



Development of a GPU-accelerated implicit material point method for geotechnical engineering

Bin Wang^{1,2} · PengLin Chen^{1,2} · Di Wang¹ · Lei-Lei Liu³ · Wei Zhang⁴

Received: 20 February 2023 / Accepted: 10 November 2023 / Published online: 16 December 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract

A graphic processing unit (GPU)-accelerated implicit material point method (IMPM) is proposed in this paper, aiming at solving large-scale geotechnical engineering problems efficiently. The Cholesky decomposition direct solution method and the preconditioned conjugate gradient (PCG) iteration method are implemented to solve the governing equation implicitly. In order to build an efficient parallel computation framework, the sequential processes in these solution methods are optimized by adopting advancing parallel computational algorithms. The risk of data race during parallel computation is avoided using atomic operation. The GPU-accelerated IMPM is firstly tested by a 1-D compress column and cantilever beam simulation to validate the accuracy of the proposed IMPM. Then, the computational efficiency is tested using the sand column collapse simulation. The solution of the governing equation is the most time-consuming process, occupying more than 95% of the computational time. The PCG iteration method shows higher efficiency compared to Cholesky decomposition direct solution method. By analysing the memory usage, it is found that memory occupation is the primary limitation on the simulation scale of IMPM, especially using the Cholesky decomposition direct solution method. Finally, the GPU-accelerated IMPM is implemented in the simulation of the Xinmo landslide, showing high accuracy and computational efficiency.

Keywords GPU-accelerated method · Implicit material point method · Large-scale landslides · Sparse matrix

1 Introduction

In geotechnical engineering, the safety and stability of a structure are the most concerning matters for engineers. The application of numerical methods in calculating the mechanical state and possible failure process of geotechnical engineering is thereby of great importance. During the last century, one of the most successful numerical methods, i.e. the finite element method (FEM), has

provided a robust and efficient way to estimate the safety of geotechnical engineering. However, for large deformation problems, the FEM may suffer from mesh distortion problems, resulting in the failure of a simulation [3, 48].

Sulsky et al. proposed a material point method (MPM) for large deformation problems by improving the particle-in-cell (PIC) method based on fluid mechanics and then applied it to simulate solid materials [38–40]. The MPM uses both Lagrangian and Eulerian descriptions [30], whereby all historically relevant state quantities are stored in material points, and the momentum equations are solved using a background grid. Due to the resetting of the background grid at each time step, the MPM does not suffer from the shortcomings of grid-based methods [11]. According to different time integration schemes, the MPM can be categorized into explicit and implicit formats. The explicit MPM uses an explicit integration method, allowing the current moment system state variable to be used to directly determine the next moment system state variable [49]. Among them, the central finite difference

✉ Lei-Lei Liu
csulll@foxmail.com

¹ State Key Laboratory of Geomechanics and Geotechnical Engineering, Institute of Rock and Soil Mechanics, Chinese Academy of Sciences, Wuhan, Hubei, China

² University of Chinese Academy of Sciences, Beijing, China

³ School of Geosciences and Info-Physics, Central South University, Changsha, China

⁴ College of Water Conservancy and Civil Engineering, South China Agricultural University, Guangzhou, China

approximation is one of the most widely used explicit integral methods and is suitable for problems with short time loads and high-frequency responses. Subsequently, the MPM was developed and applied to various fields [20, 23, 25, 41]. The explicit MPM has received much attention due to its high efficiency [24]. Wieckowski et al. [44] used MPM to solve the flow problem of granular material discharge in silos with flat bottom and conical hoppers. Shin [33] enhanced MPM using the principle of multi-field variation so that it can be applied to analyse landslide and debris flow problems. Wang et al. [42] used MPM to analyse the dynamic development process of regressive and progressive slopes. Zhao et al. [52] used 3D MPM to reproduce actual landslides that include debris and entrainment effects, allowing further understanding of the hazard.

However, IMPM breaks through the limitation of explicit MPM to the Courant–Friedrichs–Lewy (CFL) conditions and is suitable for problems that are more interested in low-frequency bulk motion [36, 37]. The IMPM can directly solve the mechanical equilibrium of the system, i.e. solving for the next moment system state variables involving both the current moment state variables and the next moment state variables. Implicit time integration methods often use Newmark’s method and Wilson’s method [49]. Guike and Weiss [19] develop and implement an implicit integration strategy in conjunction with the material point method (MPM), which proposes an incremental iterative solution strategy based on Newton’s method that utilizes Newmark integrals for updating the kinematic variables. Sulsky and Kaul [37] derive and validate a time-implicit discretization of the material point method for solving discrete equations using Newton’s method in combination with the conjugate gradient method or the generalized minimum residual method, where the solution is implemented using a matrix-free approach, which greatly improves numerical efficiency. Beuth et al. [7] used the implicit integration scheme to solve the three-dimensional quasi-static large-strain geotechnical problems using quadratic interpolation functions. Coombs et al. [11] established an IMPM with the previously converged Lagrangian approach, which does not need to map the derivatives of the basis function, thus reducing computational cost and algorithmic complexity. Wang et al. [43] compared IMPM with updated Lagrangian FEM and explicit MPM in terms of algorithm accuracy and time step, elaborating that the implicit MPM is a useful tool to capture post-damage behaviour in geotechnical engineering applications. Chandra et al. [8] imposed a penalty formulation for IMPM with non-flush, non-coordinated Dirichlet boundaries to solve the problem of inconsistent material boundaries and background grids. Yuan et al. [47] proposed a proposed static implicit generalized

interpolation material point method that eliminates the crossover noise of cells inherent in traditional MPM for large deformation problems in geotechnical engineering.

However, the IMPM requires the solution of a large system of linear equations at each time step, resulting in high single-step computational costs that prevent the application of the IMPM in practical geotechnical engineering simulations. In recent years, with the development of parallel acceleration technology, graphics processing unit (GPU) acceleration technology has provided a promising way to solve large-scale numerical simulation problems. GPU parallel schemes have been successfully applied to many advancing numerical simulation methods [14, 28]. For example, in recent decades, Chiang et al. [10] used a GPU scheme to accelerate the generalized interpolation material point method. Dong et al. [15] proposed a GPU parallel explicit MPM scheme with a maximum acceleration of 30.8 times for single precision and 24 times for double precision. Xia and Liang [45] used a GPU-accelerated smooth particle hydrodynamics (SPH) model in shallow water equations with an overall efficiency improvement of 7–10 times compared to the use of the central processing unit (CPU) acceleration scheme. Zhang et al. [51] used a GPU-accelerated explicit smoothed particle finite element method (SPFEM) simulation for large deformation problems in geotechnical engineering, which improves the efficiency by more than 10 times compared to the sequential CPU simulation. However, few studies involve the implementation of GPU acceleration technology in the IMPM. Possible difficulties include the assembly of a large sparse matrix and the solution of large systems of linear equations in parallel computing structures, which hinders the further development of the IMPM.

Therefore, to solve the computational efficiency problem that affects the development of the IMPM, a GPU-accelerated IMPM is proposed and applied to the analysis of large deformations in geotechnical engineering. This paper consists of four major parts. First, Sect. 2 introduces the basic theory of the IMPM and proposes the GPU parallel framework containing two implicit solution methods. Second, the proposed IMPM is validated by three examples in Sect. 3. Subsequently, the computational efficiency of the IMPM is evaluated and tested in Sect. 4 based on the simulation of sand column collapse example. Finally, the landslide process in Xinmo Village is simulated, demonstrating the reliability of the proposed GPU-accelerated IMPM in modelling large-scale geotechnical engineering problems.

2 Methodology

This section summarizes the governing equations and algorithm flow of the GPU-accelerated IMPM. For completeness of the paper, a brief introduction of the governing equations for the IMPM is provided first in Sect. 2.1. Then, in Sect. 2.2, the calculation steps of the GPU-accelerated IMPM are presented. Finally, Sect. 2.3 describes in detail the implementation procedure of the stiffness matrix assembly and the implicit solution of the governing equations.

2.1 Governing equations

In a continuous medium, the governing equations for MPM include the mass conservation (Eq. (1)) and momentum conservation (Eq. (2)) equations.

$$\frac{d\rho}{dt} + \nabla \cdot (\rho \mathbf{v}) = 0 \quad (1)$$

$$\rho \frac{d\mathbf{v}}{dt} = \mathbf{f} + \nabla \cdot \boldsymbol{\sigma} \quad (2)$$

where ∇ is the differential operator, ρ is the mass density, \mathbf{v} is the velocity, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and \mathbf{f} is the body force, e.g. gravity.

A continuous medium occupies an initial domain Ω of the three-dimensional space. The Lagrangian governing equations of mass conservation Eq. (1) and momentum conservation Eq. (2) must be satisfied in the region Ω . The corresponding displacement boundary conditions Eq. (3) and stress boundary conditions Eq. (4) are imposed for the solution, i.e.

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on } s_u \quad (3)$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \bar{\mathbf{t}} \quad \text{on } s_\sigma \quad (4)$$

where s_u is the displacement boundary region, s_σ is the stress boundary region, \mathbf{n} is the outwards unit normal vector, $\bar{\mathbf{u}}$ and $\bar{\mathbf{t}}$ are the Dirichlet prescribed displacement and the Neumann traction confined by the boundary region, respectively.

As the mass of the MPM is independent of time, the mass conservation equation Eq. (1) is satisfied. By introducing the principle of virtual displacement and using the divergence theorem in Eq. (2), the integral form of the equilibrium equation Eq. (5), which represents the dynamic governing equation at time $t + dt$ in weak form, is obtained [6, 43].

$$\int_{\Omega^{t+dt}} \mathbf{S}^{t+dt} \cdot \delta \boldsymbol{\varepsilon}^{t+dt} d\Omega = \int_{\Omega^{t+dt}} \mathbf{f}^{t+dt} \cdot \delta \mathbf{u}^{t+dt} d\Omega + \int_{s_\sigma^{t+dt}} \bar{\mathbf{t}} \cdot \delta \mathbf{u}^{t+dt} ds_\sigma \quad (5)$$

where \mathbf{S} is the second Piola–Kirchhoff stress tensor, $\boldsymbol{\varepsilon}$ is the Green–Lagrange strain tensor, dt is the time step, and $\delta \mathbf{u}$ is the virtual displacement.

The corresponding stress and strain terms are brought into the governing equation (Eq. (5)), and the Galerkin method is used to discretize the weak form of the governing equation at the material points to obtain a system of equilibrium equations in the form of a matrix as follows

$$\mathbf{K}^t \mathbf{u} = \mathbf{F}_{\text{total}}^{t+dt} \quad (6)$$

where \mathbf{K}^t is the global stiffness matrix, $\mathbf{F}_{\text{total}}^{t+dt} = \mathbf{F}_{\text{ext}}^{t+dt} - \mathbf{F}_{\text{int}}^t$ is the resultant force, $\mathbf{F}_{\text{ext}}^{t+dt}$ is the external force, and $\mathbf{F}_{\text{int}}^{t+dt}$ is the internal force. Considering the dynamic cases, the inertia term can be added directly to the governing equation and combined with Newmark's time integration scheme, as shown in Eq. (7) and Eq. (8), the dynamic governing equation matrix takes the form of Eq. (9).

$$\mathbf{v}_i^{t+dt} = \mathbf{v}_i^t + [(1 - \beta)\mathbf{a}_i^t + \beta\mathbf{a}_i^{t+dt}]dt \quad (7)$$

$$\mathbf{u}_i^{t+dt} = \mathbf{u}_i^t + \mathbf{v}_i^t dt + \left[\left(\frac{1}{2} - \alpha \right) \mathbf{a}_i^t + \alpha \mathbf{a}_i^{t+dt} \right] dt^2 \quad (8)$$

$$\left(\mathbf{K}^t + \frac{\mathbf{M}^t}{\alpha dt^2} \right) \mathbf{u} = \mathbf{F}_{\text{ext}}^{t+dt} + \mathbf{M}^t \left(\frac{1}{\alpha dt} \mathbf{v}^t + \left(\frac{1}{2\alpha} - 1 \right) \mathbf{a}^t \right) - \mathbf{F}_{\text{int}}^t \quad (9)$$

where \mathbf{v}^{t+dt} and \mathbf{u}^{t+dt} are the velocity and displacement at time $t + dt$, respectively, α and β are the time step parameters that affect the integral accuracy and stability, which are chosen as 0.25 and 0.5, respectively, in this study, and \mathbf{M}^t is the lumped mass matrix.

