This is a preprint of the paper published in Computers and Geotechnics
Accepted citation: Zhang, W., Zhong, Z.H., Peng, C., Yuan, W.H. and Wu, W., 2021. GPU-accelerated smoothed particle finite element method for large deformation analysis in geomechanics. Computers and Geotechnics, 129, p.103856.
DOI: https://doi.org/10.1016/j.compgeo.2020.103856

GPU-accelerated smoothed particle finite element method for large deformation analysis in geomechanics

Wei Zhang[a,b], Zhi-hao Zhong[a], Chong Peng[b], Wei-hai Yuan[c], Wei Wu[b,*]

[a] *College of Water Conservancy and Civil Engineering, South China Agricultural University, Guangzhou 510642, China*
[b] *Institut für Geotechnik, Universität für Bodenkultur, Feistmantelstrasse 4, Vienna 1180, Austria*
[c] *College of Mechanics and Materials, Hohai University, Nanjing 210098, China*

Abstract

Particle finite element method (PFEM) is an effective numerical tool for solving large-deformation problems in geomechanics. By incorporating the node integration technique with strain smoothing into the PFEM, we proposed the smoothed particle finite element method (SPFEM). This paper extends the SPFEM to three-dimensional cases and presents a SPFEM executed on graphics processing units (GPUs) to boost the computational efficiency. The detailed parallel computing strategy on GPU is introduced. New computation formulations related to the strain smoothing technique are proposed to save memory space in the GPU parallel computing. Several benchmark problems are solved to validate the proposed approach and to evaluate the GPU acceleration performance. Numerical examples show that with the new formulations not only the memory space can be saved but also the computational efficiency is improved. The computational cost is reduced by $\sim 70\%$ for the double-precision GPU parallel computing with the new formulations.

*Corresponding author. E-mail: wei.wu@boku.ac.at

Compared with the sequential CPU simulation, the GPU-accelerated simulation results in a significant speedup. The overall speedup ranges from 8.21 to 11.17 for double-precision simulations. Furthermore, the capability of the GPU-accelerated SPFEM in solving large-scale complicated problems is demonstrated by modelling the progressive failure of a long slope with strain-softening soil.
Keywords: Particle finite element method; Parallel computing; GPU; Speedup; Geomechanics; Large deformation

1. Introduction

Many problems in geotechnical engineering involve large deformation, e.g., sampling, in-situ penetration testing, pile installation, landslides and debris flow, etc. Large-deformation numerical simulations can provide a crucial perspective for these problems. However, they often pose some difficulties for conventional numerical method, e.g. the finite element method (FEM). The mesh distortion associated with large deformation may lead to reduction of accuracy or even termination of the numerical calculations.

Many numerical frameworks have been presented to solve large-deformation problems in geomechanics so far, such as Arbitrary Lagrangian-Eulerian (ALE) method [1], remeshing and interpolation technique with small strain (RITSS) method [2], Smoothed particle hydrodynamics (SPH) [3], material point method (MPM) [4], particle finite element method (PFEM) [5], etc. Among them, the PFEM attracts more and more attentions in recent years. It inherits the solid theoretical foundation of FEM which can guarantee the accuracy and convergence, whilst it possesses the flexibility of

mesh-free methods for large deformation analysis. In the PFEM, the nodes in traditional FEM are treated as Lagrangian particles. The Delaunay triangulation technique is used to re-generate the mesh when it is too distorted due to large deformation. Therefore, the PFEM is actually an updated Lagrangian FEM approach with remeshing technique to overcome the mesh distortion problem. In the field of geomechanics, the PFEM has been developed and applied to solve many different types of problems, including the ground excavation process [6, 7], granular flow [8–10], landslide and landslide-generated wave [11–13], soil penetration problem [14, 15], hydro-mechanical coupled problem [15–17], and progressive slope failure [18–20].

In the original PFEM, the numerical integration is carried out at Gasuss points while the field information is stored at nodes/particles, As a result, during the numerical computation, it is required to transfer information between Gasuss points and nodes/particles frequently, which inevitably introduces errors and increases complexity. In view of this, the authors proposed the smoothed particle finite element method (SPFEM) recently [21, 22]. In the SPFEM, a strain smoothing technique for node integration is incorporated. Thanks to this node integration technique, all the field variables are computed and stored at nodes/particles, making the method more like a Lagrangian particle-based method. There are also some extra advantages, such as utilization of low order elements without volumetric locking and insensitivity to mesh distortion. The former can assure the computational efficiency while the latter is especially beneficial for large deformation analysis. Very recently, the SPFEM has been extended to solve the fluid dynamics problem [23] and fluid–structure interaction problem [24], which has been termed

'PFEM with Nodal Integration' in their studies [25].

In recent years, using graphic processing units (GPUs) to achieve high-performance computing has gained popularity rapidly. GPUs were initially designed for three-dimensional image rendering that is not complex but highly computationally intensive. In 2007, NVIDIA released the compute unified device architecture (CUDA) programming platform which is designed for general-purpose computing. Since then, GPUs have been successfully used in many scientific fields for high-performance computing (e.g. [26–29]). GPUs, which feature much more cores, lower thread-scheduling cost and higher memory bandwidth than ordinary CPUs, are capable of executing computations with thousands of independent threads, operating in Single-Instruction-Multiple-Data (SIMD) mode. Here, SIMD is a terminology used to describe a class of parallel computers with multiple processing elements that perform the same operation on multiple data points simultaneously. In the computational process of PFEM/SPFEM, there are many procedures in which the same calculations need to be performed for all elements or nodes/particles. Meanwhile, data dependence is local and usually only extends to a few surrounding elements or nodes. Therefore, most calculations can be distributed into independent GPU threads and executed in parallel. Compared with the PFEM, the computation process of SPFEM is much simplified due to the adoption of node integration scheme. The computation process of the explicit SPFEM [22] is especially simple because the accumulation of global stiffness matrix in the implicit SPFEM is avoided. These simplifications can undoubtedly greatly facilitate GPU parallel computing. However, to the best of the authors' knowledge, GPU parallel computing for

PFEM has not been reported yet.

This study aims to develop a GPU-accelerated SPFEM for large deformation analysis in geomechanics based on CUDA, with focus on the explicit version of SPFEM [22]. The explicit SPFEM is first extended to three-dimensional cases. Then the GPU parallel computing strategy is presented after a brief view on the basic theory of SPFEM. To reduce the memory consumption during the GPU parallel computing, which is crucial for increasing the scale of problems analysed, new computation formulations related to the strain smoothing technique are used. Several benchmark examples are solved with sequential CPU simulations and GPU-accelerated simulations respectively. The speedup of the presented GPU-accelerated SPFEM is carefully evaluated.

2. Smoothed particle finite element method

2.1. Strain smoothing technique for node integration

In PFEM, the problem domain is discretized into a set of Lagrangian particles, which carry mass, velocity and other state variables (e.g. stress, hardening parameters related to constitutive models). The FEM computation mesh is generated based on this set of Lagrangian particles using the Delaunay triangulation technique combined with the alpha shape algorithm. Thus, nodes in traditional FEM correspond to particles in PFEM. The terms 'node' and 'particle' can be used interchangeably in PFEM.

In SPFEM, the strain smoothing technique for node integration is incorporated. The first step of this technique is to construct smoothing cells associated with particles. In two-dimensional cases, the smoothing cell as-

sociated with particle k is created by connecting sequentially the mid-edge points to the centroid points of the surrounding triangular elements of the particle. This approach can be easily extended to three-dimensional cases, as shown in Fig. 1. Note that it is not required to explicitly calculate the specific geometries during the computation process, which will be shown in the following.

With this approach, the whole problem domain is discretized into finite strain smoothing cells associated with particles. Thus, all the integration over the whole problem domain can be performed by accumulating over the smoothing cells associated with particles. Furthermore, all the field variables can be calculated in these smoothing cells. In SPFEM, each particle represents a smoothing cell associated with it, which makes the SPFEM more like a particle-based method.

In the smoothed strain technique, the volume of the smoothing cell associated with node k is calculated by

$$V_k = \sum_{i=1}^{E_k} \frac{1}{4} V^i \tag{1}$$

where E_k is the number of tetrahedral elements related to node k and V^i is the volume of the ith element. The smoothed strain matrix is calculated by

$$\tilde{\boldsymbol{B}}_k = \frac{1}{V_k} \sum_{i=1}^{E_k} \frac{1}{4} V^i \boldsymbol{B}^i \tag{2}$$

where $\boldsymbol{B}^i$ is the strain gradient matrix used in the standard FEM for the ith element. It can be seen that the smoothed strain is actually the average of

the strains of the elements related to node k weighted by volume. For more details about the strain smoothing technique, readers are suggested to refer to [31, 32].

2.2. Computational formulations of SPFEM

Similar to the FEM, there are two numerical solution strategies of SPFEM with different time integration schemes: the implicit SPFEM [21] and the explicit SPFEM [22]. In this study, the explicit SPFEM is considered, which has simpler formulations and thus is more suitable for GPU parallel computing.

In the computation domain, the motion of a continuum can be described as

$$\rho \boldsymbol{a} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} \tag{3}$$

where, ρ is the material density, $\boldsymbol{a}$ is the acceleration, $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\boldsymbol{b}$ is the specific body force density. Considering the principle of virtual displacement and the divergence theorem, the weak form is expressed as

$$\int_{\Omega} \delta \boldsymbol{u} \cdot \rho \boldsymbol{a} d\Omega = \int_{S} \delta \boldsymbol{u} \cdot \boldsymbol{\tau}_S dS + \int_{\Omega} \delta \boldsymbol{u} \cdot \rho \boldsymbol{b} d\Omega - \int_{\Omega} \delta \boldsymbol{u} : \boldsymbol{\sigma} d\Omega \tag{4}$$

where $\boldsymbol{u}$ is the displacement vector, Ω represents the configuration domain, S represents the boundary, and $\boldsymbol{\tau}_S$ is the prescribed traction. After the node-based discretization, the above equation reads

$$\sum_{k=1}^{T} \rho \boldsymbol{a}_k V_k = \sum_{k=1}^{T} \int_{S} \boldsymbol{N}_k \boldsymbol{\tau}_S dS + \sum_{k=1}^{T} \rho \boldsymbol{b}_k V_k - \sum_{k=1}^{T} \tilde{\boldsymbol{B}}_k^T \boldsymbol{\sigma}_k V_k \tag{5}$$

where T is the total number of nodes in the computation domain. The above

Algorithm 1 Computational cycle of SPFEM

1. Generate mesh using Delaunay triangulation and alpha shape technique
2. Get basic data of elements and nodes
3. Calculate smoothed strains of nodes
4. Update stresses of nodes through constitutive integration
5. Calculate internal forces of nodes
6. Update velocities and positions of nodes

discrete form can be then written in the vector or matrix form as

$$\boldsymbol{M}\boldsymbol{a} = \boldsymbol{F}^{ext} - \boldsymbol{F}^{int} \tag{6}$$

in which

$$\boldsymbol{F}^{ext} = \sum_{k=1}^{T} \int_{S} \boldsymbol{N}_k \boldsymbol{\tau}_S dS + \sum_{k=1}^{T} \rho \boldsymbol{b}_k V_k \tag{7}$$

$$\boldsymbol{F}^{int} = \sum_{k=1}^{T} \tilde{\boldsymbol{B}}_k^T \boldsymbol{\sigma}_k V_k \tag{8}$$

$$\boldsymbol{M} = \sum_{k=1}^{T} \rho V_k \tag{9}$$

where $\boldsymbol{F}^{ext}$ and $\boldsymbol{F}^{int}$ are termed as the external and internal forces, respectively, and $\boldsymbol{M}$ is the diagonal mass matrix.

