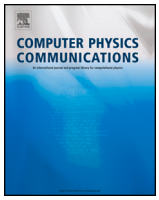
Computer Physics Communications 301 (2024) 109219

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc



Computer Programs in Physics

GeoTaichi: A Taichi-powered high-performance numerical simulator for multiscale geophysical problems ☆,☆☆

Y.H. Shi [a], N. Guo [a,*], Z.X. Yang [a,b]

[a] Computing Center for Geotechnical Engineering, Engineering Research Center of Urban Underground Space Development of Zhejiang Province, Department of Civil Engineering, Zhejiang University, Hangzhou, Zhejiang 310058, China
[b] Zhejiang Provincial Engineering Research Center for Digital & Smart Maintenance of Highway, China

ARTICLE INFO

Keywords:
Discrete element method
Material point method
Coupled material point-discrete element method
High performance computing
GeoTaichi

ABSTRACT

This study introduces GeoTaichi, an open-source high-performance numerical simulator designed for addressing multiscale geophysical problems. By leveraging the power of the Taichi parallel language, GeoTaichi maximizes the utilization of modern computer resources on multicore CPU and GPU architectures. It offers robust and reliable modules for the discrete element method (DEM), material point method (MPM), and coupled material point-discrete element method (MPDEM). These modules enable efficient solving of large-scale problems while being implemented in pure Python. The design philosophy of GeoTaichi focuses on creating a framework that is readable, extensible, and user-friendly. This paper highlights the coupling procedure of MPDEM, the code structures, and the most important features of GeoTaichi. Rigorous benchmark tests have been conducted to verify the validity and robustness of GeoTaichi. Additionally, the performance of GeoTaichi is compared with similar software tools in the field, underscoring a notable improvement in both computational efficiency and memory savings when compared to existing alternatives.

Program summary

Program title: GeoTaichi
CPC Library link to program files: https://doi.org/10.17632/858bmcf7j6.1
Developer's repository link: https://github.com/Yihao-Shi/GeoTaichi
Licensing provisions: GNU General Public License v3.0
Programming language: Python
Nature of problem: The simulations of large-deformation geophysical flows and their interaction with structures play a crucial role in the field of geophysics. To address the complexities of these nonlinear problems, the discrete element method (DEM), material point method (MPM), and their coupling (MPDEM) have proven to be highly suitable numerical schemes. However, these schemes impose substantial computational demands, necessitating the development of an efficient framework that can harness modern computer resources on multicore CPU and GPU architectures.
Solution method: The open-source code GeoTaichi implements the DEM, MPM, and coupled MPDEM, encompassing a range of constitutive models and contact laws for different geologic materials. The clump particle model is also introduced in DEM to solve granular mechanics involving complex-shaped particles. One significant advantage of GeoTaichi is its utilization of the Taichi parallel language, which is designed to be user-friendly and easily extensible for customized applications.

☆ The review of this paper was arranged by Prof. Andrew Hazel.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: nguo@zju.edu.cn (N. Guo).

https://doi.org/10.1016/j.cpc.2024.109219
Received 24 January 2024; Received in revised form 13 April 2024; Accepted 24 April 2024
Available online 30 April 2024

1. Introduction

Geophysical flows entail a vast number of discrete particles, presenting significant challenges for modeling using feasible numerical methods and accessible computational resources. The difficulties encountered by numerical methods stem from the intricate constitutive behavior of geologic materials and the large deformations associated with geophysical flows. Broadly speaking, numerical methods can be classified into two categories: continuum-based and discrete approaches.

The continuum-based approaches describe the mechanical behavior of a material using conservation and constitutive laws, which help reduce computational costs. Among these approaches, the material point method (MPM) [1] has been gaining popularity, particularly for modeling large-deformation geophysical flows. MPM discretizes a continuum body using a set of material points that can freely move on a fixed background mesh, eliminating potential mesh entanglement seen in traditional mesh-based methods. Moreover, the governing equations are solved on the background mesh, avoiding the neighbor search procedure commonly used in other meshfree methods such as the smoothed particle hydrodynamics (SPH) [2]. This significantly reduces computational costs. However, the constitutive laws in continuum-based approaches often oversimplify the mechanical responses of geologic materials, overlooking their discrete nature as typical granular materials. Therefore, discrete methods, exemplified by the discrete element method (DEM) [3], are widely used for simulating granular materials. DEM can accurately model the geometry of granular materials and calculate contact forces at the particle level, allowing for capturing the microscopic behavior of granular systems under external loading [4–6]. However, the direct application of DEM to large-scale problems is challenging due to its high computational costs [7].

Furthermore, to capture the multiscale characteristics in geophysical flows, where both macro- and microscale properties are of interest, the coupled material point-discrete element method (MPDEM) has recently garnered great interests [8–10]. Liu et al. [11] introduced a unified contact algorithm within a pure MPM framework to simulate granular flow and its impact on blocks. The model discretizes DEM blocks into several material points and adopts the contact model from MPM to detect interactions between the granular material and blocks. One major criticism of this method is the lack of accuracy in contact force calculation when multiple material boundary conditions coexist. To address this problem, Jiang et al. [12] proposed a hybrid MPM-spheropolygon DEM under a DEM-enriched contact framework. In this approach, material points are treated as ghost DEM particles during the contact force calculation process. The contact forces then act as body forces on the target particles following DEM penalty contact models. Under the same coupling framework, Ren et al. [13] extended the approach by incorporating a two-phase MPM and DEM coupling scheme to account for the influence of interstitial water. Similarly, Guilkey et al. [14] and Chen et al. [15] also employed a penalty method to address contact detection between material points and complex-shaped bodies.

Utilizing the parallel computational power of modern GPUs presents an appealing solution to address the computational requirements of particle-based numerical methods including MPM, DEM, and MPDEM. While several researchers have explored MPM or DEM algorithms on GPU architecture, less attention has been paid to GPU-accelerated MPDEM. To bridge this gap, Li et al. [16] proposed an MPDEM coupling framework that leverages GPU acceleration for MPM whereas retaining DEM on the CPU without optimization strategies. More recently, Feng et al. [17] employed a GPU-based MPDEM to enhance the computational efficiency of large-scale geologic hazard simulations. Nevertheless, their GPU-based code is developed as proprietary, in-house software.

In the past decade, there has been rapid development in open-source MPM software, including Anura3D (https://github.com/Anura3D), NairnMPM (https://osupdocs.forestry.oregonstate.edu/index.php/NairnMPM), and CB-Geo (https://github.com/cb-geo). Similarly, there are several open-source DEM projects such as Yade [18], Lammps [19], SudoDEM [20], MercuryDPM [21], and MUSEN [22]. However, open-source numerical simulators specifically designed for MPDEM, especially those based on GPU architecture, are relatively scarce. To the best of the authors' knowledge, the MPDEM approach is only implemented in the open-source software Kratos [23], while GPU-based MPDEM is currently being developed in-house and has not yet been published.

It is noteworthy that the aforementioned software packages are typically developed using low-level languages such as C++, CUDA, and Fortran to handle computationally intensive tasks. This can make it challenging for users to read and modify the source code for their specific applications. Just-in-time (JIT) compilation technology [24] has been developed to enable high-level languages like Python to run as fast as Fortran/C++. It complies Python code into native machine code during runtime, allowing for direct execution on hardware instead of within the Python virtual machine. Several high-level languages have incorporated JIT techniques, and one notable example is Taichi [25], which leverages multiple compiler technologies in its own intermediate representation layer, providing enhanced performance during both development and runtime on multicore CPU and GPU architectures. Taichi was originally designed to decouple computation from data structures and type systems. This allows developers to seamlessly shift between different data structures and efficiently pack more data per memory unit [26]. In recent years, Taichi has gained significant recognition as a powerful tool for coding high-performance solvers in scientific computing, including MPM [27], lattice Boltzmann method (LBM) [28], molecular dynamics (MD) [29], and phase field method (PFM) [30]. Additionally, Taichi provides automatic differentiation capabilities [31] and efficient mesh-based operations [32], which can be integrated with existing numerical frameworks. Therefore, the Taichi parallel language has been chosen for this study.

In summary, this paper presents GeoTaichi, an open-source multiscale numerical simulator designed for large-scale simulations of geophysical problems. GeoTaichi harnesses the power of multicore CPU and GPU architectures. The rest of the paper is organized as follows. The fundamentals of MPM and DEM are first presented in Section 2, followed by the GPU acceleration of contact detection in MPDEM in Section 3. The design philosophy and main features of GeoTaichi are introduced in Section 4. Validation and performance tests are presented in Sections 5 and 6, respectively. Finally, Section 7 concludes the paper.

2. MPM and DEM modules

2.1. Material point method

As illustrated in Fig. 1, the computational domain of MPM consists of a fixed Cartesian grid and a collection of discrete material points that represent the continuum body. The material points store physical properties of the continuum, such as mass, volume, velocity, and stress. Note that the illustration is presented in 2D for clarity. In this study, we employ the widely accepted update stress last (USL) [33] explicit time integration scheme for MPM, which involves the following three phases:

(1) Particle-to-grid (P2G) mapping:

$$m_I^t = \sum_p N_I(\mathbf{x}_p^t) m_p \tag{1}$$

$$\mathbf{p}_I^t = \sum_p N_I(\mathbf{x}_p^t) m_p \mathbf{v}_p^t \tag{2}$$

$$\mathbf{f}_I^{\text{ext},t} = \sum_p N_I(\mathbf{x}_p^t) m_p \mathbf{g} + \sum_p N_I(\mathbf{x}_p^t) \boldsymbol{\tau}_p^t V_p^t h^{-1} + \mathbf{f}^c \tag{3}$$

$$\mathbf{f}_I^{\text{int},t} = -\sum_p \nabla N_I(\mathbf{x}_p^t) \boldsymbol{\sigma}_p^t V_p^t \tag{4}$$

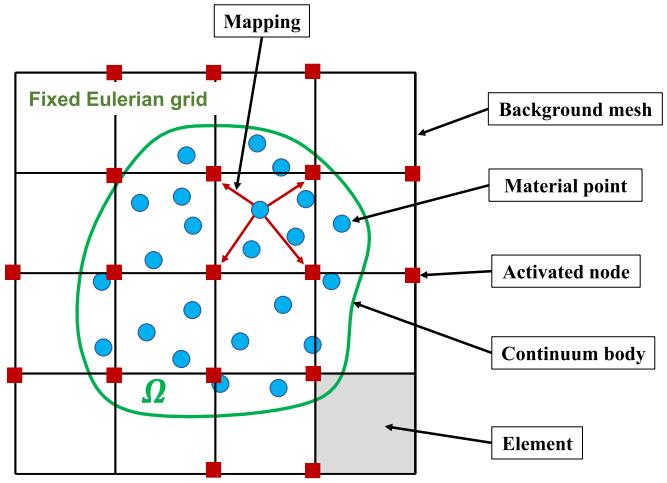


Fig. 1. Computational domain of the material point method. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

where m_p is the mass of a material point, which remains constant throughout the simulation so that the system maintains mass conservation automatically; $\mathbf{x}_p^t$, $\mathbf{v}_p^t$, $\boldsymbol{\sigma}_p^t$, $\boldsymbol{\tau}_p^t$, and V_p^t denote the material point position, velocity, stress, traction, and volume at time step t, respectively; $\mathbf{g}$ is the gravitational acceleration; h is the initial size of the material point; m_I^t, $\mathbf{p}_I^t$, $\mathbf{f}_I^{\text{ext},t}$, and $\mathbf{f}_I^{\text{int},t}$ denote the nodal mass, momentum, external force, and internal force of the grid, respectively; and $\mathbf{f}^c$ denotes the possible contact force. The shape function and its gradient are represented by N_I and ∇N_I, respectively.

(2) Update of nodal kinematics and imposition of essential boundary conditions:

$$\mathbf{f}_I^t = \mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t} - \zeta \|\mathbf{f}_I^{\text{ext},t} + \mathbf{f}_I^{\text{int},t}\| \frac{\mathbf{p}_I^t}{\|\mathbf{p}_I^t\|} \tag{5}$$

$$\mathbf{a}_I^{t+\Delta t} = \frac{\mathbf{f}_I^t}{m_I^t} \tag{6}$$

$$\mathbf{v}_I^{t+\Delta t} = \frac{\mathbf{p}_I^t}{m_I^t} + \mathbf{a}_I^{t+\Delta t}\Delta t \tag{7}$$

$$\begin{cases} \mathbf{v}_I^t = \mathbf{0} \\ \mathbf{a}_I^t = \mathbf{0} \end{cases} \quad \text{where } I \in \Gamma_D \tag{8}$$

where $\mathbf{a}_I^{t+\Delta t}$ and $\mathbf{v}_I^{t+\Delta t}$ are the updated nodal acceleration and velocity, respectively; ζ is the damping coefficient; Δt is the timestep size, and Γ_D represents the fixed Dirichlet boundary.

(3) Grid-to-particle (G2P) mapping:

$$\mathbf{v}_p^{t+\Delta t} = \alpha \underbrace{\sum_I N_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t}}_{\mathbf{v}_{p,\text{PIC}}^{t+\Delta t}} + (1-\alpha)\underbrace{(\mathbf{v}_p^t + \Delta t \sum_I N_I(\mathbf{x}_p^t)\mathbf{a}_I^{t+\Delta t})}_{\mathbf{v}_{p,\text{FLIP}}^{t+\Delta t}} \tag{9}$$

$$\mathbf{x}_p^{t+\Delta t} = \mathbf{x}_p^t + \Delta t \sum_I N_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t} \tag{10}$$

$$\mathbf{L}_p^{t+\Delta t} = \sum_I \nabla N_I(\mathbf{x}_p^t)\mathbf{v}_I^{t+\Delta t} \tag{11}$$

$$V_p^{t+\Delta t} = \det(\mathbf{I} + \mathbf{L}_p^{t+\Delta t}\Delta t)V_p^t \tag{12}$$

$$d\boldsymbol{\epsilon}_p^{t+\Delta t} = \frac{1}{2}(\mathbf{L}_p^{t+\Delta t} + (\mathbf{L}_p^{t+\Delta t})^T)\Delta t \tag{13}$$

$$\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \mathbb{D}^{ep} d\boldsymbol{\epsilon}_p^{t+\Delta t} + d\mathbf{w}_p^{t+\Delta t} \cdot \boldsymbol{\sigma}_p^t - \boldsymbol{\sigma}_p^t \cdot d\mathbf{w}_p^{t+\Delta t} \tag{14}$$

where $\mathbf{L}_p^{t+\Delta t}$, $d\boldsymbol{\epsilon}_p^{t+\Delta t}$, and $d\mathbf{w}_p^{t+\Delta t}$ are the velocity gradient, strain increment, and spin increment, respectively. Note that in Eq. (14), the Zaremba-Jaumann objective stress rate has been adopted in the case of large deformations. $\mathbf{I}$ is the identity tensor; $\mathbf{v}_{p,\text{PIC}}^{t+\Delta t}$ and $\mathbf{v}_{p,\text{FLIP}}^{t+\Delta t}$ represent the material point velocities obtained from the particle-in-cell (PIC) [34] and fluid implicit particle (FLIP) [35] methods, respectively. A linear combination of PIC and FLIP velocities is usually adopted to update the material point velocity for a balance between energy conservation and numerical stability [36], where the contributing fraction is controlled by a factor α. $\mathbb{D}^{ep}$ denotes the elastoplastic modulus, which is associated with the specific constitutive model used for the material.

