



ESPFEM2D: A MATLAB 2D explicit smoothed particle finite element method code for geotechnical large deformation analysis

Wei Zhang¹ · Yihui Liu¹ · Jinhui Li² · Weihai Yuan^{3,4}

Received: 7 September 2023 / Accepted: 17 December 2023 / Published online: 15 February 2024
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2024

Abstract

The Smoothed Particle Finite Element Method (SPFEM) has gained popularity as an effective numerical method for modelling geotechnical problems involving large deformations. To promote the research and application of SPFEM in geotechnical engineering, we present ESPFEM2D, an open-source two-dimensional SPFEM solver developed using MATLAB. ESPFEM2D discretizes the problem domain into computable particle clouds and generates the finite element mesh using Delaunay triangulation and the α -shape technique to resolve mesh distortion issues. Additionally, it incorporates a nodal integration technique based on strain smoothing, effectively eliminating defects associated with the state variable mapping after remeshing. Furthermore, the solver adopts a simple yet robust approach to prevent the rank-deficiency problem due to under-integration by using only nodes as integration points. The Drucker-Prager model is adopted to describe the soil's constitutive behavior as a demonstration. Implemented in MATLAB, this open-source solver ensures easy accessibility and readability for researchers interested in utilizing SPFEM. ESPFEM2D can be easily extended and effectively coupled with other existing codes, enabling its application to simulate a wide range of large geomechanical deformation problems. Through rigorous validation using four numerical examples, namely the oscillation of an elastic cantilever beam, non-cohesive soil collapse, cohesive soil collapse, and slope stability analysis, the accuracy, effectiveness and stability of this open-source solver have been thoroughly confirmed.

Keywords PFEM · Nodal integration · Large deformation · Remeshing · Open-source · Geotechnical engineering

1 Introduction

Numerous geotechnical engineering topics involve large deformations, such as landslides, avalanches, debris flows, cone penetration tests, pile installation, et al. With the development of geotechnical numerical analysis, the large deformation simulation has become increasingly popular in addressing these issues, garnering significant research attention in recent years. Effective numerical methods for large

deformation analysis must accurately capture the geometric changes in surface profiles and soil layers. However, traditional mesh-based numerical methods (e.g., finite element method (FEM), finite difference method (FDM)) are prone to mesh distortion during large deformation. Even if only a few meshes are severely distorted, the calculation will be interrupted.

To address the issue of mesh distortion in traditional FEM during large deformation, numerous alternative methods have been proposed. Some representative methods include Arbitrary Lagrangian–Eulerian (ALE), Coupled Eulerian–Lagrangian (CEL) and Remeshing and Interpolation Technique with Small Strain (RITSS). In ALE [1, 2], an excessively deformed mesh is substituted by another one with regularly-shaped elements and the state variables are mapped from the old mesh to the new one. This approach has been called ALE because the transfer of state variables from one mesh to another is perceived as a convection process similar to that in classical Eulerian analysis. CEL, which is a method very similar to ALE, has gained popularity in geotechnical

✉ Weihai Yuan
yuanwh@hhu.edu.cn

¹ College of Water Conservancy and Civil Engineering, South China Agricultural University, Guangzhou 510642, China

² Department of Civil and Environmental Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China

³ Department of Engineering Mechanics, Hohai University, Nanjing 210098, China

⁴ College of Mechanics and Materials, Hohai University, Nanjing 210098, China

analysis, especially due to its presence in the commercial codes Abaqus/Explicit [3, 4] and LS-DYNA [5]. The basis of CEL is that there are two domains, one Eulerian domain and one Lagrangian domain, that interact with each other via a suitable contact formulation. RITSS was first proposed by Hu and Randolph [6], which is based on implicit time integration for Lagrangian computation with mesh re-profiling techniques and state variable interpolation techniques to solve the mesh distortion problem.

Another class of numerical methods for solving large deformation problems is the particle-based methods, which discretize the geotechnical body into a series of particles that can freely move within the computational domain, thereby overcoming the mesh distortion problem encountered by traditional mesh-based methods during large deformation. Currently, the commonly used particle-based methods include Smoothed Particle Hydrodynamics (SPH), Material Point Method (MPM), and Particle Finite Element Method (PFEM). Initially proposed for solving astrophysical problems [7], SPH has found widespread applications in solving geotechnical problems [8–10]. Two key difficulties with SPH for solid modelling are the imposing of essential boundary conditions and the so-called tensile instability. MPM combines the advantages of particle-based and mesh-based methods by discretizing the continuum into material points. It preserves the state variable information on these material points while maintaining a fixed background mesh to solve the governing momentum equations [11–14]. However, MPM is prone to computational inaccuracies, particularly stress oscillations and inaccuracies [15]. PFEM was initially proposed by Oñate et al. [16–18] with the primary goal of solving fluid dynamics and fluid–structure interaction problems. In recent years, it has gained significant popularity for solid large deformation analysis [19–22]. PFEM treats the nodes in the FEM mesh as particles, and addresses mesh distortion during large deformation through frequent mesh re-discretization and state variable mapping. The updated Lagrangian FEM is utilized for solving the governing equation.

Among the three particle-based methods, PFEM inherits the sound theoretical basis of FEM while maintaining the flexibility of the particle-based method, and thus has been widely used in geotechnical large deformation analysis. However, PFEM performs numerical integration on Gaussian points while storing state variable information on particles. This necessitates frequent information transfer between Gaussian points and particles (i.e. nodes), adding complexity and introducing errors. To overcome this defect, a nodal integration technique based on strain smoothing has been introduced by the authors to improve PFEM, leading to a novel framework, known as Smooth Particle Finite Element Method (SPFEM) [23–27]. With the nodal integration technique, all state variables are calculated and stored on

particles. This approach simplifies the computational procedure, eliminates errors caused by mapping, and helps mitigate volume-locking issues.

