

UC Berkeley

UC Berkeley Previously Published Works

Title

3-D impulse-based level-set method for granular flow modeling

Permalink

<https://escholarship.org/uc/item/5v1591xz>

Journal

International Journal for Numerical Methods in Engineering, 125(17)

ISSN

0029-5981

Authors

Tan, Peng

Wijesuriya, Hasitha S

Sitar, Nicholas

Publication Date

2024-09-15

DOI

10.1002/nme.7546

Peer reviewed

3-D impulse-based level-set method for granular flow modeling

Peng Tan | Hasitha S. Wijesuriya^{ORCID} | Nicholas Sitar^{ORCID}

Civil and Environmental Engineering,
UC Berkeley, Berkeley, California, USA

Correspondence

Nicholas Sitar, Civil and Environmental
Engineering, UC Berkeley, Berkeley,
CA 94720, USA.

Email: sitar@berkeley.edu

Abstract

We explore the viability of modeling dynamic problems with a new formulation of an impulse-based Level-Set DEM (LS-DEM). The new formulation is stable, fast, and energy conservative. However, it can be numerically stiff when the assembly has substantial mass differences between particles. We also demonstrate the feasibility of modeling deformable structures in a rigid body framework and propose several enhancements to improve the convergence of collision resolution, including a hybrid time integration scheme to separately handle at rest contacts and dynamic collisions. Finally, we extend the impulse-based LS-DEM to include arbitrarily shaped topographic surfaces and exploit its algorithmic advantages to demonstrate the feasibility of modeling realistic behavior of granular flows. The new formulation significantly improves the performance of dynamic simulations by allowing larger time steps, which is advantageous for observing the full development of physical phenomena such as rock avalanches, which we present as an illustrative example.

KEYWORDS

DEM, impulse, level set, modeling, rock avalanche, rock fall

1 | INTRODUCTION

Typical DEM codes use penalty-based approach to resolve contacts between discrete bodies, which are then converted to forces, accelerations, and displacements. This approach has varying amounts of complexity in generating a contact interaction force based on the separation distance between the particles in contact.¹ It is also computationally demanding, if not prohibitively expensive, for large-scale DEM simulations with complex-shaped grains or particles. While recent research efforts have been made to accelerate these DEM codes using high-performance computing clusters or graphics processing units (GPUs),^{2–5} such resources are not generally or broadly accessible.

Alternatively, an impulse-based DEM formulation is significantly more computationally efficient since it directly computes the velocity increment of the modeled objects. This approach was originally designed to accelerate computer graphics^{6–8} and prioritizes code speed, stability, and physical plausibility over simulation fidelity. Recent work^{9–12} demonstrates that, in addition to much improved code speed and numerical stability, the impulse-based technique also produces results with comparable accuracy to the corresponding penalty-based DEM simulations. This makes the impulse-based approach intriguing, since it is capable of addressing very large dynamic problems.

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2024 The Author(s). *International Journal for Numerical Methods in Engineering* published by John Wiley & Sons Ltd.

Herein we outline the framework of impulse-based rigid body dynamics, which is fundamentally a process of kinetic energy conversion to elastic energy. Rather than providing a broad introduction to the subject, we focus on the numerical aspects of the impulse-based approaches and the possibility of parallelizing impulse-based LS-DEM using the domain decomposition approach.^{13,14} We show that in comparison to parallel implementation for penalty-based LS-DEM, impulse-based DEM saves considerable effort in data communication, but requires additional handling of multibody assemblies in direct contact. We then introduce a modified Energy Tracking Method (ETM),⁹ in order to resolve collisions between a wide range of grain shapes and sizes and to add deformable structures into rigid body dynamics. Finally, we illustrate the performance and applicability of the impulse-based LS-DEM for rock fall and rock avalanche simulations.

2 | FORMULATION OF IMPULSE-BASED RIGID BODY DYNAMICS

A negative relative velocity between two objects at the time of a collision suggests that they are about to penetrate. To avoid penetration between rigid bodies, the negative relative velocity is increased till it reaches zero by applying a sequence of impulses in the normal direction of contact. The compression phase converts kinetic energy to elastic energy, which is then stored at the contact points. Following that, previously held elastic energy is released to restore relative normal velocity, repelling the two bodies from one another and concluding the collision process. As it is almost impossible to predict the microstructure of materials during collisions in practice, we adopt Stronge's hypothesis:¹⁵ the amount of energy absorbed and released may vary according to the restitution coefficient. For example, for a perfectly elastic collision with no energy loss. Stronge's hypothesis states that energy is dissipated as follows:

$$W_{\text{release}} = \epsilon^2 W_{\text{absorbed}} \quad (1)$$

Where W_{release} and W_{absorbed} denote the energy released and absorbed during release and compression phases, respectively. ϵ ($0 \leq \epsilon \leq 1$) denotes the coefficient of restitution that determines the elasticity of the contact. Without the loss of generality, we formalize a single collision process by considering a body b_A collides with another b_B due to the i -th impulse \mathbf{P}^i , which effectively changes their linear velocities (\mathbf{V}_A and \mathbf{V}_B) and angular velocities ($\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$, as shown in Figure 1 via:

$$\mathbf{P}_A^i = M_A \Delta \mathbf{V}_A \quad (2)$$

$$\mathbf{P}_B^i = M_B \Delta \mathbf{V}_B \quad (3)$$

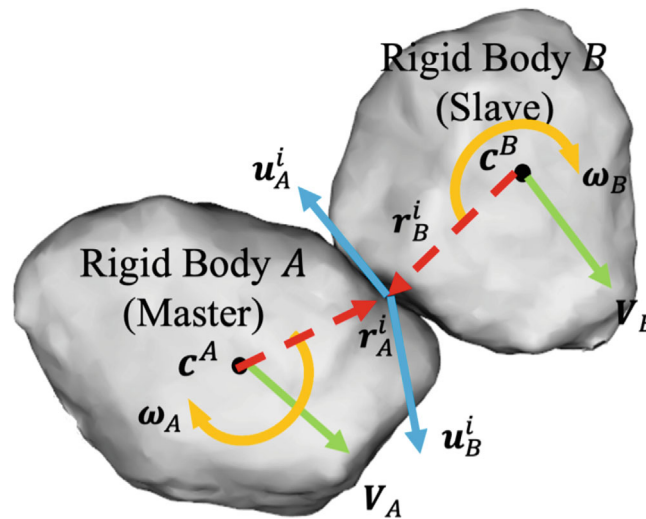


FIGURE 1 Collision between bodies b_A and b_B : \mathbf{V}_A and \mathbf{V}_B are linear velocities of the center of mass, $\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$ are angular velocities, \mathbf{r}_A^i and \mathbf{r}_B^i are the vectors from the center of mass of bodies to the contact point where the i -th impulse \mathbf{P}^i applied, \mathbf{u}_A^i and \mathbf{u}_B^i are the velocities at the contact points, and \mathbf{n}^i is the contact normal.

$$\mathbf{r}_A^i \times \mathbf{P}_A^i = \mathbf{I}_A \Delta \boldsymbol{\omega}_A \quad (4)$$

$$\mathbf{r}_B^i \times \mathbf{P}_B^i = \mathbf{I}_B \Delta \boldsymbol{\omega}_B \quad (5)$$

$$\mathbf{P}_A^i = -\mathbf{P}_B^i \quad (6)$$

Where M_A and M_B are the masses, I_A and I_B are the inertial tensors, \mathbf{r}_A^i and \mathbf{r}_B^i are the relative positions from the center of gravity of b_A or b_B to the contact point at which the i -th impulse was applied. The velocities \mathbf{u}_A^i and \mathbf{u}_B^i of contact points on b_A and b_B in global frame are:

$$\mathbf{u}_A^i = \mathbf{V}_A + \boldsymbol{\omega}_A \times \mathbf{r}_A^i \quad (7)$$

Therefore, the velocity changes at a point of contact between body b_A and body b_B due to the i -th impulse respectively is given below. Here, we suppose that the collision occurs and resolves quickly enough that the contact point i remains stationary.

$$\Delta \mathbf{u}_A^i = \Delta \mathbf{V}_A + \Delta \boldsymbol{\omega}_A \times \mathbf{r}_A^i = \frac{1}{M_A} \mathbf{P}_A^i + \mathbf{I}_A^{-1} \mathbf{r}_A^i \times \mathbf{P}_A^i \times \mathbf{r}_A^i = \left(\frac{1}{M_A} - \tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i \right) \mathbf{P}_A^i \quad (8)$$

$$\Delta \mathbf{u}_B^i = \Delta \mathbf{V}_B + \Delta \boldsymbol{\omega}_B \times \mathbf{r}_B^i = \frac{1}{M_B} \mathbf{P}_B^i + \mathbf{I}_B^{-1} \mathbf{r}_B^i \times \mathbf{P}_B^i \times \mathbf{r}_B^i = \left(\frac{1}{M_B} - \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i \right) \mathbf{P}_B^i \quad (9)$$

The second equality converts cross product between vectors into equivalent matrix-vector multiplication. In the vector space \mathcal{R}^3 , the cross product (\times) is an operator taking two vectors to a third vector:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix} = \tilde{\mathbf{a}} \mathbf{b} \quad (10)$$

Where $\tilde{\mathbf{a}}$ is a 3×3 skew-symmetric matrix. In the collision formula above:

$$\tilde{\mathbf{r}} = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \quad (11)$$

Above formulas imply that if we know the change of relative velocity at the contact points, the resulted impulse can be computed via:

$$\Delta(\mathbf{u}_A^i - \mathbf{u}_B^i) = \Delta \mathbf{u}_A^i - \Delta \mathbf{u}_B^i = \left[\left(\frac{1}{M_A} + \frac{1}{M_B} \right) \mathbf{I} - (\tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i + \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i) \right] \mathbf{P}_A^i \quad (12)$$

$$\mathbf{K} = \left(\frac{1}{M_A} + \frac{1}{M_B} \right) \mathbf{I} - (\tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i + \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i) \quad (13)$$

Where \mathbf{P}_A^i is the i -th impulse exerted on the body b_A , and \mathbf{K} is termed as the collision matrix and it is non-singular, symmetric, positive definite and defined in global frame for a given collision.¹⁶ Collision contact occurs within a very short time interval for a rigid body, implying that displacement and change in the contact area are insignificant. This indicates that the collision matrix \mathbf{K} remains constant throughout the collision, and that once the change in relative velocity is known, the collision matrix is used to compute the impulses:

$$\mathbf{P}^i = \mathbf{K}^{-1} \Delta \mathbf{u}^i \quad (14)$$

Where $\mathbf{P}^i = \mathbf{P}_A^i = -\mathbf{P}_B^i$ is the i -th impulse, $\mathbf{u}^i = \mathbf{u}_A^i - \mathbf{u}_B^i$ is the relative velocities at the contact points where the i -th impulse is applied, and $\Delta \mathbf{u}^i$ is the change of relative velocities. This formula remains true if the friction forces are ignored

or if the friction forces are constant. As a result, if there is no slip between the bodies, only the normal component of the change in relative velocity $\Delta \mathbf{u}_n^i$ is non-zero, because it is the relative normal velocity at the contact locations that drives the bodies to collide. Thus, the preceding formula is reduced to:

$$\mathbf{n}^i \cdot \mathbf{K} \mathbf{P}^i = \mathbf{n}^i \cdot \Delta \mathbf{u}^i = \Delta \mathbf{u}_n^i \quad (15)$$

$$\left| \mathbf{P}_n^i \right| = \mathbf{n}^i \cdot \mathbf{P}^i \mathbf{n}^i \quad (16)$$

$$\mathbf{P}_t^i = \mathbf{P}^i - \mathbf{P}_n^i \quad (17)$$

If sliding occurs, apart from ensuring that relative normal velocity changes are consistent with the prescribed value, the impulse must be recalculated to meet Coulomb's friction requirement, which limits the magnitude of tangential impulse:

$$\left| \mathbf{P}_t^i \right| = \mu \left| \mathbf{P}_n^i \right| \quad (18)$$

$$\mathbf{P}^i = \left| \mathbf{P}_n^i \right| \mathbf{n}^i + \mu \left| \mathbf{P}_n^i \right| \mathbf{t}^i \quad (19)$$

$$\left| \mathbf{P}_n^i \right| = \frac{\Delta \mathbf{u}_n^i}{\mathbf{n}^i \cdot (\mathbf{K} \mathbf{n}^i + \mu \mathbf{t}^i)} \quad (20)$$

Where \mathbf{n}^i , \mathbf{t}^i are the contact normal and tangential direction at the i -th collision, and \mathbf{P}_n^i and \mathbf{P}_t^i are the normal and tangential components of applied impulse. This concludes the compression phase where the corresponding impulse is generated based on the change of relative velocity and the kinetic energy is stored in the form of elastic energy at the contact.