To increase numerical stability, it is necessary to add a damping force that is proportional to the nodal velocity at the right-end term of Eq. (9) [43]. Therefore, the governing equation matrix with the same dynamic and quasi-static form is finally obtained as follows.

$$\bar{\mathbf{K}}^t \mathbf{u} = \bar{\mathbf{F}}_{\text{total}}^{t+dt} \quad (10)$$

where $\bar{\mathbf{F}}_{\text{total}}^{t+dt} = \mathbf{F}_{\text{ext}}^{t+dt} + \mathbf{M}^t \left(\frac{1}{\alpha dt} \mathbf{v}^t + \left(\frac{1}{2\alpha} - 1 \right) \mathbf{a}^t \right) + \mathbf{F}_{\text{damp}}^t$, $\bar{\mathbf{K}}^t = \mathbf{K}^t + \frac{\mathbf{M}^t}{\alpha dt^2}$. $\mathbf{F}_{\text{damp}}^t$ uses local non-viscous damping as described by Cundall [12].

2.2 IMPM format

Prior to the introduction of the parallel strategy of the IMPM, the main stages within a single time step and the manipulations involved in the parallel computing are addressed briefly. In MPM, the material region Ω is discretized by a series of material points, which carry information such as the position, momentum, stress and strain of the material. The state variables of the material points are mapped onto the background grid through the shape

function to solve the momentum equation in each time step. After the new state variables are obtained, they are mapped back to the material points by the shape function. When the current time step ends, the background grid is cleared. Therefore, the single time step of the IMPM can be divided into three calculation stages.

(a) Mapping phase

In the mapping phase, the material region is discretized into material points, and the state variable of each material point is mapped to the corresponding grid node according to the shape function.

$$m_i = \sum_p N_i(\mathbf{x}_p^t) m_p \tag{11}$$

$$\mathbf{P}_i^t = \sum_p N_i(\mathbf{x}_p^t) m_p \mathbf{v}_p^t \tag{12}$$

$$\mathbf{F}_i^{\text{ext}} = \sum_p N_i(\mathbf{x}_p^t) \mathbf{F}_p^{\text{ext}} \tag{13}$$

$$\mathbf{F}_i^{\text{int}} = - \sum_p \dot{N}_i(\mathbf{x}_p^t) \boldsymbol{\sigma}_p^t V_p^t \tag{14}$$

where i is the grid node, p are the material points in the supporting domain of the grid node, m is the mass of the material point, \mathbf{P} is the momentum of the grid node, \mathbf{v} is the velocity of the material point, and V is the volume of the material point.

(b) Lagrangian phase

In the Lagrangian phase, the governing equation Eq. (10) is solved to obtain the single-step displacement increment \mathbf{u} . Using the obtained displacement increment \mathbf{u} , the motion variables of the grid are updated with Newmark’s time integration scheme Eqs. (7) and (8).

(c) Convection phase

The convection phase includes three substeps. First, the state variables of the grid are mapped back to the material points, as described by Eqs. (15) and (16). Second, the velocity and position of the material point are updated by Eqs. (17) and (18), respectively. Finally, the information on the background grid is cleared for the next mapping phase.

$$\mathbf{u}_p^{t+dt} = \sum_{i=1}^{N_n} N_i(\mathbf{x}_p^t) \mathbf{u}_i^{t+dt} \tag{15}$$

$$\mathbf{a}_p^{t+dt} = \sum_{i=1}^{N_n} N_i(\mathbf{x}_p^t) \mathbf{a}_i^{t+dt} \tag{16}$$

$$\mathbf{v}_p^{t+dt} = \mathbf{v}_p^t + \frac{1}{2} (\mathbf{a}_p^t + \mathbf{a}_p^{t+dt}) dt \tag{17}$$

$$\mathbf{x}_p^{t+dt} = \mathbf{x}_p^t + \mathbf{u}_p^{t+dt} \tag{18}$$

where \mathbf{u} is the displacement increment, \mathbf{a} is the acceleration, and \mathbf{x} is the position of the material point.

In this paper, the GPU-accelerated IMPM is proposed based on the IMPM framework of Wang et al. [43]. All the calculation steps are parallelized on the GPU, which achieves the optimal parallel effect of the code. The implementation procedure of the GPU-accelerated IMPM is as follows (as shown in Fig. 1):

- (1) Read the initial material information and discrete material region.
- (2) Clear the information of the background grid nodes.
- (3) Calculate shape functions and their derivatives. The mass, momentum, internal forces, and external forces of the material points are obtained on the nodes of the grid through the shape function.

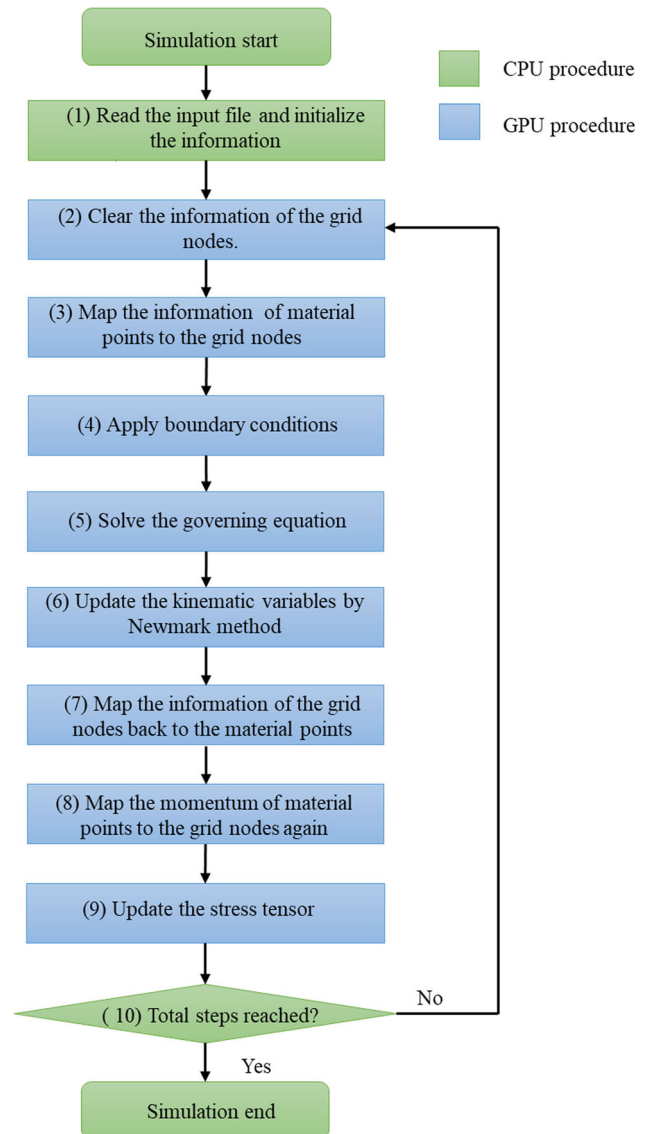


Fig. 1 GPU acceleration scheme of the IMPM

- (4) Apply the boundary conditions.
- (5) Solve the matrix-form governing equation i.e. Eq. (10), to obtain the nodal displacements (see Sect. 2.3 for details).
- (6) Update the grid node kinematic variables, i.e. the velocity and acceleration, with the nodal displacements obtained in step (5).
- (7) Map the displacement increments, velocities, and accelerations of the grid nodes back to the material points to obtain information on the material points at time $t + dt$.
- (8) Map the updated material point momentum back to the background grid and perform step (4) again.
- (9) Calculate the strain rate and strain tensor and update the stress tensor.
- (10) If the current step is greater than or equal to the total step, then the loop ends and the program are terminated. Otherwise, jump to step (2).

Compared to the explicit integral method, step (5) is special and difficult to implement in a parallelized computational framework. In Fig. 1, two solving methods are provided for step (5), namely the Cholesky decomposition direct solution method and the preconditioned conjugate gradient (PCG) iterative method. The Cholesky decomposition direct solution method can directly calculate the exact solution of each step, only requiring the matrix to be invertible. It is applicable in a wide range of cases, and the rate of solving depends only on the size of the calculation. However, this method needs to assemble the global stiffness matrix, which requires considerable memory space. Although the sparse matrix is stored optimally using the compressed sparse row (CSR) format, the memory usage of this approach is still a major constraint on large-scale calculations. The other solution method is the PCG iterative method, which does not require the assembly of the global stiffness matrix. PCG method may greatly release the memory, but the solving speed is closely related to the coefficient matrix [37]. It is suitable for the case where the element stiffness matrix is dominant. Otherwise, the excessive number of iterations will sharply increase the computational cost.

The two solution methods are described as follows (as shown in Fig. 2).

- (5.1) Preprocess of computational models. Loop over the grid nodes and neglect the nodes without any physical information to determine the number of active elements and degrees of freedom. Loop over the active element nodes and build lists to cast the local element number to grid number for efficient data access.
- (5.2) Assemble the internal and external force vectors. Calculate the resultant force of all grid nodes. Three

vectors are created with the magnitude of the degrees of freedom, i.e. the displacement increment \mathbf{u} , the resultant force \mathbf{F} , and the lumped mass matrix \mathbf{M} . The displacement increment \mathbf{u} is assigned an initial value equal to zero. The resultant force vector \mathbf{F} and the lumped mass matrix \mathbf{M} are assigned values determined by the mapping relationship between the grid nodes and the degree of freedom. The lumped mass matrix \mathbf{M} is only diagonally nonzero and can be stored in a vector form.

- (5.3) Calculate the stiffness matrix for each element. Loop over the material point in each element and calculate the elementary stiffness matrix.

Case 1: The Cholesky decomposition direct solution method:

To solve the matrix-form governing equation i.e. Eq. (10) with case 1, it is necessary to construct the global stiffness matrix. In practical engineering problems, the magnitude of the grid nodes can reach millions, leading to a tremendous global stiffness matrix that consumes too many memories of the computer and damages the computational efficiency. As the global stiffness matrix is sparse, one solution is to use the CSR format to optimize the storage problem. The detailed information about CSR format and global stiffness can be found in Sect. 2.3. Therefore, the Cholesky decomposition direct solution method includes two major steps as follows:

- (5.4) Assemble the global stiffness matrix in CSR format.
- (5.5) Solve Eq. (10) using the Cholesky decomposition direct solution method. The obtained displacement increment \mathbf{u} is mapped to the activated grid nodes.

Case 2: The preconditioned conjugate gradient iteration method:

In a PCG iterative method, the result of $\bar{\mathbf{K}}\mathbf{u}$ is obtained by sum up the result of each element, i.e. $\sum_{i=1}^{N_n} \bar{\mathbf{K}}_e^i \mathbf{u}_e$, which can avoid the assembling of global stiffness matrix and also suitable for parallel computation. The solution of \mathbf{u}_e is obtained until the iteration is convergence. The key step of PCG iterative method is to preprocess the coefficient matrix, which greatly improves the speed of iteration convergence. Diagonal preprocessing was used for preconditioning. When the coefficient matrix of the example is strictly diagonal dominant, this processing method can improve the rate of convergence. The PCG iterative method includes two major steps as follows:

- (5.4) Preprocess the element stiffness matrix of each element. The preconditioning method adopts the diagonal method.

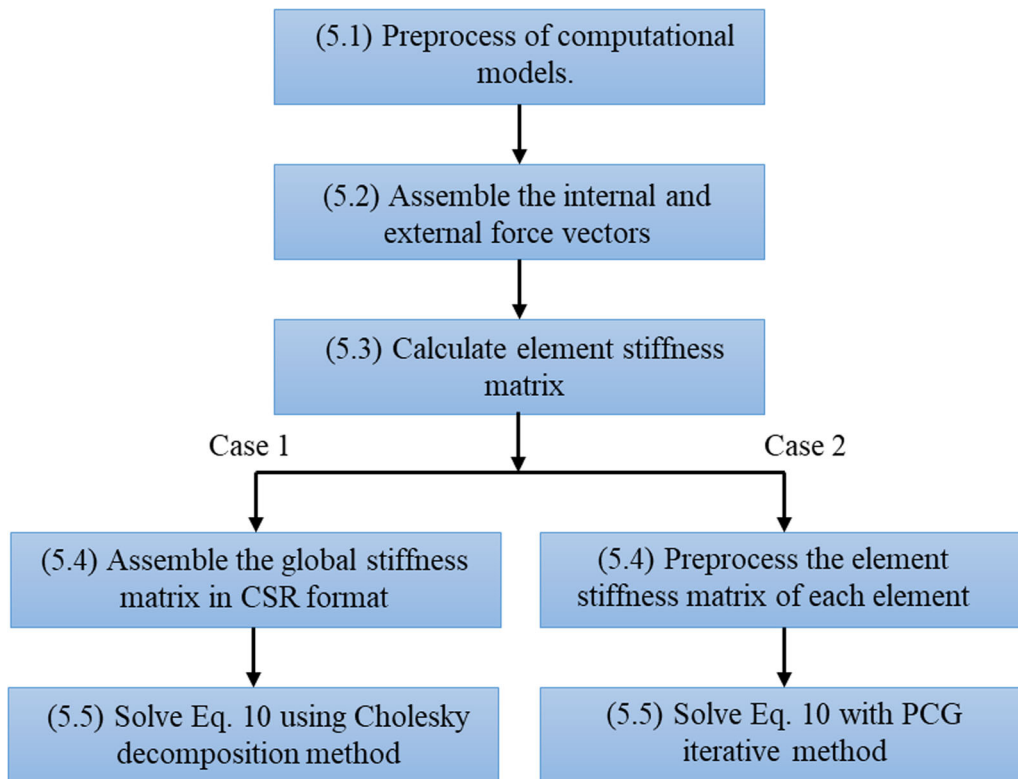


Fig. 2 Solution methods of the IMPM

- (5.5) Solve Eq. (10) with the PCG iterative method. The obtained displacement increment \mathbf{u} is mapped to the activated grid nodes.