To this end, it is important for geotechnical researchers who are already utilizing the aforementioned methods, as well as for new users looking to adopt them, to have access to an efficient open-source solver. ABAQUS software [3, 4] and LS-DYNA [5] provide options for implementing CEL, ALE, and SPH methods. However, the underlying codes are hidden from users, which hinders the promotion of applications and in-depth research on numerical methods. In terms of SPH, many open-source codes are available, such as GADGET [28, 29], GIZMO [30, 31], SPPhysics [32, 33], DualSPHysics [34, 35], and GPU SPH [36], et al. SPH open-source solvers specific to geotechnical problems have been developed as well, such as the 3D GPU-accelerated SPH open-source solver LOQUAT proposed by Peng et al. [37]. For open-source solvers of MPM, many codes are available, such as NairnMPM, Uintah and Cb-Geo [38]. Uintah is a massively parallel, multi-physics process simulation platform that implements MPM and can be used to simulate the behavior of solid materials [39]. The open-source solvers of PFEM are few in comparison with SPH and MPM. Kratos, which is an object-oriented framework of FEM, has implemented PFEM by linking with an external mesh generation library [40]. Guo and Yang [38] proposed NSPFEM2D, a 2D SPFEM open-source solver based on C++ and Python. However, the hybrid programming of C++ and Python is less readable and less easy to use for mechanical researchers in comparison to the MATLAB language. Herein, this paper introduces a MATLAB-based open-source solver for SPFEM named ESPFEM2D, aiming to enable interested researchers to more easily grasp the method and promote the research and application of SPFEM. ESPFEM2D employs an explicit time integration scheme and utilizes the Drucker-Prager model to describe the soil constitutive behavior as a demonstration. It can be easily extended and effectively coupled with other existing codes to simulate various geotechnical large deformation problems.

The following sections are organized as follows. Section 2 provides an introduction to the fundamental theory of explicit SPFEM. Section 3 elaborates on the implementation details of the ESPFEM2D code. In Sect. 4, the open-source solver is demonstrated through application examples, including simulations of the oscillation of an elastic cantilever beam, non-cohesive soil collapse, cohesive soil collapse, and slope stability analysis. These examples serve to verify the accuracy, validity, and robustness of the solver. Finally, a summary is presented in Sect. 5.

2 Theory of ESPFEM2D

2.1 Governing equations

The explicit SPFEM [25, 27] employs the same governing equations with the FEM. The linear momentum conservation of the continuum in the computational domain can be represented as

$$\rho \mathbf{a} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} \tag{1}$$

where ρ represents the material density, \mathbf{a} denotes the acceleration, $\boldsymbol{\sigma}$ represents the Cauchy stress tensor, and \mathbf{b} represents the specific body density acting on the object. By considering the principle of virtual displacement and the divergence theorem, the weak form of the governing equation can be expressed as

$$\int_{\Omega} \delta \mathbf{u} \cdot \rho \mathbf{a} d\Omega = \int_S \delta \mathbf{u} \cdot \boldsymbol{\tau} dS + \int_{\Omega} \delta \mathbf{u} \cdot \rho \mathbf{b} d\Omega - \int_{\Omega} \nabla \delta \mathbf{u} : \boldsymbol{\sigma} d\Omega \tag{2}$$

where \mathbf{u} represents the displacement vector, Ω denotes the configuration domain, S represents the boundary domain, and $\boldsymbol{\tau}$ is the prescribed traction force.

2.2 Spatial discretization

The SPFEM first discretizes the problem domain into a particle cloud and then generates a finite element mesh by combining Delaunay triangulation and α -shape techniques [41]. Furthermore, the nodal integration technique based on strain smoothing [42, 43] is used to solve Eq. (2).

As shown in Fig. 1, the problem domain Ω is divided into N_n strain-smoothed sub-cells associated with particle k such that $\Omega = \sum_{k=1}^{N_n} \Omega^k$ and $\Omega^i \cap \Omega^j = \emptyset, i \neq j$ where \emptyset denotes the empty set. Each smoothing sub-cell associated with particle k is created by sequentially connecting the midpoints of the edges of the triangular elements around particle k to their centroids. Thus, each triangular element is divided into three quadrilateral subdomains. Afterward, the strain tensor $\tilde{\boldsymbol{\epsilon}}_k$ associated with particle k is obtained as

$$\tilde{\boldsymbol{\epsilon}}_k = \sum_{I=1}^{N^k} \tilde{\mathbf{B}}_I(\mathbf{x}_k) \mathbf{u}_I \tag{3}$$

where I is the particle number, \mathbf{x}_k is the coordinate of particle k , N^k is the number of particles connected to particle k , \mathbf{u}_I is the displacement of particle I , and $\tilde{\mathbf{B}}$ is the smoothed strain matrix.

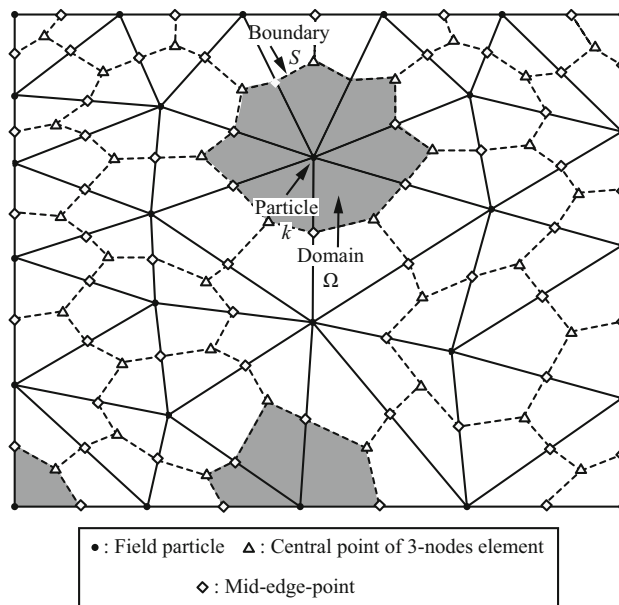


Fig. 1 Smoothing sub-cells associated with particles

For triangular elements, due to the constant element strain, the smooth strain matrix can be simply calculated as follows

$$\tilde{\mathbf{B}}_I(\mathbf{x}_k) = \frac{1}{A_k} \sum_{j=1}^{N_e^k} \frac{1}{3} A_e^j \mathbf{B}_j \tag{4}$$

where N_e^k is the number of elements around particle k , j is the element number, A_e^j and \mathbf{B}_j are respectively the area and strain gradient matrix of the j th triangular element around particle k , and A_k is the area of smoothing sub-cell associated with particle k which can be calculated by the following equation:

$$A_k = \int_{\Omega^k} d\Omega = \frac{1}{3} \sum_{j=1}^{N_e^k} A_e^j \tag{5}$$

Using the standard Galerkin method and performing the node-based discretization, the global discretized formulation of the weak form Eq. (2) can be expressed as

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

in which

$$\mathbf{F}^{ext} = \sum_{k=1}^{N_n} \int_S N_k \boldsymbol{\tau} dS + \sum_{k=1}^{N_n} \rho \mathbf{b}_k A_k \tag{7}$$

$$\mathbf{F}^{int} = \sum_{k=1}^{N_n} \tilde{\mathbf{B}}_k \boldsymbol{\sigma}_k A_k \tag{8}$$

$$\mathbf{M} = \sum_{k=1}^{N_n} \rho A_k \tag{9}$$

where \mathbf{F}^{ext} represents the external force, \mathbf{F}^{int} denotes the internal force, and \mathbf{M} represents the diagonal mass matrix.