The next step is to dissipate the elastic energy and invert the sign of the relative normal velocity to separate the bodies. This is the inverse of the compression phase: a predefined amount of elastic energy is dissipated to increase the relative normal velocity via a series of impulses. Mirtich¹⁶ proved that if the relative contact velocity \mathbf{u}^i proceeds from \mathbf{u}_0^i to \mathbf{u}_f^i during a collision and over some arbitrary path, the total work done by the collision impulse \mathbf{P}^i is independent of the path taken and is given below, which enables the change in relative velocities to be calculated from the released elastic energy.

$$\Delta W^i = \frac{1}{2} (\mathbf{u}_f^i + \mathbf{u}_0^i)^T \mathbf{K}^{-1} (\mathbf{u}_f^i - \mathbf{u}_0^i) = \frac{1}{2} (\mathbf{u}_f^i + \mathbf{u}_0^i)^T \mathbf{P}^i \quad (21)$$

Given that rigid body motion is better defined in terms of its principal or local frame, it is more convenient to address collisions in the local coordinate system via the rotation matrix \mathbf{R} , which defines the transformation from body to global coordinates. In our implementation, rotations are handled numerically via a singularity-free quaternion technique given by:

$$\mathbf{R} = \begin{pmatrix} -q_1^2 + q_2^2 - q_3^2 + q_4^2 & -2(q_1 q_2 - q_3 q_4) & 2(q_2 q_3 + q_1 q_4) \\ -2(q_1 q_2 + q_3 q_4) & q_1^2 - q_2^2 - q_3^2 + q_4^2 & -2(q_1 q_3 - q_2 q_4) \\ 2(q_2 q_3 - q_1 q_4) & -2(q_1 q_3 + q_2 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{pmatrix} \quad (22)$$

Where the q 's are the quaternions.¹⁷ The time derivatives of the orientation parameters (q_1, q_2, q_3, q_4) can be expressed in terms of the quaternions and the angular velocities.

$$\begin{aligned} \dot{q}_1 &= \frac{1}{2} (-q_3 \omega_x - q_4 \omega_y + q_2 \omega_z) \\ \dot{q}_2 &= \frac{1}{2} (q_4 \omega_x - q_3 \omega_y - q_1 \omega_z) \\ \dot{q}_3 &= \frac{1}{2} (q_1 \omega_x + q_2 \omega_y + q_4 \omega_z) \\ \dot{q}_4 &= \frac{1}{2} (-q_2 \omega_x + q_1 \omega_y - q_3 \omega_z) \end{aligned} \quad (23)$$

$$\sum_{i=1}^4 q_i^2 = 1 \quad (24)$$

These equations can be solved explicitly for the quaternion values at the new time step in terms of the old values, and for the angular velocities at the midpoint of the time step. Then

$$\Delta W^i = \frac{1}{2} \left(\mathbf{u}_f^i + \mathbf{u}_0^i \right)^T \mathbf{P}^i = \frac{1}{2} \left(\mathbf{R}\mathbf{u}_f^i + \mathbf{R}\mathbf{u}_0^i \right)^T \mathbf{R}\tilde{\mathbf{P}}^i = \frac{1}{2} \left(\mathbf{u}_f^i + \mathbf{u}_0^i \right)^T \mathbf{P}^i \quad (25)$$

Where \mathbf{u}_0 , \mathbf{u}_f and \mathbf{P}^i denote initial relative velocity, final relative velocity, and applied impulse, all with respect to the body frame. During a collision, the work done by the impulse can be decomposed into the work done by the impulse in the normal direction ΔW_n^i and the work done by the impulse in the tangential direction ΔW_t^i .

$$\Delta W^i = \Delta W_n^i + \Delta W_t^i \quad (26)$$

$$\Delta W_n^i = \frac{1}{2} (2\tilde{u}_n^i + \Delta\tilde{u}_n^i) \tilde{P}_n^i \quad (27)$$

$$\Delta W_t^i = \frac{1}{2} (2\tilde{u}_t^i + \Delta\tilde{u}_t^i) \tilde{P}_t^i \quad (28)$$

Where \tilde{u}_n^i , \tilde{u}_t^i and \tilde{P}_n^i , \tilde{P}_t^i are the normal and tangential component of relative velocities and impulses at contact points defined in the body frame. It is more usual to depict a plane perpendicular to the contact normal using two orthogonal vectors, which is also compatible with the rotation operation using three orthogonal vectors. We decided not to differentiate the two tangential directions here since we were only concerned with the normal components: tangential velocities would remain unchanged because they did not break the impenetrability requirements between rigid entities. Due to the fact that work performed in the different directions is independent, the change in the relative normal velocity has no effect on the amount of work performed in the normal direction by the impulse, which later inverts the relative normal velocity by releasing the energy absorbed during the compression phase.

The change of relative velocity $\Delta\mathbf{u}^i$ and the corresponding work done ΔW^i by the impulse implies that there should be a way to compute $\Delta\mathbf{u}^i$ (ΔW^i) if the collision work ΔW^i ($\Delta\mathbf{u}^i$) is known. Indeed, simply changing the relative normal velocity results in the generation of tangential impulses and thus the associated work. However, because the work performed in the normal directions is decoupled, the change in relative normal velocity $\Delta\mathbf{u}_n^i$ can be retrieved provided that we get access the normal impulse \tilde{P}_n^i and the absorbed/released work caused by \tilde{P}_n^i . Additionally, the tangential restitution coefficient is set to zero, which requires calibration but is plausible in many impulse-based simulators. As a result, the collision work ΔW^i and ΔW_n^i are used interchangeably, and we focus on the conversion of kinetic to elastic energy in the contact normal direction. For instance, during the compression phase, the relative normal velocity decreases and eventually converts to the elastic energy:

$$\Delta W_n^i = \frac{1}{2} (2\tilde{u}_n^i + \Delta\tilde{u}_n^i) \tilde{P}_n^i \quad (29)$$

Where \tilde{P}_n^i is the normal component of applied impulse. In the separation phase, the absorbed elastic energy reduces and changes the sign of the relative normal velocity via an impulse along the contact normal:

$$\mathbf{K}\mathbf{P}^i = \Delta\mathbf{u}^i \quad (30)$$

$$\mathbf{n}^i \cdot \mathbf{K} \left(\tilde{P}_n^i \mathbf{n}^i \right) = \mathbf{n}^i \Delta\mathbf{u}^i = \Delta\tilde{u}_n^i \quad (31)$$

$$\tilde{P}_n^i = \frac{\Delta\tilde{u}_n^i}{\mathbf{n}^i \cdot (\mathbf{K}\mathbf{n}^i)} \quad (32)$$

$$\Delta\tilde{u}_n^i = -\tilde{u}_n^i + \sqrt{(\tilde{u}_n^i)^2 + 2\Delta W_n^i \mathbf{n}^i \cdot (\mathbf{K}\mathbf{n}^i)} \quad (33)$$

3 | IMPULSE-BASED LS-DEM IMPLEMENTATION

3.1 | Contact model

Just as in a penalty-based method, the contact in impulse-based LS-DEM is handled through node-to-surface contact algorithm and the contact normal direction is also interpolated from the underlying signed distance function grid of a grain. The difference is that contact force calculation is skipped in the impulse-based formulation; however, it could be retrieved from the applied impulse with high-precision. In addition, the inter-grain tangential contact is significantly easier to manage in an impulse-based DEM because it does not track friction evolution and updates velocities directly. During the force resolution phase, the impulse-based DEM iterates through the surface nodes between colliding bodies, identifying all contact points with negative relative velocities, and calculating repulsive impulses to satisfy impenetrability constraints.

The friction type of contact is examined to ensure that the magnitude of the tangential component of the collision impulse follows Coulomb's friction law. Due to the fact that LS-DEM aims to simulate arbitrarily complex grain shapes, it is unavoidable that multiple contact points collide when two reconstructed avatars are on a collision course. To handle many collisions and various contacts, the simultaneous impulse methods (SMM)^{18,19} assembles all collisions into a system of linear equations, enforces non-penetration restrictions as a linear complementary problem (LCP) and evaluates the outcomes in a single step. However, the SMM fails to capture the propagation of contact forces during a collision. The Sequential Impulse Method (SQM)²⁰ resolves each collision within a single time interval and treats the contacts as if they occurred sequentially. This method inverts the relative velocity directly using Newton's law and prioritizes contact points based on the depth of inter-penetration; it is iterative and continues until all contact points have positive relative velocities. The primary disadvantage of SQM is that edge-surface or surface-surface contacts do not end up with similar repulsive velocities and the results depend on the sequence in which collisions are addressed. This issue can be resolved by gradually changing the relative normal velocity and elastic energy in such a way that all contact points with similar normal directions can adjust velocities to similar magnitudes.^{9,12} The compression phase involves incrementally increasing the relative normal velocities at contact points until all relative normal velocities are non-negative. During the separation phase, the absorbed energy is gradually released until no energy remains at the contact points, again in an iterative fashion.

3.2 | Discrete equations of rigid body motion in impulse-based LS-DEM

The original implementation of LS-DEM²¹ solves Newton's and Euler's governing equations of motion for translational and rotational components of motion, respectively. For the translational component of motion, the positions, forces, and velocities are known at the end of each time step so that grain motion can be explicitly updated via the governing translational equation given by Newton's law using a centered finite-difference integration scheme. For the rotational components of motion, the time derivatives of the angular accelerations in the principal frame are given by Euler's equations of motion, which is nonlinear and implicit, hence prone to numerical instability.

Again, it is much simpler to change the velocities of two rigid bodies with the impulse-based method. For example, if the contact master body b_A experiences a collision contact with b_B under the corresponding impulse \mathbf{P}^i , in accordance with Newton's law, the changes in velocity for each rigid body are given by:

$$\Delta \dot{\mathbf{x}}_{A,B} = M_{A,B}^{-1} \mathbf{P}_{A,B}^i \quad (34)$$

$$\Delta \dot{\boldsymbol{\theta}}_{A,B} = \mathbf{I}_{A,B}^{-1} (\mathbf{r}_{A,B}^i \times \mathbf{P}_{A,B}^i) \quad (35)$$

The velocities and positions for the rigid bodies are subsequently updated via symplectic Euler scheme:

$$\dot{\mathbf{x}}_{A,B} = \dot{\mathbf{x}}_{A,B} + \Delta \dot{\mathbf{x}}_{A,B} \quad (36)$$

$$\dot{\boldsymbol{\theta}}_{A,B} = \dot{\boldsymbol{\theta}}_{A,B} + \Delta \dot{\boldsymbol{\theta}}_{A,B} \quad (37)$$

$$\mathbf{x}_{A,B} = \mathbf{x}_{A,B} + \dot{\mathbf{x}}_{A,B} \Delta t \quad (38)$$

$$\theta_{A,B} = \theta_{A,B} + \dot{\theta}_{A,B} \Delta t \quad (39)$$

3.3 | Collision resolution algorithm

The ETM algorithm is demonstrated with three vertically stacked spheres as shown in Figure 2. The concurrent collisions are treated as a series of single point collisions and include an additional iteration within the compression and separation phase to make the results insensitive to the order of the impulses. When dealing with numerous contact collisions, ETM gradually delivers impulses to adjust the relative normal velocities of several collisions, yielding low angular velocity errors. The time step within the sub-cycle loop can be adjusted adaptively to obtain more stable and energy-conservative rigid body dynamic simulations.¹² In the impulse-based DEM, if any pair of rigid bodies is connected to a group of LS avatars via a chain of contacting avatars, collisions at all contact points are resolved simultaneously, as the impulse can propagate within the group. The algorithm begins by iterating through contact points created by collisions and prioritizing those with a negative relative normal velocity. During the compression phase, the priority queue is sorted by minimum relative normal velocity; after the algorithm enters the separation phase, it is sorted by elastic energy. To expedite the process, a group of adjacent contact points is glued together and treated as a single contact. Kinetic and elastic energy are converted to one another during the compression and separation phases.