2.2. Discrete element method

In DEM, each particle is assumed to be a rigid body, and their movements are governed by Newton's second law of motion. In the simulation, interactions between particles are first determined through a contact detection procedure, which can be divided into broad-phase and narrow-phase contact detection. The contact forces at each contact point are then calculated based on the specified contact model. The velocity and position of particles are next updated using a numerical integration scheme, such as the symplectic-Euler, velocity-Verlet, and Runge-Kutta scheme. The dynamic behavior of particles is described by the following Newton-Euler equations:

$$\mathbf{F}^b + \mathbf{F}^d + \sum_c \mathbf{F}^c = m\frac{d\mathbf{v}}{dt} \tag{15}$$

$$\mathbf{T}^d + \sum_c \mathbf{T}^c = \hat{\mathbf{I}}\frac{d\boldsymbol{\omega}}{dt} + \boldsymbol{\omega} \times \hat{\mathbf{I}}\boldsymbol{\omega} \tag{16}$$

where m is the DEM particle mass; $\hat{\mathbf{I}}$ represents the principal tensor of inertia in the local coordinate system; $\mathbf{v}$ and $\boldsymbol{\omega}$ are the translational and angular velocities, respectively; $\mathbf{F}^c$ and $\mathbf{T}^c$ are the contact force and torque at contact point c, respectively; $\mathbf{F}^b$ is the body force; $\mathbf{F}^d$ and $\mathbf{T}^d$ are the local damping force and torque, respectively. Note that Eq. (15) is computed in the global coordinate system, whereas Eq. (16) is calculated in the local coordinate system [37]. The damping force and torque, which are introduced to dissipate the kinetic energy during the simulation, are given by:

$$\begin{cases} \mathbf{F}^d = -\zeta_d \|\mathbf{F}^b + \sum_c \mathbf{F}^c\| \text{sign}(\mathbf{v}) \\ \mathbf{T}^d = -\zeta_d \|\sum_c \mathbf{T}^c\| \text{sign}(\boldsymbol{\omega}) \end{cases} \tag{17}$$

where ζ_d refers to the coefficient of local damping in DEM. Besides the particle-particle contact, the contact between a particle and multiple triangle patches is also considered in GeoTaichi. The resultant force is summed as [38]:

$$\mathbf{F}^{c,w} = \sum_c \frac{A_{\text{proj}}^c}{A}\mathbf{F}^c \tag{18}$$

where A_{proj}^c is the projection area of the particle on each triangle patch, and A is the sum of projection areas. Considering a circle and a triangle randomly positioned as shown in Fig. 2(a), the area of the purple region can be calculated by summing up the three standard geometries as depicted in Figs. 2(b)–(d).

The particle orientation, particularly important for non-spherical particles, can be determined from the resulting rotations. A quaternion, represented by an array $\mathbf{q} = (q_w, q_x, q_y, q_z)$, can be utilized to establish the transformation relationship between the local and global coordinate systems [39]. Accordingly, the rotation matrix $\mathbf{R}$ can be computed as follows:

$$\mathbf{R} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_w q_z) & 2(q_x q_z - q_w q_y) \\ 2(q_x q_y - q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_w q_x) \\ 2(q_x q_z + q_w q_y) & 2(q_y q_z - q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{19}$$

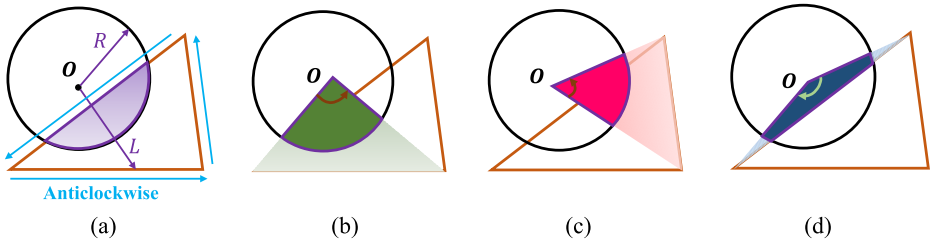


Fig. 2. Calculation of the interaction area between a circle and a triangle. The shaded area in (a) is equal to the sum of shaded area in (b), (c) and (d). As the anticlockwise direction is treated as positive, the areas of (b) and (c) are positive, whereas the area of (d) is negative.

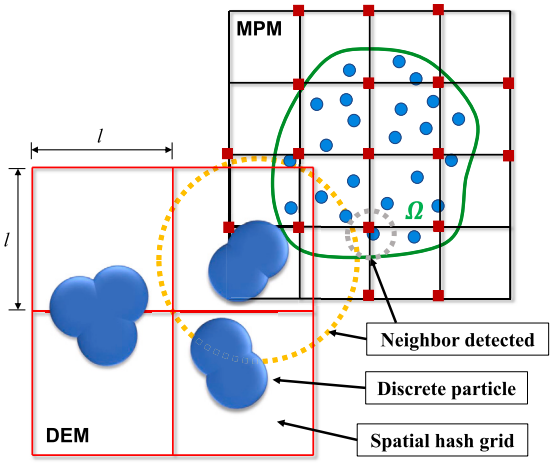


Fig. 3. Contact detection in the coupled material point-discrete element method. The yellow and gray dashed-line circles show the cut-off distance for a DEM particle and a material point, respectively.

where the four components satisfy the following constraint:

$$q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1 \tag{20}$$

3. GPU-accelerated MPDEM

3.1. Coupling of MPM and DEM

Collision detection and contact force calculation are most crucial and computationally intensive tasks in particle-based numerical methods, such as DEM and MPDEM. In this study, a DEM-inspired coupling scheme is employed to handle interactions between material points and rigid DEM particles. When material points have the potential to come into contact with surrounding DEM particles, the former are treated as ghost DEM particles. The contact detection and force calculation are then performed exactly the same way as in conventional DEM. There are several efficient neighbor search algorithms available, such as the bounding volume hierarchy (BVH) and the linked-cell method, the latter of which is reported to exhibit better computational efficiency under both CPU and GPU architectures [38,40]. Additionally, the linked-cell method has simpler data structures compared to the BVH method, making it easier to implement. To further mitigate computational costs, a contact detection algorithm combining the multilevel linked-cell method [41,42] with the Verlet list [43] has been implemented.

As shown in Fig. 3, the simulation domain is discretized using a spatial hash grid with the cell size given by:

$$l = 2r_{\max}^{\text{dem}} + L_v \tag{21}$$

where $r_{\max}^{\text{dem}}$ is the maximum radius of DEM particles, and the Verlet distance L_v is computed by $L_v = \lambda r_{\min}^{\text{dem}}$, where λ is the skin factor and $r_{\min}^{\text{dem}}$ is the minimum radius of DEM particles. To identify all potential contacts with a particle in a cell, it is sufficient to consider only the particles located in this cell and its adjacent cells, using the hash grid.

The neighbor search in MPDEM consists of two steps. The first step is identical to the potential contact search in pure DEM using the classical linked-cell method [44], as illustrated in Fig. 4(a). In the second step, the potential contacts between material points and discrete particles are identified. Following [42], the material points are viewed as small particles, while discrete particles are regarded as large particles with a broader range of size distribution. Fig. 4(b) depicts the insertion of material points into the hash grid, in which the coordinates of target cells are found as:

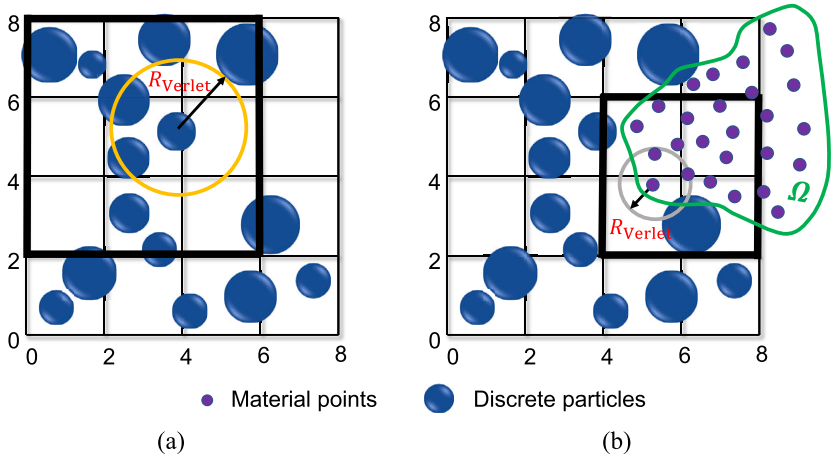


Fig. 4. Neighbor search for (a) DEM and (b) MPDEM. The neighboring cells are visually highlighted with thick lines, and the cut-off distances for a discrete particle and a material point are represented by yellow and gray circles, respectively.

$$\mathbf{C}^{\text{start}} = \left\lfloor \frac{\mathbf{x}_i^{\text{mpm}}(t) - (r_i^{\text{mpm}} + r_{\max}^{\text{dem}})}{l} \right\rfloor \tag{22}$$

$$\mathbf{C}^{\text{end}} = \left\lceil \frac{\mathbf{x}_i^{\text{mpm}}(t) + (r_i^{\text{mpm}} + r_{\max}^{\text{dem}})}{l} \right\rceil \tag{23}$$

where $\mathbf{x}_i^{\text{mpm}}(t)$ is the position of material point P_i at time t, and r_i^{mpm} is its radius. $\mathbf{C}^{\text{start}}$ (left bottom) and $\mathbf{C}^{\text{end}}$ (right top) are depicted as thick-lined cells in Fig. 4.

Furthermore, for each particle, a list of all neighboring particles or walls located within a certain cut-off distance R_{Verlet} is determined using the hash grid and stored as a potential contact list. Note that the particle can be either a DEM particle or a material point (see Fig. 4). The condition for a potential contact between particle P_i and P_j is:

$$|\mathbf{x}_j(t_v) - \mathbf{x}_i(t_v)| \le R_{\text{Verlet}} \tag{24}$$

where

$$R_{\text{Verlet}} = L_v + r_i + r_j \tag{25}$$

where r_i and r_j are the radii of P_i and P_j, respectively; $\mathbf{x}_i(t_v)$ and $\mathbf{x}_j(t_v)$ are the positions of P_i and P_j, respectively, at time t_v when the potential contact list is updated. The update occurs when the following condition is met:

$$\max |\mathbf{x}_i(t) - \mathbf{x}_i(t_v)| \ge \frac{L_v}{2} \tag{26}$$

Therefore, the frequency of broad-phase neighbor search is regulated by the Verlet distance L_v, which, in turn, is controlled by the skin factor λ. A larger skin factor implies less frequent neighbor searches during the simulation. However, it also leads to a larger potential contact list including more particles, which increases the time for narrow-phase contact detection and force calculation. Additionally, a larger potential contact list requires more GPU memory allocation [45]. Consequently, the optimal value for the skin factor may differ depending on the specific simulation scenario.

3.2. Compacted potential contact list

During the neighbor search process, the number of data reads or writes is typically much greater than the number of arithmetic operations, leading to suboptimal GPU performance. This is primarily due to the fact that a significant portion of the execution time is spent on memory transactions rather than computations. A technique known as memory coalescing [46] can be implemented to improve the efficiency of global memory loads by maximizing data transfer and reducing memory latency. The technique is effective when threads within a warp fetch contiguous data from the global memory. Instead of retrieving a single requested data value, a chunk of data is returned in a single transaction. This consolidation of multiple memory accesses into a single transaction can significantly enhance the overall efficiency of global memory loads.

Fig. 5 demonstrates how to create a potential contact list using GPU memory coalescing. A 2D schematic diagram of the linked-cell method is illustrated in Fig. 5(a), showcasing a series of equal-sized lattice cells with a few superimposed particles. Both the particles and cells are labeled using consecutive integer sequences. Before placing the particle into the hash grid, an array PC is created to store the number of particles in each cell. The array size corresponds to the number of cells. To prevent data racing, an atomic-exchange operation or mutex lock is adopted, which can impact parallelism and potentially slow down the process. Subsequently, a prefix reduce sum instruction, accelerated by shared memory and warp-level CUDA intrinsic functions, is executed with one block consisting of 64 threads:

$$SPC_m = \sum_{n=0}^{m} PC_n \tag{27}$$

Algorithm 1: Building the cell-particle list

Input: The number of particle N_p; The number of cell N_c; The length of hash grid l; The particle positions X_p; The auxiliary lists CPC; The prefix sum on particle-cell list SPC.
Output: The cell-particle list CP.

```
foreach thread i in the Block do
    while i < N_c do
        if i > 0 then
            calculate CPC_i = SPC_i;
        i = i + BlockDim;
foreach thread i in the Block do
    while i < N_p do
        cid = id of cell that particle i locates by Eq. (28);
        calculate the insertion index ind = CPC_cid
        push i into CP_ind;
        update the insertion index CPC_cid = CPC_cid − 1 in list CP for
          the cell cid;
        i = i + BlockDim;
```

where SPC is the prefix sum of PC. Note that the subscript in the variables refers to an index for accessing a list in this subsection. For example, SPC_i refers to the i-th element in SPC. Besides, a auxiliary list CPC is built, which copies elements from SPC. Afterwards, the threads are assigned to particles to find the cell id cid that particle belongs to:

$$cid = \left\lfloor \frac{X_p - X_0}{l} \right\rfloor \tag{28}$$

where X_p is the particle position and X_0 is the left down position of hash grid. If particle i is located in cell m, push the particle id i into cell-particle list CP_{ind}, where $ind = CPC_m$ is the index for insertion particle id into CP. Then CPC is updated to record the next insertion index in CP for a cell m. Note that the length of CP equals to the number of particles. The pseudocode of creating a cell-particle list is presented in Algorithm 1. This step facilitates the built of a linked list, where particle information for a given cell is stored contiguously in global memory transactions. As a result, assessing this information incurs minimal latency, as illustrated in Fig. 5(b).