2.3 Temporal discretization

The present SPFEM utilizes the explicit time integration scheme. The leapfrog approach is implemented, which is a second-order accuracy method [25]. Equation (10) is used to update the acceleration of particles at the n -th step, followed by Eqs. (11) and (12) applied to update the velocity and position of particles, i.e.

$$\mathbf{a}^n = \mathbf{M}^{-1} (\mathbf{F}^{ext} - \mathbf{F}^{int}) \tag{10}$$

$$\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^{n-\frac{1}{2}} + \Delta t \mathbf{a}^n \tag{11}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{v}^{n+\frac{1}{2}} \tag{12}$$

In addition, to simulate static problems with the dynamic relaxation method, a local damping force term is added to the system, which is calculated as

$$\mathbf{F}^\xi = -\xi \cdot \left| (\mathbf{F}^{ext} - \mathbf{F}^{int}) \right| \text{sign}(\mathbf{v}^{n-\frac{1}{2}}) \tag{13}$$

where ξ is the damping constant.

2.4 Constitutive model

To obtain a smooth yield surface that approximates the classic Mohr-Coulomb surface for numerical convenience, Drucker-Prager introduced the following yielding criterion,

$$f = \alpha I_1 + \sqrt{J_2} - k \tag{14}$$

where I_1 represents the first invariant of the stress tensor, J_2 denotes the second invariant of the deviatoric stress tensor, α and k represent the material constants, respectively. To align with the Mohr-Coulomb yield surface, three schemes can be adopted. In the first two schemes, we use

$$\alpha = \frac{2\sin\varphi}{\sqrt{3} (3\mp\sin\varphi)}, k = \frac{6c\cos\varphi}{\sqrt{3} (3\mp\sin\varphi)} \tag{15}$$

where c represents the cohesion and φ denotes the friction angle. The plus sign in Eq. (15) represents the Drucker-Prager yield surface passing through the inner apexes of the Mohr-Coulomb yield surface in the π -plane, while the minus sign represents the outer apexes. In the third scheme, we use

$$\alpha = \frac{\tan\varphi}{\sqrt{9 + 12\tan^2\varphi}}, k = \frac{3c}{\sqrt{9 + 12\tan^2\varphi}} \tag{16}$$

which indicates that the Drucker-Prager yield surface inscribes the Mohr-Coulomb yield surface in the π -plane.

The same plastic potential function as the yield function is used, which is given by

$$g = \sqrt{J_2} + \alpha_\phi I_1 \tag{17}$$

where α_ϕ represents the dilatancy coefficient, which is associated with the ratio of plastic volume change to plastic shear strain.

The classic return-mapping algorithm is used to implement stress point integration. Although more complicated soil constitutive models can be used, we use the Drucker-Prager elastic perfectly-plastic model for demonstration purposes. The extension to other constitutive models for the ESPFEM2D solver is straightforward.

2.5 Rigid boundary contact

A rigid boundary contact algorithm is employed to account for the soil interaction with a rigid boundary. As shown in Fig. 2, the interaction between soil particles and a rigid boundary must satisfy the following contact constraints, also known as Signorini conditions [44]:

$$\begin{aligned} g_n &\geq 0, \sigma_n \geq 0, \sigma_n g_n = 0 \\ |\sigma_t| - \mu \sigma_n &\leq 0 \end{aligned} \tag{18}$$

where g_n represents the gap between the particle and the boundary, σ_n denotes the normal contact pressure, σ_t represents the tangential stress, and μ is the friction coefficient.

A prediction-correction algorithm is employed to ensure the contact constraint. First, the positions and velocities of all particles are predicted assuming that no contact occurs. Then, contact is activated when the gap between particles and line segments is less than zero. It is assumed that the normal contact force can completely suppress the normal penetration velocity of the particles during the time step, resulting in:

$$\mathbf{f}_I^n = -\frac{m_I v_I^n}{\Delta t} \mathbf{e}_n \tag{19}$$

where m_I represents the mass of particle I , $v_I^n (= \mathbf{v}_I \cdot \mathbf{e}_n)$ denotes the normal projection of the relative velocity between the particle and the rigid boundary, which is negative upon contact, Δt is the time step, and \mathbf{e}_n represents the normal vector of the rigid boundary. Similarly, the tangential contact force \mathbf{f}_I^t is calculated by assuming viscous conditions as follows:

$$\mathbf{f}_I^t = -\frac{m_I (\mathbf{v}_I - v_I^n \cdot \mathbf{e}_n)}{\Delta t} \tag{20}$$

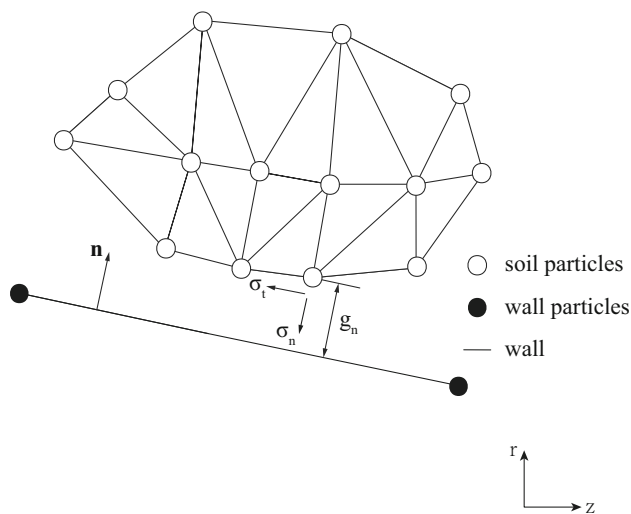


Fig. 2 The contact between a deformable body and a rigid surface

The tangential force is also subject to Coulomb friction, which sets a limit $|f_t^t| \leq \mu |f_n^t|$ to the magnitude of the tangential force, where μ is the friction coefficient. It should be pointed out that the current prediction-correction method does not involve additional penalty parameters and thus does not affect the time step.

After determining the contact force, the corrected velocity of the particle is calculated as follows:

$$\bar{v}_I = v_I + \Delta t \frac{f_n^c + f_t^c}{m_I} \tag{21}$$

where v_I represents the predicted velocity of particle I .