The kinetic energy gradually decreases to zero as a result of the application of a series of impulses in the compression phase, and the relative velocity reverts to its initial sign during the energy release phase. Following Stronge's hypothesis (setting the coefficient of restitution to 0.5), the restored elastic energy abruptly decreases at the end of the compression phase and gradually decreases to zero during the energy release phase. Although the relative velocity or elastic energy converge arbitrarily close to zero, they do not achieve it numerically. Rather, a threshold value is chosen to truncate small values and to make the algorithm more efficient. The threshold value should be less than the time step, as otherwise velocity changes may not be reflected, thus accumulating errors. In addition, While the grain interactions can be reconciled element by element in a penalty-based approach, a contact island illustrated in Figure 3, in which any pair of grains

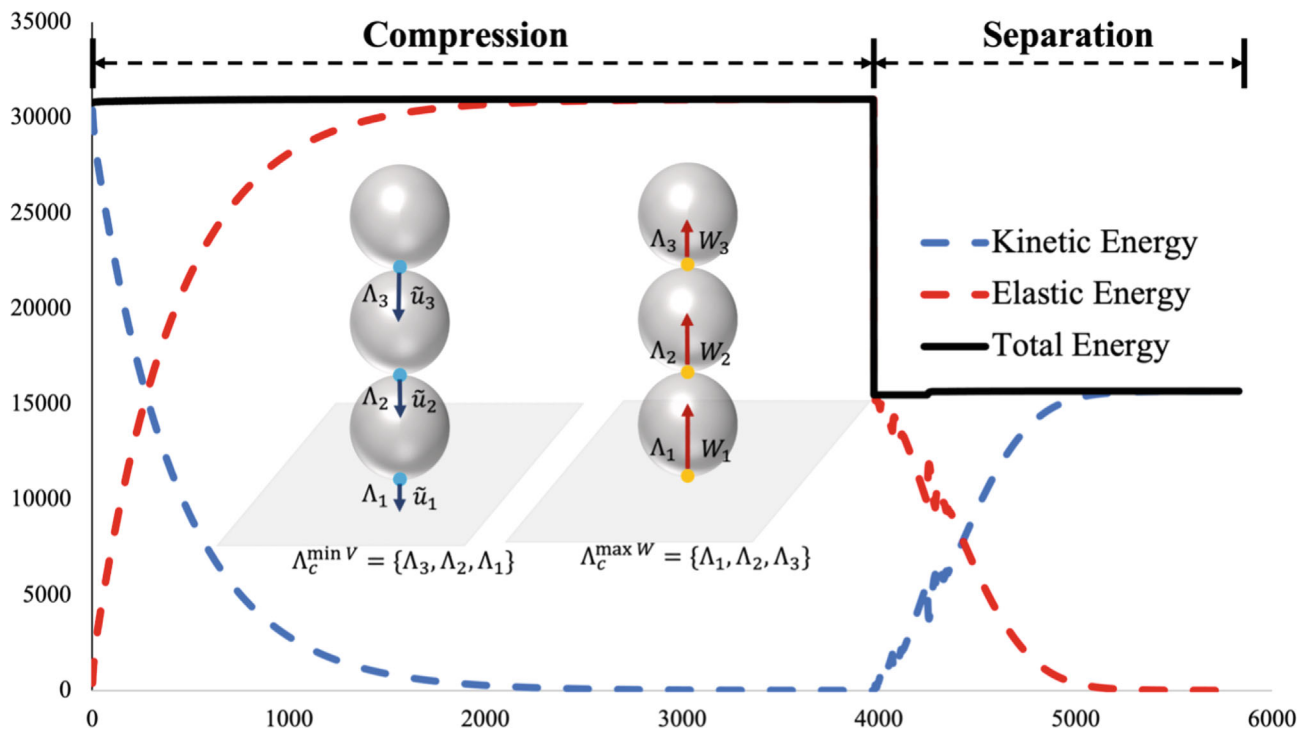


FIGURE 2 Energy conservative property of impulse-based collision resolution, kinetic energy (blue) and elastic energy (red) during compression and separation phases. The coefficient of restitution is 0.5.

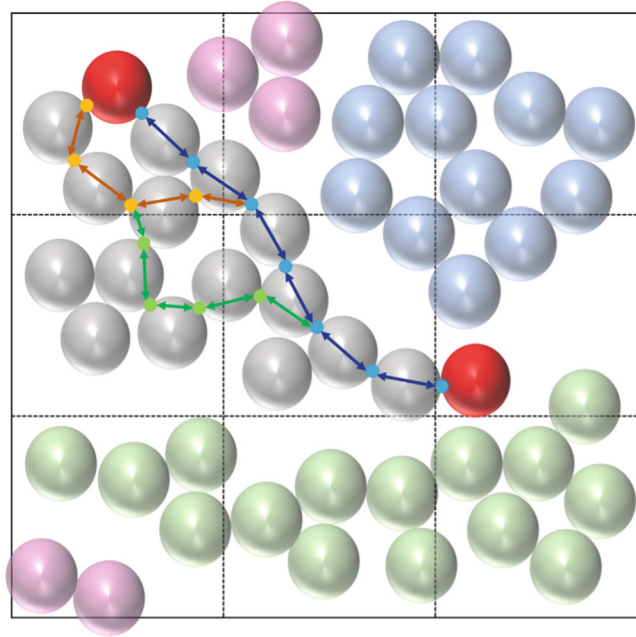


FIGURE 3 Illustration of contact islands (indicated in different colors) in which a group of bodies is associated via a contact chain which can span over several sub-domains. For example, two red bodies in a same group could influence each other via alternate paths.

is connected via a chain of immediately contacting grains, must be solved concurrently to satisfy the impenetrability constraints in the impulse-based formulation. The resulting algorithm is presented in Algorithm 1.

3.4 | Time stepping algorithm

Explicit time integration is ideally suited to dynamic contact/impact problems, as small time steps allow for the handling of contact/impact discontinuities. In a traditional penalty-based DEM, the governing equations determine the resultant force acting on objects and then update accelerations and velocities via a numerical integrator. In comparison, the impulse-based dynamic simulation modifies velocity directly using a different numerical scheme, obviating the requirement for force computations. This results in the symplectic Euler scheme of first order.

$$\mathbf{v}^{t+1} = \dot{\mathbf{x}}^t + \Delta \dot{\mathbf{x}}^t = \mathbf{v}^t + \Delta \mathbf{v}^t \quad (40)$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+1} \Delta t \quad (41)$$

$$\boldsymbol{\omega}^{t+1} = \dot{\boldsymbol{\theta}}^t + \Delta \dot{\boldsymbol{\theta}}^t = \boldsymbol{\omega}^t + \Delta \boldsymbol{\omega}^t \quad (42)$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \boldsymbol{\omega}^{t+1} \Delta t \quad (43)$$

Above formulations are different from the time integration methods used in penalty-based DEM, an example of second order finite difference scheme in the time-centered form²² is displayed below.

$$\mathbf{v}^{t+\frac{1}{2}} = \mathbf{v}^{t-\frac{1}{2}} + \dot{\mathbf{v}}^t \Delta t \quad (44)$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+\frac{1}{2}} \Delta t \quad (45)$$

Which is equivalent to the formulation of Reference 10:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \dot{\mathbf{x}}^t \Delta t + \frac{1}{2} \ddot{\mathbf{x}}^t \Delta t^2 \quad (46)$$

Algorithm 1. Collision resolution algorithm

Input: A list of contacts within a contact island including contact IDs of colliding bodies A and B; branch vectors from centers to contact point \mathbf{r} ; contact normal direction \mathbf{n} ; relative velocity $\tilde{\mathbf{u}}$; elastic energy W and body mass M .

- 1: Define parameter α to control velocity increment in sub-iteration, Coulomb's friction coefficient μ , tolerance $\beta \approx 1\text{-}5\%^{12}$, and Stronge's coefficient ϵ .
- 2: **for** all contact points **do**
- 3: Compute inertial matrix in global frame $\mathbf{I} = R\mathbf{I}_0R^T$, cross-product matrix of colliding bodies $\tilde{\mathbf{r}}$, relative normal velocity u_n , collision matrix $\mathbf{K} = \left(\frac{1}{M_A} + \frac{1}{M_B}\right)\mathbf{I} - (\tilde{\mathbf{r}}_A\mathbf{I}_A^{-1}\tilde{\mathbf{r}}_A + \tilde{\mathbf{r}}_B\mathbf{I}_B^{-1}\tilde{\mathbf{r}}_B)$, wrap computed quantities as Λ_c .
- 4: **end for**
- 5: Prepare a priority queue for all contact points with negative relative normal velocity, sorted by the minimum value. $\Lambda^{\min V} := \{\Lambda_c, \text{key} = u_n\}$.
- 6: **while** $\Lambda^{\min V}$ is not empty **do**
- 7: /* Compression Phase */
- 8: **while** $\Lambda^{\min V}$ is not empty **do**
- 9: Generate a list Γ^V containing Λ_c with minimum relative normal velocity u_n^{\min} and those with similar magnitude, $\frac{|u_n' - u_n^{\min}|}{|u_n^{\min}|} \leq \beta$.
- 10: Obtain increment of relative normal velocity: $\Delta u_n^{\min V} = \frac{u_n^{\min}}{\alpha \cdot |\Gamma^V|}$.
- 11: **for** all contact points in Γ^V **do**
- 12: $\tilde{\mathbf{P}} = \Delta u_n^{\min V} \mathbf{K}^{-1} \mathbf{n}$, $\tilde{\mathbf{P}}_n = \mathbf{n}(\tilde{\mathbf{P}} \cdot \mathbf{n})$ and $\tilde{\mathbf{P}}_t = \tilde{\mathbf{P}} - \tilde{\mathbf{P}}_n$.
- 13: **if** $|\tilde{\mathbf{P}}_t| \geq \mu |\tilde{\mathbf{P}}_n|$ **then**
- 14: $|\tilde{\mathbf{P}}_n| = \frac{\Delta u_n^{\min V}}{\tilde{\mathbf{n}} \cdot [\mathbf{K}(\tilde{\mathbf{n}} + \mu \tilde{\mathbf{t}})]}$
- 15: $\tilde{\mathbf{P}} = |\tilde{\mathbf{P}}_n| \mathbf{n} + \mu |\tilde{\mathbf{P}}_n| \tilde{\mathbf{t}}$
- 16: **end if**
- 17: Calculate change in velocity, $\Delta \tilde{\mathbf{u}} = M^{-1} \tilde{\mathbf{P}}$, $\Delta \tilde{\omega} = \mathbf{I}^{-1}(\mathbf{r} \times \tilde{\mathbf{P}})$.
- 18: Restore elastic energy, $\Delta W = \frac{1}{2}(2u_n + \Delta u_n^{\min V})|\tilde{\mathbf{P}}_n|$.
- 19: **end for**
- 20: Compute updated relative normal velocities and reset $\Lambda^{\min V}$.
- 21: **end while**
- 22: /* Restitution Phase */
- 23: **for** all contact points in the contact island **do**
- 24: Multiply restored elastic energy by Stronge's coefficient to imitate energy dissipation. $W \leftarrow \epsilon W$.
- 25: **end for**
- 26: /* Separation Phase */
- 27: Build a priority queue for all contact points with non-negative elastic energy, sorted by the maximum value. $\Lambda^{\max W} := \{\Lambda_c, \text{key} = W\}$.
- 28: **while** $\Lambda^{\max W}$ is not empty **do**
- 29: Generate a list Γ^W containing Λ_c with maximum restored elastic energy W^{\max} and those with similar magnitude, $\frac{|W' - W^{\max}|}{|W^{\max}|} \leq \beta$.
- 30: Obtain change of relative normal velocity at point with largest elastic energy. $\Delta u_n^{\max W} = -u_n^{\max W} + \sqrt{(u_n^{\max W})^2 + \mathbf{n} \cdot (\mathbf{K}\mathbf{n})}$
- 31: **for** all contact points in Γ^W **do**
- 32: $\tilde{\mathbf{P}} = \Delta u_n^{\max W} \mathbf{K}^{-1} \mathbf{n}$, $\tilde{\mathbf{P}}_n = \mathbf{n}(\tilde{\mathbf{P}} \cdot \mathbf{n})$ and $\tilde{\mathbf{P}}_t = \tilde{\mathbf{P}} - \tilde{\mathbf{P}}_n$.
- 33: **if** $|\tilde{\mathbf{P}}_t| \geq \mu |\tilde{\mathbf{P}}_n|$ **then**
- 34: $|\tilde{\mathbf{P}}_n| = \frac{\Delta u_n^{\max W}}{\tilde{\mathbf{n}} \cdot [\mathbf{K}(\tilde{\mathbf{n}} + \mu \tilde{\mathbf{t}})]}$
- 35: $\tilde{\mathbf{P}} = |\tilde{\mathbf{P}}_n| \tilde{\mathbf{n}} + \mu |\tilde{\mathbf{P}}_n| \tilde{\mathbf{t}}$
- 36: **end if**
- 37: Calculate change in velocity, $\Delta \tilde{\mathbf{u}} = M^{-1} \tilde{\mathbf{P}}$, $\Delta \tilde{\omega} = \mathbf{I}^{-1}(\mathbf{r} \times \tilde{\mathbf{P}})$.
- 38: Release elastic energy, $\Delta W = -\frac{1}{2}(2u_n + \Delta u_n^{\max W})|\tilde{\mathbf{P}}_n|$.
- 39: **end for**
- 40: Compute updated restored elastic energy and reset $\Lambda^{\max W}$.
- 41: **end while**
- 42: /* some velocities can be negative after above two iterations */
- 43: Identify all contact points with negative relative normal velocity and update priority queue $\Lambda^{\min V}$, $\Lambda^{\min V} := \{\Lambda_c, \text{key} = u_n\}$.
- 44: **end while**

For motion update, the penalty-based DEM is second order accurate, but the impulse-based dynamic simulation updates the motion via a linear change in velocities. In other words, the symplectic Euler scheme analyzes velocity changes in the ‘secant’ direction, whereas the time-centered Euler scheme considers velocity changes in the ‘tangent’ direction. Regarding the numerical stability, both schemes are conditionally stable and require a critical time step such that there is limited oscillation in the solution and any numerical errors do not accumulate.¹⁰ In addition, the rigid body dynamic simulation also requires that the numerical solution does not produce spurious energy gains in the modeled system for a sufficiently long simulation time.