A typical computational cycle of SPFEM is shown in Algorithm 1. More details are available in Ref. [21, 22].

2.3. Adaptive critical time step size

In utilizing the explicit time integration scheme, the incremental time step size should satisfy the Courant stability restriction to ensure numerical

stability. The incremental time step size is generally calculated by

$$\Delta t_{cr} = \alpha \frac{l_{min}}{c} \tag{10}$$

where Δt_{cr} is the critical time step size, l_{min} is the characteristic length of the mesh, c is the sound speed in the material and α is a scale factor. As the characteristic length of the mesh varies during the computation, an adaptive time step size is realized.

The determination of time step sizes greatly influences the computation cost of a numerical method using explicit time integration. In the original PFEM, the determination of time step sizes is the same as the standard FEM. For example, the characteristic length of the mesh in Eq. (10) can be taken to be the minimum radius of spheres inscribed in the tetrahedral elements [33]. As stated in [34], the Delaunay triangulation and the triangulation satisfying the max-min angle criterion are identical in two-dimensional cases, while in three-dimensional cases they are no longer identical in general and the Delaunay triangulation in three-dimensional cases does not seem to satisfy any optimal angle condition. Actually, many so-called silver elements may occur after Delaunay triangulation [35]. As a result, with the criterion of the minimum radius of spheres inscribed in the tetrahedrons, the time step size may reduce by several orders of magnitude during the computation [33].

In this study, as the node integration technique of strain smoothing is adopted, the integration domain for each node is the smoothing cell associated with this node. The smoothing cells generally have a more regular shape (See Fig. 1) and their volumes are generally much larger than those of the silver elements, because their volumes are the summation of the volumes

of elements related to the nodes divided by four (See Eq. (1)). The characteristic length of a smoothing cell can be obtained in analogy to an element in the standard FEM. In this study, one half of the minimum distance between the node and its related nodes is used. This value is generally much larger than the value determined by the criterion of the minimum radius of spheres inscribed in the tetrahedrons, and thus the computational efficiency can be greatly increased as a result. This can be considered as a newly discovered advantage of incorporating the strain smoothing technique for node integration into an explicit PFEM.

2.4. Contact strategies in the explicit SPFEM

In the explicit SPFEM, as the solution strategy is similar to that of the explicit FEM, the contact algorithm in the explicit FEM can be easily incorporated. Specifically, the slave node to master segment contact scheme is utilized. A simple illustration is shown in Fig. 2. The frictional contact condition can be written as:

$$\begin{aligned} g_n \geq 0, \lambda_n \geq 0, \lambda_n g_n = 0 \\ |\lambda_t| - \mu_f \lambda_n \leq 0 \end{aligned} \tag{11}$$

where g_n is the gap between node and segment, λ_n is the normal contact force which is positive corresponding to compression, λ_t is the tangential contact force, and μ_f is the friction coefficient.

The penalty method is utilized to fulfil the frictional contact condition. For the explicit time integration scheme, when a contact pair is detected, i.e.

$g_n < 0$, the contact force can be calculated as

$$\begin{aligned} \mathbf{R}_{cs} &= \boldsymbol{\lambda}, \ for\ slave\ node \\ \mathbf{R}_{cm,i} &= -N_i\boldsymbol{\lambda}, \ for\ master\ nodes \end{aligned} \tag{12}$$

where $\boldsymbol{\lambda}$ is the vector of contact force, i is the node number of contact segment and N_i is the corresponding shape function of contact segment. For the vector of contact force $\boldsymbol{\lambda}$, the normal part λ_n is obtained as $\lambda_n = -\epsilon g_n$ while the tangential part λ_t is obtained as $\lambda_t = min(-\epsilon g_t, \mu\lambda_n)$ when the Coulomb friction model is used. The above contact forces are added into $\boldsymbol{F}_{ext}$ in Equation (6) and then the accelerations, velocities and positions of nodes can be updated correspondingly.

In case that the node to rigid wall contact is considered, the numerical implementation can be further simplified. When the penetration between node k and a rigid wall occurs, i.e. $g_n < 0$, the acceleration and velocity of node k can be modified as follows to make the normal penetration g_n to be zero,

$$\begin{aligned} \mathbf{a}_{new} &= \mathbf{a}_{old} - (\mathbf{a}_{old} \cdot \mathbf{n})\mathbf{n} \\ \mathbf{v}_{new} &= \mathbf{v}_{old} - (\mathbf{v}_{old} \cdot \mathbf{n})\mathbf{n} \end{aligned} \tag{13}$$

where $\boldsymbol{n}$ is the unit normal vector of the rigid wall. When the Coulomb friction model is used, the tangential contact force can be obtained as

$$\lambda_t = min(m_k\sqrt{\mathbf{v}_{new} \cdot \mathbf{v}_{new}}/\Delta t, \mu\lambda_n) \tag{14}$$

in which m_k is the mass of node k, Δt is the time increment, and the normal

contact force λ_n can be calculated by

$$\lambda_n = m_k \sqrt{\mathbf{v}_{old}^n \cdot \mathbf{v}_{old}^n} / \Delta t \tag{15}$$

where $\mathbf{v}_{old}^n$ is the normal velocity of node k before modification.

2.5. Mesh rebuilding technique in the explicit SPFEM

An important part of SPFEM is the mesh rebuilding technique to avoid mesh distortion when large deformations occur in the continuum domain. The mesh rebuilding technique in the explicit SPFEM is basically the same with that in the original PFEM [5], which is generally based on the Delaunay triangulation technique combined with alpha shape algorithm.

When the mesh rebuilding is required, the Delaunay triangulation technique, which is widely used in FEM mesh generation [36], is first performed to rebuild the element connectivity on the basis of a cloud of nodes. Another task of mesh rebuilding is the identification of computational domain. There is generally no unique solution to this problem. The solution originally proposed by Idelsohn [37] and subsequently adopted as a standard feature of the PFEM is to use the alpha-shape method previously developed by Edelsbrunner [38] for computer graphics applications. The basic principle of the alpha-shape method is that for a cloud of points with a characteristic spacing h and a predefined value of parameter α, all nodes on an empty sphere with a radius greater than αh are considered as boundary nodes.

When the alpha-shape method is combined with the Delaunay triangulation technique, a simple two-step algorithm for mesh rebuilding is available [39]: First, Delaunay triangulation is performed to generate the convex

domain on the basis of a cloud of nodes, and thus a mesh consisting of tetrahedrons is obtained in three-dimensional case. Then, the radius of the circumsphere of each tetrahedron is examined and the tetrahedrons with a radius greater than αh are deleted, and hence the remain tetrahedrons correspond to the computational domain.

Note that in the SPFEM, thanks to the node integration technique of strain smoothing, more distorted mesh can be used without serious loss of accuracy, i.e. the requirement of mesh quality is not as strict as the original PFEM. Nevertheless, mesh smoothing technique is utilized here to improve the mesh quality. To be specific, the Laplacian smoothing technique [40, 41] is applied to a few nodes when they are too close to their neighbours. Some other mesh smoothing technique can also be used (e.g. [33, 42]), however, we choose the Laplacian smoothing technique due to its simplicity and high efficiency. Moreover, with the Laplacian smoothing technique, the critical time step size can be increased remarkably, which is useful for improving the computational efficiency.

It should be pointed out that in some PFEM simulations, new particles were introduced during the simulation to further improve the mesh quality (e.g. [14, 43, 44]). However, for the present approach, it seems that introducing new particles is not necessary and thus no new particle is introduced during the simulation for simplicity.

3. GPU parallel computing strategy

3.1. GPU acceleration scheme

The GPU parallel computing is implemented based on the Compute Unified Device Architecture (CUDA), a toolkit developed by NVIDIA to perform parallel computing using their graphics cards. The parallelisation scheme is shown in Fig. 3. The whole computation consists of a CPU part and a GPU part. The CPU part works as a process controller, which is used for loading and saving data, initializing GPU data, and controlling the GPU computing kernels to finish designated computation tasks, while the GPU part executes the actual computing.

Ideally, all the steps are expected to be parallelised on GPU to maximise the speedup of GPU parallel computing. However, in the SPFEM, the Delaunay triangulation technique is required to re-generate the computation mesh. In this study, a state-of-the-art library for Delaunay triangulation, CGAL [45], is adopted to obtain a robust Delaunay triangulation. So the Delaunay triangulation is executed on CPU. However, the workload of Delaunay triangulation is minimal, because it is performed intermittently only when the mesh is too distorted. Hence the CPU execution of the Delaunay triangulation has only a small influence on the overall speedup. Except for the Delaunay triangulation, all other calculations are executed on GPU in a parallelised manner. To reduce memory transfer as much as possible, especially the transfer between CPU and GPU, all the data are stored in the GPU global memory during the entire simulation, and they are transferred to the CPU memory only when the Delaunay triangulation is required and when the occasional data output is required.

It can be seen from Algorithm 1 that the computation cycle of SPFEM is generally quite simple, which greatly ensures the applicability of GPU parallel computing. Especially for Steps 2, Step 4 and Step 6 in Algorithm 1, the GPU parallel computing is straightforward. Because in these steps, the computations at nodes are independent to each other, they can be executed on GPU by parallelising over nodes with no race condition. The so-called race condition is defined as the situation that 2 or more threads try to change the same shared or global memory location at the same time. However, for Step 3 and Step 5, the smoothed strain and the internal force contribution of a node are related to its adjacent nodes. As a result, the race condition occurs, which should be paid more attention to. Note that Step 3 and Step 5 are both related to the strain smoothing technique.

