojun Yao, Renguo Feng, and Huamin Wang. 2014. Physics-inspired adaptive fracture refinement. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–7.

Floyd M Chitalu, Qinghai Miao, Kartic Subr, and Taku Komura. 2020. Displacement-Correlated XFEM for Simulating Brittle Fracture. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 569–583.

P. A. Cundall. 1971. A Computer Model for Simulating Progressive, Large-scale Movement in Blocky Rock System. *Proceedings of the International Symposium on Rock Mechanics* (1971).

P. A. Cundall and O. D. L. Strack. 1979. A discrete numerical model for granular assemblies. *Géotechnique* 29, 1 (1979), 47–65.

David N De Klerk, Thomas Shire, Zhiwei Gao, Andrew T McBride, Christopher J Pearce, and Paul Steinmann. 2022. A variational integrator for the Discrete Element Method. *J. Comput. Phys.* 462 (2022), 111253.

Bernhard Eberhardt, Olaf Etzmuß, and Michael Hauth. 2000. Implicit-explicit schemes for fast animation with particle systems. In *Computer Animation and Simulation 2000: Proceedings of the Eurographics Workshop in Interlaken, Switzerland, August 21–22, 2000*. Springer, 137–151.

Linxu Fan, Floyd M Chitalu, and Taku Komura. 2022. Simulating brittle fracture with material points. *ACM Transactions on Graphics (TOG)* 41, 5 (2022), 1–20.

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Transactions on Graphics* 40, 4 (2021).

Marco Fratarcangeli, Valentina Tibaldo, Fabio Pellacini, et al. 2016. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph.* 35, 6 (2016), 214–1.

Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. 2015. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics* 21, 10 (2015), 1103–1115.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

David Hahn and Chris Wojtan. 2015. High-resolution brittle fracture simulation with boundary elements. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

David Hahn and Chris Wojtan. 2016. Fast approximations for boundary element based brittle fracture simulation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

James K Hahn. 1988. Realistic animation of rigid bodies. *ACM Siggraph computer graphics* 22, 4 (1988), 299–308.

Jeffrey Hellrung, Andrew Selle, Arthur Shek, Eftychios Sifakis, and Joseph Teran. 2009. Geometric fracture modeling in Bolt. In *SIGGRAPH 2009: Talks*. 1–1.

Gentaro Hirota, Susan Fisher, A State, Chris Lee, and Henry Fuchs. 2001. An implicit finite element method for elastic solids in contact. In *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No. 01TH8596)*. IEEE, 136–254.

Couro Kane, Jerrold E Marsden, Michael Ortiz, and Matthew West. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *International Journal for numerical methods in engineering* 49, 10 (2000), 1295–1325.

Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J Guy. 2017. Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. 2009. Enrichment textures for detailed cutting of shells. In *ACM SIGGRAPH 2009 papers*. 1–10.

Min Hyung Kee, Kiwon Um, HyunMo Kang, and JungHyun Han. 2023. An Optimization-based SPH Solver for Simulation of Hyperelastic Solids. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, 225–233.

Dan Koschier, Jan Bender, and Nils Thuerey. 2017. Robust eXtended finite elements for complex cutting of deformables. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Dan Koschier, Sebastian Lipponer, and Jan Bender. 2014. Adaptive Tetrahedral Meshes for Brittle Fracture Simulation.. In *Symposium on Computer Animation*, Vol. 3.

Tassilo Kugelstadt and Elmar Schömer. 2016. Position and orientation based Cosserat rods.. In *Symposium on Computer Animation*. 169–178.

Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: accelerated incremental potential contact with medial elastics. *ACM Transactions on Graphics* 40, 4 (2021).

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020a. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (2020), 49.

Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M Kaufman. 2019. Decomposed optimization time integrator for large-step elastodynamics. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–10.

Minchen Li, Danny M Kaufman, and Chenfanfu Jiang. 2020b. Codimensional incremental potential contact. *arXiv preprint arXiv:2012.04457* (2020).

C Karen Liu and Sumit Jain. 2012. A quick tutorial on multibody dynamics. *Online tutorial, June* (2012), 7.

Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–7.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2016. Towards real-time simulation of hyperelastic materials. *arXiv preprint arXiv:1604.07378* (2016).