The symplectic Euler integration is also conditionally stable in the sense that the numerical results are bounded only if a small Δt is used that is less than Δt_{cr} . However, because the contact stiffness is omitted from the impulse-based dynamic formulation, the critical time step is determined by the Courant-Friedrichs-Lewy (CFL) condition, which is substantially larger than the criterion for the penalty-based DEM formulation. The CFL condition prevents obvious penetration errors between objects, which implies that the time step is limited by the physics of the problem, where an excessively large time step size results in objects passing through one another, but not by numerical stability issues. Additionally, it possesses the virtue of numerical stability over a long simulation period due to the fact that the total energy of a modeled system is nearly conserved even at large Δt . Thus, the impulse-based simulation may use a time step several orders greater than penalty-based DEM simulations, maintaining comparable computational fidelity. This feature is especially advantageous for modeling natural granular materials with large grain size distributions, for which standard DEM is prohibitively expensive. Also, while mass scaling is widely employed in DEM,²³ it is not advisable if a dynamic analysis requires high frequency response.

Energy conservation is ensured by the combined usage of the symplectic scheme and iterative collision resolution algorithm. This trait, however, is less desirable when modeling a granular system with a broad variety of shapes and sizes. Consider a small grain sandwiched between two considerably larger grains, both of which are attempting to crush the small grain at opposing contact locations (Figure 4 left). The small grain may elastically bounce back and forth in a protracted succession of near-simultaneous encounters. This procedure is unstable in the long run in that even a very small numerical inaccuracy is likely to accumulate and result in the simulation becoming unstable. To address this issue, global damping is introduced, and the velocity change caused by global damping is updated in the following manner:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}(1 - \xi \Delta t) \quad (47)$$

$$\dot{\theta} = \dot{\theta}(1 - \xi \Delta t) \quad (48)$$

$$0 \leq \xi \Delta t \leq 1 \quad (49)$$

Where ξ is the global damping ratio. While this approach helps accelerate convergence in the case of multiple collisions in a complicated system, it may still fail when a stiff boundary is involved. Rather being squashed by larger grains, for example, a small grain may rest on a rigid plane in one direction while colliding with a larger grain in the opposite direction (Figure 4 right). The rationale for applying damping to dynamic collisions between grains is that collisions will eventually resolve after a sufficient number of repetitions, and damping is a last choice for expediting this process. Interacting with a rigid boundary, on the other hand, is fundamentally different, as the rigid boundary’s velocity cannot be dampened, resulting in a numerically stiff problem as discussed next.

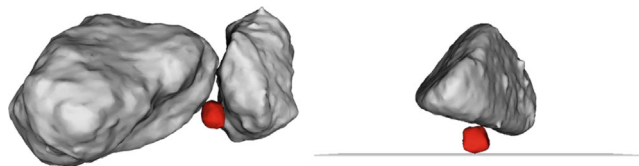


FIGURE 4 Left: A small grain is sandwiched by two larger grains. Right: A small grain is resting on rigid plane while colliding against a large grain. The collision resolution algorithm for these cases might take extremely large number of iterations.

4 | NUMERICAL CHALLENGES OF THE IMPULSE-BASED LS-DEM

4.1 | Contact with rigid boundaries

The conventional approach for impulse-based DEM is to integrate the equations for rigid body evolution forward in time and then resolve collisions to update velocities. This requires the ETM algorithm to modify the velocity discontinuously and to treat all types of interactions in the same manner to reap the algorithmic benefits. That is, regardless of the magnitude of relative velocity and friction status, both inter-grain and wall-grain interactions are considered equivalent. We will demonstrate how resting contact on rigid wall causes the ETM algorithm to enter an infinite loop when dealing with a system of complex shaped objects. We define the following terms: resting contact refers to the state of objects being stationary, whereas dynamic collision refers to the state of grains experiencing a change in force due to interference. A simplified explanation is that because the velocity of a rigid wall remains unchanged, it violates the premise of the ETM algorithm, which separates two objects by sequentially applying repulsive impulses to both. Collisions require impulses to modify the velocity, whereas contacts are more closely associated with forces and accelerations. As a result, a special treatment is required to distinguish impulse-based collision treatment and needs a penalty-springs approach for at rest contact.

We used a time sequence similar to Guendelman et al.²⁰ to distinguish the two types of contacts with the magnitude of the relative velocity suggested by Moore and Williams²⁴ and Mirtich.⁶ The critical concept behind this procedure is to detect static contacts that violate impenetrability constraints and correct velocity immediately after the ETM algorithm is used to update the velocity. To avoid an infinite loop during the collision resolution, only dynamic collisions are considered. The general application of velocity correction is dependent on two critical factors: a low relative velocity and rigid boundary interaction. If one of these two criteria is not met, a wall-grain contact is still classified as a normal collision and treated using a collision resolution algorithm. After the ETM algorithm is completed, the object is advanced in the next time step using the newly solved velocities. The interference between rigid walls and grain is then checked, and the grain velocity is corrected, as we want the objects to move to positions where there are no rigid wall interactions. To ensure this, we use the old velocities to predict the positions of the rigid bodies and use corrected velocities to progress through time steps. For instance, in the collision phase, if an object's current position and velocity are \mathbf{x} and \mathbf{v} , we test for interference with rigid walls using the predicted position $\mathbf{x}' = \mathbf{x} + \mathbf{v} \cdot \Delta t$, and then apply correction to the current velocity such that the resulting velocity \mathbf{v}^* does not interfere with rigid walls. Finally, we advance the object's position $\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^* \cdot \Delta t$. The algorithm's overall structure is shown in Figure 5: it moves all rigid bodies to their predicted locations first, and then it identifies and resolves all grains that penetrate rigid boundaries. This idea is visualized in Figure 6, that is, we consider all nodes within the interpenetrating edges of a grain and move the one with deepest penetration till it is no longer in contact with the rigid wall. Angular velocity is also corrected by integrating the repulsive forces due to temporary grain-wall overlaps. After velocity correction, new contacts with negative relative velocities are identified and included in contact islands; we then re-evolve the position using the new post-collision velocity and locate grains penetrating with rigid walls once again. We repeat the process until all contacts are either non-interpenetrating or separating. By design, this time sequence also ensures impenetrability between a grain and the rigid wall; if the velocity correction step were to occur before collision resolution, the objects would otherwise pass through the rigid wall. We note that, Wriggers and Laursen²⁵ and Zohdi²⁶ implemented a similar adaptive time-stepping algorithm with convergence criterion to resolve the interaction between the network fabric and the rigid body.

4.2 | Modeling deformable structures

In comparison to rigid bodies, dynamic simulation excludes a wide variety of methods for modeling deformable structures, as the motion of a deformable structure is frequently unpredictable using a mathematical formula, for this reason, it is not considered in a dynamic simulation. As a result, impulse-based methods are frequently ignored or avoided in dynamic simulations of deformable structures, and are instead used exclusively for rigid bodies. The assertion that impulse-based methods can be applied only to rigid body models¹⁶ is intuitive: impulses are instantaneous changes in momentum, whereas deformation is a gradual process that occurs over time. However, many applications of DEM, such as numerical modeling of experimental triaxial tests or studies of the interaction between retention barriers and boulders in debris flow simulation, require the modeling of a flexible membrane. The central idea is to represent structures using a group

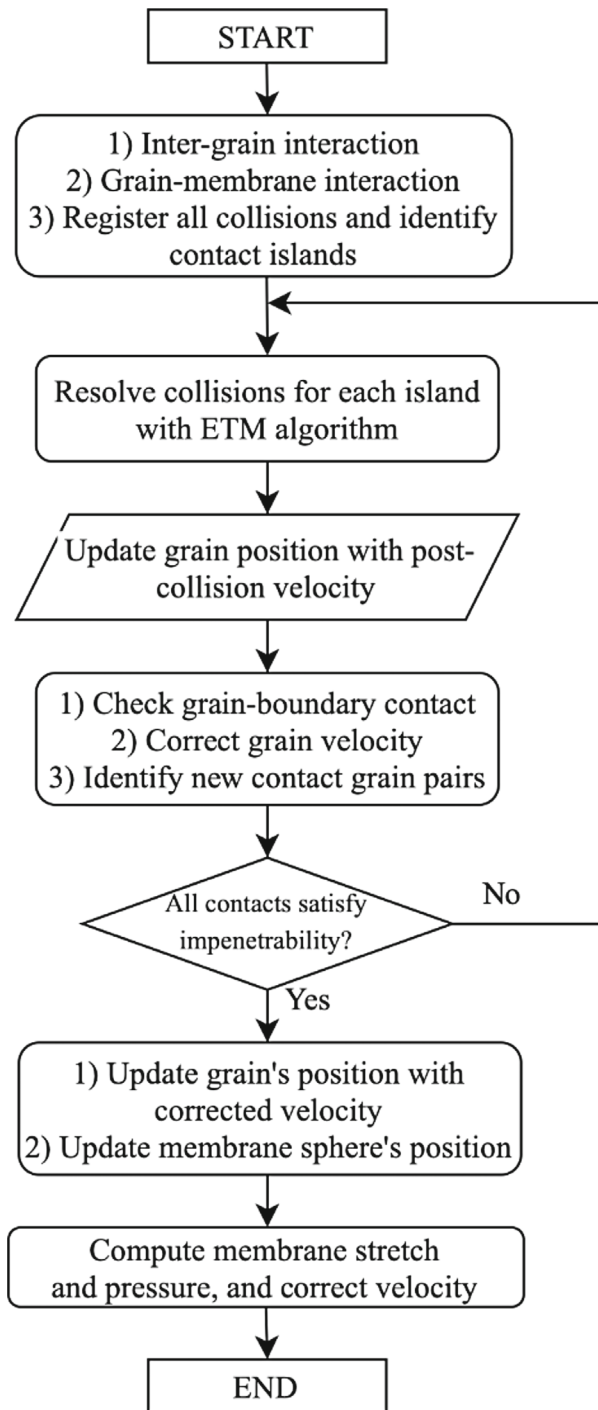


FIGURE 5 Flowchart of the time stepping integration, post-collision velocities were corrected by checking grain-boundary contact, membrane sphere velocity was corrected after it advanced to the next time step to ensure impenetrability.

of rigid balls: while each ball changes velocity in a sequence of instantaneous impulses, the configuration of the balls in the group changes in a seemingly gradual manner over time. We claim that this design does not contradict the rigid body assumption of impulse-based formulation because individual balls are still rigid. This method is different from the soft-contact model used for penalty-based DEM in that the springs used in contact detection and force resolution in penalty-based DEMs are inserted temporarily between detached colliding objects, whereas the springs used in the flexible membrane are permanent, and the distinct balls represent an entire entity. Using a similar idea, Zohdi²⁶ computationally simulated a network of coated yarn using coupled fiber-segments, which include damage and plastification of the yarn.