2.6 Remeshing

The mesh reconstruction technique is a crucial aspect of SPFEM to prevent mesh distortion during large deformations. In SPFEM, the mesh reconstruction technique combines the Delaunay triangulation technique with the α -shape algorithm, forming a two-step mesh reconstruction algorithm [45]. First, the Delaunay triangulation is performed based on the particle cloud to generate a convex domain and reconstruct the triangular mesh. Then, the α -shape algorithm is employed to calculate the radius of the sphere for each triangular element and delete the triangular elements with a radius greater than αh where α is a predefined parameter generally ranging from 1.2~1.6 and h is the characteristic spacing of particles. As a result, the remaining triangular elements correspond to the computational domain [16].

In SPFEM, due to the nodal integration technique based on strain smoothing, more distorted meshes can be utilized without significant loss of accuracy. Thus, the requirements

for mesh quality are not as stringent as in the original PFEM. However, a mesh smoothing technique is still necessary to enhance the mesh quality and to increase the critical time step. Specifically, the Laplacian smoothing technique [46, 47] is utilized when a few particles are too close to their adjacent particles. Although other mesh smoothing techniques (e.g. Meduri et al. [48], Vartziotis et al. [49]) can also be employed, we use the Laplacian smoothing technique due to its simplicity and high efficiency.

2.7 Rank-deficiency treatment

Despite the success of the nodal integration technique in many applications, in some cases, the results may be not reliable due to spurious low-energy instability. This instability associated with the under-integrating using only nodes is also known as rank-deficiency or rank-instability in the literature [23, 50–57].

To circumvent this instability associated with nodal integration, we proposed a simple yet robust coping control approach [58], which has also been implemented in ESPFEM2D. The key idea of the approach is to ensure that all the particles in the smoothing element follow a single linear velocity field by adding correction forces. The correction force on particle i due to all neighbors j in the smoothing cell is computed as

$$f_{ij} = \sum_{l=1}^{N_k} \frac{\beta}{2 \|x_{ij}^t\|} \left(E_i \Omega_i \frac{\delta_{ij}^i}{\|x_{ij}^t\|} + E_j \Omega_j \frac{\delta_{ij}^j}{\|x_{ij}^t\|} \right) n_{ij}^{t+\Delta t} \tag{22}$$

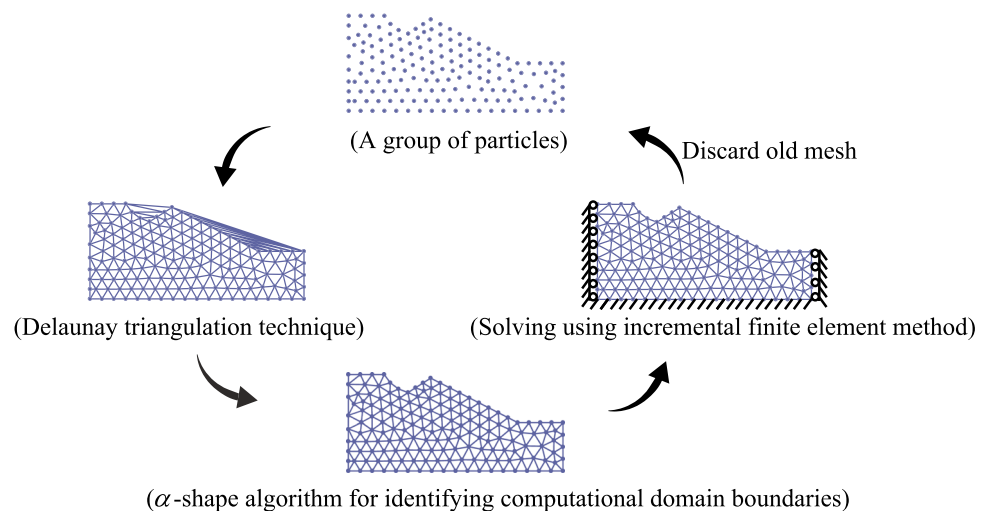
where β represents the artificial control parameter ranging from 0.1 to 1.0, $\|x_{ij}^t\|$ represents the actual distance vector of the deformed coordinates at time t , E_i and E_j are Young’s modulus of smoothing cells i and j respectively, Ω_i and Ω_j represent the area of smoothing cells i and j respectively, δ_{ij}^i represents the projected length of the error vector, $n_{ij}^{t+\Delta t}$ represents the unit normal of distance vector $x_{ij}^{t+\Delta t}$ which can be calculated as $n_{ij}^{t+\Delta t} = x_{ij}^{t+\Delta t} / \|x_{ij}^t\|$. Readers can refer to Yuan et al. [58] for a detailed description.

3 Description of the open-source solver ESPFEM2D

3.1 Overview

The primary procedures of PFEM for a typical calculation step is shown in Fig. 3. First, we discretize the region into a set of nodes/particles. Then, we obtain the FEM mesh by consequently using the Delaunay triangulation technique and

Fig. 3 Primary procedures of PFEM for a typical calculation step



the α -shape algorithm. With the FEM mesh generated, we solve the governing equation using the incremental FEM. The nodal coordinates of the FEM mesh are then updated according to the incremental displacement results. If the FEM mesh is too distorted, we discard the old mesh and generate a new one based on the set of nodes/particles.

The code workflow diagram of the open-source solver ESPFEM2D is shown in Fig. 4. All the 10 steps in the workflow diagram are implemented in the form of MATLAB functions except that the MATLAB script named *SPFEM.m* is used as the main program. The MATLAB function names corresponding to all the 10 computational steps are shown in Fig. 5.

The installation of the ESPFEM2D can be done by simply adding a path of `\ESPFEM2D` directory to the MATLAB search path. The practical usage of ESPFEM2D can be simply realized by running the main MATLAB script *SPFEM.m*.

To facilitate the users, we provide 4 demo examples, i.e., all the validation and verification examples in Sect. 4. A total of 7 simulations are available for these 4 examples, as two simulation stages are required for the last 3 examples. Each simulation has a separate ID. By changing the variable `iEX` in *SPFEM.m*, the simulation ID is selected and the corresponding pre-processing for the simulation is implemented in the function `input_data.m`. These demonstration examples can be used as the reference for the ESPFEM2D users to create new geotechnical large deformation simulation applications.