Subsequently, a potential contact list is built by performing, in parallel, the neighbor search for each particle in the target cell. With lists SPC and CPC, the particle ids in the target cell m can be acquired from index CPC_m to SPC_m in list CP. If Eq. (24) is satisfied, the contact pair, i.e., particle i and particle j ($i < j$), is pushed to the potential contact list from $i \times Z_p + offset$, where Z_p denotes the largest possible coordination number for a particle and is set to 16 in this study. $offset$ refers to the number of neighbors of particle i and is stored in an auxiliary list, as shown in Algorithm 2. Fig. 5(c) shows the procedure of building the potential contact list. It is noteworthy that the pre-allocated lengths for the potential contact and auxiliary lists are $N_p Z_p$ and N_p, respectively. At this stage, the narrow-phase contact detection and contact force calculation become feasible using the potential contact list. However, the classical linked-cell method exhibits a strided access pattern, resulting in poor global memory load efficiency. Besides, accessing a large amount of memory incurs a higher rate of cache misses [47], further impacting performance. To remedy these problems, a prefix sum is conducted on list ZL to build a compacted potential contact list CNL. Correspondingly, parallelism can be achieved for contact pairs in the stage of narrow detection. More details can be found in [48,49].

3.3. Contact models

After the broad-phase contact detection and the creation of compacted potential contact list, the narrow-phase contact detection is conducted to determine the actual contact pairs and calculate the contact forces and torques using a specific contact law. Note that these opera-

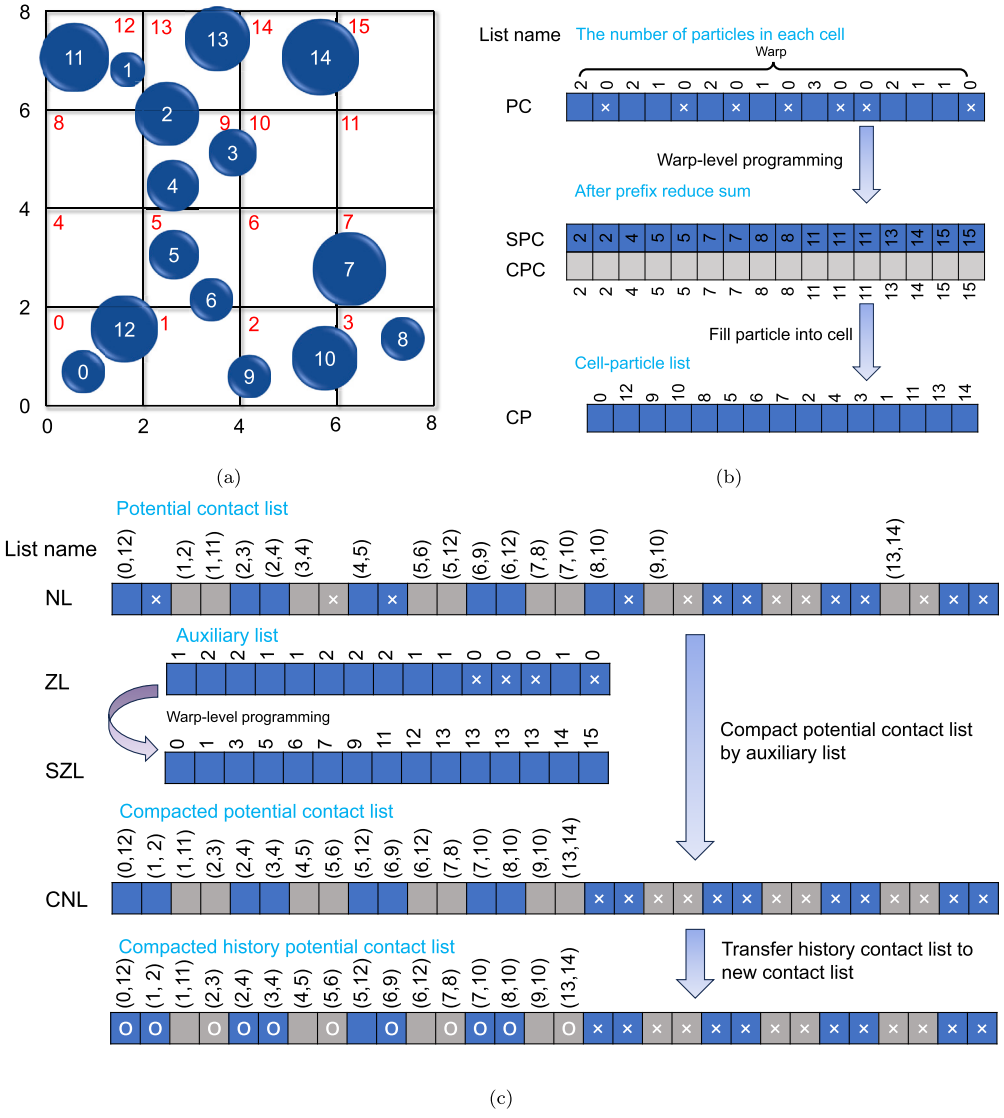


Fig. 5. (a) Illustration of randomly positioned particles on a hash grid, and building of (b) the particle-cell list and (c) the compacted potential contact list.

Algorithm 2: Building the potential contact list

Input: The number of particles N_p; The coordination number Z_p; The length of hash grid l; The particle position X_p; The auxiliary list CPC; The prefix sum on particle-cell list SPC; The cell-particle list CP.

Output: The potential contact list NL; The auxiliary list ZL.

```
foreach thread i in the Block do
    while i < N_p do
        cid = id of cell that particle i locates by Eq. (28);
        Initialize index off set = i × Z_p
        foreach target cell id k using Eq. (22) and Eq. (23) do
            foreach particle id j from CPC_cid to SPC_cid in list CP do
                if distance is less than the threshold, using Eq. (24) then
                    if i < j then
                        push j into NL_off set; update index
                        off set = off set + 1;
        push off set into ZL_i;
        i = i + BlockDim;
```

tions are assigned to individual GPU threads for each contact pair and executed simultaneously in parallel. As mentioned earlier, during the narrow-phase contact detection, the material points are transformed into DEM particles. The equivalent radius for material point i is calculated as:

$$r_i^{\mathrm{mpm}} = \sqrt[3]{\frac{3}{4\pi} V_i^{\mathrm{mpm}}} \tag{29}$$

where V_i^{mpm} is the representative volume of the material point, which is updated during the simulation using Eq. (12).

For a given contact c between either two DEM particles or between a material point and a DEM particle, the interaction force is generally split into two orthogonal components: the normal contact force $\mathbf{F}_n^c$ and the tangential contact force $\mathbf{F}_t^c$, which are computed following a force-displacement law [3]:

$$\mathbf{F}_n^{c,t+\Delta t} = (-k_n \delta - \epsilon_n \mathbf{v}_r \cdot \mathbf{n})\mathbf{n} \tag{30}$$

$$\mathbf{F}_t^{c,t+\Delta t} = \min(\mu |\mathbf{F}_n^{c,t+\Delta t}|, |\mathbf{F}_t^{c,t}| - k_t (\mathbf{v}_r \cdot \mathbf{t}) \Delta t - \epsilon_t \mathbf{v}_r \cdot \mathbf{t})\mathbf{t} \tag{31}$$

where k_n and k_t are the contact normal and tangential stiffness, respectively; ϵ_n and ϵ_t are the normal and tangential viscous damping coefficients, respectively; $\mathbf{n}$ and $\mathbf{t}$ are the contact normal and tangential directions, respectively; δ denotes the contact overlap between the two particles; μ is the frictional coefficient, and $\mathbf{v}_r$ is the relative velocity between the two particles given by:

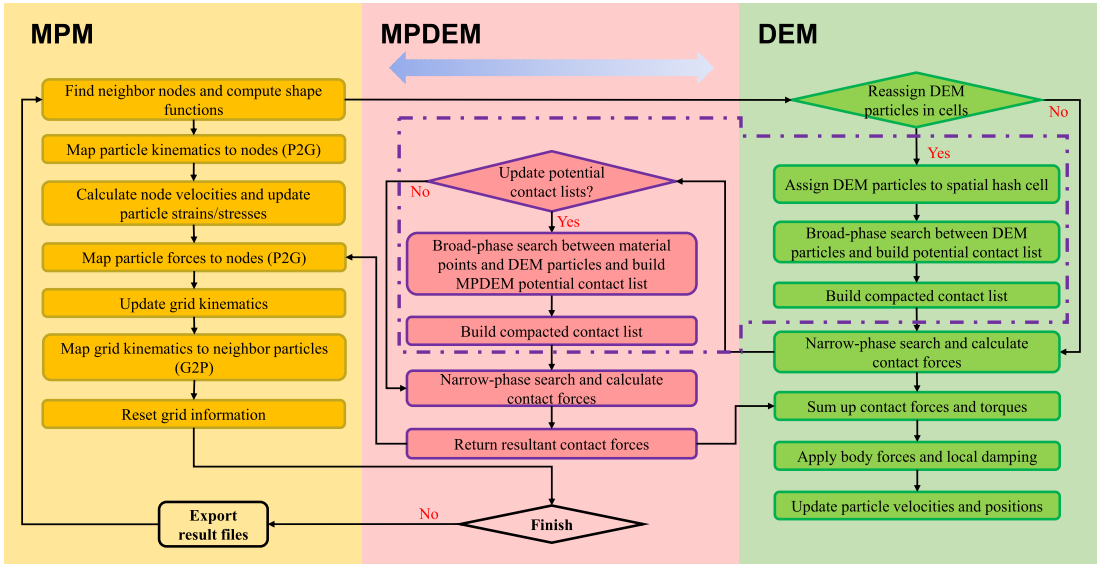


Fig. 6. Computational flow of the coupled material point-discrete element method.

$$\mathbf{v}_r = (\mathbf{v}_i - \boldsymbol{\omega}_i \times \mathbf{r}_i) - (\mathbf{v}_j - \boldsymbol{\omega}_j \times \mathbf{r}_j) \tag{32}$$

where $\mathbf{r}_i$ is the position vector from the contact point to the mass center of particle i. It is important to note that for the material point, its angular velocity is always zero, as only Cauchy stress is considered for the continuum body. The torque acting on DEM particle j is computed as:

$$\mathbf{T}_j^c = \mathbf{F}_t^c \times \mathbf{r}_j - \mathbf{M}^{rt} \tag{33}$$

where $\mathbf{M}^{rt}$ is the rolling and twisting resistance, which should be considered if the rolling resistance model is employed [50,51].

3.4. Computational flow of MPDEM

Fig. 6 presents the computational flow and main functions of MPDEM. The detailed procedure in a time step Δt is explained as follows:

(1) Initialization: The initialization process involves setting up the material points and background grid in the MPM, discrete particles and walls (or meshes composed of triangle patches) in the DEM, and the spatial hash grid. The material and contact parameters are assigned simultaneously.
(2) DEM: Determine whether the potential contact list for DEM should be updated using Eq. (26). If yes, update the potential contact list using Eq. (24). After that, narrow-phase contact detection and force calculation are executed.
(3) MPDEM: If either the displacement of DEM particles or material points meets the condition specified by Eq. (26), the potential contact list for MPDEM should be recalculated. Once the potential contact list has been updated, the contact force between DEM particles and material points can be obtained using appropriate contact models.
(4) DEM: Once the contact forces acting on the material points have been determined, they are applied to the corresponding DEM particles to calculate the particle acceleration. The velocity and position of DEM particles are then integrated using appropriate time integration schemes.
(5) MPM: Using shape functions, the properties of material points are mapped to the background grids. These properties include mass, momentum, body force, traction, stress, and contact force from DEM particles. The momentum balance equation is then solved, followed by the updating of position, velocity, strain, and stress for the material points.

To ensure computational stability, the time step Δt should satisfy the Courant-Friedrichs-Lewy (CFL) condition [52]:

$$\Delta t = \kappa \min\left(\frac{h}{c}, \sqrt{\frac{m}{k_n}}\right) \tag{34}$$

where κ is the Courant number, and c is the speed of sound, which is related to the Young's modulus E and density ρ of the material (i.e., $c = \sqrt{E/\rho}$). The first term in the parenthesis on the right-hand side of Eq. (34) is associated with the continuum body (MPM), while the second term relates to the discrete particles (DEM).

4. Design philosophy and main features

4.1. Design philosophy

GeoTaichi is developed in Python using the advanced features of the Taichi parallel language. The primary objective of GeoTaichi is to enhance code comprehensibility by leveraging the new capabilities provided by Taichi. The entire codebase adheres to a unified coding style, ensuring that class and variable names closely correspond to the specific parameters or model names they represent. GeoTaichi embraces an object-oriented approach and provides a full-featured Python Application Programming Interface (API), which grants users access to a wide range of functions, enabling seamless navigation through the source code files and facilitating future developments.