3.2. Parallelisation of the strain smoothing technique

A crucial feature of the SPFEM is the adoption of strain smoothing technique. From a computation perspective, the core of this technique lies in the calculation of the smoothed strain matrix $\tilde{\boldsymbol{B}}$ for all nodes. It is the premise of the calculation of smoothed strains and internal forces of nodes. In three-dimensional cases, the number of related nodes for a node is much higher than that in two-dimensional cases, which leads to a big smoothed strain matrix for each node and greatly increases the memory requirement during the computation of SPFEM. For example, in the kernel function for calculating the internal forces of nodes, we need either to calculate the smoothed strain matrix $\tilde{\boldsymbol{B}}$ or to read it calculated before. In both approaches, the smoothed strain matrices can be quite memory-consuming, which is a bottleneck for the application to large-scale problems.

To reduce the GPU memory consumption related to the strain smoothing technique, new formulations are used in this study. The meaning of reducing memory consumption is twofold. First, the main function of GPU parallel computing is to solve large-scale problems. Reducing memory consumption can increase the scale (i.e. number of nodes) of the problem to be simulated with the same hardware. Please note that the memory of a GPU is constant and not easy to be extended as a CPU. Meanwhile, reducing memory consumption is also meaningful for improving efficiency. As the global memory used to store the smoothed strain matrices is a high-latency memory, accesses to the global memory are time-consuming and should be reduced as much as possible to improve the efficiency.

Let's start from the original formulation. The smoothed strain of a node k is calculated by

$$\tilde{\boldsymbol{\varepsilon}}_k = \tilde{\boldsymbol{B}}\boldsymbol{u}_k \tag{16}$$

where $\boldsymbol{u}_k$ is the vector assembled by the displacements of all the nodes in the smoothing cell associated with node k. By substituting Eq. (2) into Eq. (16), the smoothed strain of the node k is calculated as

$$\tilde{\boldsymbol{\varepsilon}}_k = (\frac{1}{V_k}\sum_{i=1}^{E_k}\frac{1}{4}V^i\boldsymbol{B}^i)\boldsymbol{u}_k \tag{17}$$

When calculating the smoothed strains of nodes with the original formulation (Eq. (17)), parallelization over nodes is performed. In the thread of a specific node k, looping over the elements related to this node is performed.

Noting the equivalence between the vector assembled by the displacements of the nodes related to node k and that assembled by the displace-

ments of the elements related to node k, the above equation can be re-written as

$$\tilde{\boldsymbol{\varepsilon}}_k = \frac{1}{V_k} \sum_{j=1}^{E_k} \frac{1}{4} V^j \boldsymbol{B}^j \boldsymbol{u}^j \tag{18}$$

where E_k is the number of elements related to node k, V^j is the volume of element j, $\boldsymbol{B}^j$ is the strain matrix of element j in standard FEM, and $\boldsymbol{u}^j$ is the vector assembled by the displacements of element j. Please note the difference between the original formulation (Eq. (17)) and the new formulation (Eq. (18)). With the new formulation (Eq. (18)), parallelization over elements can be easily performed to obtain the smoothed strains of all the nodes.

The new formulation for the calculation of internal forces related to the strain smoothing technique is a bit more complicated. In the original formulation of SPFEM, the internal forces applied on its neighbour nodes for a node k can be calculated as,

$$\boldsymbol{F}_k^{int} = \sum_{i=1}^{N_k} \tilde{\boldsymbol{B}}_k^T \boldsymbol{\sigma}_k V_k \tag{19}$$

where N_k is the number of the neighbour nodes of node k. By substituting Eq. (2) into Eq. (19), we have

$$\boldsymbol{F}_k^{int} = \sum_{i=1}^{N_k} \sum_{j=1}^{E_k} \frac{1}{4} V^j \boldsymbol{B}^j \boldsymbol{\sigma}_k \tag{20}$$

where E_k is the number of elements related to node k. Since the two summations in Eq. (20) are both carried out on the elements related to node k,

Eq. (20) can be re-written as

$$\boldsymbol{F}_k^{int} = \sum_{j=1}^{E_k} \sum_{m=1}^{4} \frac{1}{4} V^j \boldsymbol{B}_m^j \boldsymbol{\sigma}_k \tag{21}$$

in which $\boldsymbol{B}_m^j$ is the strain matrix related to node m of element j.

Let us make a comparison between the original formulations (Eqs. (17) and (19)) and the new ones (Eqs. (18) and (21)) from the perspective of GPU memory consumption. In three-dimensional cases, the strain matrix of node k related to a specific node consists of 18 floating-point numbers. Hence, using the original formulation, a total of $(N_k + 1) \times 18$ floating-point numbers should be stored to obtain the smoothed strain matrix of a particular node k. As the number of the neighbour nodes N_k of a specific node is undetermined, it is sensible to allocate a memory of equal size which is slightly larger than needed. As stated above, the possible maximum value of N_k in three-dimensional cases is much larger than that in two-dimensional cases. In this study, N_k is taken to be 64 to guarantee robustness. Bearing in mind that the smoothed strains and internal forces of all the nodes are calculated in a parallelised way. As a result, $M \times (N_k + 1) \times 18$ floating-point numbers should be saved during the computation, where M is the concurrent thread number in GPU, which generally ranges from several thousands to tens of thousands for a common GPU. One may save memory by only saving the derivatives of shape functions instead of the strain matrix. This approach will surely reduce the memory requirement, but it is not an essential improvement. However, with the new formulations, if we calculate and store the strain matrices of all the elements before, which is sensible to

avoid unnecessarily repeated computation, only the strain matrix variable $\boldsymbol{B}_m^j$ is required to be loaded. Thus, in the kernel function for calculating the smoothed strains of nodes and the kernel function for calculating the internal forces of nodes, the memory occupation related to the strain matrices is greatly reduced from $M \times (N_k + 1) \times 18$ floating-point numbers to $M \times 18$ floating-point numbers. The memory occupation can be further reduced by only saving the derivatives of shape functions.

Note that as parallel computing is performed with GPU threads, the so-called race condition may occur. With the new formulations, the smoothed strain of a particular node may be changed by all the threads of the elements related to it at the same time, and the internal force of a particular node may also be changed by all the threads corresponding to its neighbour nodes at the same time. So race condition occurs for both the calculations of the smoothed strains and the internal forces. With the original formulations, the former race condition does not occur while the latter race condition exists. To deal with the race condition problem, the so-called atomic operations, which are provided by CUDA, are used to calculate the smoothed strains and the internal forces of nodes. With the atomic operations, we are capable of reading, modifying, and writing a value back to the same memory address while avoid interferences between different threads.

Please note that the time cost of atomic operations is generally notable. However, in the present approach, because atomic operations are only performed to accumulate the smoothed strains and the internal forces of nodes whereas the intensive computations of the smoothed strains and the internal forces are still parallelized, the time cost of waiting in atomic operations

is not so notable. Taking the computation of the smoothed strains as an example, although atomic operations are performed with the new formulation, the time cost is still slightly lower than that with the original formulation. The main reason is that for each thread the computational workload is much heavier with the original form although atomic operations are avoided. With the original formulation, parallelization over nodes is performed. In the thread of a specific node k, looping over the elements related to node k is required. Moreover, the identification of the sequential number of node k in its surrounding elements is also required. On the contrary, with the new formulation, parallelization over elements is performed with a rather simple procedure for each thread.

3.3. Parallelisation for getting the indices of elements and nodes related to nodes

Another crucial step in the SPFEM is to get the indices of elements and nodes related to all the nodes, although this step is not so time-consuming in the entire simulation. This step should be performed in Step 2 in Algorithm 1, before the calculation of smoothed strains. A naive method is to loop over all the elements for each node to determine if the element is related to this node. However, this method is both time-consuming and memory-consuming. A more efficient algorithm is used instead in this study.

As shown in Fig. 4, a kernel function in GPU is executed to perform the same calculation for all the elements with parallelised GPU threads. For the element i, its index i is recorded respectively to the four nodes of the element. A key step is to accumulate the serial number of current element related to each node. The so-called atomic operations are used again to do this. As

the sequence of the related elements for each node does not matter in the SPFEM, desired results can be obtained by the parallelisation over elements with GPU threads.

It is more straightforward to get the indices of nodes related to nodes once the indices of elements related to nodes are obtained. Parallelisation over nodes with GPU threads can be performed without the so-called race condition. For each node, all the nodes of the elements related to this node are picked out and the unique indices of nodes are records as the indices of nodes related to nodes.

4. Numerical examples

To validate the proposed approach and to evaluate the speedup of the GPU parallel computing, three benchmark examples are numerically solved. The first example is the vibration of an elastic continuum bar. This problem has an analytical solution and is suitable for validating the proposed approach, especially for investigating the new formulations related to the strain smoothing technique in terms of correctness and efficiency. In the second example, the laboratory experiment of soil collapse is simulated. The experiment was carried out by Bui [3] to validate their SPH approach for geo-materials. The third example is the progressive failure of a long strain-softening soil slope. Since the geometry of the problem is complicated during the whole simulation, and a strain softening constitutive model is used, it is suitable to validate the application capability of the proposed approach on an actual engineering problem.

The GPU parallel computing was conducted on a personal computer with

an NVIDIA GeForce GTX 1080Ti GPU. On this card, 3584 cores are available and the global memory is 11 GB. For comparison purpose, the first and the second example were also solve by sequential CPU simulation. The CPU is Intel i7-8650U with a frequency of 1.90 GHz. Different meshes with various number of nodes are used for the first and the second examples to evaluate the speedup thoroughly. In the first example, both single-precision and double-precision simulations are performed for comparison. For other examples, only double-precision simulations are performed, because single-precision simulation is not suitable for complicated problems, such as the strain-softening [46] and other specific behaviour of soil (e.g. [47–53]). Furthermore, the results of the first example show that the efficiency of single-precision simulation is only slightly higher than that of double-precision simulation.

4.1. Axial vibration of an elastic continuum bar

The first example considers the axial vibration of an elastic continuum bar. As shown in Fig. 5, the length, width and height of the continuum bar are 10 m, 1 m and 1 m respectively. The bar is linearly elastic and the material parameters are: Young's modulus $E = 100$ Pa, Poisson's ratio $\nu = 0$ and the density $\rho = 1\ kg/m^3$. The left end (x = 0 m) of the bar is fixed while all other surfaces of the bar are free. Free vibration of the bar along the axial direction (i.e. x-direction) is considered.