Fabian Löschner, Andreas Longva, Stefan Jeske, Tassilo Kugelstadt, and Jan Bender. 2020. Higher-order time integration for deformable solids. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 157–169.

Jia-Ming Lu, Chen-Feng Li, Geng-Chen Cao, and Shi-Min Hu. 2021. Simulating Fractures With Bonded Discrete Element Method. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 4810–4824.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. In *ACM SIGGRAPH 2011 papers*. 1–8.

Dominik L Michels, Gerrit A Sobottka, and Andreas G Weber. 2014. Exponential integrators for stiff elastodynamic problems. *ACM Transactions on Graphics (TOG)* 33, 1 (2014), 1–20.

Neil Molino, Zhaosheng Bao, and Ron Fedkiw. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 385–392.

Matthias Müller and Nuttapong Chentanez. 2011. Solid simulation with oriented particles. In *ACM SIGGRAPH 2011 papers*. 1–10.

Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2013. Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Matthias Müller, Miles Macklin, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Detailed rigid body simulation with extended position based dynamics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 101–112.

Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Computer Animation and Simulation 2001*. 113–124.

Rahul Narain, Matthew Overby, and George E Brown. 2016. ADMM⊆ projective dynamics: fast simulation of general constitutive models.. In *Symposium on Computer Animation*, Vol. 1. 2016.

Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM SIGGRAPH 2009 papers*. 1–9.

Vinh DX Nguyen, A Kiet Tieu, Damien André, Lihong Su, and Hongtao Zhu. 2021. Discrete element method using cohesive plastic beam for modeling elasto-plastic deformation of ductile materials. *Computational Particle Mechanics* 8 (2021), 437–457.

James F. O'Brien and Jessica K. Hodgins. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of ACM SIGGRAPH 1999*. 137–146.

Eric G Parker and James F O'Brien. 2009. Real-time deformation and fracture in a game environment. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 165–175.

Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J Guibas. 2005. Meshless animation of fracturing solids. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 957–964.

Tobias Pfaff, Rahul Narain, Juan Miguel De Joya, and James F O'Brien. 2014. Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–9.

David O Potyondy and PA Cundall. 2004. A bonded-particle model for rock. *International journal of rock mechanics and mining sciences* 41, 8 (2004), 1329–1364.

Saty Raghavachary. 2002. Fracture generation on polygonal meshes using Voronoi polygons. In *ACM SIGGRAPH 2002 conference abstracts and applications*. 187–187.

Witawat Rungjiratananon, Zoltan Szego, Yoshihiro Kanamori, and Tomoyuki Nishita. 2008. Real-time animation of sand-water interaction. In *Computer Graphics Forum*, Vol. 27. 1887–1893.

Sara C Schvartzman and Miguel A Otaduy. 2014. Fracture animation based on high-dimensional voronoi diagrams. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 15–22.

Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 73–80.

Carlota Soler, Tobias Martin, and Olga Sorkine-Hornung. 2018. Cosserat rods with projective dynamics. In *Computer Graphics Forum*, Vol. 37. 137–147.

Ari Stern and Eitan Grinspun. 2009. Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation* 7, 4 (2009), 1779–1794.

David E Stewart. 2000. Rigid-body dynamics with friction and impact. *SIAM review* 42, 1 (2000), 3–39.

Jonathan Su, Craig Schroeder, and Ronald Fedkiw. 2009. Energy stability and fracture for frame rate rigid body simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 155–164.

Pascal Volino and Nadia Magnenat-Thalmann. 2001. Comparing efficiency of integration methods for cloth simulation. In *Proceedings. Computer Graphics International 2001*. IEEE, 265–272.

Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–10.

Stephanie Wang, Mengyuan Ding, Theodore F Gast, Leyi Zhu, Steven Gagniere, Chenfanfu Jiang, and Joseph M Teran. 2019. Simulation and visualization of ductile fracture with the material point method. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), 1–20.

Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games*. 79–84.

Martin Wicke, Daniel Ritchie, Bryan M Klingner, Sebastian Burke, Jonathan R Shewchuk, and James F O'Brien. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on graphics (TOG)* 29, 4 (2010), 1–11.

Chris Wojtan and Greg Turk. 2008. Fast viscoelastic behavior with thin features. In *ACM SIGGRAPH 2008 papers*. 1–8.