2.3 GPU parallelization strategy

2.3.1 Data race problem

In the IMPM, processes may occur on each material point or grid point separately, such as mapping the information of the material points to the grid nodes (P2G), mapping the information of the grid nodes back to the material points (G2P), boundary conditions, and stress updating. When the information of material points and surrounding grid nodes are provided, these processes can be solved parallelized to accelerate the simulation. For example, in a P2G process, the information of each material point is assigned to a thread, and the information mapped to grid points can be solved independently using Eqs. (11–14) (shown in Fig. 3). Threads are calculated by the multiple cores of the GPU, which are invoked simultaneously, reducing the computational time significantly compared to a sequential process. The number of threads sometimes can be larger than the number of blocksize (maximum number of threads in each block). Then, these threads are divided into several blocks, and the blocks are executed sequentially by the GPU. Due

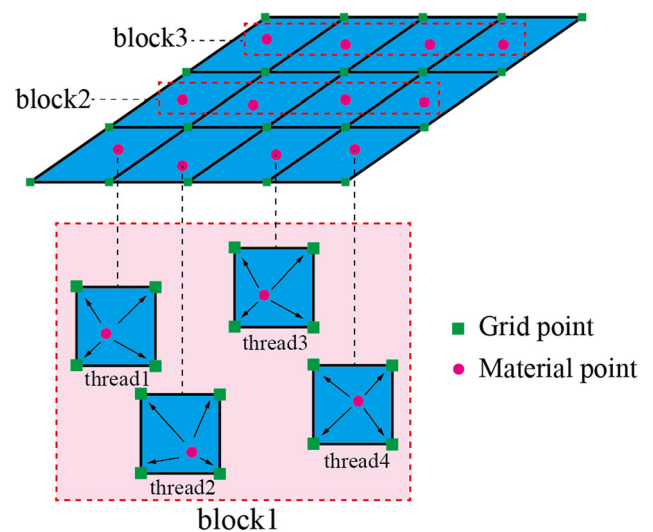


Fig. 3 GPU-accelerated IMPM parallel schemes

to a large number of GPU cores and high data transfer efficiency, this process can be solved quickly.

However, there are still some processes in IMPM that may require data communication between threads or sequential procedures, such as stiffness matrix calculation, global stiffness assembling or solving of the governing equation implicitly, which may significantly damage the parallel efficiency of IMPM. Therefore, these problems in

IMPM are studied and optimized in this paper to obtain a high-efficiency GPU-accelerated IMPM framework.

The data communication may cause a data race problem, which refers to the read–modify–write operation of data from different threads on the same memory location. When the read–modify–write operations are performed by several threads simultaneously, the result will become one of the outputs from a random thread rather than the combination of several threads. For instance, when material point mass in different threads is mapped to the same grid node, the mass on the grid should be the cumulative mass over each thread. However, during the computation, each thread gets the initial mass of the grid and sums up the mapped mass from only one material point on this thread. This result will override the memory without comparison or summation with other threads, leading to an incorrect result. Dong et al. think the benefit of GPU parallelization can be trivial when dealing with the data race problem, so they adopt a CPU sequential process to avoid data race in parallel computation [15]. However, the data transfer between CPU and GPU can be expensive, especially when the model is large. In IMPM, in addition to the operation of P2G, there is also the risk of data race during stiffness matrix calculation, global stiffness assembling, etc. The CPU sequential process may significantly damage the computational efficiency, which is not an appropriate solution to the data race problem in IMPM. Therefore, the atomic operation is introduced in this framework. The basic atomic operation in hardware changes the data race problem into a read–modify–write operation performed by a single hardware instruction on a memory location address [13]. The hardware ensures that no other threads can perform another read–modify–write operation on the same location until the current atomic operation is complete. The atomic operation limits the data race problem to the minimum scale and avoids the data transfer between CPU and GPU. The atomic manipulation method has also been applied by Hu et al. in developing MLS-MPM, showing high efficiency in explicit MPM simulation [21, 22].

2.3.2 Fast active element scan procedure

Unlike FEM, MPM has many background grids that are not involved in the calculation. Therefore, it is necessary to carry out a preprocess to divide the background grid into material area and empty area. The material area refers to the background grid that contains the material points; then, the remaining background grid is obviously the empty region. All grid nodes in the material area will be activated. The nodes with mass larger than zero are regarded as active nodes, which should be considered when calculating the global stiffness matrix. When all eight nodes of a background grid are activated, the background grid element will

be activated. The total number of degrees of freedom is determined by the activated grid nodes, and an activated node contains three degrees of freedom in the x , y and z directions, respectively. The active grid node, the degree of freedom and the activation element establish a mutual mapping relationship by constructing two mapping lists. One of the mapping lists indicate the relationship between the activated grid nodes and the degree of freedom number. The other list establishes a mapping relationship between the activated grid nodes and the activated grid elements. Through the mapping list, the relationship of elements, degrees of freedom and grid nodes can be accessed quickly by each paralleled thread in GPU independently.

2.3.3 Sequential operation optimization in IMPM

Sequential processes refer to the operation of counting, sorting or reducing as the operation of each thread may change the results of other threads, which is unsuitable for parallel computation. These sequential processes are involved in counting the number of degrees of freedom, the building of compression matrix, solution of systems of linear equations, etc. These sequential procedures are inevitable but can be optimized by adopting advancing algorithms to boost parallel efficiency. For instance, higher computational efficiency can be obtained by introducing the Radix Sort algorithm compared to comparison-based sorting algorithms such as Merge Sort or Enumeration Sort [5]. Therefore, two optimized parallel computation packages thrust and cuSOLVER are introduced to utilize the advancing parallel algorithms and obtain high computational efficiency [5, 31].

To further explain the problems and solutions in GPU-accelerated IMPM framework, the global matrix assembling process with acceleration procedure are illustrated and discussed. As shown in Fig. 4, K is the global stiffness matrix which is assembled by the element stiffness including K_e^i and K_e^j . The variable V represents the value of element stiffness matrix, where the subscript indicates the position in the global stiffness matrix, and the superscript indicates the element number. When assembling the global stiffness matrix, the values that share the same row and col are sum up as shown in Fig. 4a.

The global stiffness matrix is sparse, which contains many zero values. The storage of these zero values may take up too much memory and is impossible for large-scale modelling. Therefore, only nonzero values should be stored with their corresponding row and column, forming a compressed sparse matrix format. In this study, a CSR format is adopted to store the global spare matrix in a compressed way. First, the element stiffness matrix is calculated in each element, and the row and column index

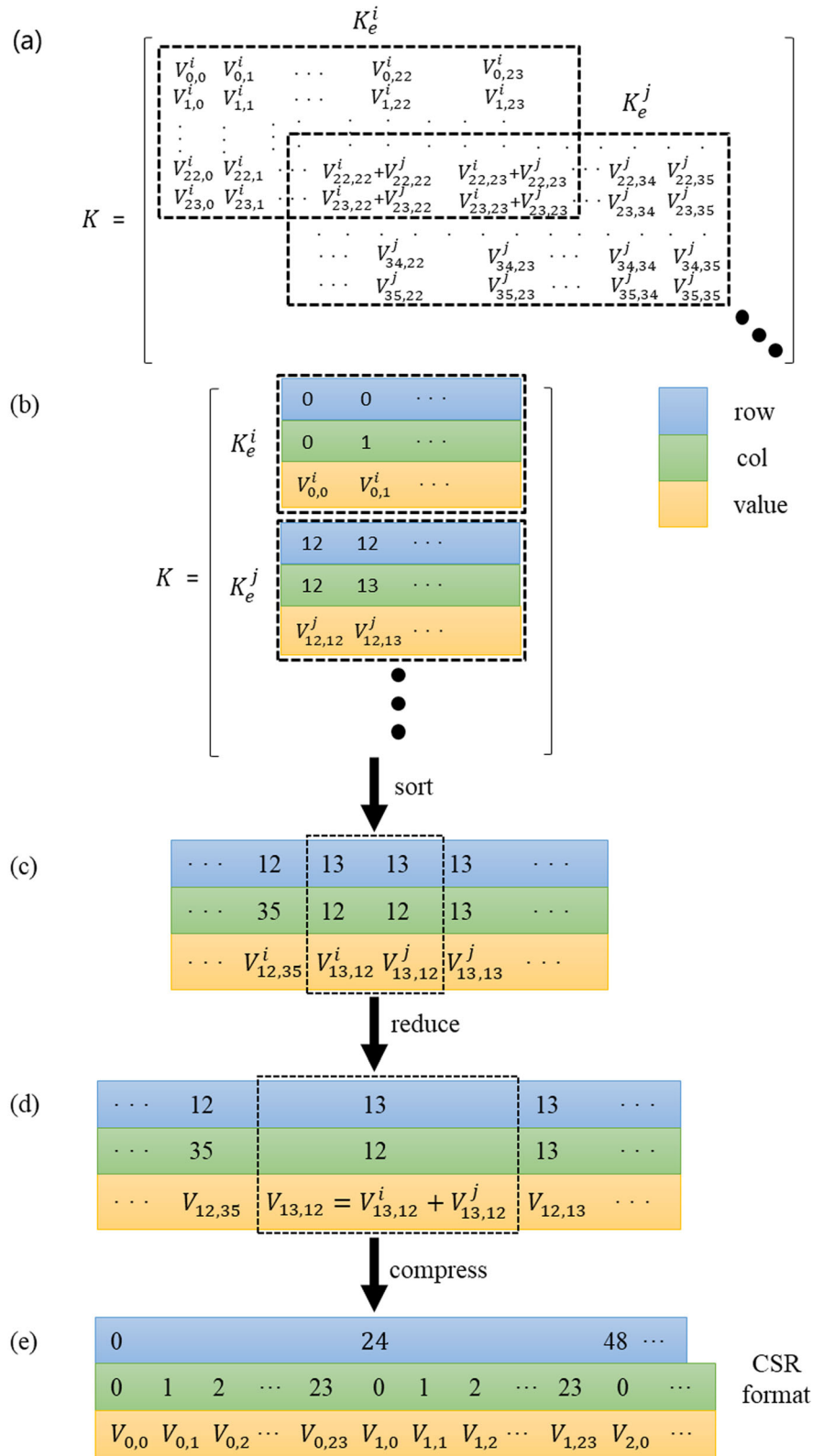


Fig. 4 The process of assembling the global stiffness matrix in CSR format

in global stiffness matrix is derived by the grid number. The element stiffness matrices are stored as three arrays, *row*, *col*, and *value*, as in Fig. 4b. *Row* and *col* denote the row and column numbers of in global stiffness matrix, and *Value* indicates the element stiffness value. Then, these arrays are then combined for the following operation. Because the compressed format requires an ascending order of row and column index, a sort operation is then adopted. After the sorting operation, the arrays are organized in ascending order, and the items with the same position (i.e. the same row and column index) in the global matrix are placed next to each other, as in Fig. 4c. Subsequently, a reduce operation is carried out to sum up these items, which is the assembling procedure shown in Fig. 4d. This procedure includes a comparison between different threads and complex memory management. Finally, the row compression operation is introduced to further save the memory usage where the row number is changed to represent the total number of nonzero values above this row, as in Fig. 4e. This operation requires a cumulative counting over each item in the arrays, which is also difficult to implement directly in parallel. After these three steps, the global stiffness matrix in CSR format is formed with all nonzero items in three one-dimensional arrays. The assembling process takes advantage of parallel acceleration as much as possible. Compared to a CPU sequential procedure, these parallel optimization processes save roughly 66% computation time, showing a significant improvement in efficiency.

Besides the global stiffness matrix assembling, these sequential processes exist in nearly all the procedures when solving the governing equation, i.e. step (5) in Sect. 2.2, including both the Cholesky decomposition direct solution method and the PCG iteration method. By adopting advancing parallelized algorithms, the computational cost of these sequential processes is reduced, contributing to a high-efficient GPU-accelerated IMPM framework.