This problem may be solved using separation of variables [54]. The bar is found to oscillate in modes that are dependent on the initial conditions. A specific mode can be excited by using initial conditions which are multiples of this natural mode of the system. Here, the first mode is considered, and

thus the analytical solution of this problem is given by [54, 55]:

$$\begin{aligned} v(x,t) &= v_0 cos(\omega_1 t) sin(\beta_1 x) \\ u(x,t) &= v_0 sin(\omega_1 t) sin(\beta_1 x)/\omega_1 \end{aligned} \tag{22}$$

where $v(x,t)$ is the velocity at time t and $u(x,t)$ is the displacement at time t. The eigenvalue is $\beta_1 = \pi/2L$, and the frequency of oscillation related to the eigenvalue is $\omega_1 = \beta_1 c$ where $c = \sqrt{E/\rho}$ is the speed of elastic wave. The corresponding initial velocity and displacements are given by

$$\begin{aligned} v(x,0) &= v_0 sin(\beta_1 x) \\ u(x,0) &= 0 \end{aligned} \tag{23}$$

A total of 5 meshes with different numbers of nodes (12221, 88641, 289261, 674081, 1303101) are utilized in the simulations. For each mesh, both sequential CPU simulation and GPU accelerated simulation are conducted, and both single-precision and double-precision simulations are performed. Moreover, the original formulations and the new formulations related to the strain smoothing technique are used respectively. A total time of 10.0 s is simulated. The incremental time step size remains constant during each simulation. However, for different mesh density, the incremental time step size is different to maintain numerical stability. Correct results are obtained with all the simulations. The numerical results obtained by the single-precision GPU simulation with the coarsest mesh and the new formulations are compared with the analytical solution, as shown in Fig. 6. They are almost identical to each other.

Figure 7 shows the comparison of the computational time cost spent with the original formulations and with the new ones. The time cost ratio, which is defined as the ratio between the time cost with the original formulations and that with the new formulations, is defined here to make a clear comparison. It can be seen that less time cost is spent with the new formulation during all the simulations. The ratio between the time cost with the original formulation and that with the new formulation ranges from 1.71 to 2.54 for single-precision GPU computation and ranges from 3.69 to 4.20 for double-precision GPU computation. It is clear that adopting the new formulation can not only save memory occupation but also greatly improve computational efficiency. Therefore, the new formulations are adopted throughout all the following simulations due to their superiority.

The computational costs for different meshes with different simulation measures are listed in Table 1. The double-precision CPU simulation with the largest number of nodes is taken as a reference to evaluate the distribution of the workload across the steps of the present approach. As shown in Fig. 8, for this example, most of the computational effort is allocated to Step 3 and Step 5 (please see Algorithm 1), which respectively account for 29.89% and 64.87% of the total workload. The computational effort of Step 1 and Step 2 is minimal because no re-meshing is performed. The computational effort of Step 4 is low because a simple elastic constitutive model is used.

The speedup ratios between the GPU simulation and CPU simulation are plotted in Fig. 9. The overall speedup ranges from 7.82 to 10.89 for single-precision computation and ranges from 8.40 to 12.30 for double-precision computation. It is clear that with the GPU-accelerated approach, a speedup

around 10 can be achieved. Moreover, the speedup with finer mesh is generally slightly higher than that with coarser mesh. As shown in Fig 9, the performance of GPU simulation varies for different steps in Algorithm 1. The speedup of Step 4 is the largest because all the threads can be executed without mutual effect. For Step 3 and Step 5, the speedup is smaller. However, the speedup performance is still rather obvious. The speedup for Step 6 is the lowest, which may be due to the algorithm complexity related to boundary condition application and the relatively low computation intensity. Note that since Step 1 is executed on the CPU, no speedup is recorded. Moreover, Step 2 only takes a minimal time even for the finest mesh, which increases the randomness of the value of speedup, so the speedup of Step 2 is also not plotted in Fig 9.

4.2. Laboratory experiment of soil collapse

Bui et al. [3] conducted a simple laboratory experiment of soil collapse to validate their SPH approach for large-deformation analysis of geomaterials. Besides the SPH simulation [3], the experiment was also simulated by MPM [56] and two-dimensional SPFEM [22], so it can serve as a good benchmark. In the laboratory experiment, the soil continuum was modelled by a large number of small aluminium bars, which is a common approach to model cohesiveless soil physically. Two types of aluminium bars were used with a diameter of 1 mm and 1.5 mm respectively and a length of 50 mm. These aluminium bars were piled upon a flat surface in parallel, and thus a plane-strain condition is mimicked. The soil continuum was modelled by arranging the aluminium bars into a rectangular area (200 mm x 100 mm) which is generated by standing two flat solid walls on the flat surface. The

soil collapse was triggered by quickly removing the right supporting wall. A reference grid was set behind the soil continuum, out of contact with the aluminium bars, to measure surface configuration of soil continuum after collapse. Moreover, square grids (20 mm x 20 mm) were plotted on the soil continuum to visualize the failure pattern.

A critical problem is to determine the mechanical parameters of the soil continuum modelled by aluminium bars that can be used in the following numerical simulations. In the laboratory experiment, a shear box test apparatus was designed to perform tests on a group of aluminium bars, and then four experiments under different normal loading conditions were conducted to obtain the parameters of the soil continuum.

A total of 5 meshes with different number of nodes (10999, 25535, 50338, 75720, 100072) are utilized for simulation (Fig. 10). For each mesh, double-precision GPU simulation and sequential CPU simulation are conducted respectively. The soil is treated as an elastic perfect plastic material with Mohr-Coulomb yield criterion. A zero dilation angle is assumed, corresponding to a non-associated flow rule. The following model parameters are used according to [3]: shear modulus G = 0.7 MPa, Poisson's ratio $\nu = 0.3$, density $\rho = 2650\ kg/m^3$, cohesion strength c = 0 kPa and friction angle $\varphi = 19.8°$. Before the simulation of soil collapse, the initial stress due to gravity is generated by a dynamic relaxation simulation, in which the bottom boundary is fully fixed and the four side boundaries are constrained at their normal directions. Then the soil collapse is triggered by removing the restraint of the right boundary. A Coulomb contact model between soil and ground is used, in which no tangential relative motion is assumed. A total time of 1.0

s is simulated, after which a stable slope is formed. Adaptive time step size is utilized during the simulation, and the scale factor α in Eq. (10) is taken to be 0.9.

The final surface configurations after collapse obtained with different mesh densities are compared with the experimental result, as shown in Fig. 11(a). All numerical results show a similar response, and they all match well with the experimental results. With the increase of the number of nodes, the runout distance increases slightly. The failure line, which is defined as the line between motionless soil and the moved soil, is also compared against the experimental result and good agreement can also be found. As shown in Fig. 11(b), the numerical results are compared with the results obtained by other numerical methods (i.e. SPH [3] and MPM [56]), as well as two-dimensional SPFEM [22]. All the numerical methods can well capture the experimental result. The run-out distance obtained by this study is slightly shorter than the two-dimensional SPFEM [22]. The reason may be that the nodes in the three-dimensional case interact with more nodes in comparison to two-dimensional cases. The final mesh configuration with the coarsest mesh is chosen to make a visual comparison with the experimental result, as shown in Fig. 12. It is clear that the three-dimensional SPFEM is capable of reproducing the experimental result.

The computational costs of double-precision GPU simulation and sequential CPU simulation are listed in Table 2. The latter with the largest number of nodes is taken as a reference to evaluate the distribution of the workload across the steps of the present approach. As shown in Fig. 13, the time-consuming steps are Step 2, Step 3 and Step 5, which respectively account

for 19.33%, 20.84% and 36.66% of the total workload, while the computational effort of Step 6 is minimal. The above workload distribution may be considered as a typical case for geotechnical applications.

As shown in Fig. 14, an overall speedup ranging from 8.21 to 11.17 is achieved by GPU parallel computing. A speedup from 35.31 to 66.49 can be achieved for the constitutive integration associated with plastic problems, which is the highest in all the steps, because the parallel operations over nodes are independent of other nodes. The speedup of Step 6 is the lowest among all the steps, which is similar to the first example, i.e. the axial vibration of a continuum bar. Compared with the first example, more Delaunay triangulation operations are performed in Step 1, which are not accelerated by GPU parallel computing. Meanwhile, the constitutive integration in Step 4 achieves a higher speedup. These two factors together lead to an overall speedup similar to the first example.

4.3. progressive failure of a long strain-softening soil slope

This example considers the progressive failure of a long strain-softening soil slope. As shown in Fig. 15, the slope is initially 25 m long and 5 m high and has a slope angle of 45°. A width of 5 m is chosen to generate a three-dimensional computation domain. The soil is considered to be a strain-softening soil to mimic the progressive failure of a sensitive clay slope, which is commonly found in nature. A main feature of sensitive clay is the decrease of undrained shear strength with the increase of plastic strain. This feature can be captured by a simple strain-softening Tresca model. As shown in Fig. 16, the cohesion equals to its peak value (c_p) at the beginning and decreases proportionally to the equivalent plastic strain invariant defined by

$\bar{\varepsilon} = \sqrt{2/3\boldsymbol{e}_p : \boldsymbol{e}_p}$. where $\boldsymbol{e}_p$ is the deviatoric part of the plastic strain tensor. When the invariant achieve a certain value ($\bar{\varepsilon}_{pr}$), the cohesion reduces to its residual value (c_r). This example, feature by the complicated geometry during the whole simulation process and the strain-softening constitutive model, has been well simulated by two-dimensional MPM [57], PFEM [18] and SPFEM [13].

For the long strain-softening soil slope, the failure mode, run-out distance and retrogression distance are greatly influenced by the degree of the strength reduction (i.e. c_r/c_p). As stated in Ref. [59], lower residual strength corresponds to more times of retrogression failure, longer run-out distance and longer retrogression distance. In this study, a relatively severe strength reduction (i.e. 0.2) is considered to generate multiple retrogression failure.

It should be pointed out that the utilization of the strain-softening model may result in the solution becoming mesh dependent. This stems from the fact that the related boundary value problem is no longer elliptic in statics or hyperbolic in dynamics [58]. Special regularisation techniques, such as nonlocal theory or gradient plasticity, have to be incorporated to overcome this issue. As this study focuses on the GPU acceleration of SPFEM, the regularisation is not involved.

Double-precision GPU simulation is conducted to validate the capability of the GPU-accelerated SPFEM on solving large-scale geotechnical problems. A total number of 250584 nodes are used in the simulation. The following model parameters are used: Young's modulus E = 1.0 MPa, Poisson's ratio ν = 0.33, density ρ = 2000 kg/m^3, peak cohesion c_p = 20 kPa, residual cohesion c_r = 4 kPa, and the softening modulus H = -50.0 kPa. A Coulomb

contact model between soil and ground is used and the friction coefficient is taken to be 0.3. As shown in Fig. 15, the bottom boundary is fully fixed and the left side boundary is constrained at x-direction. All the motions along y-direction are constrained to mimic a plane strain response. The soil is first assumed to be elastic and the initial stress due to gravity is generated by a dynamic relaxation simulation. After the initial stress is generated, the soil plasticity is considered. Since the slope can not maintain stability with these material parameters, the landslide is triggered. A total of 10 seconds are simulated to reproduce the whole progressive failure process. Adaptive time step size is utilized and the scale factor α in Eq. (10) is taken to be 0.9.