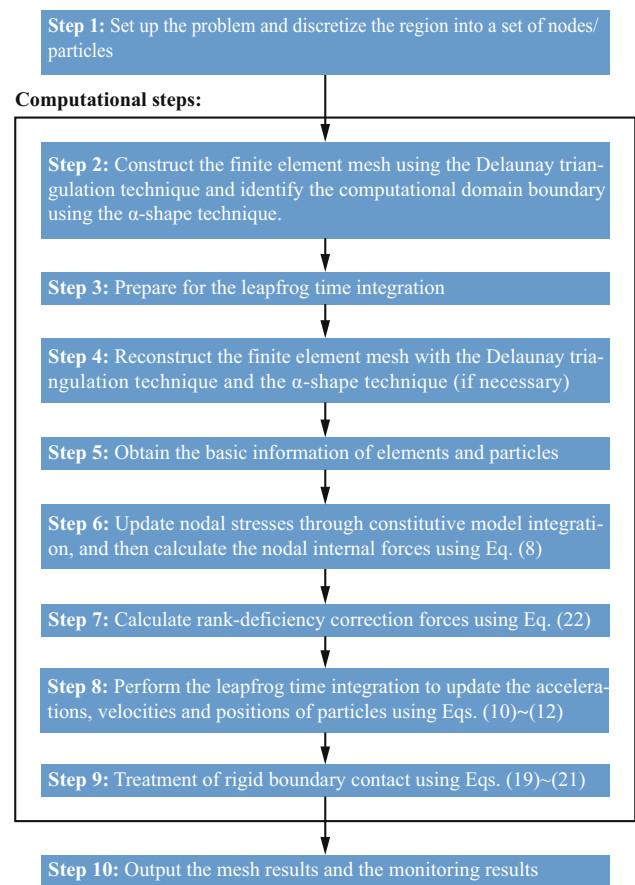


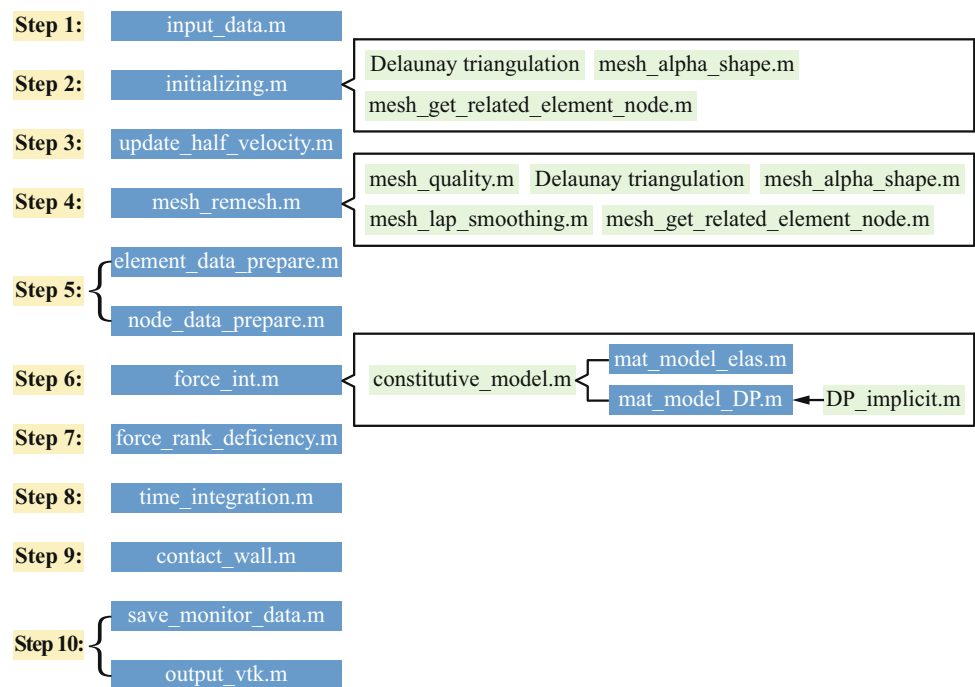
Fig. 4 Code workflow diagram of the open-source solver ESPFEM2D

3.2 The ESPFEM2D repository

The MATLAB open-source solver ESPFEM2D is freely available from the repository: <https://github.com/WeiZhang-2023/ESPFEM2D>. The structure of the repository, illustrated in Fig. 6, is described in this section.

The directory ESPFEM2D comprises four subdirectories of the comprehensive open-source solver ESPFEM2D, and the main script *SPFEM.m*, which can be executed to initiate the simulation.

Fig. 5 MATLAB function names corresponding to all the computational steps



The directory `comp` contains the MATLAB functions associated with the core computational procedure of the explicit SPFEM, including Step 1, Step 10, and Steps 4~9 in Fig. 4.

The directory `constitutive_model` contains the MATLAB functions associated with the numerical implementation of constitutive models. The linear elastic constitutive model and the Druker-Prager elastic perfect-plastic model are available. New constitutive models can be implemented by simply adding a new MATLAB function similar to the MATLAB function `mat_model_DP.m` (see Fig. 5).

The directory `mesh` contains the MATLAB functions associated with the mesh generation and remeshing, including the Delaunay triangulation and α -shape techniques, the calculation of element quality, the nodal integration technique based on strain smoothing, and the retrieval of related particle and element data. It should be pointed out that in PFEM, it is beneficial to add new nodes in those areas we are interested in. However, we choose to use a simple yet robust remeshing algorithm in the open-source code ESPFEM2D for demonstration purposes. With the basic remeshing algorithm mastered, the readers can realize complex remeshing algorithms (including adding nodes, deleting nodes, mesh optimization, Yuan et al. [59, 60]) by modifying the MATLAB codes in the directory `mesh`.

The directory `example` includes the MATLAB functions associated with the pre-processing for simulations. A MATLAB function corresponds to a simulation, including the particle distribution, the material model and parameters, the

definition of the boundary conditions, the initial condition settings, and the monitoring and output settings.

The directory `documentation` contains a `.pdf` file that provides a systematic tutorial on setting up a basic problem, along with explanations of code functions and variable definitions.