The main components of GeoTaichi are organized into four layers, as presented in Fig. 7. From the top layer to the bottom, GeoTaichi is composed of the base layer, kernel layer, solver layer, and user layer. The base layer contains abstract classes representing fundamental components such as `particles`, `clumps`, `walls`, `elements`, and `boundary conditions`. Additionally, abstract classes for key functionalities like `contact detection`, `constitutive models`, and `contact models` are also included. This allows for adding new models and algorithms without altering the general structure of these components. The kernel layer consists of several kernel functions that correspond to the DEM and MPM solvers. These kernel functions form the core computational elements of the framework. The solver layer is constructed by combining the kernel functions from the kernel layer and comprises the executables that call the kernel functions to perform simulations. In the user layer, users can set up simulations by specifying process parameters and configuring the desired simulation settings. This layer provides a user-friendly interface for setting up and running simulations. In

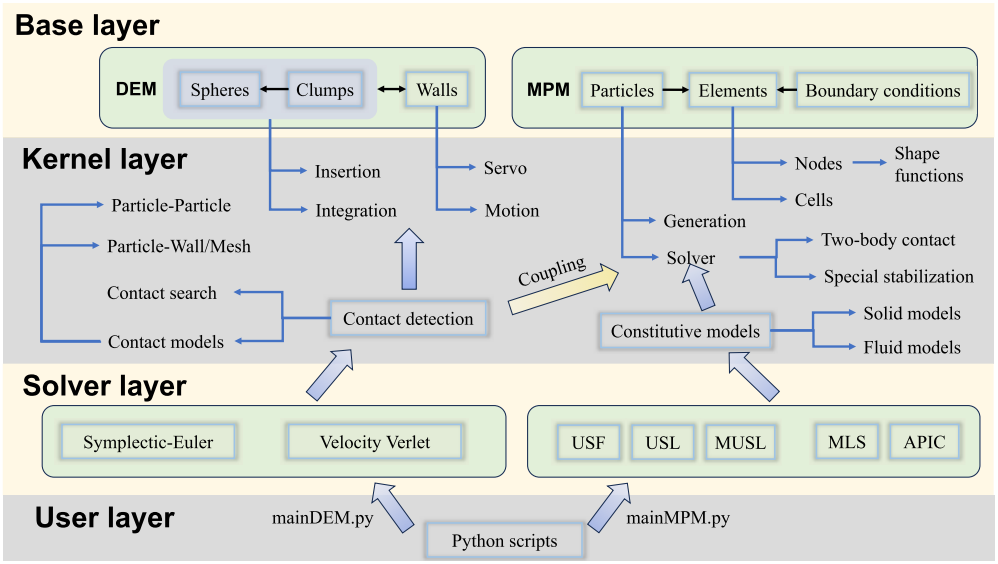


Fig. 7. Organization of the main components of GeoTaichi.

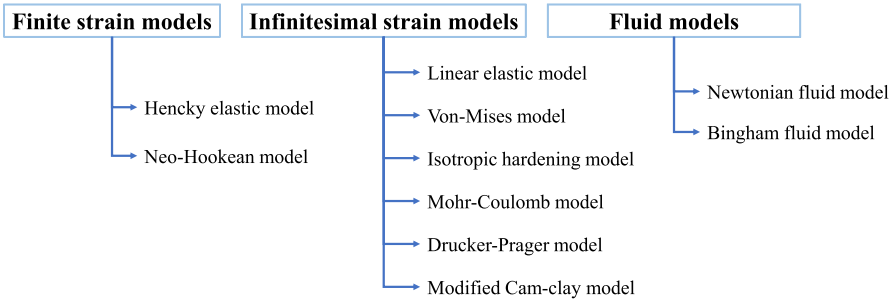


Fig. 8. Available constitutive models in GeoTaichi.

Appendix A, a typical Python script for modeling column collapse of granular materials is provided, which demonstrates all the essential ingredients for an MPM simulation in GeoTaichi. More examples of MPM, DEM, and MPDEM simulations can be found in the example folder.

4.2. Stabilization in MPM

Several numerical techniques are employed to enhance computational stability and accuracy of MPM. The standard linear quadrilateral elements are susceptible to volumetric locking as a result of the use of low-order elements and a large number of material points within each element [53]. To address this issue, the B-bar [54] and F-bar methods [27] are exploited in GeoTaichi. Users have the flexibility to choose between the two methods to alleviate volumetric locking and improve the overall performance of MPM.

Furthermore, the classical MPM formulation suffers from cell-crossing instability dealing with large deformations, which arises primarily from the insufficient smoothness of the shape functions. To address this issue, GeoTaichi adopts the generalized interpolation material point (GIMP) method [55], in which the standard shape functions are replaced with their average values over the volume of each material points. Besides, some higher order shape functions are also available in GeoTaichi, such as quadratic B-splines (C^1-continuous) and cubic B-splines (C^2-continuous).

In addition to the USL scheme presented in Section 2.1, GeoTaichi offers two other stress update schemes, i.e., update stress first (USF) [33] and modified update stress last (MUSL) [56], as well as two improved velocity projection schemes, i.e., moving least squares (MLS) [57] and affine particle in cell (APIC) [58], for better energy conservation during the simulation.

4.3. Constitutive and contact models

GeoTaichi offers a range of constitutive models for MPM simulations, including the finite strain and infinitesimal strain theory based elastoplastic models for solids, and Newtonian and Bingham models for fluids, as shown in Fig. 8. Additionally, for DEM and MPDEM simulations, three contact models, including the linear contact model [50], Hertz-Mindlin model [50], and linear rolling resistance model [50,51], are available in GeoTaichi.

4.4. Clump model in DEM

Particle shape has been recognized as one of the most critical factors that significantly impact the mechanical responses of granular assemblies. GeoTaichi provides support for modeling non-spherical particles using the clump model. A clump represents a cluster of overlapping spheres arranged with fixed relative positions, approximating the arbitrary geometry of complex particle shapes. The elementary spheres composing a clump can possess varying radii. Due to the overlap of elementary spheres, the volume and moment of inertia of the clump are computed internally by the Monte Carlo method. However, users also have the option to manually override these computed values if desired. The assembly with clumps can be generated from clump templates that prescribe the number and initial positions of elementary spheres. Contact forces involving clumps are calculated the same way as that of spherical particles, where the resultant forces and torques are summed

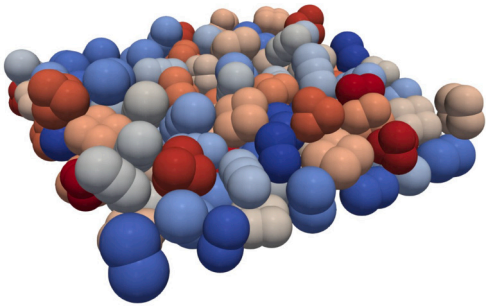


Fig. 9. Assembly of clumps generated from four different templates.

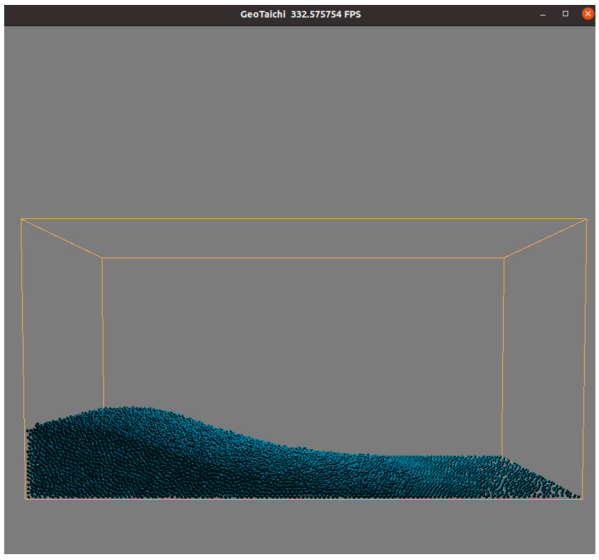


Fig. 10. Visualization of the heap formation in real-time DEM simulation using GGUI.

from those acting on elementary spheres and applied to the center of mass of the clump. It is worth mentioning that the efficient and robust integration of the rotational motion is challenging for clumped particles. GeoTaichi provides two integration schemes to address this challenge: the leap frog method proposed by Fincham [59], and the `SPIRAL` method proposed by del Valle et al. [37]. Fig. 9 depicts an assembly of clumps generated from four different templates.

4.5. Visualization

Thanks to the new user interface system GGUI powered by Taichi, GeoTaichi offers modules that greatly simplify real-time GPU rendering of 3D scenes (see Fig. 10). This enables users to interactively manipulate the motion of both the render camera and objects within the scene, enhancing the visualization experience. To ensure computational efficiency, GeoTaichi also provides the capability to export simulation results into vtk files, which can be post-processed and rendered using compatible third-party software such as ParaView.

4.6. Restart from a specific timestep

In some circumstances, it may be necessary to interrupt a simulation and subsequently restart it from a specific timestep rather than from the beginning. GeoTaichi addresses this requirement by providing a write function for each basic data structure, allowing the current state of simulation to be stored in a binary file, and a read function to reload the simulation state from a saved file.

5. Validation tests

In this section, we present four benchmark tests to showcase the robustness and versatility of GeoTaichi. All tests presented in this study were conducted on an Ubuntu 20.04 system equipped with an AMD AM4 5800X CPU (8 cores @ 3.8 GHz, 16 GB memory) and a GeForce RTX 3070 GPU (8 GB memory).

5.1. DEM simulation of granular packing

A packing of non-spherical particles settling under gravity is first simulated. The clump particle is constructed by two equal-sized spheres with an aspect ratio of 1.5. The equivalent radii of clumps span uniformly from 9 to 15 cm. All clumps share the same solid density of 2,650 kg/m^3. A linear contact model is employed with both normal and tangential contact stiffness set to $k_n = k_s = 10^5$ N/m. Additionally, a viscous damping coefficient of $\epsilon_n = \epsilon_t = 0.05$ and a frictional coefficient of $\mu = 0.5$ are chosen. A total of 23,168 clumps with arbitrary initial orientations are generated and allowed to freely fall into a cubic box measuring 7 m×7 m×7 m under gravity. During the deposition process, the clumps will collide with their neighbors, leading to tumbling and oscillatory behaviors before a stable granular packing is achieved, as shown in Fig. 11. Fig. 12 presents the temporal evolution of energies throughout the simulation, where the total mechanical energy of the system encompasses both kinetic and potential energies. It is observed that the kinetic energy exhibits an initial surge followed by a decrease and eventually reaches zero at approximately $t = 1$ s, whereas the potential energy shows a monotonic decrease during the simulation. The total mechanical energy decreases slightly during the process due to the dissipation caused by friction and viscous damping.

5.2. DEM simulation of triaxial compression

Three numerical specimens consisting of 28,242 spherical particles are generated within a cubic box bounded by six rigid, frictionless walls. The particle radii are uniformly distributed between 0.25 and 0.55 mm.

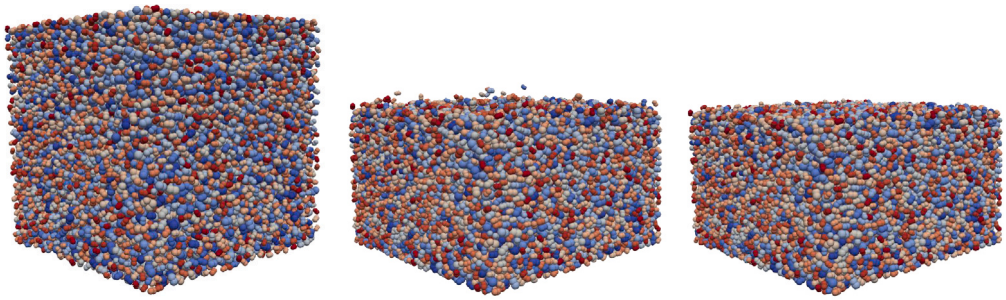


Fig. 11. Deposition of granular packing under gravity: (a) $t = 0$ s, (b) $t = 1.02$ s, and (c) $t = 2$ s.

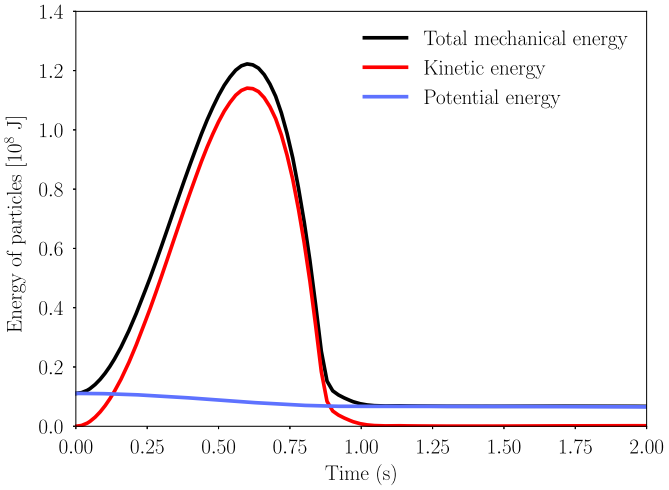


Fig. 12. Evolution of the kinetic, potential, and total energies of granular packing during the deposition process.

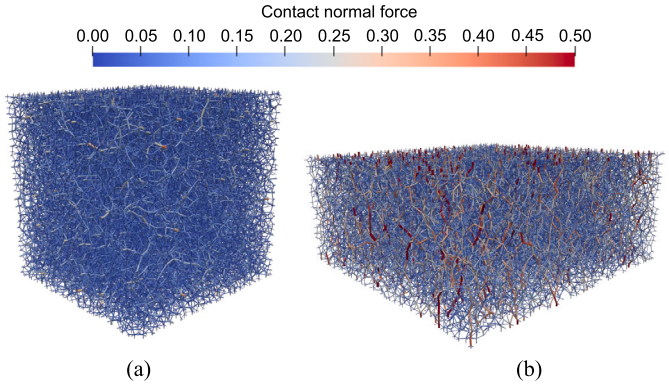


Fig. 13. Snapshots of force chains in the dense packing at the (a) initial and (b) final states during the triaxial shearing process.