The simulation is completed in around 5.66 hours. The initial vertical stress is shown in Fig. 17(a). The development process of the progressive failure is shown in Fig. 17(b)-(g), with coloured contours representing the equivalent plastic strain invariant $\bar{\varepsilon}$. Many shear bands occur during the whole simulation process due to the adoption of the strain-softening soil constitutive model. As shown in Fig. 17(b), the first major shear band (MS1) initiate from the bottom and propagate towards the top surface. During the sliding, two crossed shear bands (S1, S2) occur in the middle of the landslide body. One (S1) propagates towards the top surface and the other (S2) propagates towards the front inclined surface, resulting in a graben (Fig. 17(c)). With the advancement of the landslide body, a new major shear band (MS2) initiate from the bottom and propagate towards the top surface due to lack of support, leading to the second retrogressive failure (Fig. 17(d)). And then, a new shear band (S3) forms in the middle of the second landslide body, which initiates from the bottom and propagate towards the front inclined

surface (Fig. 17(e)). Then a new shear band (S4) initiates from the bottom of the landslide body and propagates towards the top surface (Fig. 17(f)). The retrogressive failure finally stops with the help of the friction between the landslide bodies and the ground. Fig. 17(g) shows the final configuration of the slope. Eventually, the retrogressive failure results in a deposit with a run-out distance of 12.0 m and a retrogression distance of 13.6 m (Fig. 17(g)).

A crucial factor affecting the run-out distance and the retrogression distance is the friction coefficient between soil and ground. To investigate this, a new simulation is conducted with a friction coefficient of 0.1. The comparison of the final configuration of the slope with different friction coefficients is shown in Fig. 18. A smaller friction coefficient corresponds to larger run-out distance and retrogression distance. The final run-out distance is 21.0 m while the retrogression distance is 18.4 m. Taken together, the GPU-accelerated SPFEM is capable of solving large-scale complicated large-deformation problem in geomechanics.

5. Conclusions

PFEM is an attractive numerical framework for large deformation analysis in geomechanics. By incorporating the strain smoothing technique for node integration, the authors proposed an improved PFEM, i.e. SPFEM. The primary advantages of SPFEM is that all the field viables are computed and stored at nodes/particles. Some additional advantages include no volumetric locking when low order elements are used, and insenstivity to element distortion. The objectives of the present work are twofold. First, the explicit SPFEM is extended to three dimensional cases. Then, a GPU accelerated

SPFEM is developed.

A GPU parallelisation strategy for the three-dimensional explicit SPFEM is presented in detail. All the steps are executed on GPU except that the Delaunay triangulation is executed on CPU with a state-of-the-art library. Since the numerical implementation of the operations related to strain smoothing technique is crucial to memory consumption, new formulations to accumulate the smoothed strains and the internal forces of nodes are proposed. The computational time cost ratio between the original formulation and the new formulation ranges from 1.71 to 2.54 for single-precision GPU computation and from 3.69 to 4.20 for double-precision computation. Compared with the sequential CPU simulation, the GPU-accelearted simulation results in a significant speedup. The overall speedup ranges from 7.82 to 10.89 for single-precision computation for elastic problems and from 8.40 to 12.30 for double-precision computation. A speedup from 35.31 to 66.49 can be achieved for the constitutive integration associated with plastic problems, which is the highest in all types of calculations in the SPFEM. Finally, the developed GPU-accelerated SPFEM is employed to model the progressive failure of a long strain-softening soil slope. It is found that the presented method can model large-scale complex problems efficiently and delivers reliable results.

Acknowledgements

The research is supported by the Natural Science Foundation of Guangdong Province (Grant No. 2018A030310346), the H2020 Marie Skłodowska-Curie Actions RISE 2017 HERCULES (778360) and FRAMED (734485), the Erasmus+ KA2 project Re-built (2018-1-RO01-KA203-049214), the Nazarbayev

University Research Fund (SOE2017001), and the Natural Science Foundation of China (Grant No. 41807223).

References

[1] Nazem M., Sheng D.C., Carter J.P., et al. Arbitrary Lagrangian-Eulerian method for large-strain consolidation problems. International Journal for Numerical and Analytical Methods in Geomechanics, 2008, 32(9): 1023-1050.

[2] Hu Y., Randolph M.F. A practical numerical approach for large deformation problems in soil. International Journal for Numerical and Analytical Methods in Geomechanics, 1998, 22(5): 327–350.

[3] Bui H.H., Fukagawa R., Sako K., et al. Lagrangian meshfree particles method (SPH) for large deformation and failure flows of geomaterial using elastic-plastic soil constitutive model. International Journal for Numerical and Analytical Methods in Geomechanics, 2008, 32(12): 1537-1570.

[4] Abe K., Soga K., Bandara S. Material point method for coupled hydromechanical problems. Journal of Geotechnical Engineering, 2014, 140(3): 04013033.

[5] Oñate E, Idelsohn SR, Del Pin F, Aubry R (2004) The particle finite element method: an overview. Int J Comput Methods 1(2):267–307.

[6] Carbonell JM, Oñate E, Suárez B. Modeling of ground excavation with

the particle finite-element method. Journal of Engineering Mechanics. 2010;136(4):455-463.

[7] Carbonell JM, Oñate E, Suárez B. Modelling of tunnelling processes and rock cutting tool wear with the particle finite element method. Computational Mechanics. 2013;52(3):607-629.

[8] Zhang X, Krabbenhoft K, Pedroso DM, Lyamin AV, Sheng D, da Silva MV, et al. Particle finite element analysis of large deformation and granular flow problems. Computers and Geotechnics. 2013;54:133-142.

[9] Zhang X, Krabbenhoft K, Sheng D. Particle finite element analysis of the granular column collapse problem. Granular Matter. 2014;16(4):609-619.

[10] Dávalos C, Cante J, Hernández JA, Oliver J. On the numerical modeling of granular material flows via the Particle Finite Element Method (PFEM). International Journal of Solids and Structures. 2015;71:99-125.

[11] Zhang X, Krabbenhoft K, Sheng D, Li W. Numerical simulation of a flow-like landslide using the particle finite element method. Computational Mechanics. 2015;55(1):167-177.

[12] Salazar F, Irazábal J, Larese A, Oñate E. Numerical modelling of landslide-generated waves with the particle finite element method (PFEM) and a non-Newtonian flow model. International Journal for Numerical and Analytical Methods in Geomechanics. 2016;40(6):809-826.

[13] Yuan W H, Liu K, Zhang W, et al. Dynamic modeling of large deformation slope failure using smoothed particle finite element method. Landslides, 2020; DOI: 10.1007/s10346-020-01375-w.

[14] Monforte L, Arroyo M, Carbonell JM, Gens A. Numerical simulation of undrained insertion problems in geotechnical engineering with the Particle Finite Element Method (PFEM). Computers and Geotechnics. 2017;82:144-156.

[15] Monforte L, Carbonell JM, Arroyo M, Gens A. Performance of mixed formulations for the particle finite element method in soil mechanics problems. Computational Particle Mechanics. 2016;4(3):269-284.

[16] Monforte L, Arroyo M, Carbonell JM, Gens A. Coupled effective stress analysis of insertion problems in geotechnics with the Particle Finite Element Method. Computers and Geotechnics. 2018;101:114-129.

[17] Yuan W-H, Zhang W, Dai B-B, Wang Y. Application of the particle finite element method for large deformation consolidation analysis. Engineering Computations. 2019;36(9):3138-3163.

[18] Zhang X, Sheng D, Sloan SW, Bleyer J. Lagrangian modelling of large deformation induced by progressive failure of sensitive clays with elastoviscoplasticity. International Journal for Numerical Methods in Engineering. 2017;112(8):963-989.

[19] Zhang X, Sloan SW, Oñate E. Dynamic modelling of retrogressive landslides with emphasis on the role of clay sensitivity. Interna-

tional Journal for Numerical and Analytical Methods in Geomechanics. 2018;42(15):1806-1822.

[20] Zhang X, Oñate E, Torres SAG, Bleyer J, Krabbenhoft K. A unified Lagrangian formulation for solid and fluid dynamics and its possibility for modelling submarine landslides and their consequences. Computer Methods in Applied Mechanics and Engineering. 2019;343:314-338.

[21] Zhang W, Yuan W, Dai B. Smoothed Particle Finite-Element Method for Large-Deformation Problems in Geomechanics. International Journal of Geomechanics. 2018;18(4):04018010.

[22] Yuan W-H, Wang B, Zhang W, Jiang Q, Feng X-T. Development of an explicit smoothed particle finite element method for geotechnical applications. Computers and Geotechnics. 2019;106:42-51.

[23] Franci A, Cremonesi M, Perego U, et al. A Lagrangian nodal integration method for free-surface fluid flows. Computer Methods in Applied Mechanics and Engineering. 2020;361:112816.

[24] Franci A. Lagrangian finite element method with nodal integration for fluid–solid interaction. Computational Particle Mechanics. 2020;https://doi.org/10.1007/s40571-020-00338-1.

[25] Cremonesi M, Franci A, Idelsohn S, et al. A State of the Art Review of the Particle Finite Element Method (PFEM). Archives of Computational Methods in Engineering. 2020;https://doi.org/10.1007/s11831-020-09468-4.

[26] Dong Y, Wang D, Randolph M F. A GPU parallel computing strategy for the material point method. Computers and Geotechnics, 2015, 66: 31-38.

[27] Xia X, Liang Q. A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations. Environmental Modelling & Software, 2016, 75: 28-43.

[28] Peng C, Wang S, Wu W, et al. LOQUAT: an open-source GPU-accelerated SPH solver for geotechnical modeling. Acta Geotechnica, 2019, 14(5): 1269-1287.

[29] Chen J Y, Lien F S, Peng C, et al. GPU-accelerated smoothed particle hydrodynamics modeling of granular flow. Powder Technology, 2020, 359: 94-106.

[30] Wu S C, Liu G R, Zhang H O, et al. A node-based smoothed point interpolation method (NS-PIM) for three-dimensional heat transfer problems. International Journal of Thermal Sciences, 2009, 48(7): 1367-1376.

[31] Chen J S, Wu C T, Yoon S, et al. A stabilized conforming nodal integration for Galerkin mesh-free methods. International journal for numerical methods in engineering, 2001, 50(2): 435-466.

[32] Liu G R, Nguyen-Thoi T, Nguyen-Xuan H, et al. A node-based smoothed finite element method (NS-FEM) for upper bound solutions to solid mechanics problems. Computers & structures, 2009, 87(1-2): 14-26.

[33] Meduri S, Cremonesi M, Perego U. An efficient runtime mesh smoothing technique for 3D explicit Lagrangian free-surface fluid flow simu-

lations. International Journal for Numerical Methods in Engineering, 2019, 117(4): 430-452.

[34] Joe B. Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation. International Journal for Numerical Methods in Engineering, 1991, 31(5): 987-997.