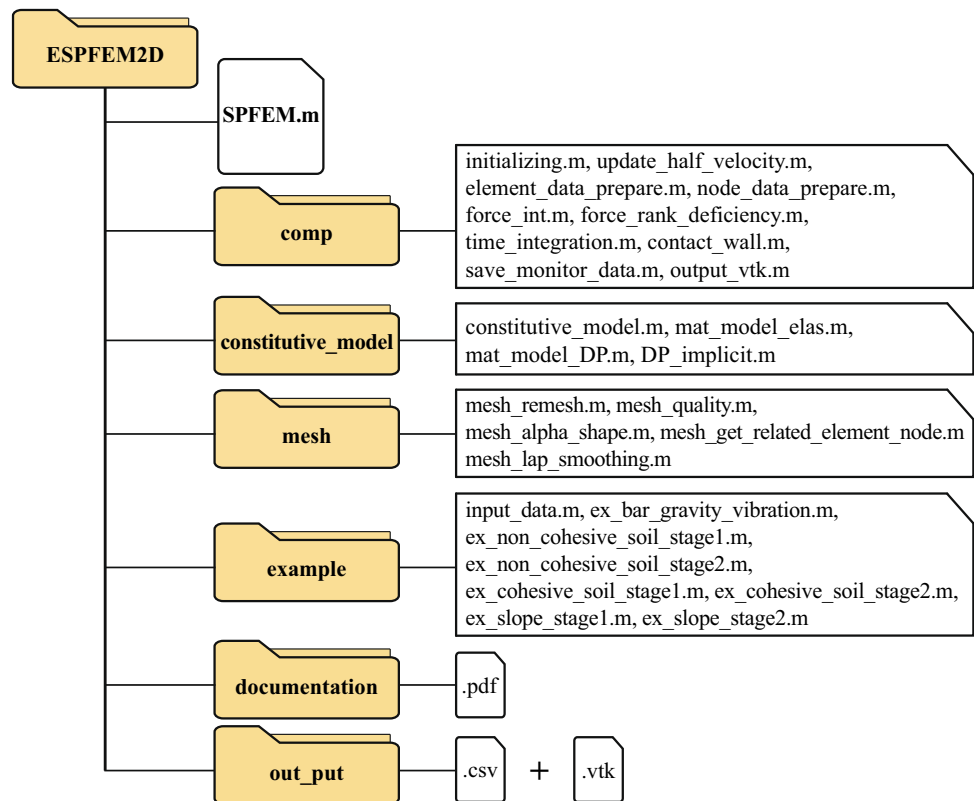
The directory `out_put` is used to save computational result files. `.vtk` files are created to save the field information intermittently, while a `.csv` file is created to monitor the evolution of the interested results for particular particles.

3.3 Variables

The primary variables in ESPFEM2D are introduced as follows:

- `iEX`: a separate ID of each simulation.
- `nCoor`: array of dimension `par-> node_cnt×2` containing the `x` - and `y` - coordinates of each node/ particle.
- `eNode`: array of dimension `par-> element_cnt×3` containing three node/particle IDs corresponding to each element.
- `nMat`: array of dimension `par-> node_cnt×1` containing the type of constitutive model for each node/particle.
- `nAccel`: array of dimension `par-> node_cnt×2` containing the accelerations in the `x` - and `y` - directions of each node/particle.
- `nVel`: array of dimension `par-> node_cnt×2` containing the velocities in the `x` - and `y` - directions of each node/particle.

Fig. 6 Structure of the open-source solver ESPFEM2D repository



- $nDisp$: array of dimension $par \rightarrow node_cnt \times 2$ containing the total displacements in the x - and y - directions of each node/particle.
- $nDdisp$: array of dimension $par \rightarrow node_cnt \times 2$ containing the incremental displacements in the x - and y - directions of each node/particle.
- $nStress$: array of dimension $par \rightarrow node_cnt \times 4$ containing the stresses of each node/particle.
- $nPstrain_eq$: array of dimension $par \rightarrow node_cnt \times 1$ containing the equivalent plastic strain of each node/particle.
- $nBoundT$: array of dimension $par \rightarrow node_cnt \times 2$ specifying the boundary condition type in the x - and y - directions of each node/particle.
- $nBoundV$: array of dimension $par \rightarrow node_cnt \times 2$ containing the value of boundary conditions in the x - and y - directions of each node/particle.
- $nREN$: array of dimension $par \rightarrow node_cnt \times 1$ containing the number of elements around each node/particle.
- $nREs$: array of dimension $par \rightarrow node_cnt \times par \rightarrow MN$ containing the element IDs around each node/particle.
- $nRNN$: array of dimension $par \rightarrow node_cnt \times 1$ containing the number of nodes/particles around each node/particle.
- $nRNs$: array of dimension $par \rightarrow node_cnt \times par \rightarrow MN \cdot 3$ containing the node/particle IDs around each node/particle.
- $nFrd$: array of dimension $par \rightarrow node_cnt \times 2$ containing the rank-deficiency correction forces in the x - and y - direction for each node/particle.
- $nFint$: array of dimension $par \rightarrow node_cnt \times 2$ containing the internal forces in the x - and y - directions of each node/particle.
- $eArea$: array of dimension $par \rightarrow element_cnt \times 1$ containing the area of each element.
- $eDNdx$, $eDNdy$: array of dimension $par \rightarrow element_cnt \times 3$ containing the partial derivatives of the shape function concerning x and y for the three nodes/particles corresponding to each element, respectively.
- $nArea$: array of dimension $par \rightarrow node_cnt \times 1$ containing the area of each strain smoothing cell associated with node/particle.
- $nDNdx$, $nDNdy$: array of dimension $par \rightarrow element_cnt \times par \rightarrow MN \cdot 3$ containing the partial derivatives of the shape function concerning x and y for each node/particle, respectively.

3.4 Functions

The primary functions in ESPFEM2D are summarized as follows:

- `input_data.m`: set up a problem.
- `initializing.m`: prepare for the calculation process of ESPFEM2D.
- `mesh_alpha_shape.m`: identify the computational domain boundary using the α -shape technique (see Sect. 2.6).
- `mesh_get_related_element_node.m`: get related particle and element information (see Sect. 2.6).
- `mesh_remesh.m`: reconstruct the finite element mesh (see Sect. 2.6).
- `mesh_quality.m`: get the mesh quality (see Sect. 2.6).
- `element_data_prepare.m`: get element information (see Sect. 2.2).
- `node_data_prepare.m`: get node/particle information (see Sect. 2.2).
- `mesh_lap_smoothing.m`: perform Laplacian smoothing for the mesh (see Sect. 2.2).
- `constitutive_model.m`: select constitutive model (see Sect. 2.4).
- `mat_model_elas.m`: implement the elasticity constitutive model (see Sect. 2.4).
- `mat_model_DP.m`: implement the Drucker-Prager constitutive model (see Sect. 2.4).
- `force_int.m`: calculate the nodal internal forces (see Sect. 2.2).
- `update_half_velocity.m`: prepare for the leapfrog time integration (see Sect. 2.3).
- `force_rank_deficiency.m`: calculate the rank-deficiency correction forces (see Sect. 2.7).
- `time_integration.m`: perform time integration (see Sect. 2.3).
- `contact_wall.m`: treatment of rigid boundary contact (see Sect. 2.5).
- `output_vtk.m`: output the field results.
- `save_monitor_data.m`: output the monitoring results.