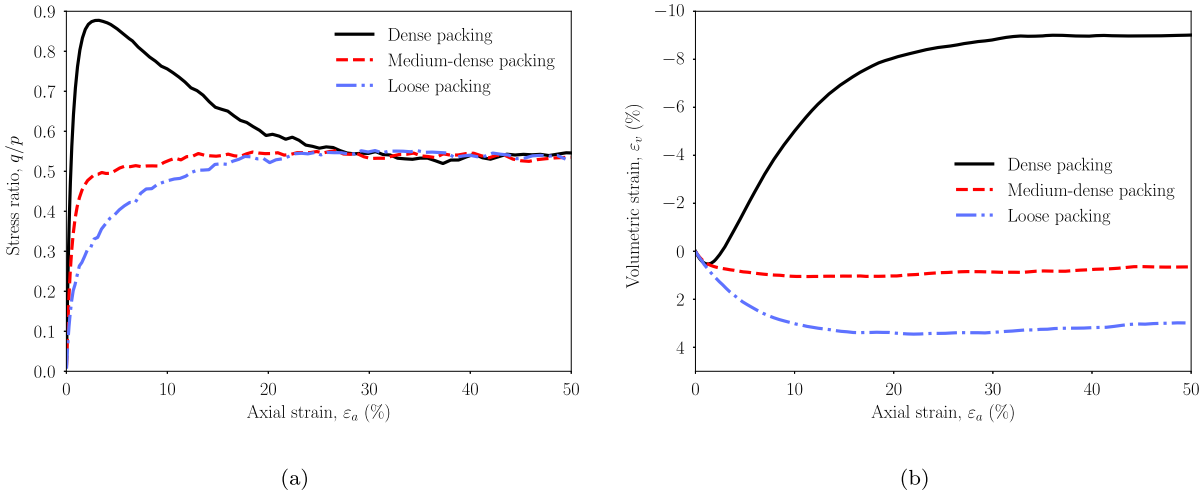


Fig. 14. Macroscopic responses of specimens with different initial void ratios under drained triaxial compression: evolution of (a) stress ratio and (b) volumetric strain in relation to axial strain.

The contact parameters follow Zhao et al. [60], with the contact stiffness set to $k_n = k_s = 4.25 \times 10^4$ N/m, and the frictional coefficient is set to 0.5. The three specimens, referred to as dense, medium-dense, and loose packings, possess initial void ratios of 0.55, 0.707, and 0.747, respectively, after consolidation under a confining stress of 200 kPa. During shearing, the specimens sustain a constant horizontal stress while the top and bottom walls move inwards at a constant small axial strain rate to ensure a quasi-static shear condition [61]. Fig. 13 depicts snapshots of force chains in the dense packing at the initial and final states during the triaxial shearing process, where force chain segments connect the centers of two particles in contact. The width of each segment is proportional to the magnitude of contact normal force, providing a visual representation of force transmission within the packing.

Fig. 14 presents the macroscopic responses of the three specimens with different initial void ratios, depicting the evolution of stress ratio q/p and volumetric strain ε_v in relation to axial strain ε_a, where p and

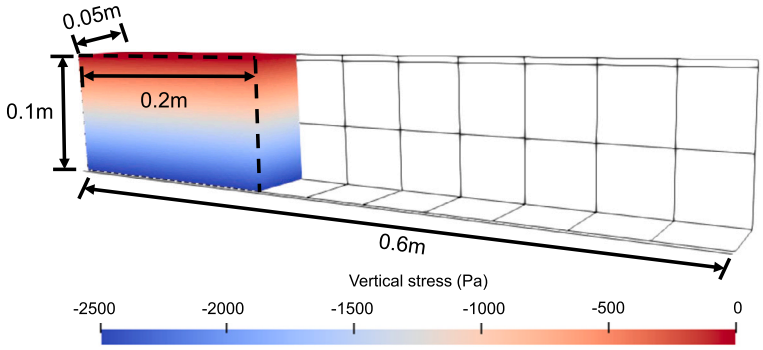


Fig. 15. Initial geometry and vertical stress distribution of the granular column.

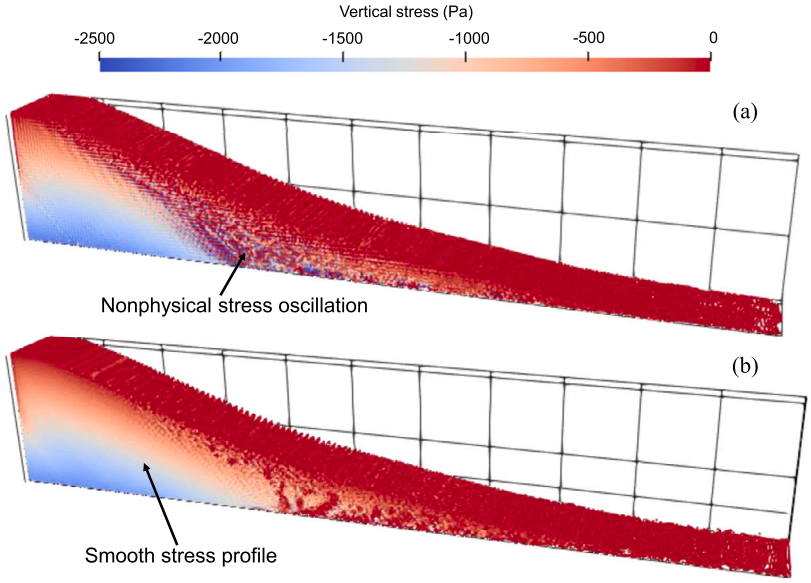


Fig. 16. Final vertical stress profile of granular column using (a) standard GIMP and (b) GIMP with B-bar stabilization.

Table 1
Model parameters for the granular column collapse test.

Definition	Value
Density (kg/m^3)	2,650
Young's modulus (Pa)	8.4×10^5
Poisson's ratio	0.3
Friction angle (°)	19.2
Cohesion (Pa)	0

q are the mean and deviatoric stresses, respectively. The stress of the packing is averaged based on the Love-Weber formula [62]. It is seen from Fig. 14 that the dense packing exhibits a strain-softening behavior accompanied by volumetric dilation, while the loose packing shows strain-hardening along with volumetric contraction. Notably, all specimens ultimately reach a distinct critical state at the end of shearing, consistent with the well-established experimental observations on sand in soil mechanics.

5.3. MPM simulation of granular column collapse and water dam break

These two benchmark tests serve to validate the implementation of MPM and associated constitutive models for solids and fluids in GeoTaichi. The first test, known as the granular column collapse test, follows the experimental setup by [63]. The material properties are obtained from a shear box test and summarized in Table 1. The initial configuration of the test is illustrated in Fig. 15, and the simulation domain is discretized using a Eulerian grid with a cell size of 0.005 m. Each grid element initially contains eight material points. The Drucker-Prager model with tension cutoff is utilized to simulate the dynamic behavior of the granular column. Note that the GIMP shape function is employed to alleviate the cell-crossing error. In the simulation, the bottom and left boundaries are frictional with a frictional coefficient of 0.5, while the front and back boundaries are set to be frictionless.

Fig. 16 compares the final vertical stress profiles obtained using the standard GIMP and GIMP with B-bar stabilization. It is seen that the standard GIMP predicts significant nonphysical stress oscillation, which could be noticeably reduced by the B-bar method. In Fig. 17(a), simulation results of the free surface and internal failure surface of the final deposit are presented alongside experimental data [63] for comparison. A remarkable agreement between these results is observed, which indicates satisfactory accuracy of the present code implementation in GeoTaichi. Furthermore, the evolution of runout distance is plotted against normalized time $T = t/t_0$ in Fig. 17(b), where t_0 represents the total simulation time of 0.6 s. Both the simulation and experimental results exhibit a consistent trend. It is also noticed that the application of B-bar method can improve the stress results but have insignificant influence on the deformation pattern.

The second test, known as water dam break is simulated to demonstrate the capability of GeoTaichi in simulating free surface fluid flows. The detailed numerical setup is shown in Fig. 18. The artificial bulk modulus and viscosity of water are assigned as 0.36 MPa and 10^{-3} Pa · s, respectively [64]. The water density is set to 1,000 kg/m^3. The discretization employs a cell size of 0.05 m, with eight material points per

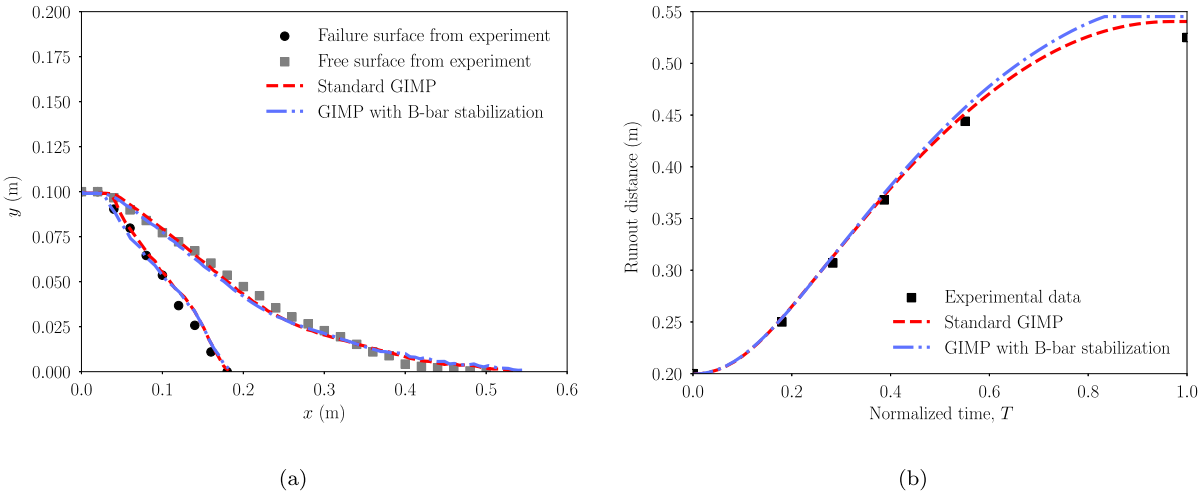


Fig. 17. Comparison between numerical and experimental results of the granular column collapse test: (a) final deposit geometry and (b) runout distance.

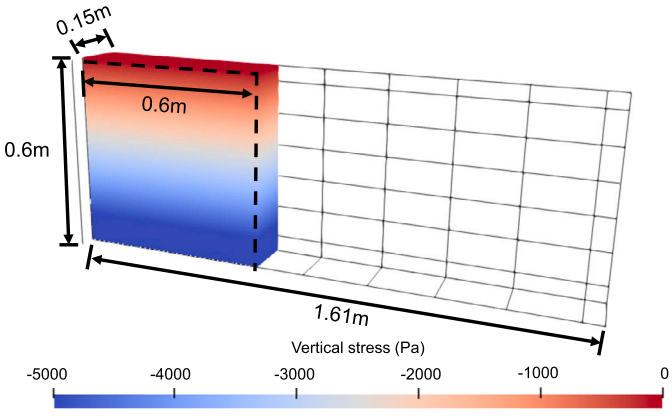


Fig. 18. Initial geometry and vertical stress distribution of the dam break test.

cell. In addition, a slip boundary condition is imposed, which constrains the particle velocity component normal to the boundaries to be zero but allows the particle to slide freely on the boundaries.

Fig. 19 shows several snapshots of the flow profiles at different time instances, capturing the occurrence of a roll-up along the downstream wall and a second splash. Furthermore, the evolution of normalized position of the surge-wave front is plotted in Fig. 20 against the normalized time. Quantitative comparisons are made with experimental result [65] as well as other numerical predictions [64,66]. The results obtained from GeoTaichi show good agreement with other simulation results. However, similar to previous simulations [66–68], there is a slight tendency for the numerical models to underpredict the early-stage propagation of the water front compared to the experimental observations.

5.4. MPDEM simulation of particle impacting on a granular bed

The reliability of the MPDEM coupling scheme is examined by studying the impact of a spherical particle on a granular bed. The problem setup is illustrated in Fig. 21, which closely resembles the experimental setup described in [69]. Particularly, the granular bed is modeled using MPM, while the projectile is modeled by DEM. In the simulation, the MPM resolution (cell size) is set to 0.01 m, and each element has eight material points. The bottom boundary is fixed, and the side boundaries are constrained in their respective normal directions. A

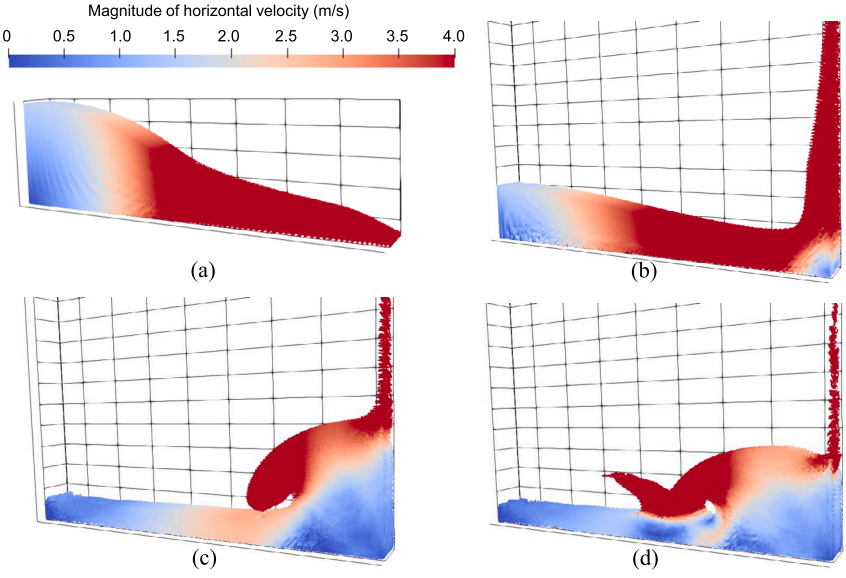


Fig. 19. Snapshots of flow profiles at (a) $t = 0.36$ s, (b) $t = 0.64$ s, (c) $t = 1.16$ s, and (d) $t = 1.26$ s.