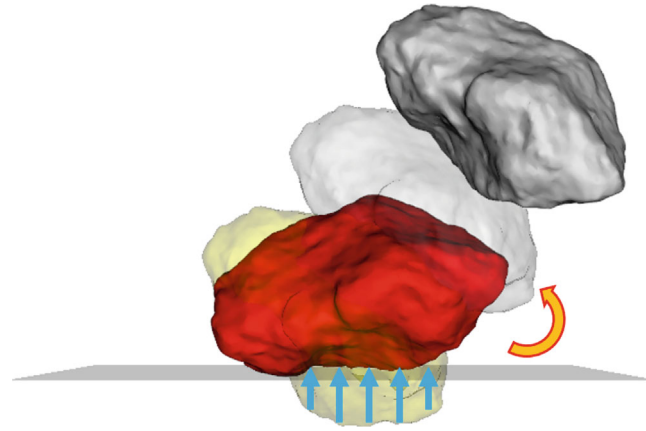


FIGURE 6 Collision time sequence: gray grain obtains post-collision velocity, temporarily translates to the position marked in yellow, subject to repulsive forces and correct velocities to satisfy impenetrability constraints.

To demonstrate the flexible membrane model, a representative example of several complex shaped grains falling into a flexible net is considered. The flexible nature of the net is modeled as a blanket of linked balls organized in hexagonal patterns with a stiffness of $K = 0.1$ for the internal springs between balls. The grains are subjected to vertical body force and the time step is $t = 2 \times 10^{-4}$; the simulation takes approximately 16,000 iterations to reach an equilibrium state equivalent to 3.2 physical seconds. Figure 7 depicts the simulation results after 0 iteration, 4000 iterations, 8000 iterations, and 12,000 iterations. The flexible membrane or net is square and has four clamped edges. All balls along the clamped edges are given an initial velocity of zero, and forces are zeroed out, effectively immobilizing them. External downward forces are applied to grains at regular time intervals via impulses and the grains begin to fall and contact the initially flat membrane. When the membrane comes into contact, it begins to sag and then exhibits a wave-like pattern as a result of internal velocity propagation triggered by the initial fall via a chain of springs. After 12,000 iterations, the induced waves have mostly subsided, and the grains have settled to form a depression in the membrane. This simulation took only 2 min to complete, which is compelling because it eliminated the need to select regular time intervals. The selection of an appropriate time interval in the case of granular systems is complicated by the fact that interacting entities can differ in size and momentum by many orders of magnitude. Because the size and mass of the membrane ball are much smaller than grains in this example, they would have governed the time step in the conventional penalty-based DEM for the sake of numerical stability, necessitating significantly more iterations to complete the simulation.

Another numerical example is illustrated in Figure 8, where an assembly of grains is initially seated on an inclined plane and then subjected to gravity force. Grain collisions are first resolved using the ETM algorithm and then the grain positions are temporarily updated using our new time stepping scheme with the post-collision velocities. When grains collide with a rigid plane as the result of gravity and interference, post-collision velocities are corrected to enforce impenetrability constraints. Then, additional pairs of grains that may collide at new velocities are identified and resolved with the ETM algorithm. When the assembly descends and rolls into the protection net, the membrane spheres interact with the grains, registering and resolving their collisions using the ETM algorithm. The membrane spheres update their positions after evolving velocities to avoid colliding with other spheres and grains. Following that, the velocity of the sphere is altered to account for spring and pressure forces acting on the membrane's surface. The sphere velocity must be corrected immediately following position advancement via post-collision velocity, and the order is critical, as the grain may otherwise pass through the membrane. A network of membrane spheres propagates a velocity wave and causes the entire entity to appear deformable in the presence of sequential collisions. Interestingly, some small grains pass through the protection net due to the stretched spheres creating gaps among them. It is worth noting that the domain re-decomposition strategy improves the efficiency of the parallel code by ensuring that the entire computational domain only encompasses the assembly's geometric configuration and distributes workload evenly across all processors. The mass difference is greater than 400 in this simulation; however, it does not cause the algorithm to stall because this is a dynamic problem which has relatively fewer resting contacts.

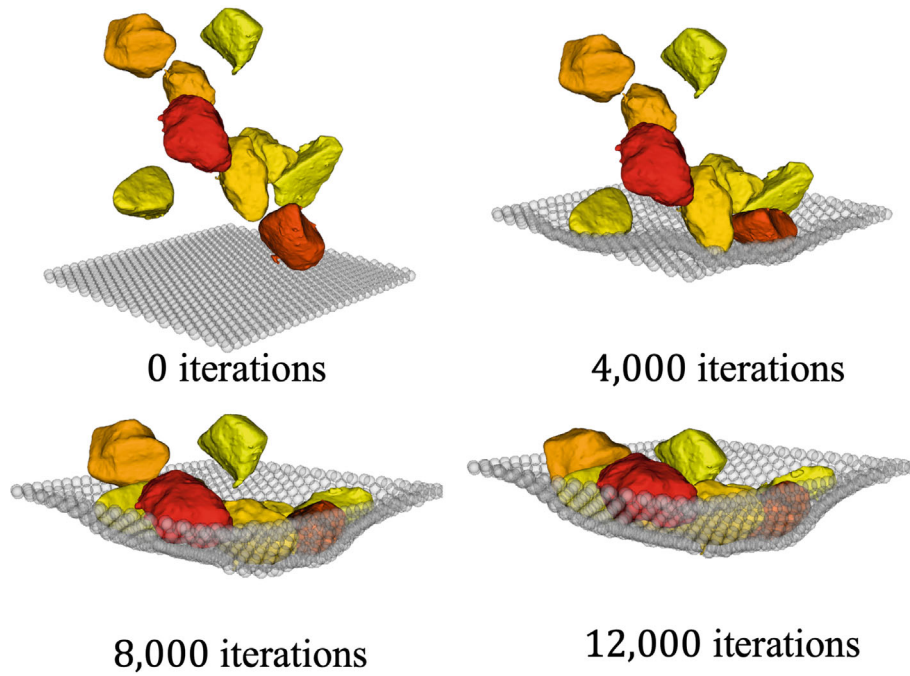


FIGURE 7 Flexible membrane model: Grains falling into a flexible net.

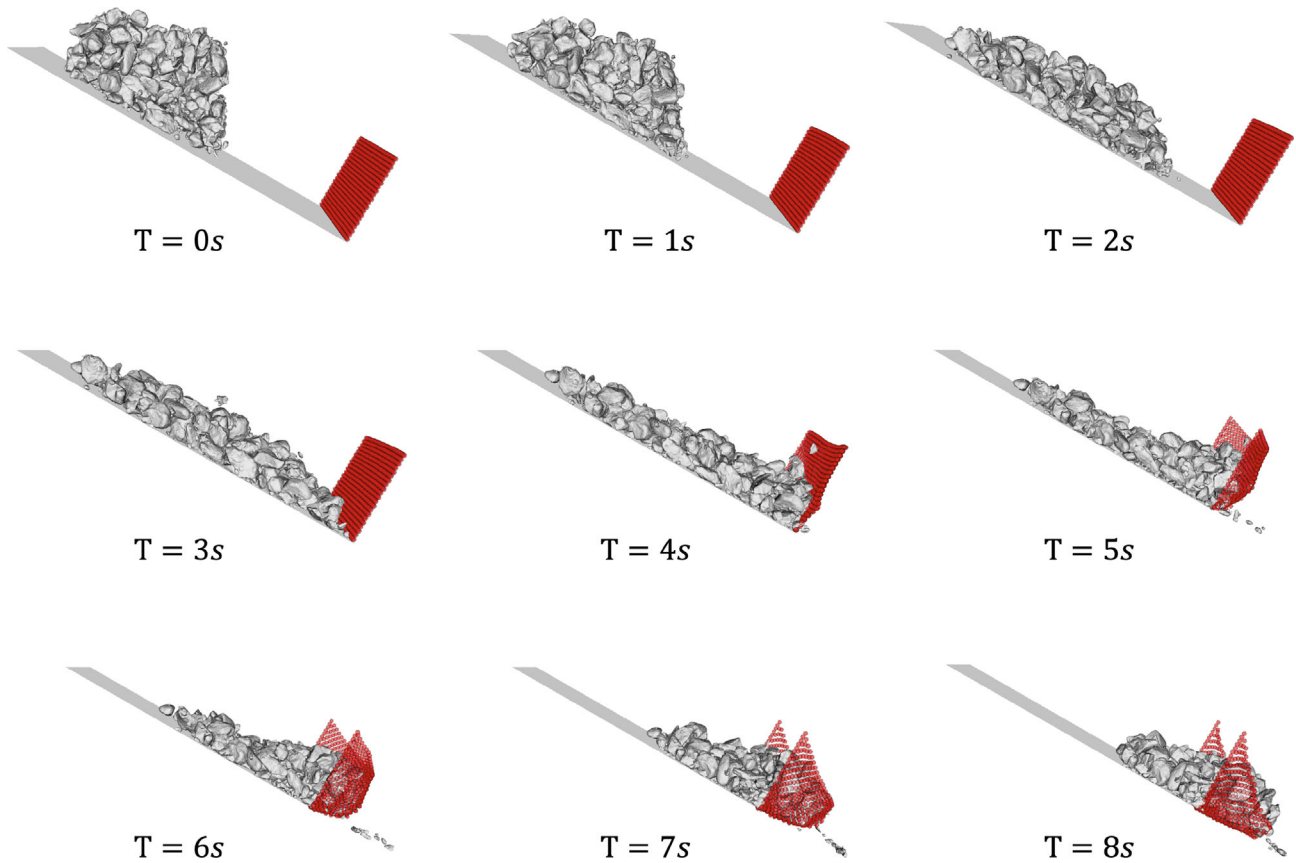


FIGURE 8 Demonstration of the rigid boundary-grain interaction and flexible membrane interaction with the impulse-based LS-DEM code.

4.3 | Weighted relative velocity implementation

Typically, the impulse-based approach is used to model similar-sized objects or to decompose large objects into uniform elements,¹¹ implying that the collision matrix between attached objects is of the same order of magnitude. This does not hold for our application, as we attempted to simulate a granular system of particles with a wide range of grain shapes and sizes, resulting in colliding pairs having significantly different masses. Indeed, this may introduce numerical oscillation, causing small grains to be bounced back and forth unnecessarily due to their greater sensitivity to applied impulses. To address this issue, we modified the original ETM algorithm by considering weighted relative normal velocity, defined as: $\tilde{\mathbf{u}} = \frac{m_1 m_2}{m_1 + m_2} \mathbf{u}$, where m_1 and m_2 are the masses of the colliding pairs. In this way, the contact points between larger grains are resolved earlier. This approach is superior to simply relying on relative velocity to determine the sequence of contacts, because larger grains respond more gently to impulses, whereas tiny grains tend to oscillate violently even with insignificant amounts of impulses. In addition, this change enables larger grains to achieve desired post-collision velocities more quickly, and when smaller grains take control of the algorithm the resulting update has a negligible effect on larger grains.

5 | PARALLEL IMPLEMENTATION OF IMPULSE-BASED LS-DEM

To improve performance, we employ a domain decomposition strategy and a binning algorithm is implemented through a series of linked lists that map relationships between grains and bins with a search complexity of $O(1)$.^{13,14} There are two types of data communication to consider: border communication and grain migration. The first type is treated using an efficient algorithm consisting of three sequential calls to the optimally tuned MPI routine MPI_Sendrecv to update exchange halo layers and automatically handle edge cases. While grain migration across sub-domains is a complicated step in a penalty-based LS-DEM,¹⁴ it is much easier to handle in an impulse-based method because it does not track the evolution of friction forces, allowing all quantities to be packed and communicated collectively, thereby lowering communication overheads.

The domain decomposition strategy divides a large computation task into smaller tasks and assumes that each sub-domain has access to all necessary resources to run independently. Because the collision reaction is associated with a change in the geometric configuration at the contact point, the change in acceleration must occur over a positive time interval. Indeed, the forces take some time to propagate throughout the body due to its elasticity. Collision contact, however, occurs within a very short time interval for a rigid body, which means that the displacement and the change in the contact area are negligible or remain unchanged. However, in the impulse-based formulation a contact island can span an arbitrary number of sub-domains as illustrated by the grid in Figure 3, resulting in grains within a sub-domain becoming indistinguishable from those in other sub-domains. Thus, the impulse-based method requires one processor to gather and resolve all collisions in a contact island.