Joshuah Wolper, Yunuo Chen, Minchen Li, Yu Fang, Ziyin Qu, Jiecong Lu, Meggie Cheng, and Chenfanfu Jiang. 2020. Anisompm: Animating anisotropic damage mechanics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 37–1.

Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. 2019. CD-MPM: continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–15.

Yonghao Yue, Breannan Smith, Peter Yichen Chen, Maytee Chantharayukhonthorn, Ken Kamrin, and Eitan Grinspun. 2018. Hybrid grains: adaptive coupling of discrete and continuum simulations of granular media. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–19.

Changxi Zheng and Doug L James. 2010. Rigid-body fracture sound with precomputed soundbanks. In *ACM SIGGRAPH 2010 papers*. 1–13.

Yufeng Zhu, Robert Bridson, and Chen Greif. 2015. Simulating rigid body fracture with surface meshes. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

A Quaternion Operations

Quaternions are generally represented in the following form,

$$q = u\mathbf{i} + v\mathbf{j} + w\mathbf{k} + s = \{u, v, w, s\} = \begin{pmatrix} u \\ v \\ w \\ s \end{pmatrix}, \tag{39}$$

where u, v, w and s are real numbers, and 1, $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the basis vectors or basis elements. A unit quaternion can denote a rotation in three-dimensional space. Compared to other forms of rotation representations such as Euler angles, axis-angle and rotation matrices, quaternions have advantages in computational simplicity and efficiency. Hence, quaternions are widely used in computer graphics to represent rotation, including BDEM [Lu et al. 2021] where they are used to describe the rotational status of discrete elements. As quaternions play a key role in formulating the proposed implicit BDEM, we briefly recap the key information related to quaternion operations in this section.

Quaternion Multiplication. The multiplication of quaternions represents the composition of rotations: the left multiplication represents the order of rotations, and the right multiplication represents the subsequent rotation of the coordinate system. Rotations can also be represented in the matrix form with quaternions:

$$q_i q_j = Q_l(q_i) q_j = Q_r(q_j) q_i, \tag{40}$$

where $Q_l(q)$ is called the left multiplication matrix and $Q_r(q)$ the right multiplication matrix. Let vector $t = \{u, v, w\}$ denote the imaginary part of quaternion $\{u, v, w, s\}$, the left and right multiplication matrices can be expressed as follows:

$$Q_l(q) = \begin{pmatrix} sI_3 + t^\times & t \\ -t^T & s \end{pmatrix} \tag{41}$$

$$Q_r(q) = \begin{pmatrix} sI_3 - t^\times & t \\ -t^T & s \end{pmatrix}, \tag{42}$$

where the skew symmetric matrix $t^\times$ is the cross matrix of vector t

$$t^\times = \begin{pmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{pmatrix}. \tag{43}$$

Direction Rotation Operator. Let d_0 denote a direction in the 3D space, the new direction d after a rotation can be expressed as:

$$\begin{pmatrix} d \\ 0 \end{pmatrix} = q \begin{pmatrix} d_0 \\ 0 \end{pmatrix} \overline{q}, \tag{44}$$

where quaternion q denotes the rotation and $\overline{q} = \{-u, -v, -w, s\}$ is its conjugate. The above rotation operation can be rewritten as:

$$d = q \odot d_0, \tag{45}$$

where $\odot$ is the direction rotation operator.

Nullspace Operator. The composition of a quaternion its and conjugate is the unit quaternion, i.e. $\overline{q}q = \{0, 0, 0, 1\}$. In the matrix form, this becomes

$$\overline{q}q = Q_l(\overline{q})q = \begin{pmatrix} sI_3 - t^\times & -t \\ t^T & s \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{46}$$

We can define the upper matrix of $Q_l(\overline{q})$ as a nullspace operator:

$$G_l(q) = \begin{pmatrix} sI_3 - t^\times & -t \end{pmatrix}, G_r(q) = \begin{pmatrix} sI_3 + t^\times & -t \end{pmatrix}. \tag{47}$$