[35] Cheng S W, Dey T K, Edelsbrunner H, et al. Silver exudation. Journal of the ACM (JACM), 2000, 47(5): 883-904.

[36] Ho-Le K. Finite element mesh generation methods: a review and classification. Computer-aided design, 1988, 20(1): 27-38.

[37] Idelsohn S R, Oñate E, Pin F D. The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. International journal for numerical methods in engineering, 2004, 61(7): 964-989.

[38] Edelsbrunner H, Mücke E P. Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG), 1994, 13(1): 43-72.

[39] Cremonesi M, Frangi A, Perego U. A Lagrangian finite element approach for the analysis of fluid–structure interaction problems. International Journal for Numerical Methods in Engineering, 2010, 84(5): 610-630.

[40] Field D A. Laplacian smoothing and Delaunay triangulations. Communications in applied numerical methods, 1988, 4(6): 709-712.

[41] Freitag L A, Ollivier-Gooch C. A comparison of tetrahedral mesh im-

provement techniques[R]. Argonne National Lab., IL (United States), 1996.

[42] Vartziotis D, Wipper J, Schwald B. The geometric element transformation method for tetrahedral mesh smoothing. Computer Methods in Applied Mechanics and Engineering, 2009, 199(1-4): 169-182.

[43] Rodriguez J M, Carbonell J M, Cante J C, et al. The particle finite element method (PFEM) in thermo-mechanical problems. International Journal for Numerical Methods in Engineering, 2016, 107(9): 733-785.

[44] Rodríguez J M, Carbonell J M, Cante J C, et al. Continuous chip formation in metal cutting processes using the Particle Finite Element Method (PFEM). International Journal of Solids and Structures, 2017, 120: 81-102.

[45] Boissonnat J D, Devillers O, Teillaud M, et al. Triangulations in CGAL[C]//Proceedings of the sixteenth annual symposium on Computational geometry. 2000: 11-18.

[46] Murakami S, Liu Y. Mesh-dependence in local approach to creep fracture. International Journal of Damage Mechanics, 1995, 4(3): 230-250.

[47] Wu Y, Li N, Hyodo M, et al. Modeling the mechanical response of gas hydrate reservoirs in triaxial stress space. International Journal of Hydrogen Energy, 2019, 44(48): 26698-26710.

[48] Wu Y, Yamamoto H, Cui J, et al. Influence of Load Mode on Particle Crushing Characteristics of Silica Sand at High Stresses. International Journal of Geomechanics, 2020, 20(3): 04019194.

[49] Medicus G, Schneider-Muntau B, Kolymbas D. Second-order work in barodesy. Acta Geotechnica, 2019, 14(5): 1483-1493.

[50] Luo Y, Luo B, Xiao M. Effect of deviator stress on the initiation of suffusion. Acta Geotechnica, 2020, 15: 1607–1617.

[51] Nitzsche K, Herle I. Strain-dependent slope stability. Acta Geotechnica, 2020, doi: 10.1007/s11440-018-0749-z.

[52] Luo Y, Huang Y. Effect of open-framework gravel on suffusion in sandy gravel alluvium. Acta Geotechnica, 2020, 15: 2649–2664

[53] Wang, S., Wu, W. A simple hypoplastic model for overconsolidated clays. Acta Geotech, 2020: https://doi.org/10.1007/s11440-020-01000-z

[54] L. Meirovitch, Analytical Methods in Vibrations, Macmillan, New York, 1967.

[55] Bardenhagen S G. Energy conservation error in the material point method for solid mechanics. Journal of Computational Physics, 2002, 180(1): 383-403.

[56] Sołowski W T, Sloan S W. Evaluation of material point method for use in geotechnics. International Journal for Numerical and Analytical Methods in Geomechanics, 2015, 39(7): 685-701.

[57] Wang B, Vardon P J, Hicks M A. Investigation of retrogressive and progressive slope failure mechanisms using the material point method. Computers and Geotechnics, 2016, 78: 88-98.

[58] Belytschko T, Lasry D. A study of localization limiters for strain-softening in statics and dynamics. Computers & structures, 1989, 33(3): 707-715.

[59] Zhang X, Sloan S W, Oñate E. Dynamic modelling of retrogressive landslides with emphasis on the role of clay sensitivity. International Journal for Numerical and Analytical Methods in Geomechanics, 2018, 42(15): 1806-1822.

Table 1: Computation time cost for the problem of axial vibration of a continuum bar (second)

Simulations	number of nodes	number of time step	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Total
sing-precision CPU simulation	12221	2000	2.36	0.02	13.40	1.54	27.38	1.24	45.93
	88641	4000	8.22	0.75	375.39	32.71	728.99	24.12	1170.18
	289261	6000	25.25	7.70	2063.50	165.82	4111.97	122.75	6496.98
	674081	8000	66.94	25.82	6498.63	510.78	13145.04	372.83	20620.04
	1303101	10000	154.29	134.80	16296.07	1240.30	34079.17	919.40	52824.02
double-precision CPU simulation	12221	2000	2.81	0.02	18.11	2.81	36.54	1.86	62.15
	88641	4000	9.89	0.51	424.05	47.22	841.97	31.81	1355.44
	289261	6000	25.48	5.20	2278.51	238.39	4591.31	162.28	7301.16
	674081	8000	74.94	20.70	7406.92	754.28	15675.08	512.34	24444.26
	1303101	10000	142.11	43.24	19435.08	1920.80	42176.64	1296.54	65014.41
sing-precision GPU simulation	12221	2000	2.36	0.01	1.01	0.22	2.00	0.28	5.88
	88641	4000	8.22	0.03	22.05	1.39	83.27	5.92	120.89
	289261	6000	25.25	0.19	107.80	4.96	459.48	27.06	624.73
	674081	8000	66.94	0.54	377.90	14.74	1676.92	93.64	2230.68
	1303101	10000	154.29	0.94	807.00	32.52	3662.06	194.78	4851.59
double-precision GPU simulation	12221	2000	2.81	0.03	1.14	0.16	2.71	0.55	7.40
	88641	4000	9.89	0.05	22.29	2.17	109.12	6.36	149.88
	289261	6000	25.48	0.20	101.90	7.04	516.84	25.02	676.49
	674081	8000	74.94	0.58	326.66	21.86	1701.70	78.44	2204.18
	1303101	10000	142.11	1.14	784.38	50.34	4121.88	184.04	5283.89

Table 2: Computation time cost for the problem of laboratory experiment of soil collapse (second)

Simulations	number of nodes	number of time step	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Total
double-precision CPU simulation	10999	54024	112	862	660	532	1221	118	3506
	25535	74025	320	2645	2234	1470	3708	401	10777
	50338	93215	772	5265	6030	4021	10138	1045	27271
	75720	112842	5559	29744	11923	8382	21327	1969	78904
	100072	122841	6244	15548	16760	9733	29484	2665	80434
double-precision GPU simulation	10999	54024	62	106	29	15	72	29	314
	25535	74025	244	253	102	36	477	70	1183
	50338	93215	742	599	259	61	1323	126	3110
	75720	112842	4260	955	491	108	2549	220	8583
	100072	122841	2422	1228	721	140	3642	307	8459

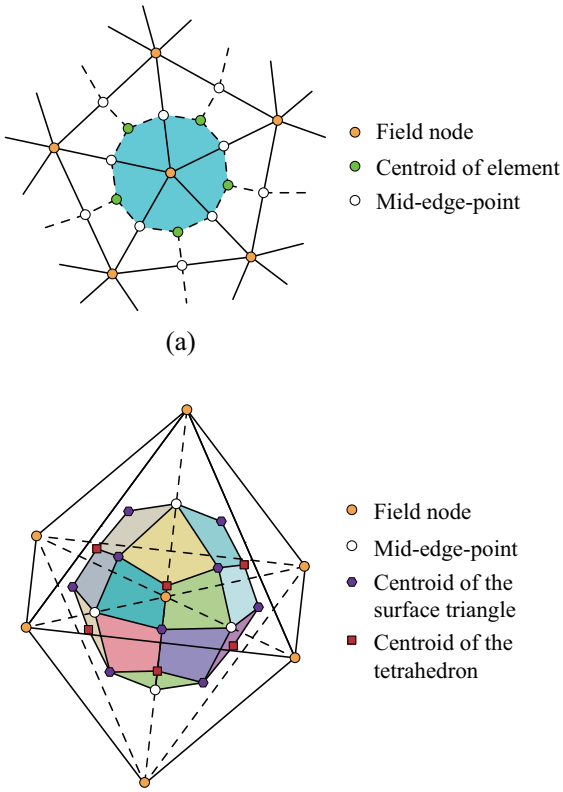


Figure 1: Smoothing cell construction (after [30]): (a) two-dimensional case; (b) three-dimensional case