The force resolution step of the impulse-based method uses an $O(n)$ binning algorithm. This step identifies and registers all contact points that violate impenetrability constraints, including contact IDs, branch vectors, and the normal direction of a collision between two avatars. Following that, the master MPI processor (rank 0) collects these contacts and summarizes a list of contact IDs, determining the number of contact islands and grains contained within. The concept of obtaining contact islands was inspired by the fact that colliding bodies make contact with one another, and that the system of all colliding pairs forms an undirected graph. As a result, the problem of determining contact islands becomes a graph partition problem of determining all connected components. Additionally, we can consider the graph as a sparse, symmetric square matrix whose dimension equals the number of grains and whose entries are non-zero only when the corresponding pair of grains collides. When the problem is rephrased as permuting such a matrix into a band matrix with a small bandwidth, the Cuthill-McKee algorithm²⁷ is used, which is based on the graph's Breath First Search (BFS) algorithm. Algorithm 2 contains the Cuthill-McKee algorithm optimized for our application. Following the identification of contact islands, collisions between rigid bodies are resolved within each contact island, which may span multiple sub-domains. To minimize data communication, a processor gathers a contact island that already contains the majority of collisions. Due to the small and specific size of the data being communicated, it can be packed and transferred efficiently and collectively. After collision resolution is complete, the change in the velocity of each grain is computed and distributed back to the grain's original process, which updates the velocity of the grain and advances the time step. This procedure formalizes the parallel strategy used in the Algorithm 3 for impulse-based LS-DEM.

Algorithm 2. CutHill-Mckee algorithm

Input: A list L^{ID} with length of N_{grains} including IDs of contact neighbors for all grains in assembly.

Output: the number of contact islands in an assembly N_{islands} , a list Λ^{islands} of size N_{islands} such that $\Lambda^{\text{islands}}[i]$ denotes IDs of elements in the i -th contact island.

```

1: From  $L^{\text{ID}}$  generate a list  $L^{\text{degree}}$  denoting the degree of an element, and a list  $L^{\text{checked}}$  denoting if an element is looked up already.
2: Set the number of contact islands  $N_{\text{islands}} = 0$ , the total number of checked elements  $N_{\text{checked}} = 0$ ; and prepare a priority queue  $\Lambda^d$  for elements in the assembly, sorted by the minimum degree.
3: while  $N_{\text{checked}} < N_{\text{grains}}$  do
4:   Iterate  $L^{\text{degree}}$  to find an element  $i$  with minimum degree among all unchecked elements.
5:   Sort all elements in  $L^{\text{ID}}[i]$  into  $\Lambda^d$  if they are not already, and set  $L^{\text{checked}}[i] = \text{true}$ .
6:   while  $\Lambda^d$  is not empty do
7:     Pop out the first element  $j$  from  $\Lambda^d$ .
8:     if  $j$  is not already in  $\Lambda^{\text{islands}}[N_{\text{islands}}]$  then
9:       Insert  $j$  into  $\Lambda^{\text{islands}}[N_{\text{islands}}]$ , and set  $L^{\text{checked}}[j] = \text{true}$ .
10:      Sort all elements in  $L^{\text{ID}}[j]$  into  $\Lambda^d$  if not already.
11:    end if
12:  end while
13:   $N_{\text{checked}} \leftarrow N_{\text{checked}} + |\Lambda^{\text{islands}}[N_{\text{islands}}]|$ 
14:   $N_{\text{islands}} \leftarrow N_{\text{islands}} + 1$ 
15: end while

```

Algorithm 3. Algorithm to parallel contact islands

Input: MPI rank i maintains a list Λ_C^i of contact details Φ_C including contact IDs of colliding bodies A and B; branch vectors from centers to contact point \mathbf{r} ; contact normal direction $\hat{\mathbf{n}}$; relative velocity $\hat{\mathbf{u}}$; elastic energy W and body mass M , such that $\Lambda_C^i := \{\Phi_C : \Phi_C \in \text{rank } i\}$.

```

1: /* Start Graph Partition to Detect Contact Islands */
2: if  $\text{rank\_id} \neq 0$  then
3:   Generate a list  $\Lambda_{\text{ID}}^i$  contains IDs of contacting neighbors for each grain belongs to the rank.
4:   MPI_Send  $\Lambda_{\text{ID}}^i$  to rank 0.
5: else
6:   for  $\text{rank\_id} = 1 : N_{\text{ranks}}$  do
7:     MPI_Recv  $\Lambda_{\text{ID}}^i$  from rank  $\text{rank\_id}$ .
8:   end for
9:   /* Prepare For the CutHill-McKee Algorithm */
10:  Build a list  $L^{\text{ID}}$  with length of  $N_{\text{grains}}$  to register contact IDs for all grains in assembly.
11:  Prepare a list  $\Lambda^{\text{islands}}$  of size  $N_{\text{islands}}$  such that  $\Lambda^{\text{islands}}[i]$  denotes IDs of elements in the  $i$ -th contact island.
12:   $[N_{\text{islands}}, \Lambda^{\text{islands}}] = \text{CutHill-McKee}(L^{\text{ID}})$ .
13:  for  $\text{island\_id} = 1 : N_{\text{islands}}$  do
14:    determine the host rank that contains most collisions of the  $i$ -th contact island, and register the host ranks in list  $L^{\text{host}}$  with length of  $N_{\text{islands}}$ .
15:  end for
16: end if
17: /* Broadcast Contact Islands Information */
18: MPI_Bcast  $\Lambda^{\text{island}}$  and  $L^{\text{host}}$ .
19: /* All-To-All Communicate Contact Detail  $\Phi_C$  */
20: MPI_Alltoall number of contact details that each rank should receive from others.
21: MPI_Alltoallv specific contact details that each rank should receive from others.
22: /* Collision-Resolution Phase */
23: for  $\text{island\_id} = 1 : N_{\text{islands}}$  do
24:   if  $L^{\text{host}}[\text{island\_id}] == \text{rank\_id}$  then
25:     Resolve collisions via Collision-Resolution Algorithm.
26:   end if
27: end for
28: /* Broadcast and Update Velocities */
29: MPI_Allgatherv linear and angular velocities from other ranks.

```

The flowchart of the parallel impulse-based LS-DEM code is shown in Figure 9. There are several implementation details worth noting. As the basic element of impulse-based method is an island of collisions which could spread over multiple processors, each processor still needs to generate a simplified avatar containing mass, moment of inertia for those that do not belong to the sub-domain, and it has to keep track of the position, rotation, velocity, and angular velocity of all grains. In addition, both post-collision velocity that is attained from solving a contact island and the corrected velocity after interacting with rigid boundaries must be broadcast to all processors before updating a grain's motion. Moreover,

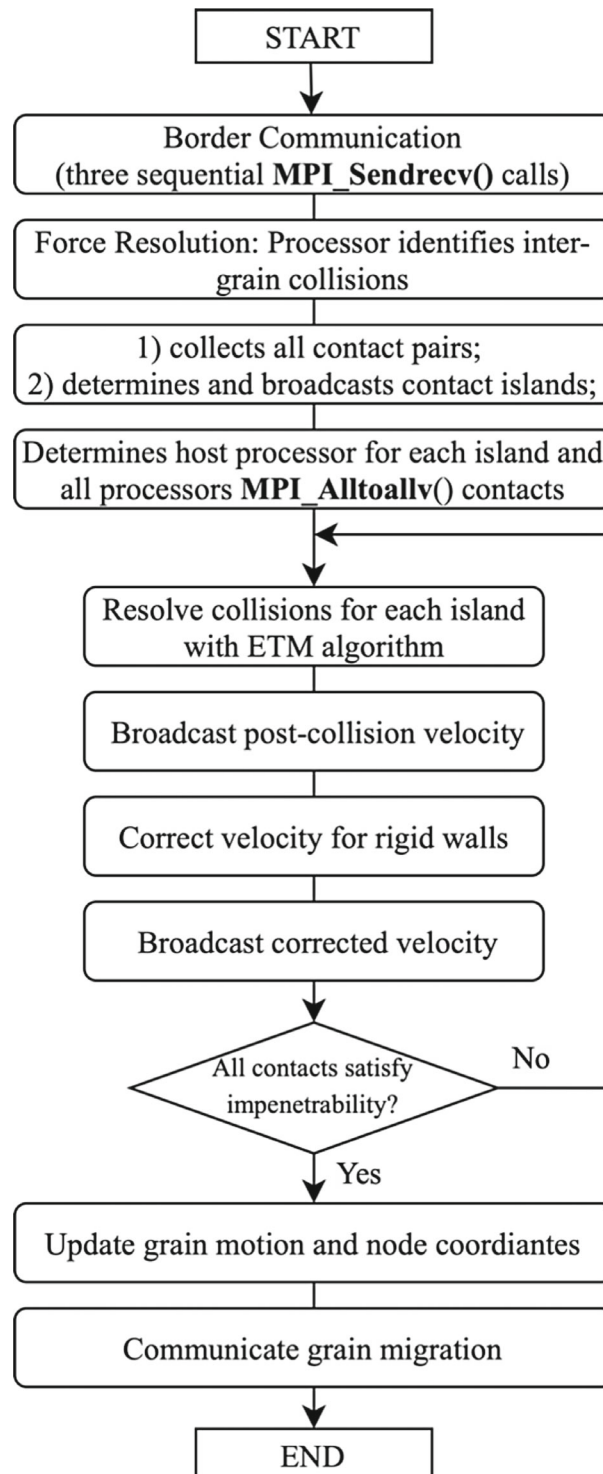


FIGURE 9 Flowchart of parallelized impulse-based method.

since each processor interacts with its neighbors via halo layers, the grain-halo contacts are duplicated when contacts in an island that spreads across multiple processors are gathered to one processor, hence half of interactions with halo layers need to be removed.

6 | NUMERICAL TESTS

6.1 | Block sliding on a plane

The accuracy of the code is demonstrated using a block sliding on an inclined surface, which has an exact analytical solution. In this case we model a cubic block sliding on a surface inclined at 30° and coefficient of friction equivalent to 20° and 25° , respectively. The results of the simulations are presented in Figure 10. In this case we used very low modulus to allow the block to slide along a rigid surface, as has been suggested by Reference 12. As the plots show, the numerical and analytical solutions are in excellent agreement, as is required. In this context the results validate both the contact algorithm and the frictional interface model.

6.2 | Performance speedups with impulse-based LS-DEM

The speed-up of the impulse-based formulation was investigated by modeling the flow of 1600 arbitrarily complex shaped grains under artificial gravity from a smooth container, as shown in Figure 11. The grains in the assembly have wide range of grain shapes and sizes, forcing the time-step of the conventional penalty-based approach to be very small. Table 1 shows model parameters for the penalty-based and the impulse-based LS-DEM. The values for inter-grains friction coefficient μ , normal contact stiffness K_n , and shear contact stiffness K_s were adopted from Reference 21. The coefficient of normal restitution according to Stronge's hypothesis is taken $R_n = 0.65$, which was obtained by Reference 12 by conducting experiments. The coefficient of tangential restitution R_t needs also to be calibrated, but many impulse-based simulators show good physical plausibility with a default value of zero.¹⁰ Global damping is $\xi_g = 0.5$ for both approaches. Different time steps for impulse-based formulation (Δt_i) were chosen to study the speed up as well as the simulation fidelity over the reference penalty-based simulation. The time step of penalty-based LS-DEM was computed by assuming a factor of safety 0.1 and the value 2.5×10^{-4} is selected for the ease of comparison.