3.5 Main program

This section concentrates on the essential component of the code, `SPFEM.m`, which orchestrates and executes all functions required for simulation. Listing 1 presents the MATLAB code snippet contained within `SPFEM.m`. The code workflow diagram for the open-source solver ESPFEM2D shown in Fig. 4 can be better understood by reading the codes in `SPFEM.m`.

Listing 1: MATLAB code in `SPFEM.m`

```

1  %example input
2  iEX=1;
3  [par, monitor, nCoor,nMat,nAccel,nVel,nStress,
   nBoundT,nBoundV]...
4  =input_data(iEX);
5  [par,eNode,nREN,nRES,nRNN,nRNs,nDisp,
   nPstrain_eq,nF,nFrd]=initializing(par,nCoor);
6  par.ctime=0.;
7  par.istep=0;
8  %prepare for the leapfrog time integration
9  [nVel, nDisp, nCoor]=update_half_velocity(par,
   nBoundT,nBoundV,nVel,nAccel,nDisp,nCoor
   );
10 while (par.ctime < par.totaltime + 1.e-9)
11   %remeshing
12   if par.if_remesh==1
13     [par,nCoor,nMat,nBoundT,nBoundV,nREN,
      nRES,nRNN,nRNs,nDisp,nVel,nStress,
      nPstrain_eq,eNode]...
14     =mesh_remesh(par,nCoor,nMat,
      nBoundT,nBoundV,nREN,nRES,
      nRNN,nRNs,nDisp,nVel,nStress,
      nPstrain_eq,eNode);
15   end
16   %obtain the basic information of elements and
      nodes
17   [eArea,eDNdx,eDNdy]=element_data_prepare(
      nCoor,par,eNode);
18   [nArea,nDNdx,nDNdy]=node_data_prepare(par,
      nREN,nRES,nRNN,nRNs,eNode,eArea,
      eDNdx,eDNdy);
19   %calculate the nodal internal forces
20   [nFint,nStress,nPstrain_eq,nF]=...
21   force_int(par,nRNN,nRNs,nArea,nMat,
      nDNdx,nDNdy,nVel,nStress,
      nPstrain_eq);
22   %rank deficiency treatment
23   if (par.rank_deficiency>1.e-6)
24     nFrd=force_rank_deficiency(par,nRNN,
      nRNs,nMat,nArea,nCoor,nVel,nF);
25   end
26   %time integration
27   [nVel,nDisp,nCoor]=time_integration(par,nMat,
      nArea,nBoundT,nBoundV,nVel,nDisp,
      nCoor,nFint,nFrd);
28   %rigid boundary contact
29   if par.rigid_wall>0.5
30     [par,nVel,nDisp,nCoor] = contact_wall(par,
      nCoor,nMat,nArea,nVel,nDisp);
31   end
32   par.ctime=par.ctime+par.dtime;

```

```

33   par.istep=par.istep+1;
34   end

```

4 Verification and demonstration examples

To verify the accuracy, effectiveness, and stability of the open-source solver ESPFEM2D, four numerical examples are given in this section, i.e., the oscillation of an elastic cantilever beam, non-cohesive soil collapse under gravity loading, cohesive soil collapse, and the failure of Mohr-Coulomb soil slope. All these examples verify the accuracy and effectiveness of the ESPFEM2D. Moreover, with the first example, we primarily aim to demonstrate the effectiveness of the rank-deficiency treatment technique to maintain numerical stability. Meanwhile, we compare the computational efficiency between the ESPFEM2D and an existing SPFEM open-source code.

4.1 Oscillation of an elastic cantilever beam

An undamped elastic cantilever beam experiencing vibrations induced by a sudden application of gravitational acceleration is considered. The problem's geometry and model setup are depicted in Fig. 7. The cantilever beam measures 2.0 m in length and 0.2 m in height, with its left end fixed and other edges of the beam free. The material parameters are as follows: Young's modulus $E = 80.0$ MPa, Poisson's ratio $\nu = 0.25$, and density $\rho = 1850$ kg/m³. We investigate the problem using a total time of $t = 4.0$ s and a constant time step of $\Delta t = 1 \times 10^{-5}$ s. The domain is discretized into 1111 particles. The rank-deficiency correction parameter β in Eq. (22) is taken to be 0.0 and 0.1 sequentially to solve this problem, corresponding to SPFEM without rank-deficiency correction and SPFEM with rank-deficiency correction, respectively. The results obtained by the FEM [61] are used as a reference.

The final stress distribution is depicted in Fig. 8. It is evident that the SPFEM with rank-deficiency correction exhibits a smoother stress field compared to the SPFEM without rank-deficiency correction, effectively mitigating stress distribution oscillations (e.g., inhomogeneity of stress distribution) caused by rank-instability associated with the nodal integration technique [50–57]. Figures 9 and 10 display the vertical displacements and velocities to time at the end of the cantilever beam. Figure 11 displays the evolution of the axial stress at the fixed end. Remarkably, the results obtained through the SPFEM with rank-deficiency treatment show excellent agreement with the corresponding FEM results [61]. In contrast, the SPFEM without rank-deficiency treatment demonstrates a gradual deviation, which clearly demonstrates the rank-instability. It is clear that

Table 1 Efficiency comparison between ESPFEM2D and NSPFEM2D

Number of nodes/particles	Average time cost per step (s)	
	ESPFEM2D	NSPFEM2D
205	0.0046	0.0024
306	0.0077	0.0036
729	0.0140	0.0132
1111	0.0227	0.0302
2737	0.0421	0.1312
6526	0.1030	0.6584
10593	0.1672	1.6704

the rank-instability issue is mitigated and smoother stress and displacement fields can be obtained by employing the SPFEM with rank-deficiency treatment.

We note that in terms of explicit SPFEM, an open-source code NSPFEM2D [38], which is built using hybrid C++ and Python programming, is also available to the public. It would be interesting to compare the computational efficiency between the NSPFEM2D and the ESPFEM2D presented in this paper. For the same problem, a total of 7 simulations with different numbers of nodes/particles are performed using NSPFEM2D and ESPFEM2D respectively, using a personal computer with an Intel(R) Core(TM) i5-8500 CPU, the main frequency of which is 3.00 GHz. The computational time costs are compared in Table 1. The results clearly show that when the number of nodes exceeds 729, the computational efficiency of ESPFEM2D outperforms that of NSPFEM2D. When the number of nodes is relatively large, the efficiency advantage is more apparent. The chief reason may be that in NSPFEM2D the computational procedure is driven by Python which runs much slower than MATLAB.