3 Validation

This section verifies the robustness and applicability of the proposed GPU-accelerated IMPM through three numerical examples, including both quasi-static and dynamic problems. First, the GPU-accelerated IMPM is verified by simulating one-dimensional weight compression and compared with the analytical solution. Second, the large deformation cantilever beam is modelled to verify the

ability of the program to accommodate large deformation. Third, by comparing the Lube sand column collapse experiment [29], the ability of the proposed algorithm to simulate large deformation processes of dynamic geotechnical problems is further proven.

3.1 One-dimensional weight compression column

One-dimensional self-weight compression is a quasi-static problem that applies a body force to a column. The self-weight compression column test follows the example described by Bardenhagen and Kober [2]. The initial height of the column $H = 50$ m, with a fixed boundary at the bottom and roller support at the side, as shown in Fig. 5. The linear elastic material model is implemented with Young's modulus $E = 1$ MPa and Poisson's ratio $\mu = 0.0$. Damping factor set to 0.05. The applied body force increases linearly from 0 to 200 N/m³ in 20 time steps. For the parameter selection of this model, the maximum vertical stress $\sigma = 10$ kPa when the column is stabilized. The analytical solution of stress under arbitrary body force is described as follows:

$$\sigma(x) = E \left\{ \sqrt{\frac{2\rho b}{E}(\Delta - x) + 1} - 1 \right\} (0 \leq x \leq \Delta)$$

where x represents the position of the current configuration, ρ represents the density of the column, and Δ represents the height of the current column. The material region is

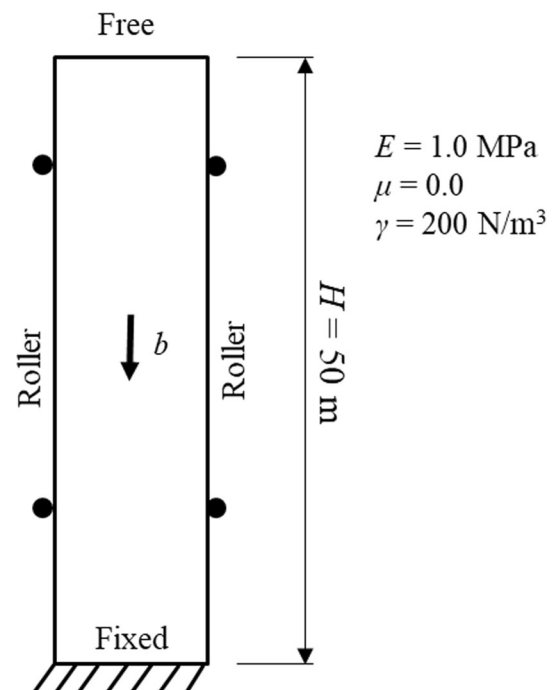


Fig. 5 One-dimensional self-weight column model

discretized into 400 material points, and the background element grid is $1\text{ m} \times 1\text{ m} \times 1\text{ m}$. To prove the robustness of the GPU-accelerated IMPM, the results are compared with the static equilibrium analytical solution of Bardenhagen and Kober [2], as shown in Fig. 6.

Figure 6a shows that both explicit MPM and IMPM may correctly simulate the stress field of the self-weight column. However, the IMPM is more stable than the explicit MPM with the same time step. When the time step is set as $1 \times 10^{-4}\text{ s}$, the IMPM shows no stress oscillations, but the explicit MPM produces a large stress oscillation, and the stress is seriously deteriorated. Until the time step is reduced to two orders of magnitude smaller than the IMPM, the stress oscillations is basically eliminated.

In Fig. 6b, with increasing time step dt , it is shown that the relative error of the explicit MPM top displacement increases gradually and that the relative error of the GPU-accelerated IMPM remains unchanged. Therefore, the IMPM shows better stability and accuracy than the explicit MPM with varying time steps. Regarding the computational cost, one-dimensional compression is a small-scale example of robustness verification, and none of them are computationally expensive. Although the explicit MPM do have significant computational advantages in small-scale example, the implicit material point still has some advantages in order to achieve the same accuracy. When time step of explicit MPM is $2 \times 10^{-6}\text{ s}$, the relative error and computation time of IMPM (case 1) are basically the same as that of explicit MPM, while IMPM (case 2) using a PCG method cost less calculation time, i.e. nearly 75% calculation time saved, as in Table 1.

3.2 Cantilever beam of concentrated force

In this section, the linear elastic cantilever beam example is presented to verify the ability of the GPU-accelerated IMPM to solve large deformation problems. A large deformation occurs by applying a concentrated force P at the right end of the cantilever beam. The cantilever beam applies a downward concentrated force on the material points in the middle of the right end, and the left end is fixed. The model has a size of $10 \times 1 \times 1\text{ m}^3$ ($L \times W \times H$), as shown in Fig. 7. The linear elastic material parameters are Young's modulus $E = 1.0\text{ GPa}$ and Poisson's ratio $\mu = 0.0$ [26]. The time step is set to $1 \times 10^{-7}\text{ s}$. The damping factor was chosen to be 0.05. The element grid size is set to 0.2 m, and each element has 64 material points.

Figure 8 shows the stress distribution of the beam in a horizontal direction at different times for a downward concentrated force $P = 3000\text{ KN}$, and the position of the neutral axis can be clearly seen. The distribution of the horizontal stress is perfectly symmetrical, and the transition is smooth. In Fig. 8a, when the load is applied in the middle, the lower part of the beam is compressed while the upper part is stretched. Figure 8b, c shows that the upper tension and lower compression begin to appear clearly. Then, the beam bends continuously until it reaches an equilibrium state, as shown in Fig. 8d, showing a very large deformation.

The cantilever beam tests are further carried out with different applied concentrated forces. According to Bathe and Bolourchi [4], the analytical solution of a cantilever beam considering geometric nonlinearity can be described as the relationship between the tip deflection ratio w/L at

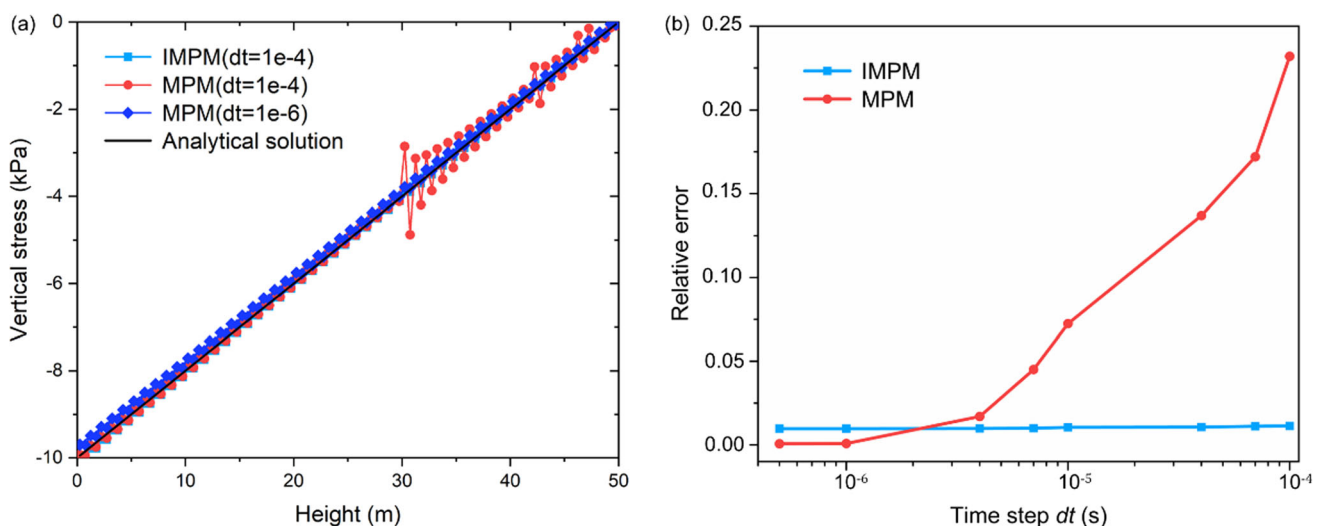


Fig. 6 One-dimensional compression column. **a** Numerical and analytical solution of MPM; **b** Relative error of displacement in different time steps of MPM

Table 1 One-dimensional compression calculation cost

Method	Total time	Time step	Computational cost	Relative error
IMPMM (Case 1)	4 s	1×10^{-4} s	421.657 s	0.014
IMPMM (Case 2)		1×10^{-4} s	95.408 s	0.018
Explicit MPM		1×10^{-4} s	11.586 s	0.227
Explicit MPM		2×10^{-6} s	408.893 s	0.012

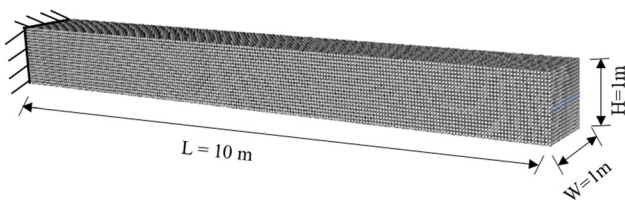


Fig. 7 Cantilever beam model

the stabilized state and the dimensionless load parameter k , where w is the tip displacement, and L is the initial length of the cantilever beam. The load parameter is determined by the geometry of the beam and material parameters $k = PL^2/EI$, where P is the concentrated force applied to

the material point at the neutral axis of the tip and I is the moment of inertia of the beam cross section. The comparison between the simulation results of the proposed IMPMM and the analytical solutions [50] is shown in Fig. 9. The simulation result is shown a good consistence with the analytical solution with varying loading parameters. This result demonstrates that our proposed IMPMM can accurately reproduce structure large deformations.

To demonstrate the convergence of the GPU-accelerated IMPMM, the material points were encrypted and sparse, respectively. Each background grid contains 8 material points with five grid sizes of 0.125, 0.2, 0.3, 0.4 and 0.5 m, respectively. We choose the loading parameter $k = 3.6$ for this convergence analysis. The relative errors are $\delta = \left| \frac{w^{IMPMM} - w^{Analytical}}{w^{Analytical}} \right|$, where w^{IMPMM} is the result of IMPMM and $w^{Analytical}$ is the analytical solution. Since the time step dt needs to be changed when considering the small size grid, the unified small time step dt is 1×10^{-7} s, and the damping factor is 0.005. The other parameters are consistent in the simulation. It is shown that when the grid is encrypted, the error reduces and finally converges towards a constant value at around 1.4%. The idea is to choose a grid size of 0.2 m to achieve the highest computational efficiency while ensuring high computational accuracy (Fig. 10).

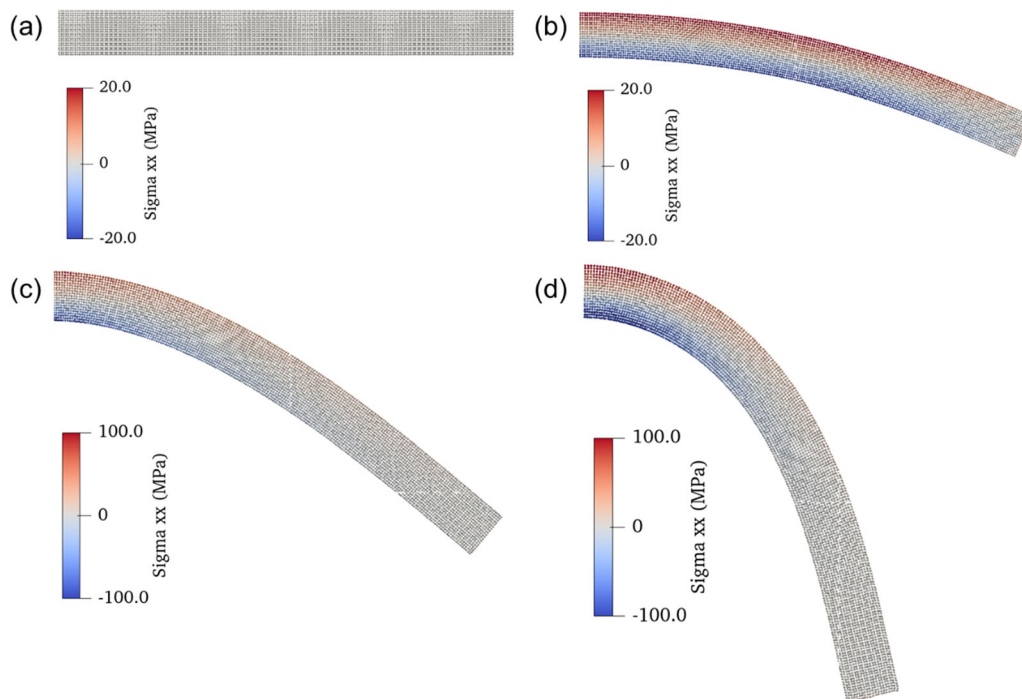


Fig. 8 Configuration and horizontal stress σ_{xx} of the cantilever beam at different times. **a** $t = 0.0$ s; **b** $t = 2 \times 10^{-3}$ s; **c** $t = 4 \times 10^{-3}$ s; and **d** $t = 0.01$ s