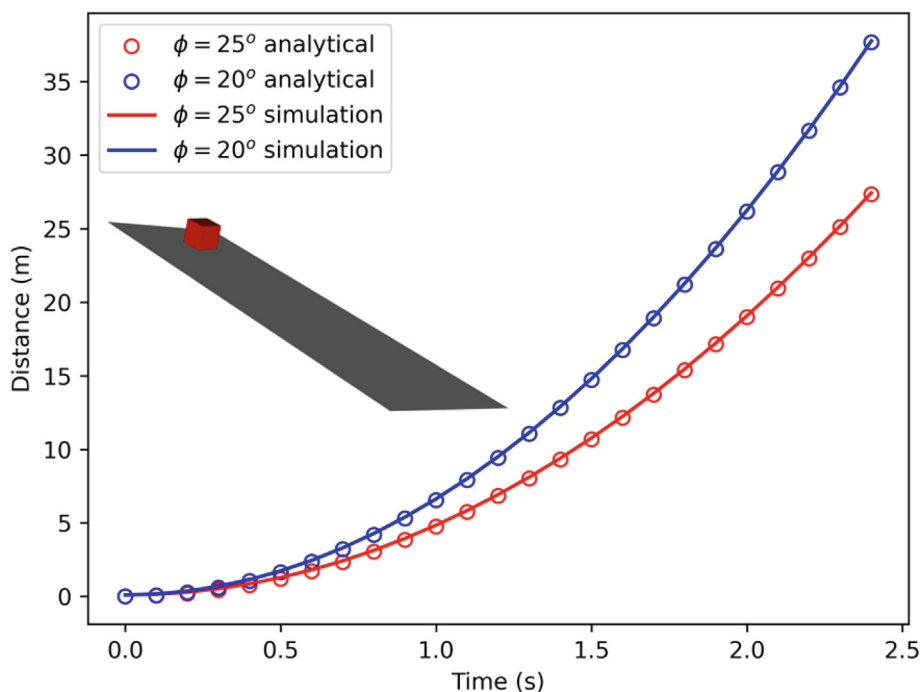


FIGURE 10 Velocity versus distance for a block sliding on a plane inclined at 30 degrees as a function of interface friction angle, ϕ .



FIGURE 11 Initial configuration of 1600 very high-resolution grains in $800 \times 800 \times 800$ domain for impulse-based method with large time steps.

TABLE 1 List of model parameters and values used for the simulations.

Common parameters					
Specific gravity of solids		G_s		2.65	
Inter-grain friction coefficient		μ		0.55	
Global damping coefficient		ξ_g		0.5	
Penalty-based LS-DEM			Impulse-based LS-DEM		
Normal contact stiffness	K_n	3×10^4	Coefficient of normal restitution	R_n	0.65
Tangential contact stiffness	K_s	2.7×10^4			

$$\Delta t = 0.1 \times \sqrt{\frac{K_n}{M_{\min}}} \approx 2.98 \times 10^{-4} \quad (50)$$

In the simulations, the outer walls of the container were removed and the sand was allowed to flow out. The geometries of the resulting mounds, approaching the angle of repose, were compared at the end of 8 s time interval. In each time step, the post-collision velocities were corrected for rigid boundary contact until the computed velocities did not violate either inter-grain or grain-wall impenetrability. In this example, we repeated this procedure several times within a single time step, and we found 10 iterations were enough to stabilize the velocity of grains. The code speed-up was measured in terms of central processing unit (CPU) time. Five simulations using impulse-based formulation were conducted, with the time step used varying from $\Delta t_i = 1\Delta t = 2.5 \times 10^{-4}$ to $\Delta t_i = 16\Delta t = 4.0 \times 10^{-3}$. Figure 12 shows the mound geometry for each simulation and Table 2 tabulates the speed up times for the different time steps in the impulse-based simulations. The simulation fidelity can be checked in terms of final height and spread of mound. Although all simulations produced similar shaped mounds, discrepancies in positions and rotations of individual grain can be large. Calibrating impulse-based method to achieve reasonable agreement to the reference penalty-based simulation is not easy, as it requires the identification of actual or ad hoc internal variables and constitutive relations. In addition, there are too many degrees of freedom even for a single grain and to make a one-to-one comparison is not feasible. Instead, we aimed to explore to what extent both methods agree with each other in a qualitative sense. At the same time we found that the simulation results, even

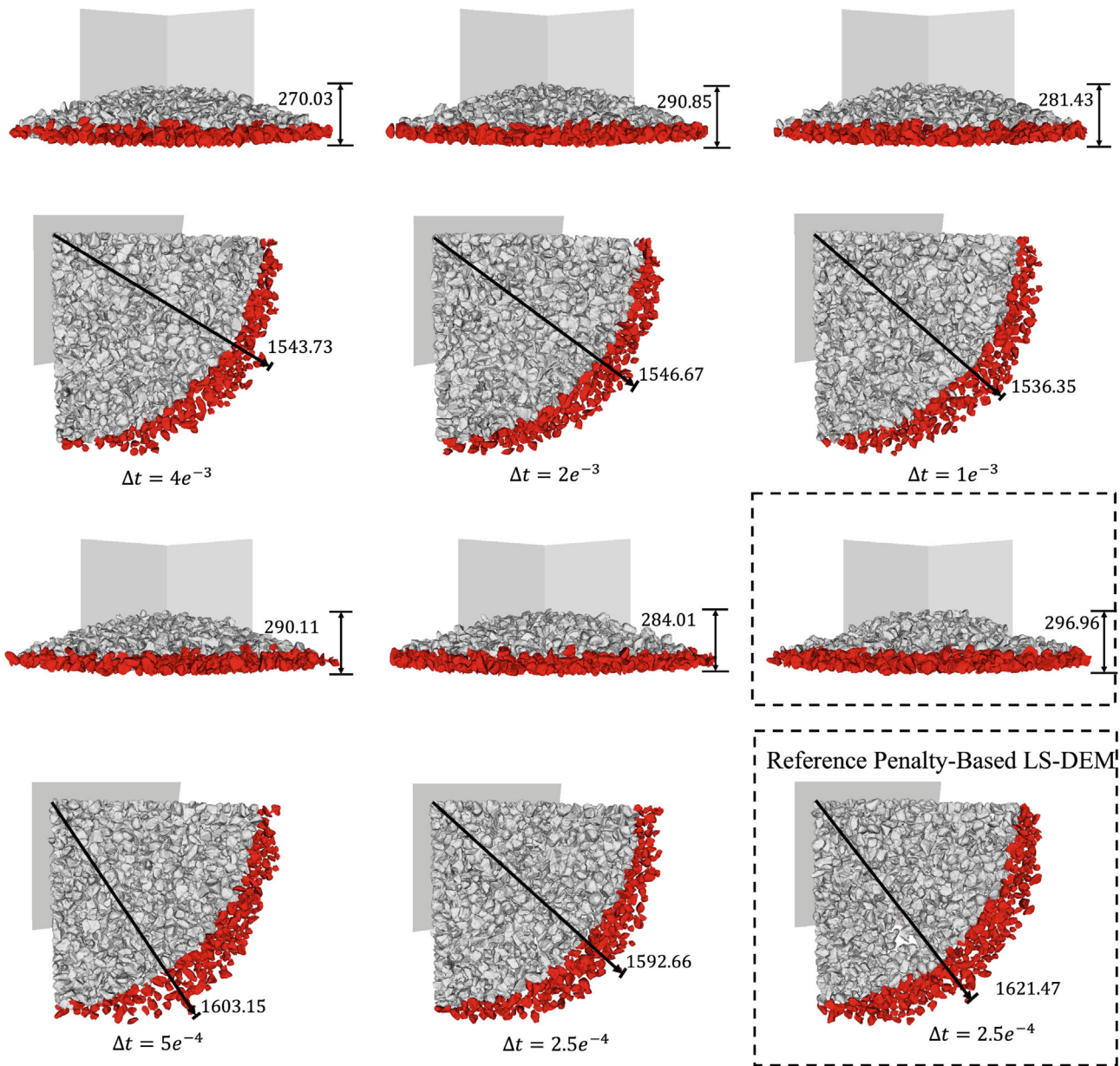


FIGURE 12 Specimen height and spread after 8s settlement under gravity with varying time steps.

TABLE 2 Comparison of CPU time and speed up between the different methods.

	Δt (sec)	CPU time (mins)	Time step diff	Speed-up
Penalty-based	2.5×10^{-4}	~ 4000		
Impulse-based	2.5×10^{-4}	265	$1\Delta t$	15.1
	5.0×10^{-4}	132	$2\Delta t$	29.4
	1.0×10^{-3}	72	$4\Delta t$	55.6
	2.0×10^{-3}	42	$8\Delta t$	95.2
	4.0×10^{-3}	42	$16\Delta t$	95.2

using a very large time steps, showed no obvious penetration errors between particles and the geometric fidelity was maintained across all simulations.

In this simulation, the particles were reconstructed from a very high-resolution data set ($3.4\mu\text{m}/\text{pixel}$) and each grain was represented by 2561 nodes on the average and storing 1600 grains required 3.36 GB of memory. Using such high-resolution avatars for large scale simulation with the penalty-based method is challenging in with an efficient parallel code,¹⁴ because actual workload corresponds to the total number of nodes rather than the number of grains. In contrast, even though impulse-based method still interpolates normal direction from the LS grids, the impulse-based formulation does not compute, update, and track complicated history-dependent friction. This simplified force resolution and helps to keep LS grids in a cache and enables continuous, fast access. In addition, the amount of time required for sub-iterations to resolve multiple collisions is less demanding because doing arithmetic operations is 100 times faster than moving data around in modern machines.

Overall, significant speed up is achieved in impulse-based method with larger time steps, which is not likely for conventional penalty-based DEM formulation. As grains settle down, the number of contact islands decreases and the number of contacts per island increases. We observed that as many as 300 contact islands still remained at the end of the simulation, the largest number of contacts in an island was more than 2000, which implies that most of the grains were grouped into several clusters and the time used to resolve collisions increased. We also found that the number of contact islands and contacts increases with the time step, as a larger time step changes configuration more rapidly and adds more collisions. The ratio of mass differences between the largest and the smallest grains in this example was around 30, and we found that at this ratio the algorithm remained stable and converged at different time steps, accelerations, and boundary conditions.¹⁴

6.3 | Example rock avalanche simulation

Even though the impulse-based formulation allows using much larger time steps and eliminates substantial amount of the demand to compute forces, the fundamental issue of a particle-based method to model an assembly with wide range of object sizes still remains. In that context, modeling the topography presents an analogous problem.

Any arbitrarily shaped surface, such as a real topography with ridges and valleys, can be captured using the LS framework. In this example, the LS algorithm was first used to locate the zero-valued contour, from which the fast-marching method²⁸ was used to construct the underlying signed distance grid by solving a boundary value problem of Eikonal equation:

$$|\nabla\phi(x)| = \frac{1}{f(x)} \quad \text{for } x \in \Omega \quad (51)$$

$$\phi(x) = 0 \quad \text{for } x \in \partial\Omega \quad (52)$$

Such a problem describes the evolution of a surface as a function of LS ϕ with speed $f(x)$ in the normal direction at a point x on the propagating surface. Alternatively, $\phi(x)$ can be thought of as the shortest time required to reach x starting from the zero contour, of course, $\phi(x) = 0$ for x on the zero LS contour $\partial\Omega$. The algorithm is similar to Dijkstra's algorithm and uses the fact that information only flows outward from the area that is already labelled with a value. Thus, a digitized topography and its LS representation are constructed as a large LS-DEM object to exploit algorithmic advantages and we treat grain-topography interaction identically to grain-grain interaction. In this case, the digitized topography occupies the entire computational domain and consumes as much as 10GB computer memory. To avoid large memory demand, we only saved a narrow band of the grid adjacent to the zero LS contours, which is sufficient to interpolate the amount of penetration between objects and keep the memory requirement low.

We show three examples of simulated rock avalanches in which a granular deposit assembly was allowed to slide as gravity overcame the internal friction of the deposit. The digitized topography is inclined at 25° , and the coefficient of internal friction between the particle and the topography is 0.2. Larger particles are red and smaller particles are orange. In all cases, the particle mass moves fluid-like down the stream channel. In the first example (Figure 13), the avalanche ran out on a flat plane and formed fan-shaped deposit, as is typical of rock avalanche deposits. In the second example, shown in Figure 14, the deposit ran out onto a surface sloping at only 3° , again similar to many natural settings, and was deflected down slope. In the third example (Figure 15, we simulated a flexible barrier, such as is often used for rock fall protection. In this case, the avalanche was initially contained until it overflowed the barrier. Our interest in these

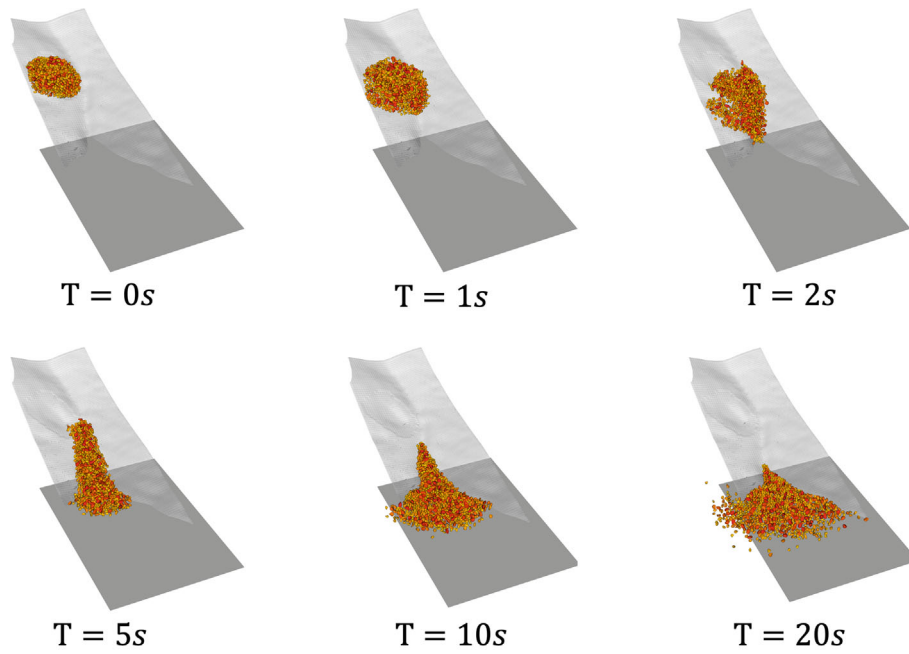


FIGURE 13 Rock avalanche runout onto a horizontal surface.