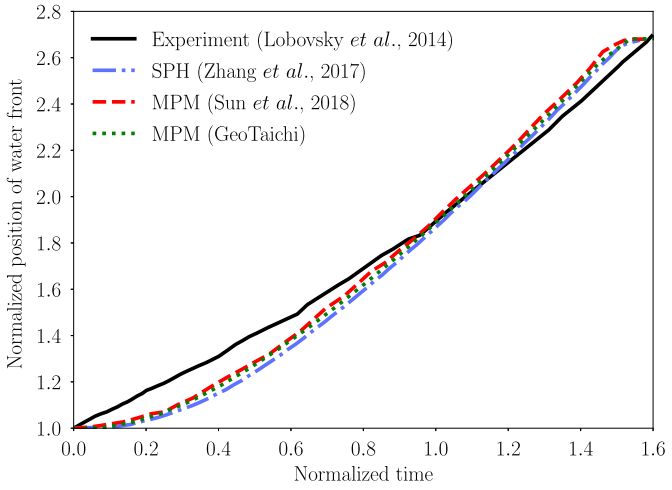


Fig. 20. Comparison between experimental data (Lobovsky et al. [65]) and numerical predictions (SPH simulation by Zhang et al. [66], MPM simulations by Sun et al. [64] and GeoTaichi) of the evolution of surge-wave front.

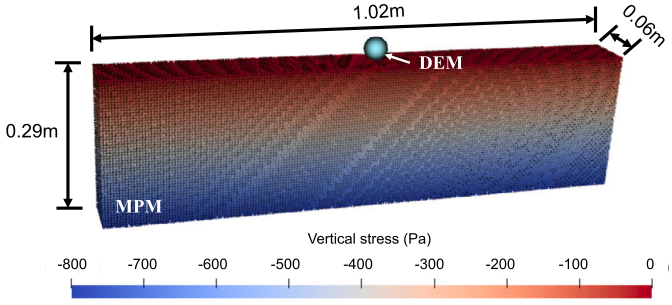


Fig. 21. Initial geometry and vertical stress distribution of a granular bed subjected to projectile impact.

projectile with a radius of 0.0223 m is released with varying initial velocities. The Drucker-Prager model is used to simulate the behavior of granular bed, while the linear elastic model is employed to describe the contact behavior between the projectile and material points. The model parameters, as summarized in Table 2, are calibrated by matching the numerical results to the corresponding experimental data for the case with an initial impacting velocity of 3.30 m/s. Subsequently, simulations with different initial velocities are performed to assess the consistency of GeoTaichi.

Table 2
Model parameters for projectile impacting on a granular bed.

Granular bed modeled by MPM	
Density (kg/m^3)	600
Young's modulus (Pa)	6.1×10^7
Poisson's ratio	0.2
Cohesion (Pa)	0
Friction angle (°)	16
Dilation angle (°)	3
Projectile modeled by DEM	
Density (kg/m^3)	7,850
Normal contact stiffness (N/m)	6.5×10^6
Tangential contact stiffness (N/m)	3.3×10^6
Frictional coefficient	0.28

Fig. 22 displays 2D vertical slices of the simulation domain at its center, showcasing the process of particle projection into the granular bed with the initial velocity of 3.30 m/s. These steps closely align with the experimental procedure described in [69]. Furthermore, Fig. 23 provides a quantitative comparison between the numerical and experimental results for various initial velocity cases. It is observed that the deviations between the simulated and experimental results are marginal in terms of the penetration depth and projectile velocity, which suggests that the MPDEM coupling scheme effectively captures the intricate interactions between discrete particles and material points, even under large deformations of the continuum body.

6. Evaluation of code efficiency

Enhancing computational efficiency is a primary objective in the development of GeoTaichi. In this section, we evaluate the computational efficiency of GeoTaichi by investigating two typical problems, namely DEM simulation of heap formation and MPDEM simulation of a granular column collapsing and impacting stacked cubic boxes. We specifically compare with other GPU-powered codes that have been previously reported in the literature, including the open-source DEM package MUSEN [22] and the coupling software CoSim developed by [17]. Note that all floating-point operations are executed using double precision.

6.1. DEM simulation of heap formation

Packings of monosized spheres with a radius of 0.075 m and a density of 2,650 kg/m^3 are generated. The Hertz-Mindlin contact law is utilized with a shear modulus of 4.3 MPa and a Poisson's ratio of 0.3.

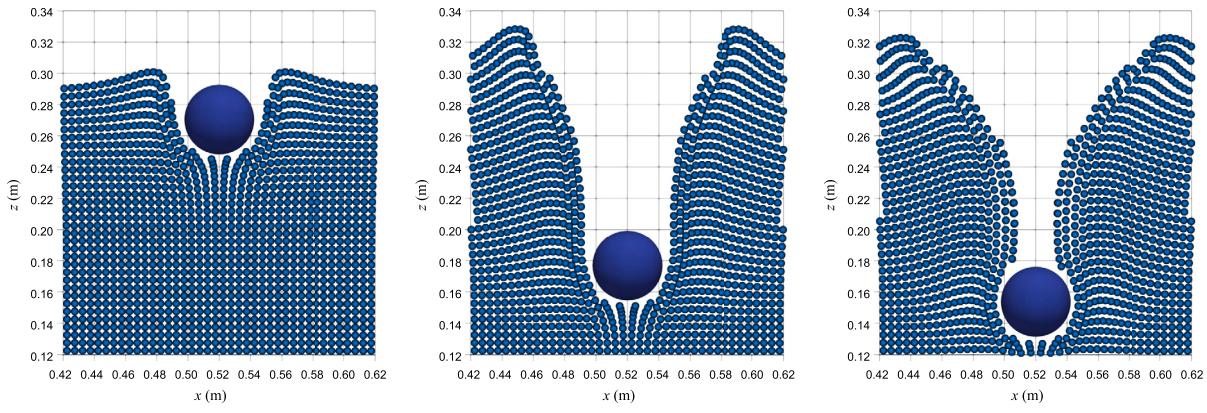


Fig. 22. 2D slices showing the process of particle impacting on a granular bed: (a) $t = 0.016$ s, (b) $t = 0.08$ s and (c) $t = 0.15$ s.

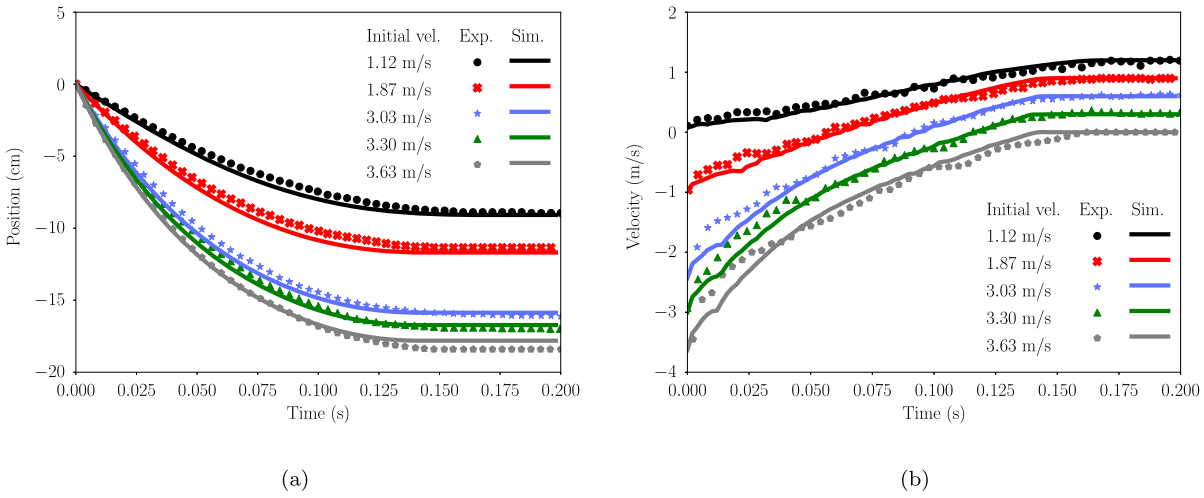


Fig. 23. Evolution of (a) penetration depth and (b) projectile velocity (the ordinate in (b) of each successive simulation has been shifted by 0.3 m/s to enhance clarity).

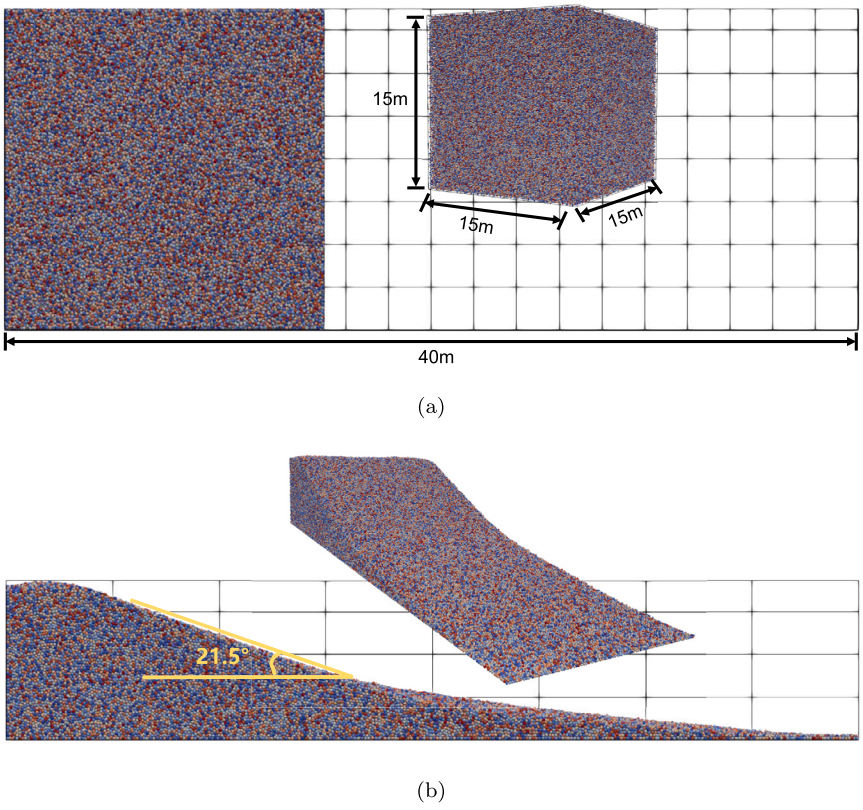


Fig. 24. Heap formation involving 600,000 particles: (a) initial and (b) final states.

The frictional and restitution coefficients are set to 0.5 and 0.6, respectively. For the boundary walls, the same contact properties including the shear modulus, Poisson's ratio, and frictional coefficient are applied. Initially, the spheres are randomly placed within a cubic container, and then allowed to settle under the influence of gravity until a stable heap is formed. A total of ten simulations have been conducted, with particle numbers ranging from 10,000 to 6,250,000. Fig. 24(a) and (b) illustrates the initial and final states, respectively, of a simulation case involving 600,000 spheres.

The same heap formation simulations are conducted using the open-source code MUSEN [22], with particle numbers ranging from 100,000 to 3,500,000. Note that our GPU has an available memory capacity of 7.35 GB, excluding the necessary system usage. The memory required by MUSEN exceeds the upper bound of our GPU memory when the particle number is larger than 3,500,000. Comparatively, GeoTaichi demonstrates a significant improvement in performance in terms of both speed and memory cost, achieving a notable average speedup of 337% and a memory saving of 38% compared to MUSEN, as shown in Fig. 25.

6.2. MPDEM simulation of a granular column collapsing and impacting stacked cubic boxes

The test of a granular column collapsing and impacting on six stacked cubic boxes is modeled following the experiment conducted by [17]. The granular column has dimensions of $0.1\,\text{m} \times 0.1\,\text{m} \times 0.2\,\text{m}$

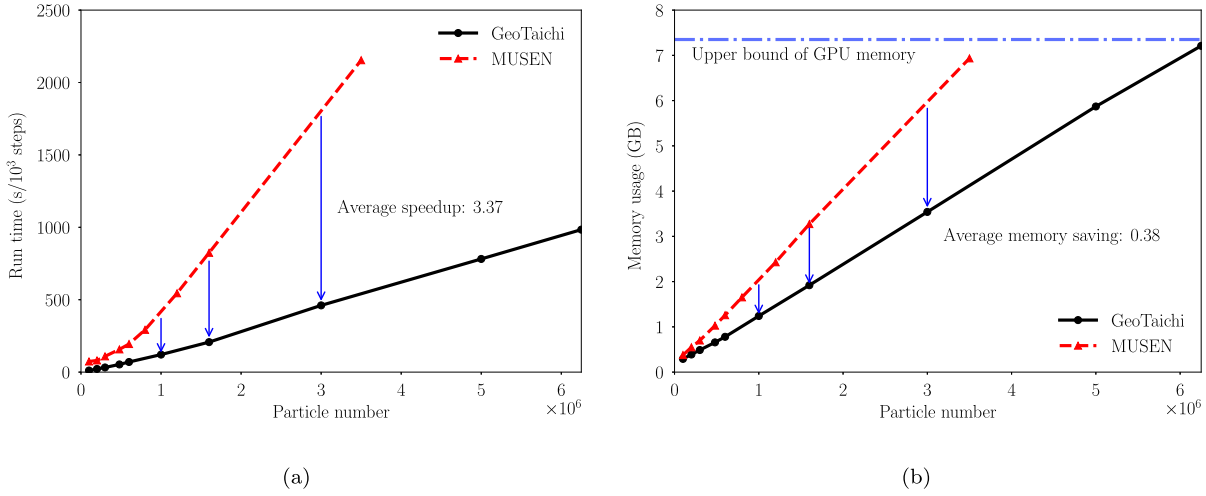


Fig. 25. Performance comparison between GeoTaichi and MUSEN: (a) run time per 10^3 computational steps and (b) memory usage.

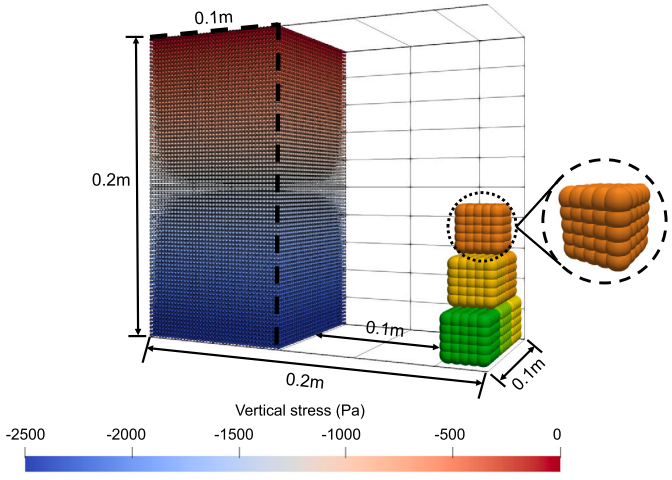


Fig. 26. Initial geometry and vertical stress distribution of a granular column collapsing and impacting stacked cubic boxes.