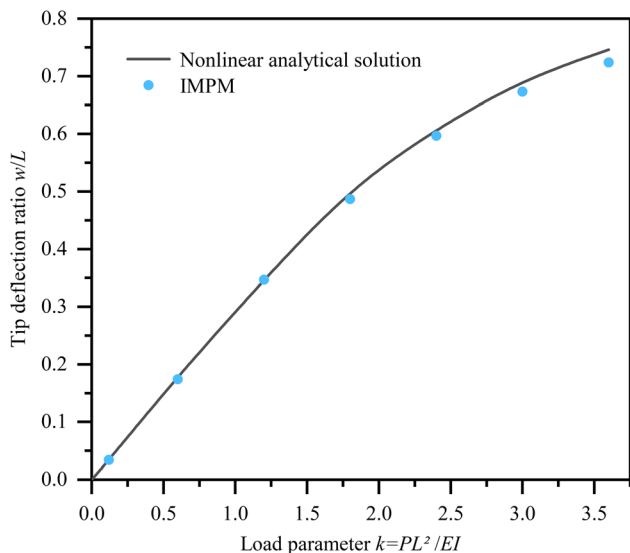


Fig. 9 Results at different load parameters k

3.3 Sand column collapse

In this section, the dynamic large deformation problem is verified by simulating the sand column collapse experiment carried out by Lube et al. [29]. The initial geometric size of the sand column is 0.63 m high and 0.09 m wide, as shown in Fig. 11. To improve the numerical stability of the dynamic problems, the described local non-viscous damping is used to overcome the energy loss caused by the friction between sand particles. The damping factor is chosen as 0.10. The discretization of the model includes 36,288 material points, and the background grid is set to 0.01 m × 0.01 m × 0.01 m, with 64 material points in each element.

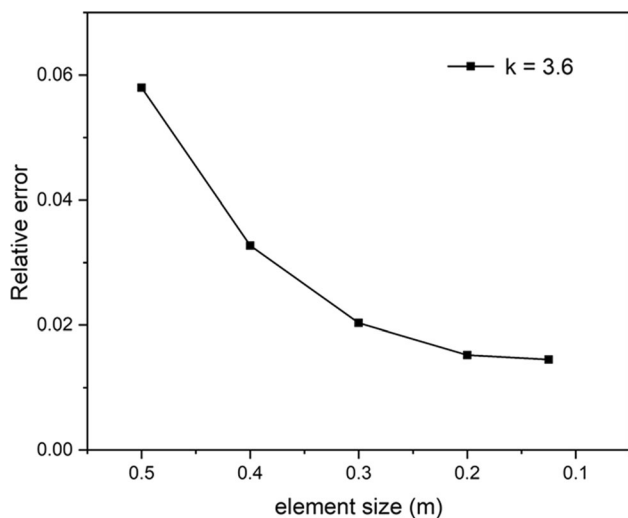


Fig. 10 Relative error analysis of different grids

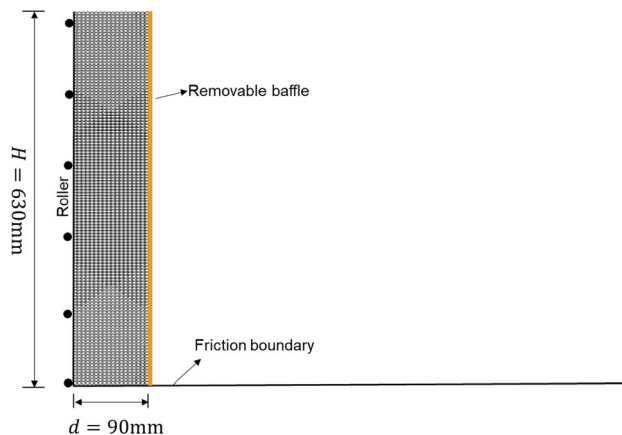


Fig. 11 Sand column collapse model

The mechanical properties of sand columns are described by the Drucker–Prager (DP) constitutive model. The parameters include Young’s modulus $E = 1.82$ MPa, Poisson’s ratio $\mu = 0.3$, cohesion $c = 0$ MPa, friction angle $\varphi = 31^\circ$, dilatancy angle $\phi = 1^\circ$ and density $\rho = 2650$ kg/m³ (see Table 2) [29, 34, 46]. The simulation is divided into two stages. In the first stage, the column is vertically deformed under gravity load. The baffle is set on the right side of the model. The left side of the model is constrained with rollers, and the bottom is a fixed boundary until the equilibrium state is reached. In the second stage, the baffle boundary is removed, and then, the bottom is given as a friction boundary to simulate the response between the soil and floor in the process of soil flow. The friction boundary refers to Bandara [1], and the friction coefficient μ_0 between the soil and the base of the model is 0.55. The time step is set as 5×10^{-5} s.

The collapse process is shown in Fig. 12, which is in good agreement with the experimental results of Lube et al. [29]. At 1.2 s, the collapse process ends, and the sand column becomes stationary. It can be seen from the final subgraph of Fig. 12 that the free surface and run-out distance of the column collapse are basically consistent with

Table 2 Material parameters for the sand column collapse example

Parameter (Unit)	Symbol	Value
Young’s modulus (MPa)	E	1.82
Poisson’s ratio (–)	μ	0.3
Cohesion (kPa)	c	0
Friction angle (°)	φ	31
Dilation angle (°)	ϕ	1
Density (kg/m ³)	ρ	2650
Damping factor (–)	α	0.1
Friction coefficient (–)	μ_0	0.50

the experimental results. The height and final run-out distance of the sand column are 0.196 m and 0.8145 m, respectively [46]. In Fig. 12d, the stacking height is 0.187 m, and the final run-out distance is 0.820 m. The stacking height of the numerical simulation is slightly lower than that of the experiment, while the final run-out distance is nearly identical to that of the experiment. The standard DP model neglects the energy loss due to particle collisions [52]. It leads to a lower height of the collapsed sand column than the experimental values, which can be improved by employing an advanced constitutive model [34, 46].

According to Lube et al. [29], the run-out distance d_∞ is 0.81454 m for this case. The constitutive parameters are referenced from Yuan et al. and experiment by Lube et al. [29, 34, 46]. The damping factor and friction coefficient are calibrated to realize a consistent result with the experimental observation. A parameter sensitivity analysis is provided in Fig. 13a, showing the influence of the damping factor and friction coefficient. It can be seen that the increasing damping factor and friction coefficient reduce the run-out distance d_∞ of the deposit. When the friction coefficient become 0.55 or higher, the reduction coverage

and the run-out distance d_∞ become constant with increasing friction coefficient.

For grid refinement analysis, the grid sizes were chosen to be 0.0025, 0.005, 0.01, 0.02 and 0.04, respectively. Eight material points were placed in each grid. The relative error is defined as $\delta = \left| \frac{d_\infty^{\text{IMPM}} - d_\infty^{\text{experiment}}}{d_\infty^{\text{experiment}}} \right|$, where d_∞^{IMPM} is the result of the IMPM and $d_\infty^{\text{experiment}}$ is the result of the experiment. The results in Fig. 13b show that the error decreases after encrypting the grid, and the error remains essentially unchanged at a grid of 0.01 m for the same selected parameters.

4 Efficiency study

In this section, the acceleration effect of the GPU-accelerated IMPM is tested by counting the computational cost of each time step during the simulation of sand column collapse. The number of particles is continuously increased to the maximum scale that can be calculated by a single personal computer (PC) with an i5-12600KF CPU and an NVIDIA GeForce RTX 3060Ti GPU.

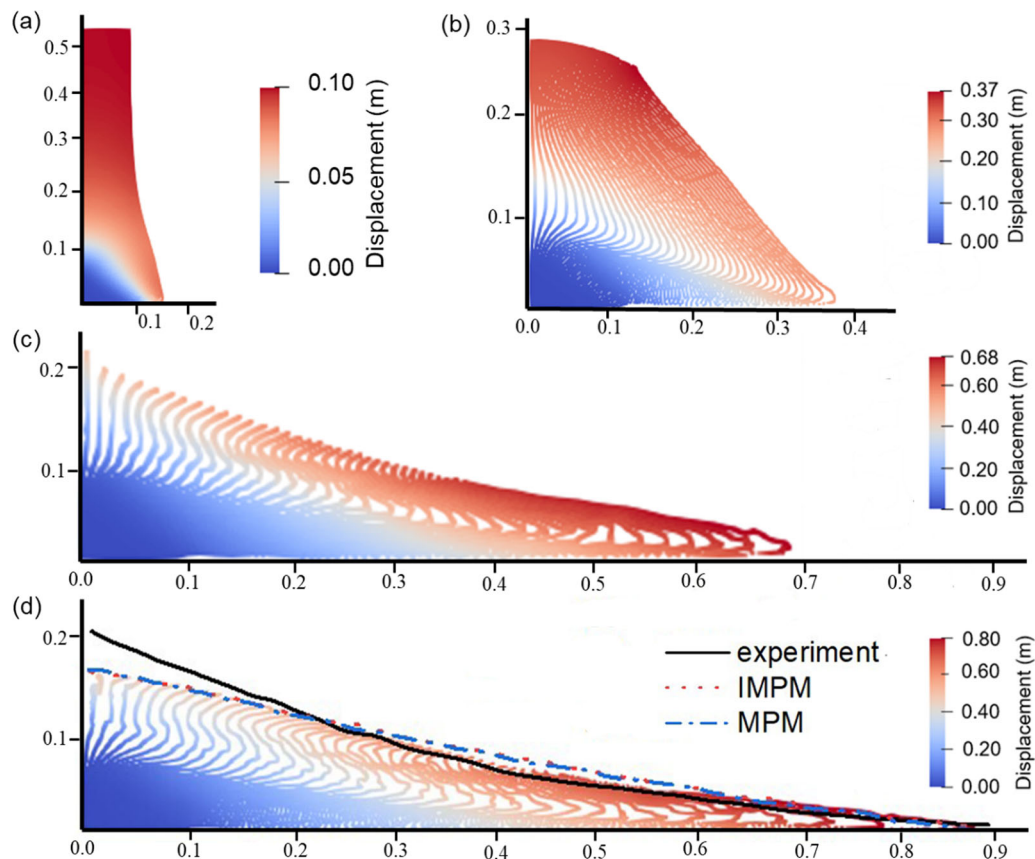


Fig. 12 Collapse column configurations at different times. **a** $t = 0.15$ s; **b** $t = 0.3$ s; **c** $t = 0.6$ s; and **d** $t = 1.2$ s

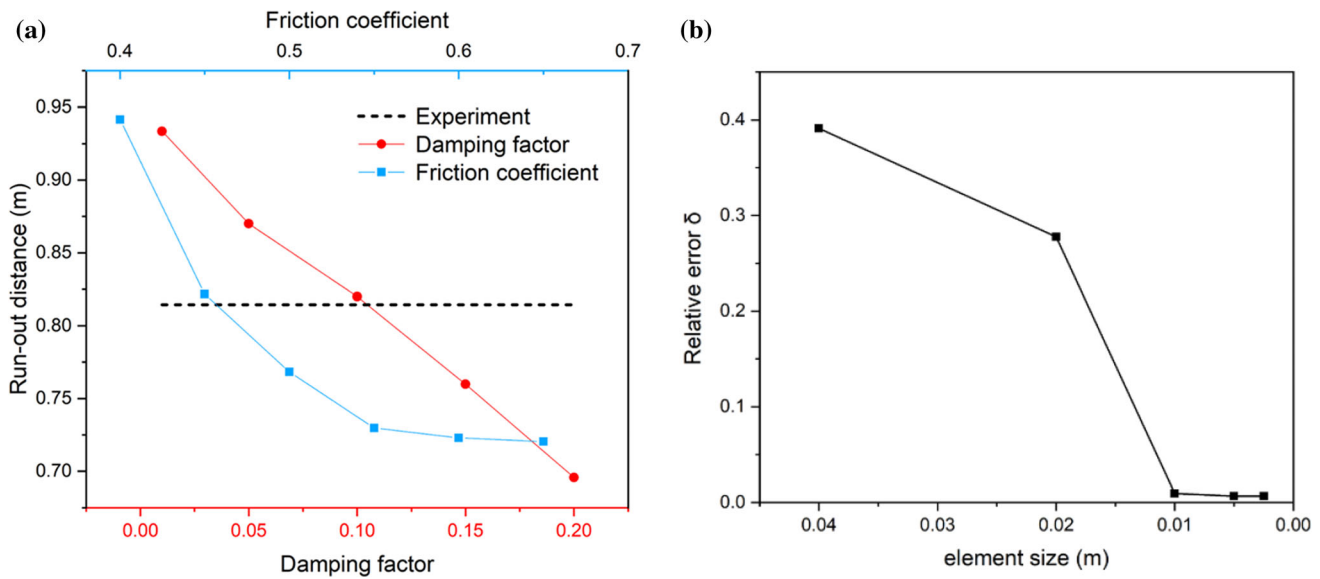


Fig. 13 Parametric analysis. **a** Runout distance of different damping factors and friction coefficients; **b** Grid refinement analysis

Column collapse examples with varying material points are used to analyse the calculation cost of the proposed IMPM. The material parameters are the same as those shown in the model in Sect. 3.3. The model is enlarged by increasing the thickness of the column, where the geometry in height and width remains unchanged. The time of each step in the calculation process using the GPU-accelerated IMPM is counted. Five samples are tested with 36,288, 72,576, 108,864, 145,152, and 181,440 material points, and the corresponding numbers of grids are 21,780, 29,040, 36,300, 43,560 and 50,820, respectively. The total number of steps is 10,000. Meanwhile, to obtain the maximum scale for both solution methods, the scale is continuously scaled up in the same way until it is limited because of computer power.