The nullspace operator can be used to reduce the computational cost of quaternion systems and the detailed strategy is provided in Section 4.2. With the nullspace operator, the following relations hold:

$$G_l(q)q = 0, G_r(q)q = 0, \tag{48}$$

and

$$\begin{aligned} G_l(q)^T G_l(q) &= |q|^2 I_4 - qq^T, \\ G_l(q)G_l(q)^T &= |q|^2 I_3. \end{aligned} \tag{49}$$

The multiplication $G_l(q_i)q_j$ removes the component of q_i in q_j and projects the vector to the direction of angle axis of quaternion $\overline{q_i}q_j$. The quaternion matrix can also be related to the nullspace operator matrix as

$$Q_l(q) = \begin{pmatrix} G_l(q)^T & q \end{pmatrix}, Q_r(q) = \begin{pmatrix} G_r(q)^T & q \end{pmatrix}. \tag{50}$$

Derivation and Angular Velocity. Let w_4 denote the angular velocity in four dimensions:

$$w_4 = \begin{pmatrix} w \\ 0 \end{pmatrix}. \tag{51}$$

The derivative of a quaternion and the extended angular velocity w_4 satisfy the following relations [Betsch and Siebert 2009]:

$$\dot{q} = \frac{1}{2} w_4 q, \tag{52}$$

$$w_4 = 2\dot{q}\overline{q}. \tag{53}$$

Quaternion Constraint. A rotation can be represented by a unit quaternion. To ensure the quaternion remains as a unit during calculations, the unit constraint must be applied:

$$C(q) = \frac{1}{2}(q^T q - 1) = 0. \tag{54}$$

B Hamiltonian System with Backward Euler Formulation

We present a derivation of the kinetic equation for rotational systems using a conventional approach: first establishing the continuous formulation, then applying a time discretization method to obtain the integrator form. As described by [Betsch and Siebert 2009], the continuous form of the Hamiltonian system equation is given by:

$$\begin{aligned}\dot{q} &= \frac{1}{4}(\mathbb{J}_0^{-1}(q\cdot p)q + G(q)^T J^{-1} G(q)p),\\ \dot{p} &= -\frac{1}{4}(\mathbb{J}_0^{-1}(q\cdot p)p + G(p)^T J^{-1} G(p)q) - \nabla V(q),\end{aligned} \tag{55}$$

where q represents the rotational state, p denotes the extended momentum, and $\mathbb{J}_0 = \mathrm{tr}(J)$. Given the quaternion unit length constraint with property $q \cdot p = 0$ [Betsch and Siebert 2009], we can simplify the equation to:

$$\begin{aligned}\dot{q} &= \frac{1}{4}(G(q)^T J^{-1} G(q)p)\\ \dot{p} &= -\frac{1}{4}(G(p)^T J^{-1} G(p)q) - \nabla V(q)\end{aligned} \tag{56}$$

For a spherical element, $J = \frac{2}{5}mr^2 I_3$. Let $\mathbb{I} = \frac{2}{5}mr^2$. Applying the result from Equation (49), we further simplify the equation:

$$\begin{aligned}\dot{q} &= \frac{1}{4\mathbb{I}}((I_4 - qq^T)p),\\ \dot{p} &= -\frac{1}{4\mathbb{I}}(|p|^2 I_4 - pp^T)q - \nabla V(q).\end{aligned} \tag{57}$$

Maintaining $q \cdot p = 0$, we obtain a more concise form:

$$\begin{aligned}\dot{q} &= \frac{1}{4\mathbb{I}}p\\ \dot{p} &= -\frac{1}{4\mathbb{I}}|p|^2 q - \nabla V(q)\end{aligned} \tag{58}$$

Integrating the first equation into the second and simplifying, we derive an equation solely in terms of q:

$$4\mathbb{I}(\ddot{q} + |\dot{q}|^2 q) + \nabla V(q) = 0 \tag{59}$$

Note the new term $|\dot{q}|^2 q$, which distinguishes this formulation from non-rotational cases. To align with our manifold optimization approach, we left-multiply by $(I_4 - qq^T)$. Since $(I_4 - qq^T)q = 0$, the residual on the manifold becomes:

$$(I_4 - qq^T)(4\mathbb{I}\ddot{q} + \nabla V(q)) = 0 \tag{60}$$

Observe that $4\mathbb{I}\ddot{q} + \nabla V(q)$ and $4\mathbb{I}(\ddot{q} + |\dot{q}|^2 q) + \nabla V(q)$ yield identical residuals on the manifold. Consequently, we can employ the simpler form, and discretization using backward Euler results in the same expression as in Equation (5).