Table 3
Model parameters for the granular column collapsing and impacting stacked cubic boxes.

Granular column (MPM)	
Density (kg/m^3)	1,350
Young's modulus (Pa)	8×10^4
Poisson's ratio	0.25
Cohesion (Pa)	0
Friction angle (°)	32
Dilation angle (°)	0
Normal contact stiffness (N/m)	3.24×10^3
Tangential contact stiffness (N/m)	5×10^2
Cubic boxes (DEM)	
Density (kg/m^3)	850
Normal contact stiffness (N/m)	4×10^4
Tangential contact stiffness (N/m)	2×10^6
Normal viscous damping coefficient	0.15
Tangential viscous damping coefficient	0.55
Frictional coefficient	0.18
Local damping coefficient	0.25
Walls (DEM)	
Normal contact stiffness (N/m)	4×10^4
Tangential contact stiffness (N/m)	2×10^6
Normal viscous damping coefficient	0.25
Tangential viscous damping coefficient	0.65
Frictional coefficient	0.65

and is discretized into 0.128 million material points with a spacing of 0.005 m. It is noteworthy that the cubic boxes, with each side measuring 3 cm, are modeled using the clump model in GeoTaichi, as opposed to the polyhedral particle model adopted in [17]. In this study, each box is composed of 125 (5^3) overlapping spheres. A total of six boxes are positioned to the right of the granular column with a separation distance of 0.1 m, as illustrated in Fig. 26. The simulation domain is bounded by six frictional walls, serving as fixed boundaries. The Drucker-Prager constitutive model is employed for MPM, and a linear elastic contact model is utilized to describe the interaction between material points and DEM particles, as well as among DEM particles themselves. The model parameters are provided in Table 3, which are consistent with those suggested in [17].

Fig. 27 presents the flow process of the granular column and the motion of cubic boxes, which aligns with the typical motion phases analyzed in the previous study [17]. Furthermore, the evolution of translation and rotation of three selected boxes are exhibited in Fig. 28, in comparison with the experimental and numerical results reported in [17]. Note that the orientation in Fig. 28(b) is defined as the angle between the central vertical axis of the top box and the horizontal direction. The results demonstrate good agreement with the experimental data when $t \leq 0.3$ s. However, beyond that point, noticeable differences in horizontal displacement and orientation of the measured boxes become apparent with elapsed time. This discrepancy may be ascribed to the simplified representation of cubic boxes as clumps in this study, which introduces artificial surface bumpiness and roughness. As a result, the contact characteristics may be altered compared to the cubic boxes used in the experiment.

The computational efficiency of MPDEM coupling is evaluated by measuring the average time required for every 10^3 computational steps, as plotted in Fig. 29. A series of simulations with varying numbers of material points ranging from 35,828 to 1,024,000 is conducted. The simulations performed in this study are performed using a GeForce RTX 3070 GPU, whereas the CoSim study [17] utilized a Tesla V100 GPU. Note that the CoSim results depicted in Fig. 29 have been digitized from [17]. Interestingly, it is observed that GeoTaichi exhibits significantly superior performance compared to CoSim, achieving an average speedup of approximately 493%. Despite the difference in the DEM models used in the two studies (polyhedral model in CoSim versus clump model in GeoTaichi), the performance of GeoTaichi is still impressive considering that the Tesla V100 GPU, with a memory bandwidth of 900 GB/s, is more powerful than the GeForce RTX 3070 GPU with a memory bandwidth of 448 GB/s.

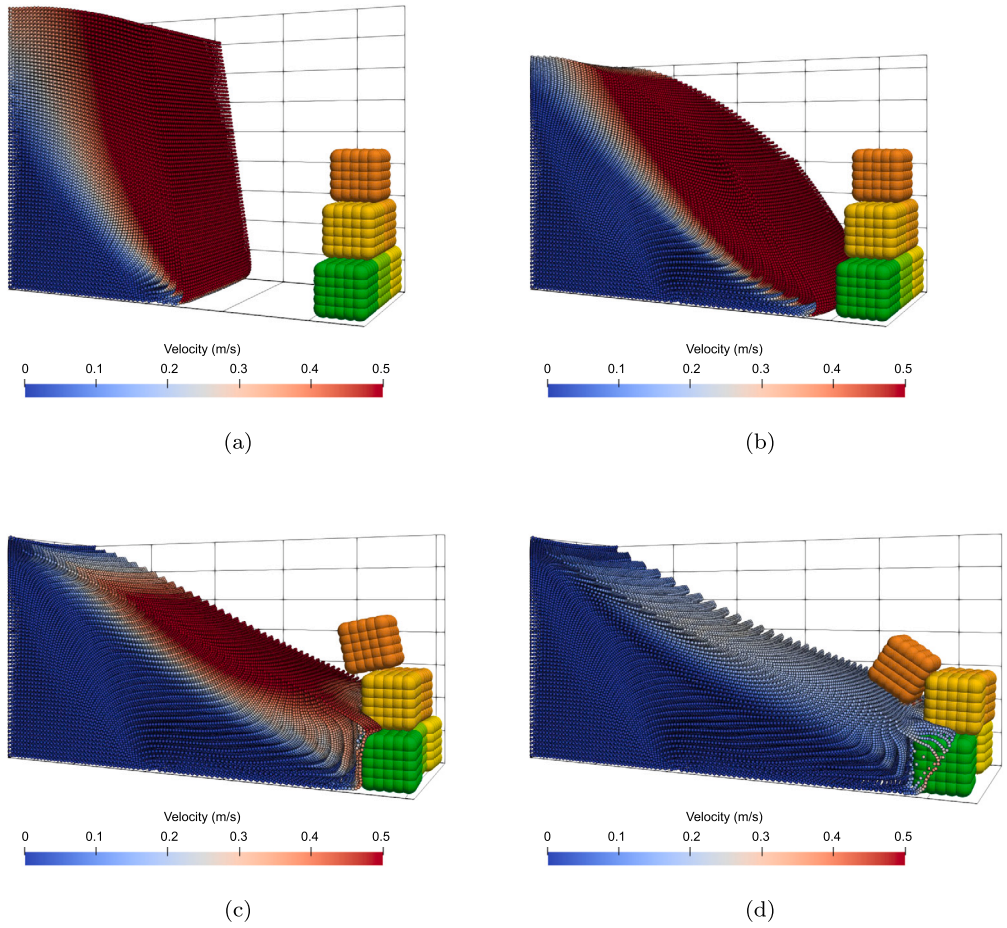


Fig. 27. Interaction between collapsed granular column and stacked cubic boxes at different time instances: (a) $t = 0.1$ s, (b) $t = 0.2$ s, (c) $t = 0.3$ s, and (d) $t = 0.4$ s.

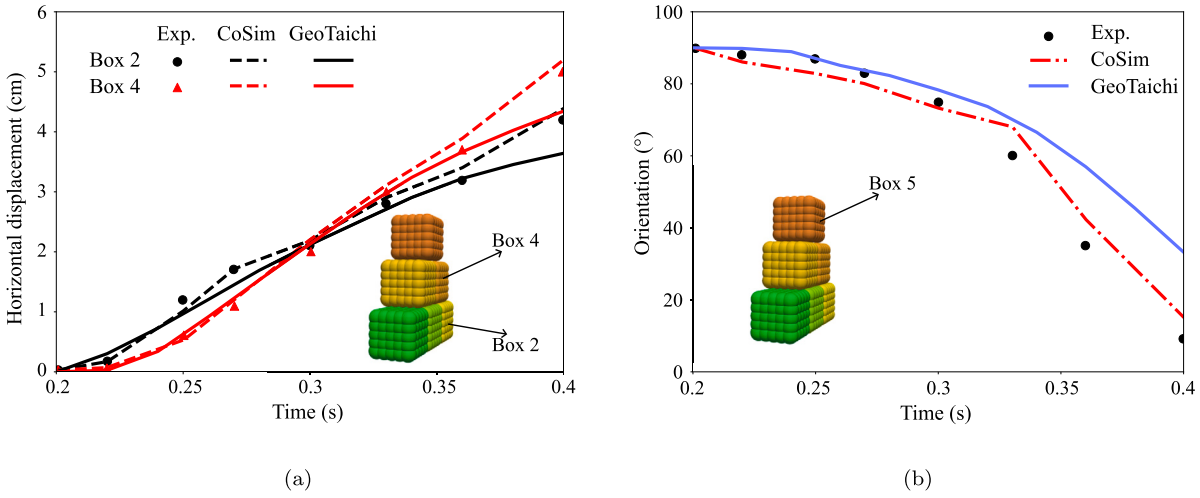


Fig. 28. Evolution of (a) translation and (b) rotation of the selected boxes.

7. Conclusions

This paper presents GeoTaichi, an open-source high-performance code specifically designed for multiscale modeling in the field of geophysics. The main characteristics of GeoTaichi can be summarized as follows:

(1) GeoTaichi features an easy-to-comprehend and easy-to-extend framework along with a user-friendly Python-based API, which empowers users to focus more on scientific problems rather than the programming language;

(2) Leveraging the Taichi parallel language, GeoTaichi is optimized using JIT compilation techniques, allowing it to be efficiently implemented on multicore CPU and GPU architectures;

(3) GeoTaichi currently offers robust and reliable modules for DEM, MPM, and MPDEM. The performance and accuracy of these modules have been validated through various benchmark tests;

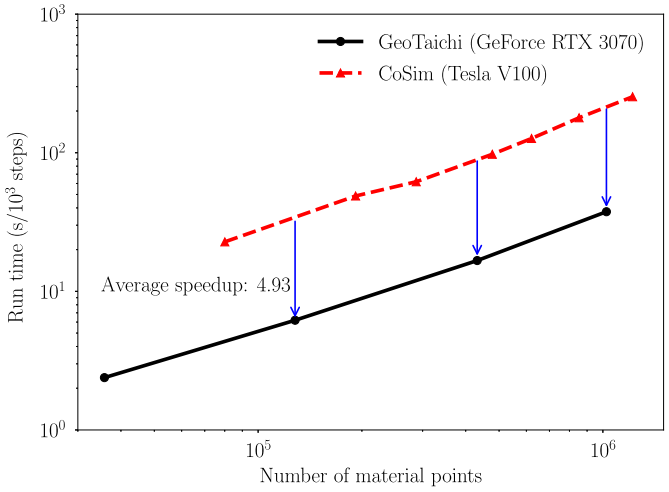


Fig. 29. Comparison of efficiency between GeoTaichi and CoSim in MPDEM simulation.

(4) When compared to the open-source DEM software MUSEN, GeoTaichi demonstrates a remarkable 337% speedup in execution time and 38% reduction in memory usage. Additionally, the MPDEM coupling scheme implemented in GeoTaichi surpasses the performance of CoSim reported in the literature [17].

In summary, GeoTaichi is a Python-based numerical simulator for multiscale geophysical problems. It is licensed under GPL v-3.0 or later. The source code is publicly hosted on GitHub at the following repository (https://github.com/Yihao-Shi/GeoTaichi). Future studies would prioritize adding support for multi-GPU architecture to achieve greater scalability and performance on systems with multiple GPUs. Contributions to GeoTaichi are highly encouraged to promote its continued development and foster its application in various domains.

CRediT authorship contribution statement

Y.H. Shi: Writing – original draft, Software, Investigation, Formal analysis. **N. Guo:** Writing – review & editing, Validation, Supervision, Investigation, Conceptualization. **Z.X. Yang:** Writing – review & editing, Resources, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the code/data at the Attach File step.

Acknowledgements

The study has been financially supported by the National Key R&D Program of China (No. 2023YFB2604200), Key R&D Program of Zhejiang Province (No. 2022C03180), Zhejiang Provincial Natural Science Foundation of China (No. LR23E080001), National Natural Science Foundation of China (No. 52078456), and Fundamental Research Funds for the Central Universities (No. 2021FZZX001-14).

Appendix A

The Python script to run MPM modeling of granular column collapse, as reported in Section 5.3, is provided in Listing 1 to demonstrate

```python
from geotaichi import *
# The simulation is run on GPU
init(arch="gpu")
# Activate MPM modulus in GeoTaichi
mpm = MPM()
# Define computational domain and background damping
mpm.set_configuration(domain=[0.55, 0.2, 0.11],
     background_damping=0.02)
# Define simulation time and time step
mpm.set_solver({"Timestep": 1e-5, "SimulationTime": 0.6,
                "SaveInterval": 0.01
               })
# Allocation GPU memory before starting simulation
mpm.memory_allocate({
                     "max_material_number": 1,
                     "max_particle_number": 5.12e5,
                     "max_constraint_number": {
                     "max_velocity_constraint": 80935,
                     "max_friction_constraint": 53703}
                     })
# Choose constitutive model and its corresponding material
     parameters
mpm.add_material("DruckerPrager",
                 {"MaterialID": 1, "Density": 2650,
                  "YoungModulus": 8.4e5, "PossionRatio": 0.3,
                  "Friction": 19.8}
                  )
# Set parameters for background grid
mpm.add_element({"ElementType": "R8N3D",
                 "ElementSize": [0.0025, 0.0025, 0.0025]
                 })
# Generate a region for material points generation
mpm.add_region({"Name": "region1",
                "Type": "Rectangle",
                "BoundingBoxPoint": [0.005, 0.005, 0.005],
                "BoundingBoxSize": [0.2, 0.05, 0.1]
                })
# Create the soil body
mpm.add_body({"Template": {"RegionName": "region1",
                           "nParticlesPerCell": 2,
                           "BodyID": 0,
                           "MaterialID": 1,
                           "ParticleStress": {"GravityField":
     True}
                           }})
# Define the boundary condition
mpm.add_boundary_condition({"BoundaryType": "FrictionConstraint",
                            "Norm": [0., 0., -1.],
                            "Friction": 0.3,
                            "StartPoint": [0., 0., 0.],
                            "EndPoint": [0.55, 0.2, 0.005]
                            })
mpm.add_boundary_condition({"BoundaryType": "VelocityConstraint",
                            "Velocity": [0., 0., 0.],
                            "StartPoint": [0., 0, 0],
                            "EndPoint": [0.005, 0.2, 0.11]
                            })
mpm.add_boundary_condition({"BoundaryType":"VelocityConstraint",
                            "StartPoint": [0., 0., 0.],
                            "EndPoint": [0.55, 0.005, 0.11],
                            "Velocity": [None, 0., None]
                            })
mpm.add_boundary_condition({"BoundaryType": "VelocityConstraint",
                            "StartPoint": [0., 0.055, 0.],
                            "EndPoint": [0.55, 0.0575, 0.11],
                            "Velocity": [None, 0., None]
                           })
# Output data in VTU format for visualization and binary files
     for postprocessing
mpm.select_save_data()
# Run the simulation
mpm.run()
# Postprocessing
mpm.postprocessing()
```

Listing 1: Python script for MPM modeling of granular column collapse.

the implementation specifics of GeoTaichi. Note that the scripts exclude certain parameters, which are set to their default values. More details can be found in the example folder.