4.1 Computational cost of the Cholesky decomposition direct solution method

The computational cost of the Cholesky decomposition direct solution method with different simulation scales is shown in Table 3. It is found that the computational time increases nearly linearly with the number of material points. To test the computation efficiency, a parameter η , which is the average computational time per material point in each step, is introduced. From Table 3, it is found that η of the GPU-accelerated implicit MPM is approximately 5×10^{-6} s, while for a GPU-accelerated explicit MPM, η is 1.6×10^{-8} s.

In the IMPM, the solution of the governing equation is the most time-consuming. For further analysis, Fig. 14 shows the calculation cost of step (5) with case 1. With an increasing number of material points, the proportion of the

Table 3 Cost of the different scales of calculation (Case 1)

Model number	Material points	Cholesky solve computational time (s)	GPU memory usage (G)	η (s)
1	36,288	6419.4	0.4	4.53×10^{-6}
2	72,576	11,773.3	1.0	4.41×10^{-6}
3	108,864	18,168.0	1.7	5.19×10^{-6}
4	145,152	25,891.2	2.3	4.88×10^{-6}
5	181,440	31,217.3	3.0	5.08×10^{-6}
6	217,728	38,910.2	3.6	5.23×10^{-6}
7	254,016	45,511.8	4.3	5.15×10^{-6}
8	290,304	52,138.1	5.0	5.34×10^{-6}

calculation cost of each part to the total time cost is basically unchanged. The main computational time is taken in two parts: assembling the CSR form stiffness matrix and solving the displacement. The calculation cost of assembling the CSR stiffness matrix is mainly consumed by the sort and reduce operations of three CSR arrays. Solving the displacement is the inevitable computational cost required to use the CUDA solver.

The Cholesky decomposition direct solution method requires the global stiffness matrix to be assembled, which may occupy a large memory space. Table 3 gives the GPU memory cost in different simulation samples as well. It is found that with an increasing number of material points, the memory cost increases quickly and approaches the maximum memory of the device (8 G). Therefore, the primary limitation on the simulation scale of the Cholesky decomposition direct solution method is the memory usage.

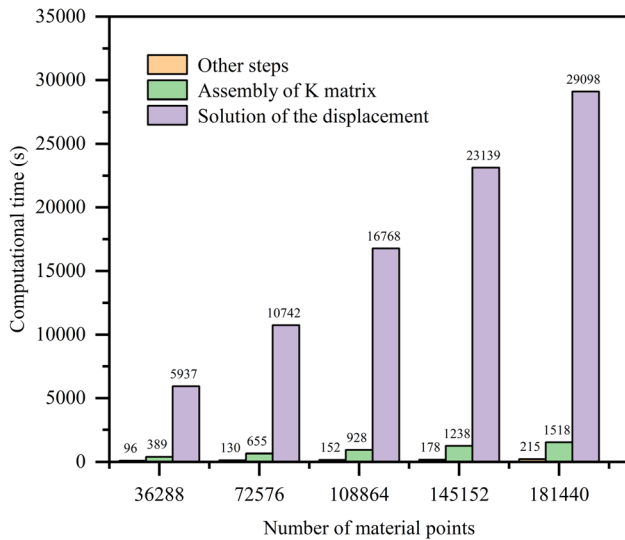


Fig. 14 The statistics of computational cost for case 1

Using this computer, the maximum scale of this method can reach approximately three hundred thousand material points.

4.2 Computational cost of the PCG iterative method

The computational cost of the PCG iterative method is provided in Table 4. Similar to the Cholesky decomposition direct solution method, a linear increase in computational time is observed with a growing number of material points. However, the computational time is significantly lower, and the computational efficiency η is approximately 2×10^{-7} s, which is roughly 10 times larger than that of the explicit MPM. As the IMPM can use a larger time step compared to the explicit MPM, for instance, 100 times

Table 4 Cost of different scales of calculation (case 2)

Model number	Material points	PCG solve calculate time (s)	GPU memory usage (G)	η (s)
1	36,288	786.9	0.7	2.17×10^{-7}
2	72,576	1489.4	0.7	2.05×10^{-7}
3	108,864	2179.8	0.9	2.00×10^{-7}
4	145,152	2791.8	0.9	1.92×10^{-7}
5	181,440	3476.4	1.0	1.91×10^{-7}
6	217,728	4171.4	1.0	1.91×10^{-7}
7	254,016	4838.4	1.1	1.90×10^{-7}
8	290,304	5641.0	1.1	1.94×10^{-7}
9	580,608	14,104.3	1.4	2.40×10^{-7}
10	1,161,216	31,215.7	2.1	2.60×10^{-7}

larger in the self-weight column test in Sect. 3.1, the simulation cost with IMPM can then be significantly reduced compared with the explicit MPM.

Although with higher computational efficiency compared to the Cholesky decomposition direct solution method, the solution of the governing equation still occupies more than 95% of the time during the simulation. Figure 15 shows the computational cost of step (5) with the PCG iterative method. Different from the Cholesky decomposition direct solution method, the computational cost of the PCG iterative method is mainly consumed by the calculation of the element stiffness matrix, while the solving of the governing equation only takes a small proportion. The performance variation between the global stiffness matrix assembly and the PCG iteration may be attribute to the memory access. The memory allocation of the global stiffness is a one-dimensional sequential memory that conforms to the spatial and temporal localization of the hardware cache line. The sequential access pattern leads to loading multiple elements into the cache line (global stiffness matrix) at once, thereby significantly reducing cache misses. By contrast, the scattered memory access used by the PCG iteration may deteriorate the performance due to the inefficient cache or cache pollution. The high memory latency, delayed pipeline and the inefficient hardware prefetching result in the improvement gap between the calculation of element stiffness matrix in PCG iteration method and global stiffness matrix assembly.

The PCG iterative method does not require the assembly of a global stiffness matrix. Therefore, the memory usage of the PCG iterative method is much lower than that of the Cholesky decomposition direct solution method. For instance, in Model 8, the memory usage is 5.0 G for the Cholesky decomposition direct solution method, while the

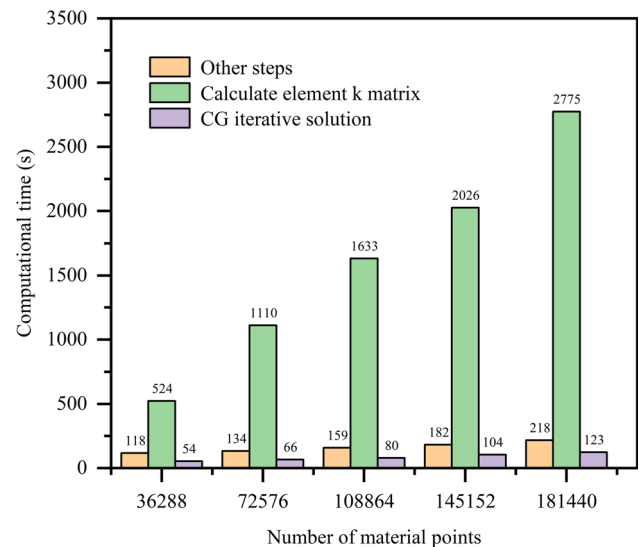


Fig. 15 The statistics of the computational cost for case 2

PCG iterative method only takes 1.1 G. Therefore, it is found that the PCG iterative method shows both higher efficiency and lower memory usage compared to the Cholesky decomposition direct solution method, as well as better performance in parallel computational systems.

5 Xinmo slope failure

The numerical study of practical engineering or natural hazards can be challenging, facing possible obstacles such as large numbers of elements, long-term physical time and low-resolution results. With the application of the GPU-accelerated IMPM, these problems can be relieved, making it possible to simulate large-scale problems with a single PC and satisfactory computational time. In this section, the slope failure and deposition process of the Xinmo landslide are simulated, proving the feasibility of the GPU-accelerated IMPM in large-scale simulation.

On June 24th, 2017, a catastrophic landslide with a volume of $5.08 \times 10^6 \text{ m}^3$ occurred in Xinmo village, Maoxian County, Sichuan Province. According to the description of Fan et al. [16], the cracks in the deep layer of the rock mass are suddenly separated, forming a landslide in the sliding source area. The movement process can be divided into three phases: (1) The rockslide phase: the surface rock mass cracks and slides under the earthquake and gravity. (2) The debris flow phase: the downward-moving rock mass impacts the surface rock along the path and becomes fragmented, forming a high-speed debris flow. (3) The deposit phase: the debris flow reaches Xinmo village near the river and accumulates in the fluvial valley, forming a deposit area.

The landslide body and sliding surface are established according to the digital elevation model before and after the earthquake. The failure area consists of 46,893 material points, representing a total volume of $5.8 \times 10^6 \text{ m}^3$. The background grid size is set as 10 m, generating 7.62 million grid nodes in total. The GPU-accelerated IMPM with a PCG iterative method is applied in this study to acquire a higher simulation efficiency. A comparison of digital elevation model data before and after the landslide shows that the source area of the landslide is briefly divided into two major parts [9]: the collapsed area (area A in Fig. 16) and the entrained area during the landslide (area B in Fig. 16). The velocity-weakening friction law described by Zhao et al. [52] was used between the landslide bed and the collapsed rock mass. The constitutive model is chosen for Drucker–Prager. The relevant parameters are from Zhao et al. [52] and are shown in Table 5.

The simulation is carried out under a gravitational acceleration of 9.8 m/s^2 with a simulation duration of 100 s. The computation of this modelling cost was

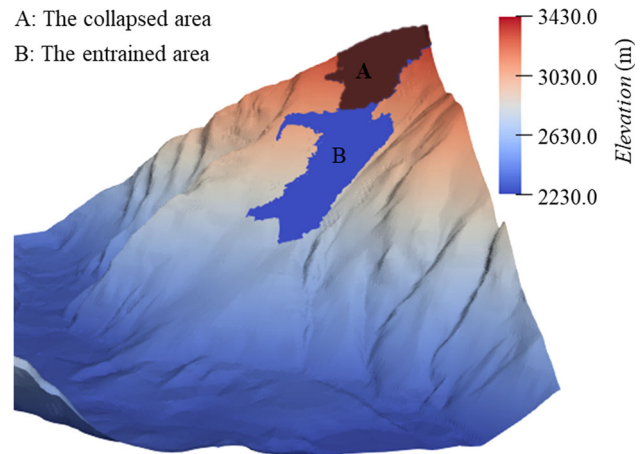


Fig. 16 Xinmo landslide three-dimensional model

approximately 2 h using an NVIDIA GeForce RTX 3060Ti GPU. A similar simulation carried out by Zhao et al. [52] is reported to cost more than 20 h, and the number of elements is half of our model. Therefore, the GPU-accelerated IMPM shows significant improvement in the computational efficiency.