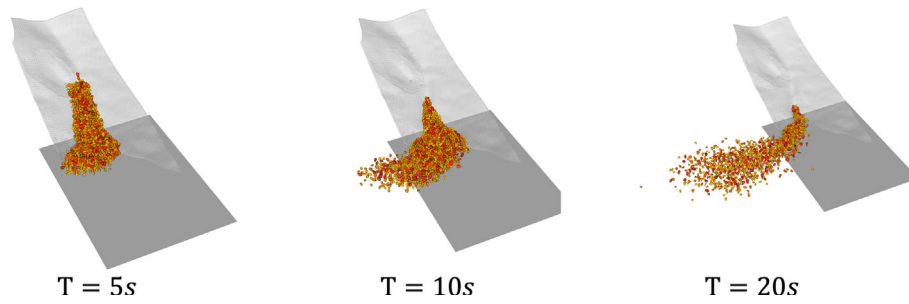


FIGURE 14 Rock avalanche runout onto a sloping surface.

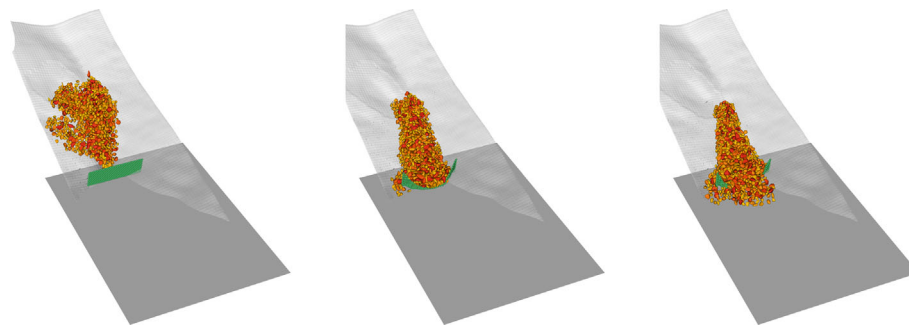


FIGURE 15 Rock avalanche impact on a protection net.

simulations was to explore the potential of the impulse-based DEM to realistically handle field-size problems rather than solving a particular problem. The first two simulations ran on a single CPU and completed 20,000 steps in 2 h, while the third simulation completed the same number of steps in 5 h due to the more complicated membrane-grain interaction. Overall, the results are very promising, as rock avalanches have been typically modeled as an equivalent viscous fluid (see e.g., References 29,30), rather than as an assemblage of particles, in order to make the simulation tractable.

7 | CONCLUSIONS

We demonstrate that the impulse-based DEM is attractive in large part because it alleviates the need to choose regular time intervals and saves substantial amount of force computation for complex shaped objects. The combined effects speed-up the simulation by at least factor of 10 with reasonable levels of fidelity. We modified the ETM algorithm making it more suitable for dealing with a system with wide range of particle shapes and sizes. Instead of removing the smallest relative velocity from priority queue each time, we prioritize the weighted relative velocity which considers the masses of colliding bodies for better convergence. This measure is effective because it first considers larger objects, as later updates of small objects trivially influence the larger objects. Compared to SMM and SQM, the ETM algorithm is advantageous in that it is less sensitive to the order of impulses and tends to be more energy conservative. By applying impulses gradually to the contact points, the impulses can influence multiple contact points at the same time and enable propagation of impulse effects and improve the overall ability of the method to capture propagation of forces during dynamics simulations. We further incorporated the impulse-based formulation into an LS framework. We also used a novel time integration scheme that separates the treatment of inter-grain collision and wall-grain resting contact. This improves the robustness of the collision resolution algorithm, thereby avoiding numerical oscillation and unrealistic energy dissipation. Also, the simulation of protection nets demonstrates that impulse-based methods can be used to simulate the dynamics of deformable structures.

The major limitation of the impulse-based method still lies in modeling a system of irregular, non-convex, non-uniform objects especially in a highly confined quasi-static setting, whereby objects with very different shapes and sizes interact with each other at many contact points and have small velocities. However, our modified ETM algorithm and the time integration scheme are capable of simulating a range of very different objects and a range of dynamic phenomena. Thus, the current work provides a starting point for further enhancement and development of a more robust analysis tool to probe contact forces and impulses for wider range of problems.

ACKNOWLEDGMENTS


We thank Prof. José Andrade and Dr. Reid Kawamoto for sharing their LS-DEM code and for their assistance in mastering its contents. We also thank Bodhinanda Chandra for illuminating many aspects of the impulse-based formulation and the ETM algorithm in particular. Funding for this research was provided by the National Science Foundation under grant CMMI-1853056, the Edward G. and John R. Cahill Chair, the Berkeley-France Fund, the US Geological Survey Cooperative Agreement G17AC00443, and the Pacific Earthquake Engineering Center (PEER), solicitation TSRP 2018-01. The opinions, findings, and conclusions expressed in this publication are those of the authors and do not necessarily reflect the views of the study sponsors: the National Science Foundation (NSF), the Pacific Earthquake Engineering Research Center (PEER), the Regents of the University of California, or the US Geological Survey.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Hasitha S. Wijesuriya  <https://orcid.org/0000-0003-3937-1629>

Nicholas Sitar  <https://orcid.org/0000-0001-7253-5985>

REFERENCES

1. Zohdi TI. *Modeling and Simulation of Functionalized Materials for Additive Manufacturing and 3d Printing: Continuous and Discrete Media: Continuum and Discrete Element Methods*. Vol 60. Springer; 2017.
2. Owen D, Feng Y. Parallelised finite/discrete element simulation of multi-fracturing solids and discrete systems. *Eng Comput*. 2001;18(3/4):557-576. doi:10.1108/02644400110387154

3. Zheng J, An X, Huang M. GPU-based parallel algorithm for particle contact detection and its application in self-compacting concrete flow simulations. *Comput Struct*. 2012;112:193-204. doi:10.1016/j.compstruc.2012.08.003
4. Yan B, Regueiro RA. A comprehensive study of MPI parallelism in three-dimensional discrete element method (DEM) simulation of complex-shaped granular particles. *Comput Particle Mech*. 2018;5(4):553-577. doi:10.1007/s40571-018-0190-y
5. Yan B, Regueiro R. Comparison between $O(n^2)$ and $O(n)$ neighbor search algorithm and its influence on superlinear speedup in parallel discrete element method (DEM) for complex-shaped particles. *Eng Comput*. 2018;35(3):2327-2348. doi:10.1108/EC-01-2018-0023
6. Mirtich B, Canny J. Impulse-based simulation of rigid bodies. Proceedings of the 1995 Symposium on Interactive 3D Graphics. 1995;181-188. doi:10.1145/199404.199436
7. Chang B, Colgate JE. Real-time impulse-based simulation of rigid body systems for haptic display. Proceedings of the 1997 ASME International Mechanical Engineering Congress and Exhibition. 1997;145-152. doi:10.1115/IMECE1997-0389
8. Bender J. Impulse-based dynamic simulation in linear time. *Comput Animat Virtual Worlds*. 2007;18(4-5):225-233. doi:10.1002/cav.179
9. Tang X, Paluszny A, Zimmerman RW. An impulse-based energy tracking method for collision resolution. *Comput Methods Appl Mech Eng*. 2014;278:160-185. doi:10.1016/j.cma.2014.05.004
10. Lee SJ, Hashash YM. iDEM: An impulse-based discrete element method for fast granular dynamics. *Int J Numer Methods Eng*. 2015;104(2):79-103. doi:10.1002/nme.4923
11. Asai M, Li Y, Chandra B, Takase S. Fluid-rigid-body interaction simulations and validations using a coupled stabilized ISPH-DEM incorporated with the energy-tracking impulse method for multiple-body contacts. *Comput Methods Appl Mech Eng*. 2021;377:113681. doi:10.1016/j.cma.2021.113681
12. Li Y, Asai M, Chandra B, Isshiki M. Energy-tracking impulse method for particle-discretized rigid-body simulations with frictional contact. *Comput Particle Mech*. 2021;8(2):237-258. doi:10.1007/s40571-020-00326-5
13. Tan P, Sitar N. Parallel implementation of LS-DEM with hybrid MPI+OpenMP. *Comput Geotech*. 2024;172:106408. doi:10.1016/j.compgeo.2024.106408
14. Tan P, Sitar N. *Parallel Level-Set DEM (LS-DEM) Development and Application to the Study of Deformation and Flow of Granular Media*. Tech. Rep. 2022/06. Pacific Earthquake Engineering Center (PEER), UC Berkeley; 2022.
15. Stronge WJ. *Impact Mechanics*. Cambridge University Press; 2018.
16. Mirtich B. *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis. University of California; 1996.
17. Evans DJ, Murad S. Singularity free algorithm for molecular dynamics simulation of rigid polyatomics. *Molecul Phys*. 1977;34(2):327-331. doi:10.1080/00268977700101761
18. Barzel R, Barr AH. A modeling system based on dynamic constraints. Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques. 1988;179-188. doi:10.1145/54852.378509
19. Baraff D. Analytical methods for dynamic simulation of non-penetrating rigid bodies. Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques. 1989;223-232. doi:10.1145/74334.74356
20. Guendelman E, Bridson R, Fedkiw R. Nonconvex rigid bodies with stacking. *ACM Trans Graph*. 2003;22(3):871-878. doi:10.1145/882262.882358
21. Kawamoto R, Andò E, Viggiani G, Andrade JE. Level set discrete element method for three-dimensional computations with triaxial case study. *J Mech Phys Solids*. 2016;91:1-13. doi:10.1016/j.jmps.2016.02.021
22. Walton O, Braun R. *Simulation of Rotary-Drum and Repose Tests for Frictional Spheres and Rigid Sphere Clusters*. Lawrence Livermore National Lab; 1993.
23. Thornton C. Numerical simulations of deviatoric shear deformation of granular media. *Géotechnique*. 2000;50(1):43-53. doi:10.1680/geot.2000.50.1.43
24. Moore M, Wilhelms J. Collision detection and response for computer animation. Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques. 1988;289-298. doi:10.1145/378456.378528
25. Wriggers P, Laursen TA. *Computational Contact Mechanics*. Vol 2. Springer; 2006.
26. Zohdi TI. Impact and penetration resistance of network models of coated lightweight fabric shielding. *GAMM-Mitteilungen*. 2014;37(1):124-150. doi:10.1002/gamm.201410006
27. Cuthill E, McKee J. Reducing the bandwidth of sparse symmetric matrices. Proceedings of the 1969 24th National Conference. 1969;157-172. doi:10.1145/800195.805928
28. Sethian JA. Fast marching methods. *SIAM Rev*. 1999;41(2):199-235. doi:10.1137/S0036144598347059
29. Setiasabda EY. *Material Point Method for Large Deformation Modeling in Geomechanics Using Isoparametric Elements*. PhD thesis. University of California; 2020.
30. Ho KK, Koo RC, Kwan JS. Mitigation of debris flows—research and practice in Hong Kong. *Environ Eng Geosci*. 2021;27(2):231-243. doi:10.2113/EEG-D-20-00009

How to cite this article: Tan P, Wijesuriya HS, Sitar N. 3-D impulse-based level-set method for granular flow modeling. *Int J Numer Methods Eng*. 2024;e7546. doi: 10.1002/NME.7546