C Discrete Lagrangian

Hamilton's principle is a fundamental relation in physics to describe the motion of a dynamic system. By taking the variation of the Lagrangian functional $S = \int_{t_0}^{t_1} L(x(t), \dot{x}(t))dt$, the equations of motion can be obtained as the stationary point of the action functional. For our system, the Lagrangian function given by the sum of the kinetic energy T and the potential energy V:

$$L(x, \dot{x}) = T(x, \dot{x}) - V(x, \dot{x}) = \frac{1}{2}\dot{x}^T M \dot{x} - V(x). \tag{61}$$

The true evolution of the physical system satisfies $\frac{\partial S}{\partial x(t)} = 0$. For quaternion-based Hamiltonian system, we can articulate the equations within which the extended kinetic energy form is derivable from [Betsch and Siebert 2009].

$$T(q, \dot{q}) = \dot{q}^T M_q(q)\dot{q}. \tag{62}$$

Here, for one element, $M_q(q) = 4G_l(q)JG_l(q)^T + 2\mathrm{tr}(J)qq^T$, J is the moment of inertia matrix and $G_l(q)$ is the nullspace operator in appendix A. Given that the mass matrix $M_q(q)$ varies with q, direct differentiation of the equations yields a form akin to the extended momentum in rigid body motion, which is complicated for the derivation of an incremental potential form. In this context, we employ the Discrete Lagrangian formulation to re-derive the equations and perform simplifications based on some special properties of the BDEM.

In a small time interval, the action functional can be approximated by a discrete Lagrangian:

$$S_d = \sum_{i=0}^{n-1} L_d(t_i, t_{i+1}, x_i, x_{i+1}) \approx \int_{t_0}^{t_n} L(t, x(t))dt. \tag{63}$$

At the stationary point of the above discrete action functional, each term of the sum must be set to zero:

$$\frac{\partial S_d}{\partial x_i} = 0 = \frac{\partial L_d(t_{i-1}, t_i, x_{i-1}, x_i)}{\partial x_i} + \frac{\partial L_d(t_i, t_{i+1}, x_i, x_{i+1})}{\partial x_i}. \tag{64}$$

Here, with mid-point approximation,

$$\begin{aligned} T_d(x_i, v_i) &= \Delta t T(\frac{x_{i+1} + x_i}{2}, \frac{x_{i+1} - x_i}{\Delta t}) \\ V_d(x_i) &= \Delta t V(\frac{x_{i+1} + x_i}{2}). \end{aligned} \tag{65}$$

While this formulation is general for any element shape, our BDEM system employs spherical discrete elements exclusively. This choice allows us to leverage the symmetry properties of spheres, significantly simplifying the computational process. It's important to note that these optimizations are specific to spherical elements and do not apply to other geometries.

For a sphere with mass m and radius r, the mass matrix $M_p = mI_3$, whilst its moment of inertia is $J = \frac{2}{5}mr^2I_3$, here I_3 stands for the 3×3 identity matrix. With vector p denotes its position and quaternion q its rotation, the discrete energy term for position, can be easily computed with linear velocity approximation:

$$T_d^p = \Delta t \frac{1}{2}(\frac{p_{i+1} - p_i}{\Delta t})^T mI_3(\frac{p_{i+1} - p_i}{\Delta t}). \tag{66}$$

For rotation, we approximate the mass matrix and velocity by a mid-point approximation:

$$T_d^q = \Delta t \frac{1}{2}(\frac{q_{i+1} - q_i}{\Delta t})^T M_q(\frac{q_{i+1} + q_i}{2})(\frac{q_{i+1} - q_i}{\Delta t}). \tag{67}$$