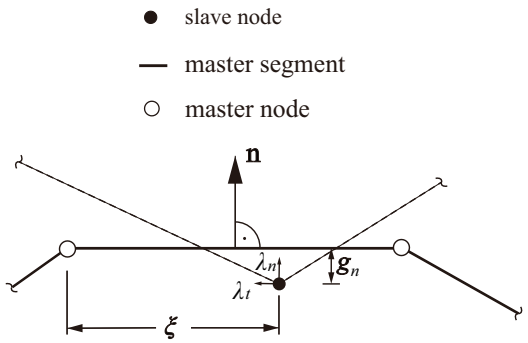


Figure 2: Illustration of the slave node to master segment contact scheme

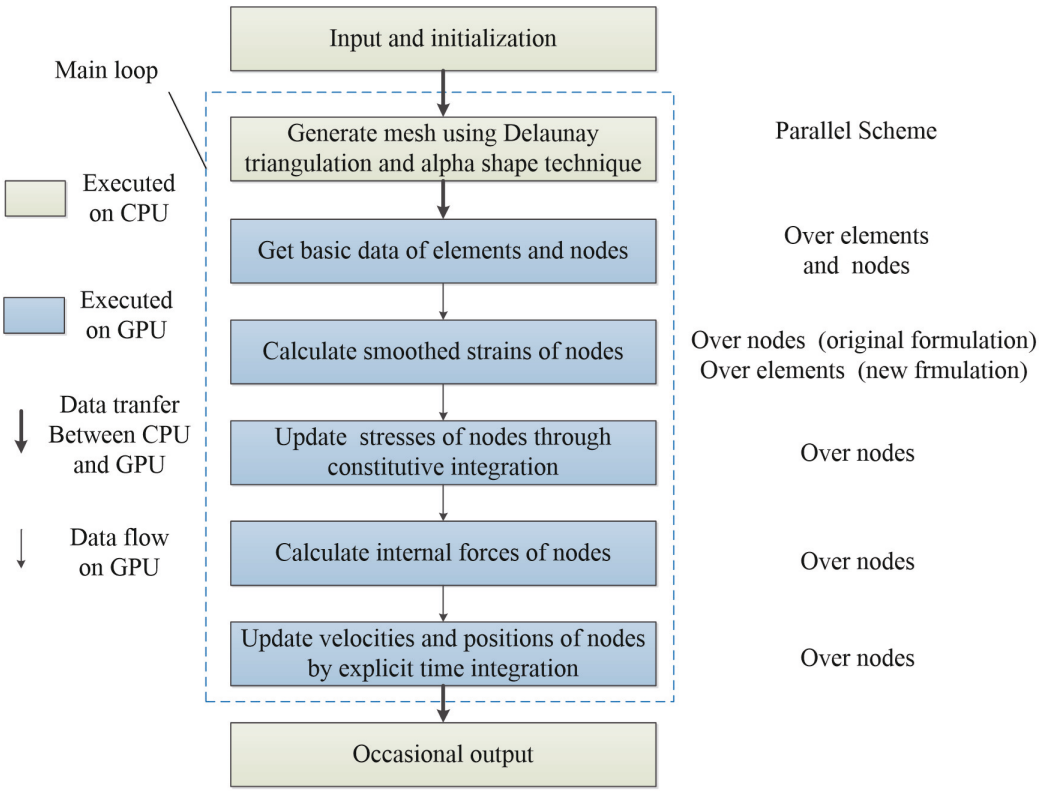
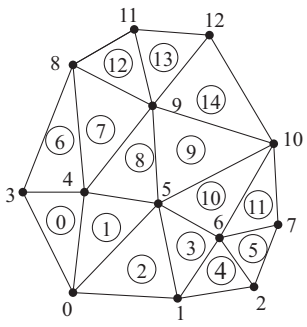


Figure 3: GPU acceleration scheme of SPFEM



Loop over the Elements (0) ~ (14)

Node	0	1	2	3	4	5	6	7	8	9	10	11	12
1st Related elements	(0)	(2)	(4)	(0)	(0)	(1)	(3)	(5)	(6)	(7)	(9)	(12)	(13)
2nd Related elements	(1)	(3)	(5)	(6)	(1)	(2)	(4)	(11)	(7)	(8)	(10)	(13)	(14)
3rd Related elements	(2)	(4)			(6)	(3)	(5)		(12)	(9)	(11)		
4th Related elements					(7)	(8)	(10)			(12)	(14)		
5th Related elements					(8)	(9)	(11)			(13)			
6th Related elements						(10)				(14)			
Total Number	3	3	2	2	5	6	5	2	3	6	4	2	2

Figure 4: Illustration of the algorithm to obtain the indices of elements related to nodes

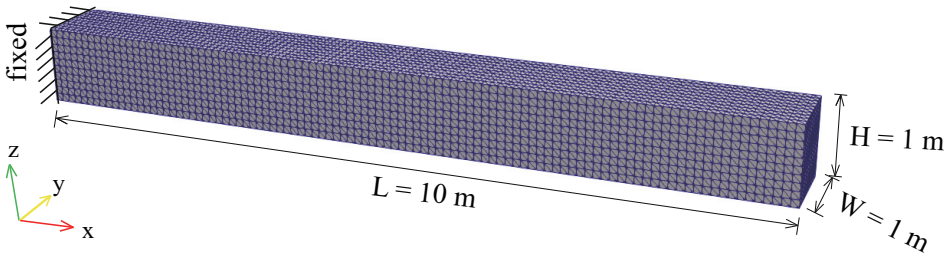


Figure 5: Problem description of the axial vibration of a continuum bar

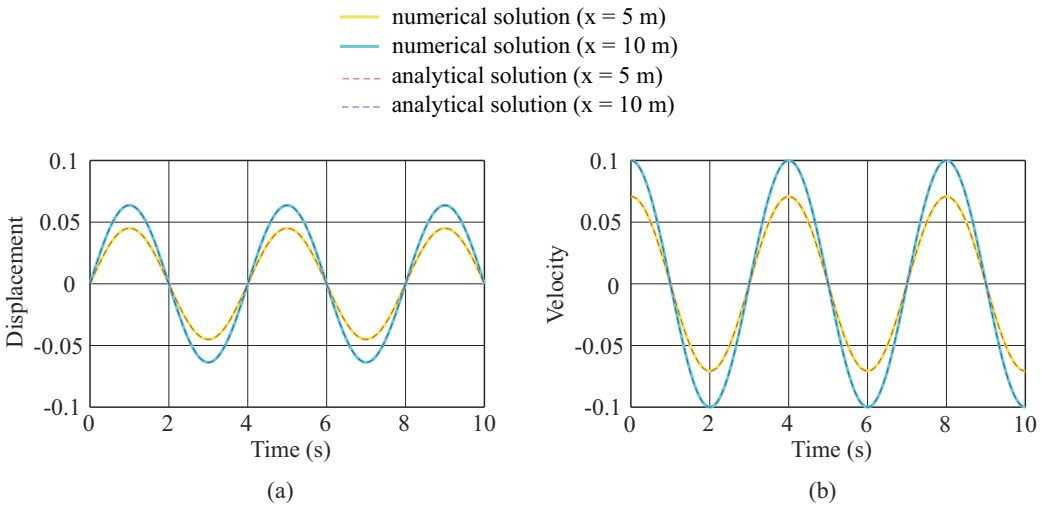


Figure 6: Comparison between the numerical results and the analytical results at the middle of the bar (x = 5 m) and at the end of the bar (x = 10 m)

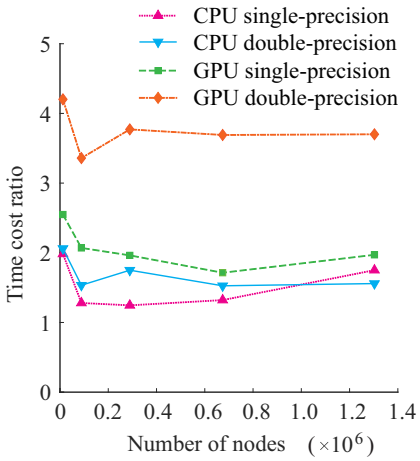


Figure 7: Ratio between the time cost with the original formnulations and that with the the new formnulations

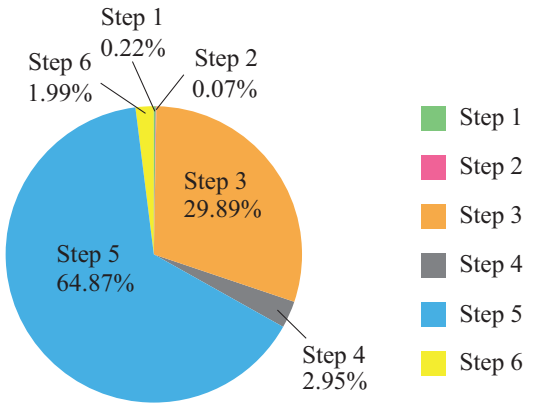


Figure 8: Distribution of workload across the steps in the double-precision CPU simulation

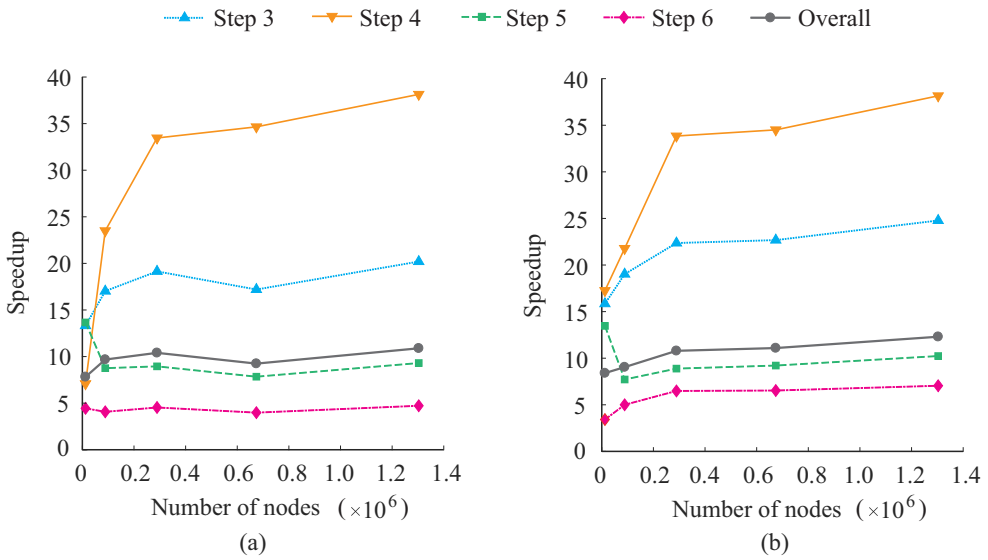


Figure 9: Speedup of the GPU simulations over the sequential CPU simulations: (a) single-precision simulations; (b) double-precision simulations