References

[1] D. Sulsky, Z. Chen, H.L. Schreyer, A particle method for history-dependent materials, Comput. Methods Appl. Mech. Eng. 118 (1–2) (1994) 179–196.

[2] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. Astron. Soc. 181 (3) (1977) 375–389.

[3] P.A. Cundall, O.D.L. Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1) (1979) 47–65.

[4] J. Zhao, N. Guo, Unique critical state characteristics in granular media considering fabric anisotropy, Geotechnique 63 (8) (2013) 695–704.

[5] N. Guo, J. Zhao, Local fluctuations and spatial correlations in granular flows under constant-volume quasistatic shear, Phys. Rev. E 89 (4) (2014) 042208.

[6] J. Zhao, N. Guo, The interplay between anisotropy and strain localisation in granular soils: a multiscale insight, Geotechnique 65 (8) (2015) 642–656.

[7] S. Pancheshnyi, P. Ségur, J. Capeillère, A. Bourdon, Numerical simulation of filamentary discharges with parallel adaptive mesh refinement, J. Comput. Phys. 227 (13) (2008) 6574–6590.

[8] P.Y. Chen, M. Chantharayukhonthorn, Y. Yue, E. Grinspun, K. Kamrin, Hybrid discrete-continuum modeling of shear localization in granular media, J. Mech. Phys. Solids 153 (2021) 104404.

[9] V. Singer, K.B. Sautter, A. Larese, R. Wüchner, K.-U. Bletzinger, A partitioned material point method and discrete element method coupling scheme, Adv. Model. Simul. Eng. Sci. 9 (1) (2022) 1–24.

[10] B. Wang, D. Wang, M.A. Hicks, X.-T. Feng, Interplay between friction and cohesion: a spectrum of retrogressive slope failure, J. Geophys. Res., Solid Earth 128 (2) (2023) e2022JB026008.

[11] C. Liu, Q. Sun, G.G.D. Zhou, Coupling of material point method and discrete element method for granular flows impacting simulations, Int. J. Numer. Methods Eng. 115 (2) (2018) 172–188.

[12] Y. Jiang, M. Li, C. Jiang, F. Alonso-Marroquin, A hybrid material-point spheropolygon-element method for solid and granular material interaction, Int. J. Numer. Methods Eng. 121 (14) (2020) 3021–3047.

[13] S. Ren, P. Zhang, S.A. Galindo-Torres, A coupled discrete element material point method for fluid–solid–particle interactions with large deformations, Comput. Methods Appl. Mech. Eng. 395 (2022) 115023.

[14] J. Guilkey, R. Lander, L. Bonnell, A hybrid penalty and grid based contact method for the material point method, Comput. Methods Appl. Mech. Eng. 379 (2021) 113739.

[15] H. Chen, S. Zhao, J. Zhao, X. Zhou, DEM-enriched contact approach for material point method, Comput. Methods Appl. Mech. Eng. 404 (2023) 115814.

[16] J. Li, B. Wang, D. Wang, P. Zhang, P.J. Vardon, A coupled MPM-DEM method for modelling soil-rock mixtures, Comput. Geotech. 160 (2023) 105508.

[17] Z.-K. Feng, W.-J. Xu, K.U.J. Khan, A GPU based hybrid material point and discrete element method (MPDEM) algorithm and validation, Comput. Geotech. 159 (2023) 105462.

[18] J. Kozicki, F.V. Donze, YADE-OPEN DEM: an open-source software using a discrete element method to simulate granular material, Eng. Comput. 26 (7) (2009) 786–805.

[19] A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, R. Shan, M.J. Stevens, J. Tranchida, C. Trott, S.J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Comput. Phys. Commun. 271 (2022) 108171.

[20] S. Zhao, J. Zhao, SudoDEM: unleashing the predictive power of the discrete element method on simulation for non-spherical granular particles, Comput. Phys. Commun. 259 (2021) 107670.

[21] T. Weinhart, L. Orefice, M. Post, M.P. van Schrojenstein Lantman, I.F.C. Denissen, D.R. Tunuguntla, J.M.F. Tsang, H. Cheng, M.Y. Shaheen, H. Shi, P. Rapino, E. Grannonio, N. Losacco, J. Barbosa, L. Jing, J.E.A. Naranjo, S. Roy, W.K. den Otter, A.R. Thornton, Fast, flexible particle simulations — an introduction to MercuryDPM, Comput. Phys. Commun. 249 (2020) 107129.

[22] M. Dosta, V. Skorych, MUSEN: an open-source framework for GPU-accelerated DEM simulations, SoftwareX 12 (2020) 100618.

[23] P. Dadvand, R. Rossi, E. Oñate, An object-oriented environment for developing finite element codes for multi-disciplinary applications, Arch. Comput. Methods Eng. 17 (2010) 253–297.

[24] C. Lattner, V. Adve, LLVM: A compilation framework for lifelong program analysis & transformation, in: International Symposium on Code Generation and Optimization, 2004, CGO 2004, IEEE, 2004, pp. 75–86.

[25] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, F. Durand, Taichi: a language for high-performance computation on spatially sparse data structures, ACM Trans. Graph. 38 (6) (2019) 201.

[26] Y. Hu, J. Liu, X. Yang, M. Xu, Y. Kuang, W. Xu, Q. Dai, W.T. Freeman, F. Durand, QuanTaichi: a compiler for quantized simulations, ACM Trans. Graph. 40 (4) (2021) 182.

[27] Y. Zhao, C. Jiang, J. Choo, Circumventing volumetric locking in explicit material point methods: a simple, efficient, and general approach, arXiv preprint, arXiv:2209.02466, 2022.

[28] J. Yang, Y. Xu, L. Yang, Taichi-LBM3D: a single-phase and multiphase lattice Boltzmann solver on cross-platform multicore CPU/GPUs, Fluids 7 (8) (2022) 270.

[29] Y.-C. Wu, J.-L. Shao, mdapy: a flexible and efficient analysis software for molecular dynamics simulations, Comput. Phys. Commun. 290 (2023) 108764.

[30] X. Peng, Z. Fu, Z. Zhang, S. Chen, E. Ji, Q. Zhong, Two different phase field models of the explicit material point method for brittle dynamic fracture, Eng. Fract. Mech. 290 (2023) 109449.

[31] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, F. Durand, DiffTaichi: differentiable programming for physical simulation, arXiv preprint, arXiv:1910.00935, 2019.

[32] C. Yu, Y. Xu, Y. Kuang, Y. Hu, T. Liu, MeshTaichi: a compiler for efficient mesh-based operations, ACM Trans. Graph. 41 (6) (2022) 252.

[33] S.G. Bardenhagen, Energy conservation error in the material point method for solid mechanics, J. Comput. Phys. 180 (1) (2002) 383–403.

[34] F.H. Harlow, The particle-in-cell computing method for fluid dynamics, Methods Comput. Phys. 3 (1964) 319–343.

[35] J.U. Brackbill, D.B. Kothe, H.M. Ruppel, FLIP: a low-dissipation, particle-in-cell method for fluid flow, Comput. Phys. Commun. 48 (1) (1988) 25–38.

[36] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, A. Selle, A material point method for snow simulation, ACM Trans. Graph. 32 (4) (2013) 102.

[37] C.A. del Valle, V. Angelidakis, S. Roy, J.D. Muñoz, T. Pöschel, SPIRAL: an efficient algorithm for the integration of the equation of rotational motion, Comput. Phys. Commun. 297 (2024) 109077.

[38] Q. Zhou, W.-J. Xu, G.-Y. Liu, A contact detection algorithm for triangle boundary in GPU-based DEM and its application in a large-scale landslide, Comput. Geotech. 138 (2021) 104371.

[39] G. Lu, J.R. Third, C.R. Müller, Discrete element models for non-spherical particle systems: from theoretical developments to applications, Chem. Eng. Sci. 127 (2015) 425–465.

[40] K. Han, Y.T. Feng, D.R.J. Owen, Performance comparisons of tree-based and cell-based contact detection algorithms, Eng. Comput. 24 (2) (2007) 165–181.

[41] V. Ogarko, S. Luding, A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects, Comput. Phys. Commun. 183 (4) (2012) 931–936.

[42] H. Mio, A. Shimosaka, Y. Shirakawa, J. Hidaka, Cell optimization for fast contact detection in the discrete element method algorithm, Adv. Powder Technol. 18 (4) (2007) 441–453.

[43] L. Verlet, Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1) (1967) 98.

[44] M.P. Allen, D.J. Tildesley, et al., Computer simulation of liquids, Clarendon-0.12, 1987.

[45] V. Skorych, M. Dosta, Parallel CPU–GPU computing technique for discrete element method, Concurr. Comput., Pract. Exp. 34 (11) (2022) e6839.

[46] R. Fernando, et al., GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics, vol. 590, Addison-Wesley, Reading, 2004.

[47] H.R. Norouzi, PhasicFlow: a parallel, multi-architecture open-source code for DEM simulations, Comput. Phys. Commun. 291 (2023) 108821.

[48] S. Zhao, J. Zhao, W. Liang, A thread-block-wise computational framework for large-scale hierarchical continuum-discrete modeling of granular media, Int. J. Numer. Methods Eng. 122 (2) (2021) 579–608.

[49] S. Wang, Q. Zhang, S. Ji, GPU-based parallel algorithm for super-quadric discrete element method and its applications for non-spherical granular flows, Adv. Eng. Softw. 151 (2021) 102931.

[50] S. Luding, Introduction to discrete element methods: basic of contact force models and how to perform the micro-macro transition to continuum theory, Eur. J. Environ. Civ. Eng. 12 (7–8) (2008) 785–826.

[51] M. Jiang, Z. Shen, J. Wang, A novel three-dimensional contact model for granulates incorporating rolling and twisting resistances, Comput. Geotech. 65 (2015) 147–163.

[52] R. Courant, K. Friedrichs, H. Lewy, On the partial difference equations of mathematical physics, IBM J. Res. Dev. 11 (2) (1967) 215–234.

[53] W.M. Coombs, T.J. Charlton, M. Cortis, C.E. Augarde, Overcoming volumetric locking in material point methods, Comput. Methods Appl. Mech. Eng. 333 (2018) 1–21.

[54] T.J.R. Hughes, J. Winget, Finite rotation effects in numerical integration of rate constitutive equations arising in large-deformation analysis, Int. J. Numer. Methods Eng. 15 (12) (1980) 1862–1867.

[55] S.G. Bardenhagen, E.M. Kober, The generalized interpolation material point method, Comput. Model. Eng. Sci. 5 (6) (2004) 477–495.

[56] D. Sulsky, S.-J. Zhou, H.L. Schreyer, Application of a particle-in-cell method to solid mechanics, Comput. Phys. Commun. 87 (1–2) (1995) 236–252.

[57] D. Sulsky, M. Gong, Innovative Numerical Approaches for Multi-Field and Multi-Scale Problems: In Honor of Michael Ortiz's 60th Birthday, 2016, pp. 217–240.

[58] C. Jiang, C. Schroeder, A. Selle, J. Teran, A. Stomakhin, The affine particle-in-cell method, ACM Trans. Graph. 34 (4) (2015) 51.

[59] D. Fincham, Leapfrog rotational algorithms, Mol. Simul. 8 (3–5) (1992) 165–178.

[60] S. Zhao, J. Zhao, N. Guo, Universality of internal structure characteristics in granular media under shear, Phys. Rev. E 101 (1) (2020) 012906.

[61] GDR MiDi, On dense granular flows, Eur. Phys. J. E 14 (2004) 341–365.

[62] J. Christoffersen, M.M. Mehrabadi, S. Nemat-Nasser, A micromechanical description of granular material behavior, J. Appl. Mech. 48 (1981) 339–344.
[63] H.H. Bui, Lagrangian mesh-free particle method (SPH) for large deformation and post-failure of geomaterial using elasto-plastic constitutive models, PhD thesis, Ritsumeikan University, 2007.
[64] Z. Sun, H. Li, Y. Gan, H. Liu, Z. Huang, L. He, Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: a comparative study, Prog. Comput. Fluid Dyn. 18 (1) (2018) 1–18.
[65] L. Lobovskỳ, E. Botia-Vera, F. Castellana, J. Mas-Soler, A. Souto-Iglesias, Experimental investigation of dynamic pressure loads during dam break, J. Fluids Struct. 48 (2014) 407–434.
[66] C. Zhang, X. Hu, N.A. Adams, A weakly compressible SPH method based on a low-dissipation Riemann solver, J. Comput. Phys. 335 (2017) 605–620.
[67] S. Adami, X.Y. Hu, N.A. Adams, A generalized wall boundary condition for smoothed particle hydrodynamics, J. Comput. Phys. 231 (21) (2012) 7057–7075.
[68] C. Zhang, M. Rezavand, Y. Zhu, Y. Yu, D. Wu, W. Zhang, J. Wang, X. Hu, SPHinXsys: an open-source multi-physics and multi-resolution library based on smoothed particle hydrodynamics, Comput. Phys. Commun. 267 (2021) 108066.
[69] M.P. Ciamarra, A.H. Lara, A.T. Lee, D.I. Goldman, I. Vishik, H.L. Swinney, Dynamics of drag and force distributions for projectile impact in a granular medium, Phys. Rev. Lett. 92 (19) (2004) 194301.