With the definition of extended matrix and moment of inertia for sphere, we can get:

$$\begin{aligned} M_q(q) &= 4G(q)JG(q)^T + 2\mathrm{tr}(J)qq^T \\ &= \frac{8}{5}mr^2(I_4 - qq^T) + \frac{12}{5}mr^2qq^T \\ &= \frac{8}{5}mr^2(I_4 + \frac{1}{2}qq^T), \end{aligned} \tag{68}$$

here I_4 stands for the 4×4 identity matrix. With unit length property of quaternion during our simulation, we have $(q_{i+1} + q_i)^T(q_{i+1} - q_i) = q_{i+1}^T q_{i+1} - q_i^T q_i = 0$. With the mid-point mass matrix as in [Betsch and Siebert 2009],

the discrete kinetic energy is:

$$\begin{aligned} T_d^q &= \frac{1}{2\Delta t}(q_{i+1}-q_i)^T M_q(\frac{q_{i+1}+q_i}{2})(q_{i+1}-q_i) \\ &= \frac{1}{2\Delta t}(q_{i+1}-q_i)^T \frac{8}{5}mr^2(I_4+\frac{1}{2}\frac{q_{i+1}+q_i}{2}(\frac{q_{i+1}+q_i}{2})^T)(q_{i+1}-q_i) \\ &= \frac{1}{2\Delta t}(q_{i+1}-q_i)^T \frac{8}{5}mr^2 I_4(q_{i+1}-q_i). \end{aligned} \tag{69}$$

Now the mass matrix for quaternion becomes constant here. The extended mass matrix M_s is

$$M_s = \begin{pmatrix} mI_3 & \\ & \frac{8}{5}mr^2 I_4 \end{pmatrix}. \tag{70}$$

Concatenating positions and rotations for all elements as system state x, the discrete kinetic energy of the spherical element can be expressed as:

$$T_d = \frac{1}{2}\Delta t(\frac{x_{i+1}-x_i}{\Delta t})^T M_s(\frac{x_{i+1}-x_i}{\Delta t}), \tag{71}$$

The discrete Lagrangian can be computed by

$$L_d(t_i, t_{i+1}, x_i, x_{i+1}) = \Delta t\frac{1}{2}(\frac{x_{i+1}-x_i}{\Delta t})^T M_s(\frac{x_{i+1}-x_i}{\Delta t}) - \Delta t V(\frac{x_{i+1}+x_i}{2}), \tag{72}$$

the terms in Eq. (64) can be expressed as:

$$\frac{\partial L_d(t_{i-1}, t_i, x_{i-1}, x_i)}{\partial x_i} = \frac{1}{\Delta t}M_s(x_i - x_{i-1}) - \frac{1}{2}\Delta t\nabla V(\frac{x_{x-1}+x_i}{2}), \tag{73}$$

$$\frac{\partial L_d(t_i, t_{i+1}, x_i, x_{i+1})}{\partial x_i} = \frac{1}{\Delta t}M_s(x_i - x_{i+1}) - \frac{1}{2}\Delta t\nabla V(\frac{x_i+x_{i+1}}{2}). \tag{74}$$

Let $\hat{x} = 2x_i - x_{i-1}$, the mid-point form of our variational integrator can be obtained as:

$$\frac{1}{\Delta t^2}M_s(x_{i+1}-\hat{x}) + \frac{1}{2}\nabla V(\frac{x_{i+1}+x_i}{2}) + \frac{1}{2}\nabla V(\frac{x_{i-1}+x_i}{2}) = 0. \tag{75}$$

Now we have developed a variational integrator for BDEM system simulations. The following part outlines the derivation of our optimization formulation, which forms the foundation of our subsequent optimization process. We introduce a new variable $\hat{z} = \hat{x} - \frac{1}{2}M_s^{-1}\Delta t^2\nabla V(\frac{x_{i-1}+x_i}{2})$, calculable from known previous states. This allows us to reformulate the equation as:

$$\frac{1}{\Delta t^2}M_s(x_{i+1}-\hat{z}) + \frac{1}{2}\nabla V(\frac{x_{i+1}+x_i}{2}) = 0. \tag{76}$$

Observing this equation, we can recast it as an optimization problem with the objective function:

$$\Phi(\boldsymbol{x}_{i+1}) = \frac{1}{2\Delta t^2}(x_{i+1}-\hat{z})^T M_s(x_{i+1}-\hat{z}) + V(\frac{x_{i+1}+x_i}{2}). \tag{77}$$

Received 21 August 2023; revised 15 October 2024; accepted 12 December 2024