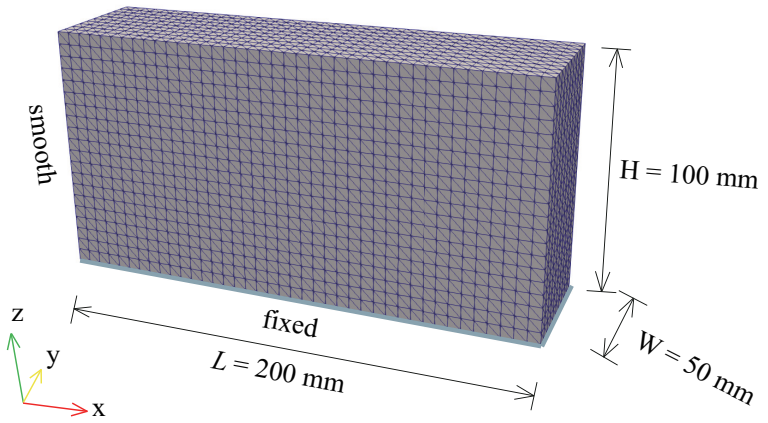


Figure 10: Computation mesh of the laboratory experiment of soil collapse

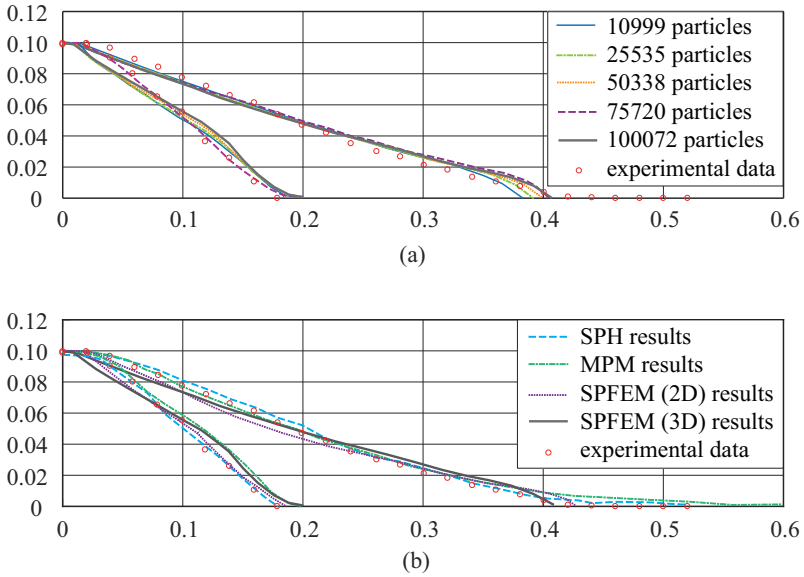
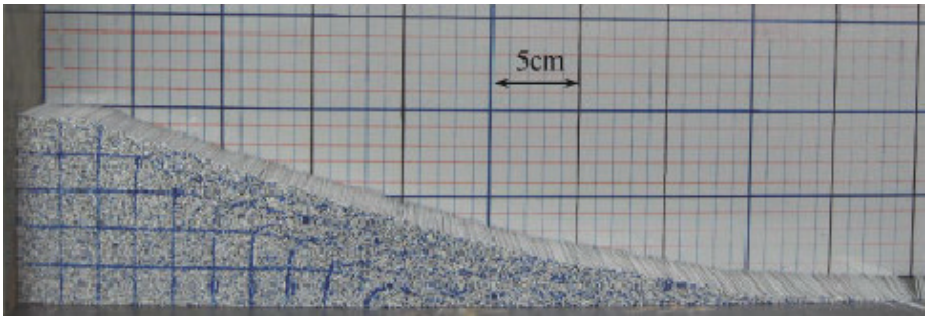
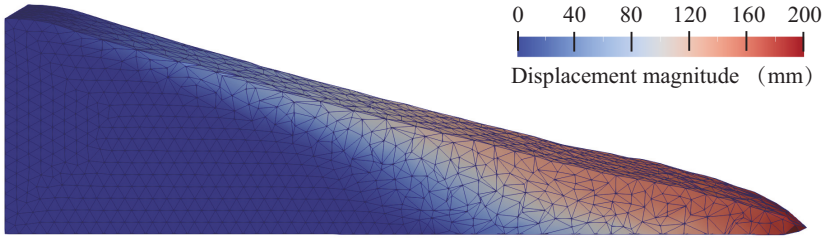


Figure 11: Final surface configurations after collapse: (a) Comparison between the numerical results and the experimental results; (b) Comparison of the numerical results with those obtained by others [3, 22, 56]



(a)



(b)

Figure 12: Comparison of the final configuration with the experimental result

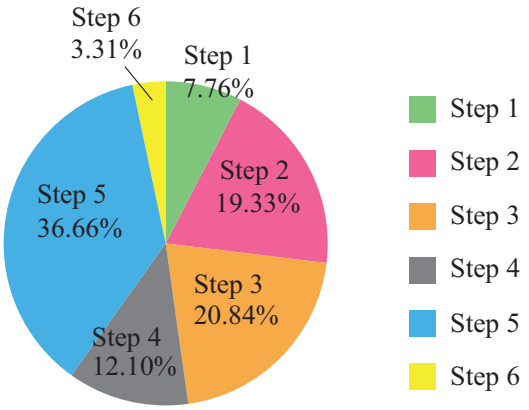


Figure 13: Distribution of workload across the steps in the double-precision CPU simulation

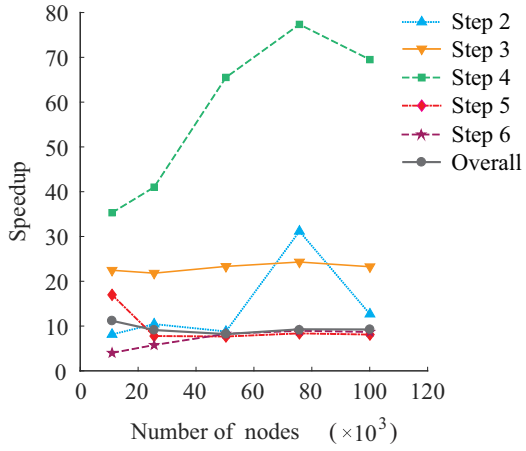


Figure 14: Speedup of the double-precision GPU simulations over sequential CPU simulations

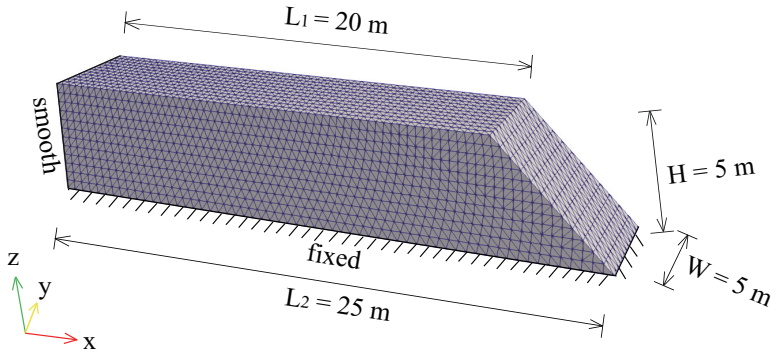


Figure 15: Problem description of the progressive failure of long strain-softening soil slope

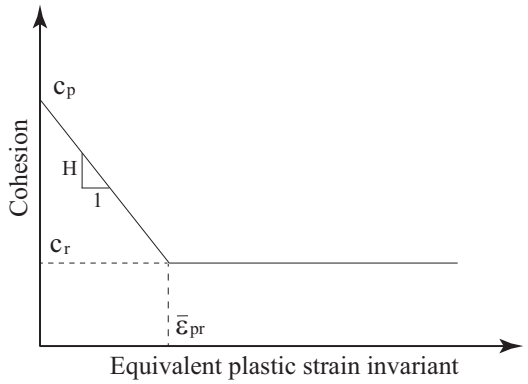


Figure 16: Consitutive model concept of the strain-softening soil

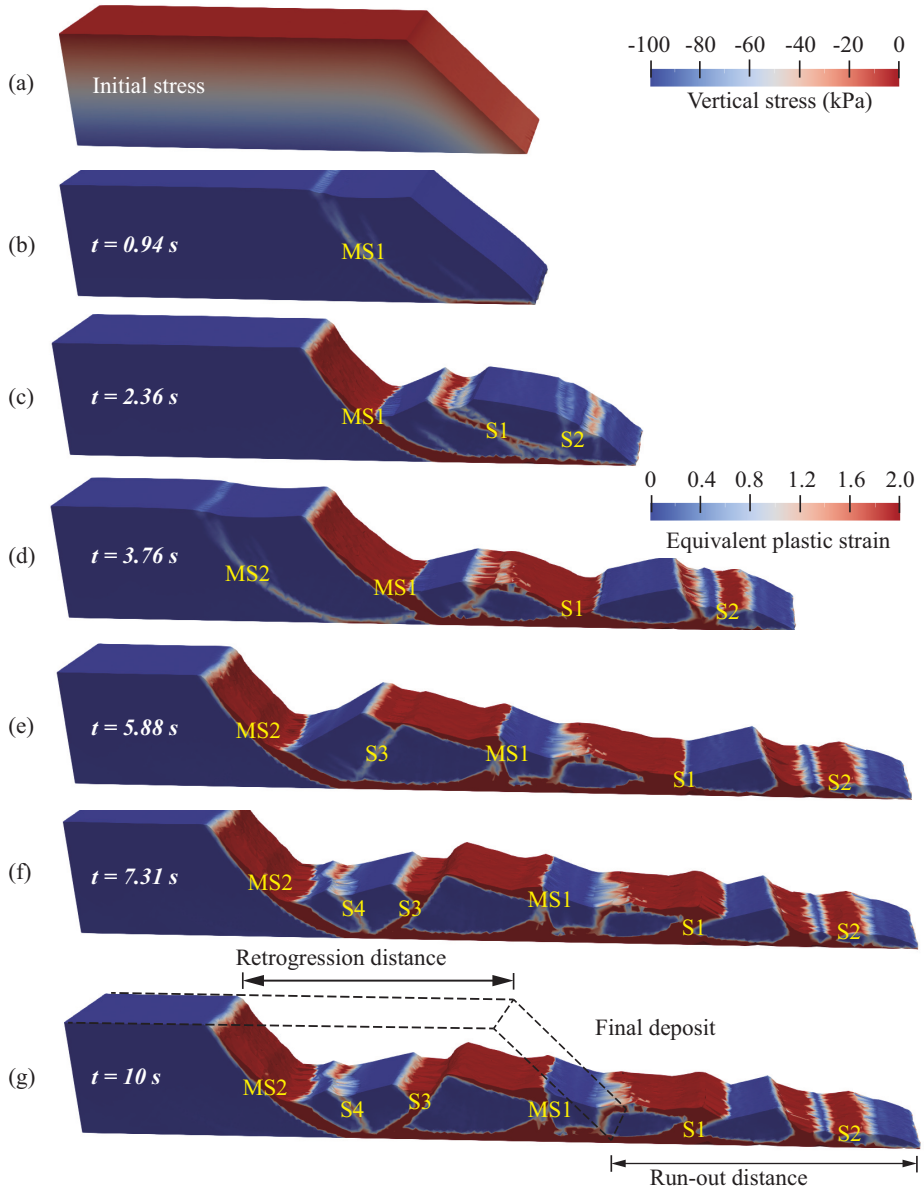


Figure 17: Numerical results of the whole progressive failure process of long strain-softening soil slope

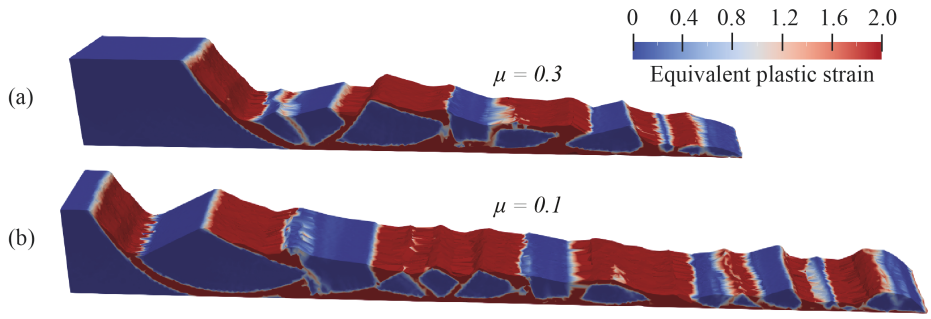


Figure 18: Final deposit of the slope for various friction coefficients: (a) $\mu = 0.3$; (b) $\mu = 0.1$