The simulated landslide lasted for 100 s, which is basically consistent with the landslide duration recorded in the practical engineering [35]. The development and sliding processes of the landslide are shown in Fig. 17. The initiation of the landslide body in Fig. 17a can be divided into two parts: the sliding source area and the deformation area. The sliding area is located at the top of the mountain, showing a rapid increase in mass velocity. The lower particles are relatively stable, showing slow movement and deformation. At approximately 40 s, the rock continues to slide with increasing velocity, forming a debris flow (see Fig. 17b). At 70 s, the moving debris flow reaches the riverbed and starts to slow down and deposit. Finally, the

Table 5 Model parameters for the Xinmo slope

	Parameter (unit)	Symbol	Value
Material parameters	Young's modulus (GPa)	E	8.1
	Poisson's ratio (–)	μ	0.18
	Cohesion (kPa)	c	0
	Dilation angle (°)	ϕ	1
	Friction angle (°)	ρ	30
	Density (kg/m^3)	ρ	2200
Contact parameters	Damping factor (–)	α	0.05
	Peak friction coefficient	μ_p	0.7
	Steady-state friction coefficient	μ_s	0.23

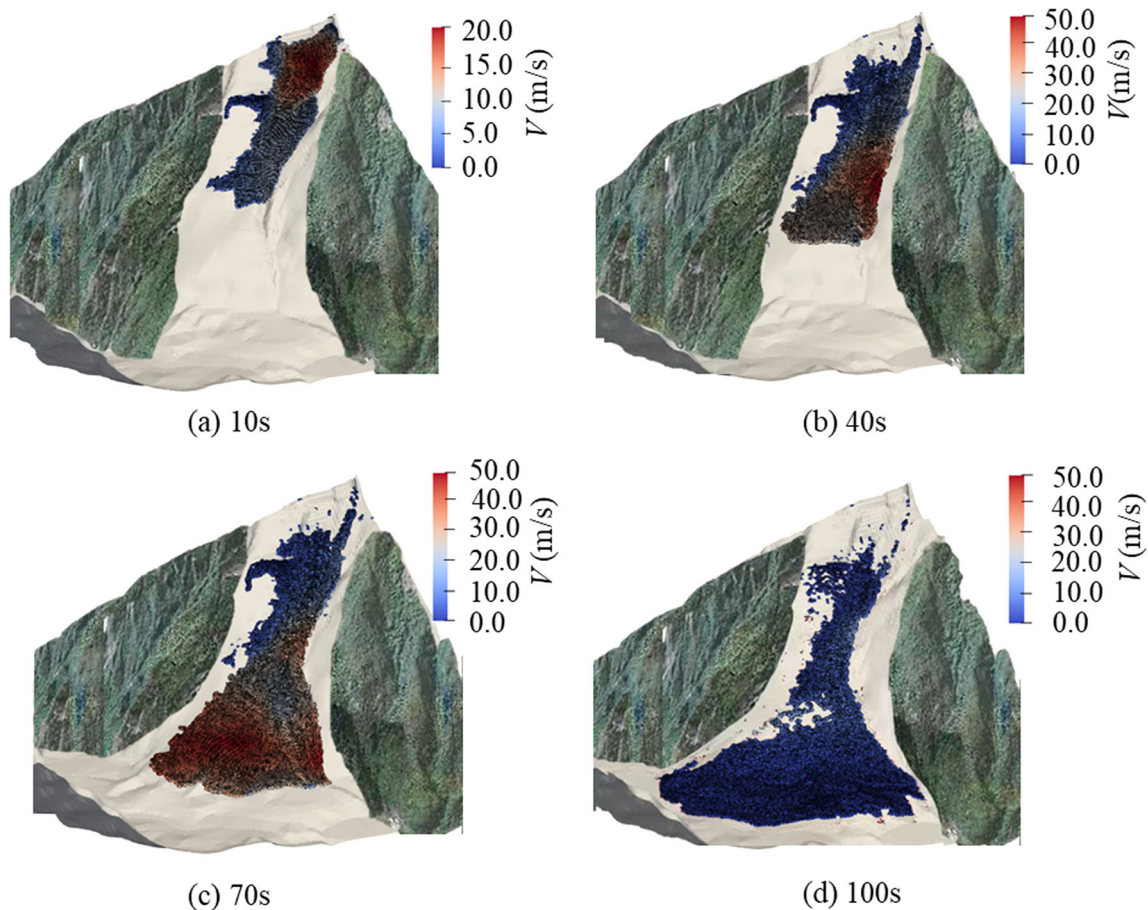


Fig. 17 Movement process of the Xinmo landslide

debris flow stopped moving at approximately 100 s, forming a deposit area in the river valley.

According to Fan et al. [16], the rock mass developed a deep fracture in the early stage, as shown in the red area of Fig. 18a, distinguishing the collapsed and entrained area A and B. Then, collapsed rock mass moves downward, causing the failure of the entrain area B, as shown in Fig. 18b, c. At the time of $t = 40$ s, the total rock mass fails, forming a debris flow as shown in Fig. 18d. Finally, the fragmented rock mass is accumulated at the bottom of the valley.

The final deposit area and thickness of the accumulation with IMPM are presented in Fig. 19a. The results of the IMPM capture the area of deposits in both the riverbed and the sliding surface, which is basically consistent with the actual deposit area. Moreover, the deposit shows a larger thickness around the eastern side and gradually decreases towards the west, which also agrees with the actual deposit thickness distribution shown in Fig. 19b. The larger deposit thickness is mainly caused by a gully at the edge between the slope and riverbed. It is notable that the thickness simulated by IMPM is more concentrated and a bit larger than the actual deposit, which may be attributed to the

limitation of the Drucker–Prager (DP) constitutive model in the simulation of high-speed landslide [52]. The DP constitutive model is a rate-independent constitutive model, which cannot consider the energy loss due to particle collision. Meanwhile, the simple constitutive models choose parameters at residual state value and neglecting the mechanical behaviour of peak state and the influence of stress path, leading to errors in the results. These problems can be solved by adopting a strain-softened Drucker–Prager or more advanced constitutive models, which may better capture the complex stress–strain response of the material and lead to a more accurate results [17, 52]. In Fig. 19c, the GPU-accelerated IMPM is compared with several other numerical simulation results. From the perspective of the accumulation area, it shows the reliability of the simulation results of IMPM, which proves the ability of GPU-accelerated IMPM in large-scale actual landslide simulation. However, there are many other limitations of this simulation that may contribute to the discrepancy. For instance, the presence of water in the place where the deposit ease is not considered, which may lead to a smaller range of deposition. The base of the landslide is set as rigid boundary and the interaction between landslide and

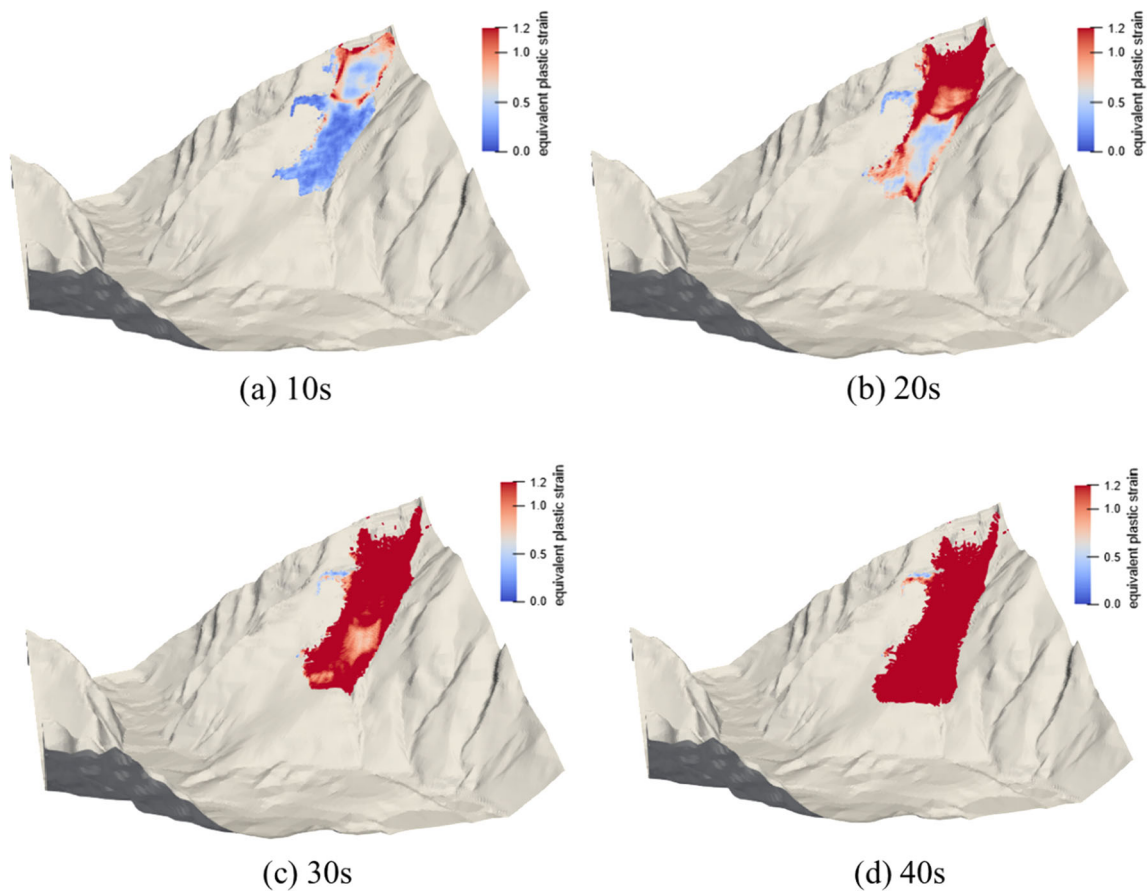


Fig. 18 Failure process of the Xinmo landslide

basement is neglected. As a continuum mechanism, the problem of contact between different objects is very complex, and the simplification of the contact model in numerical simulation may lead to the difference with the practical engineering [52]. Therefore, it is necessary to construct a more refined model after improving the computational efficiency in simulating large-scale actual disasters.

6 Conclusions

This paper develops a GPU-accelerated IMPM for large-scale practical engineering applications. Two different methods, the Cholesky decomposition direct solution method of assembling the global stiffness matrix and the PCG iterative method, are used to solve the linear equations. The risk of the data race occurring in the IMPM process is also studied and avoided by adopting atomic operations. For the frequent reassembly of stiffness matrix, we also constructed an efficient search technique. Statistically, the time to assemble the stiffness matrix is only about 5% for the constructed code. Therefore, the proposed

IMPM framework takes advantage of the parallel acceleration in GPU and avoids all the data transfers or sequential processes in the CPU domain, showing high performance in computation.

In the study, one-dimensional compression is used to compare the efficiency of explicit MPM and IMPM. It is found that to achieve the same accuracy, the IMPM solved by the PCG iterative method is about 4 times faster than the explicit MPM. It is found in the IMPM that assembling the stiffness matrix and solving the linear equations take more than 95% of the total computation time. The computational efficiency η is defined as the average computational time cost by each material point in each step. It is found that IMPM shows a large η compared to explicit MPM, showing a ratio of 300 and 10 for the Cholesky decomposition direct solution method and the PCG iterative method, respectively. However, this does not mean the IMPM is low efficiency when solving a practical engineering problem because the time step in the IMPM can be much larger (generally ten times or larger) than the explicit MPM, which may lead to a shorter computational time in the simulation. By modelling the Xinmo landslide, the

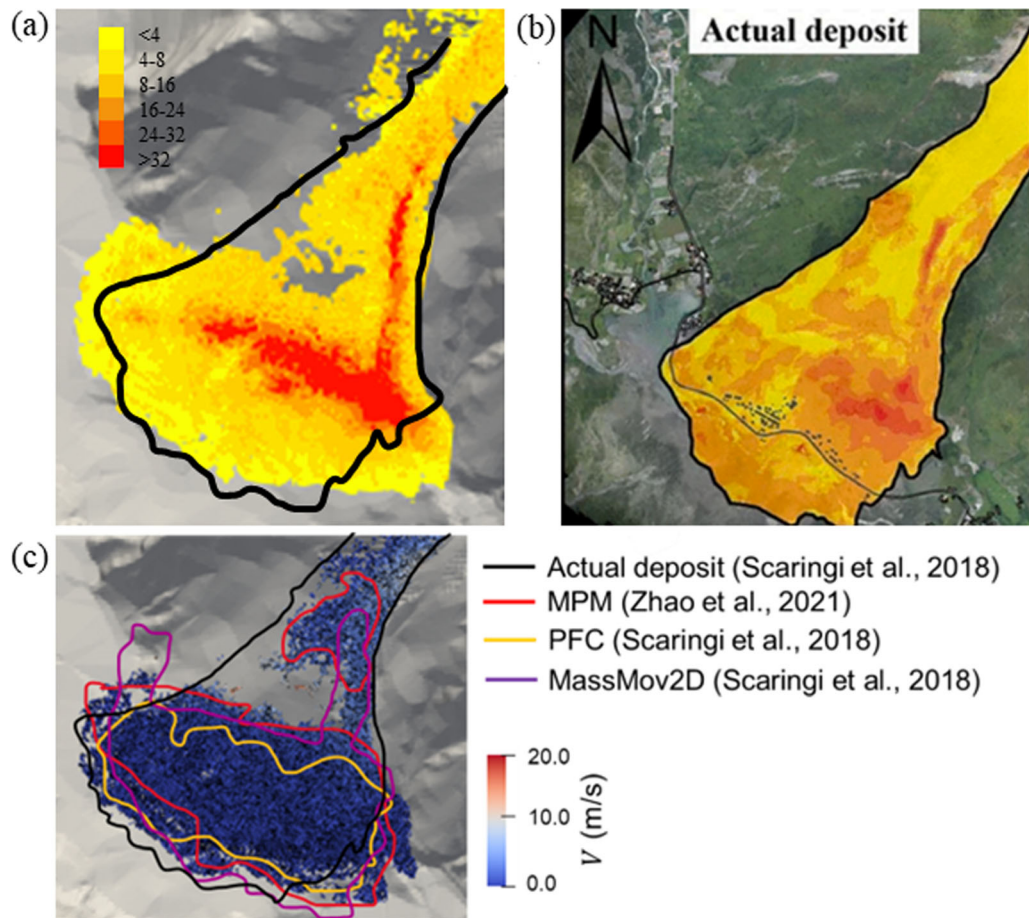


Fig. 19 Final position of the deposit area of the Xinmo landslide. **a** Thickness map of IMPM simulation; **b** Thickness map of actual deposit [32]; **c** The final accumulation area of different numerical methods [32, 52]

prospect of GPU-accelerated IMPM in practical engineering applications is demonstrated.

Despite the improvement of the GPU-accelerated IMPM, there are still many obstacles in the application of IMPM in engineering. The memory storage is one of the reasons to limit the growth of GPU-accelerated IMPM computing scale. We can introduce the data storage formula of sparsely populated grid to improve memory storage and break the limit of computing scale [27]. When simulating real engineering problems, the mechanical behaviour of geotechnical bodies is very complex, and more advanced constitutive models should be further used to approximate the engineering, such as the low-plasticity soil model [18].

Appendix A: Nomenclature

Nomenclature

Abbreviations

1-D	One dimension
MPM	Material points method
IMPMM	Implicit material points method
FEM	Finite element method
PIC	Particle-in-cell
GPU	Graphics processing unit
PCG	Preconditioned conjugate gradient
SPH	Smooth particle hydrodynamics
CPU	Central processing unit
SPFEM	Smoothed particle finite element method
CSR	Compressed sparse row
P2G	The material points to the grid nodes
G2P	The grid nodes back to the material points
PC	Personal computer
CUDA	Computer unified device architecture

Nomenclature

Case 1	The Cholesky decomposition direct solution method
Case 2	The PCG iterative method
dt	Time step
DP	Drucker–Prager
PFC	Particle flow code
GIMP	Generalized interpolation material point
GMRES	Generalized min residual
<i>Symbol</i>	
k	Load parameter
w	Tip displacement
L	Initial length
I	The moment of inertia
η	The average computational time per material point in each step
α	Damping factor
H	Initial height of the column
u	The displacement increment
F	The resultant force
M	The lumped mass matrix
u_e	The incremental displacement obtained for each particle
η	The average computational time cost by each material point in each step
E	Young's modulus
μ	Poisson's ratio
c	Cohesion
φ	Friction angle
ϕ	Dilation angle
ρ	Density
μ_0	Friction coefficient
μ_p	Peak friction coefficient
μ_s	Steady-state friction coefficient

Acknowledgements The authors wish to acknowledge the National Nature Science Foundation of China (Grant Nos. 51979270 and 51709258) and the CAS Pioneer Hundred Talents Program for their financial supports.

Data availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

- Bandara SS (2013) Material point method to simulate large deformation problems in fluid-saturated granular medium. Doctoral dissertation, University of Cambridge
- Bardenhagen SG, Kober EM (2004) The generalized interpolation material point method. *Comput Model Eng Sci* 5(6):477–496
- Bardenhagen SG, Brackbill JU, Sulsky D (2000) The material-point method for granular materials. *Comput Methods Appl Mech Eng* 187(3–4):529–541
- Bathe KJ, Bolourchi S (1979) Large displacement analysis of three-dimensional beam structures. *Int J Numer Methods Eng* 14(7):961–986
- Bell N, Hoberock J (2012) Thrust: a productivity-oriented library for CUDA. In: GPU computing gems Jade edition. Morgan Kaufmann, pp 359–371
- Beuth L, Benz T, Vermeer PA, Wieckowski Z (2008) Large deformation analysis using a quasi-static material point method. *J Theor Appl Mech* 38(1–2):45–60
- Beuth L, Więckowski Z, Vermeer PA (2011) Solution of quasi-static large-strain problems by the material point method. *Int J Numer Anal Methods Geomech* 35(13):1451–1465
- Chandra B, Singer V, Teschemacher T, Wuechner R, Laesele A (2021) Nonconforming Dirichlet boundary conditions in implicit material point method by means of penalty augmentation. *Acta Geotech* 16(8):2315–2335
- Chen KT, Wu JH (2018) Simulating the failure process of the Xinmo landslide using discontinuous deformation analysis. *Eng Geol* 239:269–281
- Chiang WF, DeLisi M, Hummel T, Prete T, Tew K, Hall M et al (2010) GPU acceleration of the generalized interpolation material point method
- Coombs WM, Augarde CE, Brennan AJ, Brown MJ, Charlton TJ, Knappett JA et al (2020) On Lagrangian mechanics and the implicit material point method for large deformation elastoplasticity. *Comput Methods Appl Mech Eng* 358:1126
- Cundall PA (1987) Distinct element models, of rock and soil structure. Analytical and computational method in engineering rock mechanics, pp 129–163I
- Dang HV, Schmidt B (2013) CUDA-enabled sparse matrix-vector multiplication on GPUs using atomic operations. *Parallel Comput* 39(11):737–750
- Dong Y, Grabe J (2018) Large scale parallelisation of the material point method with multiple GPUs. *Comput Geotech* 101:149–158
- Dong Y, Wang D, Randolph MF (2015) A GPU parallel computing strategy for the material point method. *Comput Geotech* 66:31–38
- Fan X, Xu Q, Scaringi G, Dai L, Li W, Dong X et al (2017) Failure mechanism and kinematics of the deadly June 24th 2017 Xinmo landslide, Maoxian, Sichuan, China. *Landslides* 14:2129–2146
- Fern EJ, Soga K (2016) The role of constitutive models in MPM simulations of granular column collapses. *Acta Geotech* 11(3):659–678
- Giridharan S, Gowda S, Stolle DF, Moormann C (2020) Comparison of UBCSAND and hypoplastic soil model predictions using the material point method. *Soils Found* 60(4):989–1000
- Guilkey JE, Weiss JA (2003) Implicit time integration for the material point method: quantitative and algorithmic comparisons with the finite element method. *Int J Numer Methods Eng* 57(9):1323–1338
- Henderson TC, McMurtry PA, Smith PJ, Voth GA, Wight CA, Pershing DW (2000) Simulating accidental fires and explosions. *Comput Sci Eng* 2(2):64–76
- Hu Y, Fang Y, Ge Z, Qu Z, Zhu Y, Pradhana A, Jiang C (2018) A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans Graph (TOG)* 37(4):1–14
- Hu Y, Li TM, Anderson L, Ragan-Kelley J, Durand F (2019) Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Trans Graph (TOG)* 38(6):1–16

23. Huang P, Zhang X, Ma S, Huang X (2011) Contact algorithms for the material point method in impact and penetration simulation. *Int J Numer Methods Eng* 85(4):498–517
24. Jiang C, Schroeder C, Selle A, Teran J, Stomakhin A (2015) The affine particle-in-cell method. *ACM Trans Graph (TOG)* 34(4):1–10
25. Lemiale V, Nairn J, Hurman A (2010) Material point method simulation of equal channel angular pressing involving large plastic strain and contact through sharp corners. *Comput Model Eng Sci* 70(1):41
26. Li J, Wang B, Jiang Q, He B, Zhang X, Vardon PJ (2021) Development of an adaptive CTM–RPIM method for modeling large deformation problems in geotechnical engineering. *Acta Geotechnica* 1–19
27. Liu H, Hu Y, Zhu B, Matusik W, Sifakis E (2018) Narrow-band topology optimization on a sparsely populated grid. *ACM Trans Graph (TOG)* 37(6):1–14
28. Liu Y, Yin K, Wu E (2011) Fast GMRES-GPU solver for large scale sparse linear systems
29. Lube G, Huppert HE, Sparks RSJ, Hallworth MA (2004) Axisymmetric collapses of granular columns. *J Fluid Mech* 508:175–199
30. Ma S, Zhang X, Lian Y, Zhou X (2009) Simulation of high explosive explosion using adaptive material point method. *Comput Model Eng Sci (CMES)* 39(2):101
31. NVIDIA cuSOLVER, Direct Linear Solvers on NVIDIA GPUs, 2023, Available at <https://developer.nvidia.com/cusolver>
32. Scaringi G, Fan X, Xu Q, Liu C, Ouyang C, Domènech G et al (2018) Some considerations on the use of numerical methods to simulate past landslides and possible new failures: the case of the recent Xinmo landslide (Sichuan, China). *Landslides* 15:1359–1375
33. Shin WK (2009) Numerical simulation of landslides and debris flows using an enhanced material point method. University of Washington
34. Sołowski WT, Sloan SW (2015) Evaluation of material point method for use in geotechnics. *Int J Numer Anal Methods Geomech* 39(7):685–701
35. Su LJ, Hu KH, Zhang WF, Wang J, Lei Y, Zhang CL et al (2017) Characteristics and triggering mechanism of Xinmo landslide on 24 June 2017 in Sichuan, China. *J Mt Sci* 14(9):1689–1700
36. Sugai R, Han J, Yamaguchi Y, Moriguchi S, Terada K (2023) Extended B-spline-based implicit material point method enhanced by F-bar projection method to suppress pressure oscillation. *Int J Numer Methods Eng*
37. Sulsky D, Kaul A (2004) Implicit dynamics in the material-point method. *Comput Methods Appl Mech Eng* 193(12–14):1137–1170
38. Sulsky D, Schreyer HL (1996) Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems. *Comput Methods Appl Mech Eng* 139(1–4):409–429
39. Sulsky D, Chen Z, Schreyer HL (1994) A particle method for history-dependent materials. *Comput Methods Appl Mech Eng* 118(1–2):179–196
40. Sulsky D, Zhou SJ, Schreyer HL (1995) Application of a particle-in-cell method to solid mechanics. *Comput Phys Commun* 87(1–2):236–252
41. Wang B, Vardon PJ, Hicks MA (2013) Implementation of a quasi-static material point method for geotechnical applications. In: *Computational geomechanics, COMGEO III*, Krakow, Poland. IC2E-International centre for computational engineering, pp 305–313
42. Wang B, Vardon PJ, Hicks MA (2016) Investigation of retrogressive and progressive slope failure mechanisms using the material point method. *Comput Geotech* 78:88–98
43. Wang B, Vardon PJ, Hicks MA, Chen Z (2016) Development of an implicit material point method for geotechnical applications. *Comput Geotech* 71:159–167
44. Więckowski Z, Youn SK, Yeon JH (1999) A particle-in-cell solution to the silo discharging problem. *Int J Numer Meth Eng* 45(9):1203–1225
45. Xia X, Liang Q (2016) A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations. *Environ Model Softw* 75:28–43
46. Yuan WH, Wang B, Zhang W, Jiang Q, Feng XT (2019) Development of an explicit smoothed particle finite element method for geotechnical applications. *Comput Geotech* 106:42–51
47. Yuan WH, Wang HC, Liu K, Zhang W, Wang D, Wang Y (2021) Analysis of large deformation geotechnical problems using implicit generalized interpolation material point method. *J Zhejiang Univ Sci A* 22(11):909–923
48. Yuan WH, Wang HC, Zhang W, Dai BB, Liu K, Wang Y (2021) Particle finite element method implementation for large deformation analysis using Abaqus. *Acta Geotechnica* 1–14
49. Zhang X, Chen Z, Liu Y (2016) *The material point method: a continuum-based particle method for extreme loading cases*. Academic Press
50. Zhang W, Yuan W, Dai B (2018) Smoothed particle finite-element method for large-deformation problems in geomechanics. *Int J Geomech* 18(4):04018010
51. Zhang W, Zhong ZH, Peng C, Yuan WH, Wu W (2021) GPU-accelerated smoothed particle finite element method for large deformation analysis in geomechanics. *Comput Geotech* 129:103856
52. Zhao S, He S, Li X, Deng Y, Liu Y, Yan S, Xie Y (2021) The Xinmo rockslide-debris avalanche: an analysis based on the three-dimensional material point method. *Eng Geol* 287:1061

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.