4.2 Non-cohesive soil collapse

The following examples involve geotechnical applications. The collapse of non-cohesive soil is considered first. The problem setup and geometric model involve a rectangular soil column with dimensions of 0.2 m in width and 0.1 m in height, as illustrated in Fig. 12. The boundary conditions consist of a fixed constraint at the bottom and a normal constraint on the left side. The corresponding physical experiment has been conducted by Nguyen et al. [62], and their results can serve as a reference for comparison in this analysis. The soil material properties are as follows: Young's modulus $E = 5.84$ MPa, Poisson's ratio $\nu = 0.3$, density $\rho = 26.5$ kN/m³, friction angle $\varphi = 19.8^\circ$, dilation angle $\Phi = 0^\circ$, and cohesion $c = 0$ kPa. We investigate the problem using a total time of $t = 0.6$ s and a constant time step of $\Delta t = 1 \times 10^{-6}$ s. The domain is discretized into 5894 particles.

Fig. 7 Oscillation of an elastic cantilever beam: setup and geometry

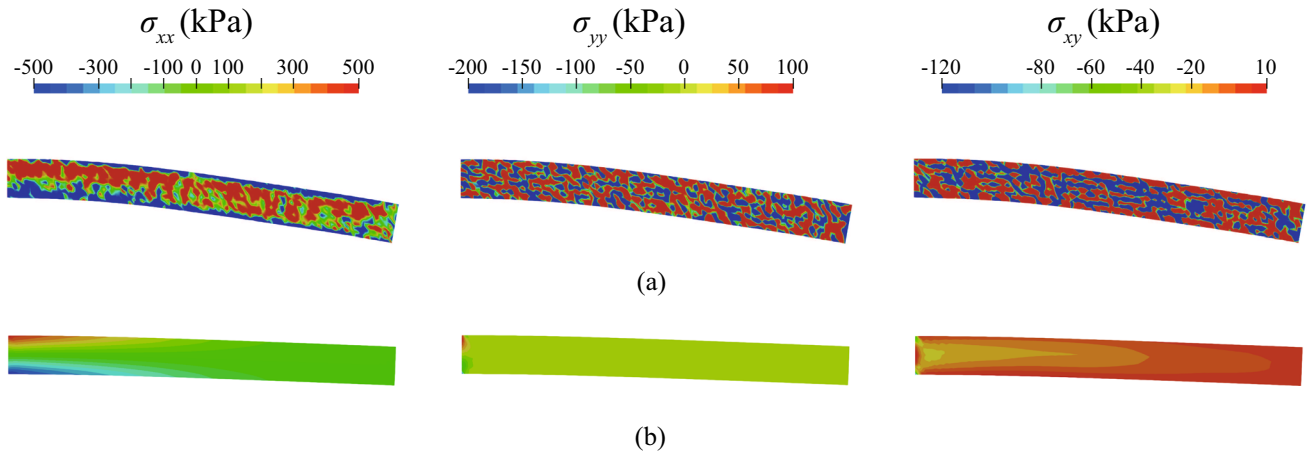
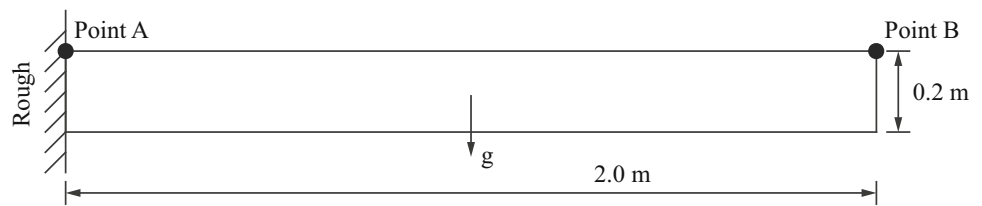
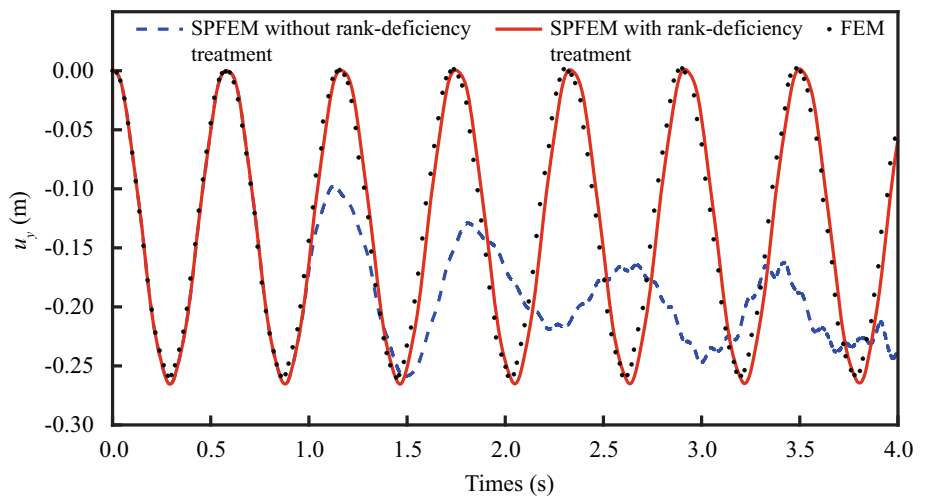


Fig. 8 Stress distributions at $t = 4$ s. **a** SPFEM without rank-deficiency treatment. **b** SPFEM with rank-deficiency treatment

Fig. 9 Vertical displacement of point B (see Fig. 7) on the end of the cantilever as a function of time



The simulation results of the non-cohesive soil collapse process using ESPFEM2D are illustrated in Fig. 13, which shows the distribution of equivalent plastic strain during non-cohesive soil collapse at different time steps, and compares the simulation results with experimental observations from the literature [62]. The simulation results obtained from ESPFEM2D are in good agreement with the results reported by Nguyen et al. [62] at different time steps. Under the influence of gravity, the right end of the non-cohesive soil column initiates the collapse process, with deformation starting in the top region while a portion remains undisturbed. As the collapse progresses, the undisturbed area gradually decreases,

and the granular material flows and undergoes deformation, forming a slope-like deposit that propagates from the lower right corner toward the top. Throughout the collapse process, a stationary region is observed in the lower left corner of the soil column, with the surface of this region defined as the slip line. The damage state of the non-cohesive soil column in Fig. 13 shows that the slip line obtained from the ESPFEM2D simulation aligns with the experimental observations.

Fig. 10 Vertical velocity of point B (see Fig. 7) on the end of the cantilever as a function of time

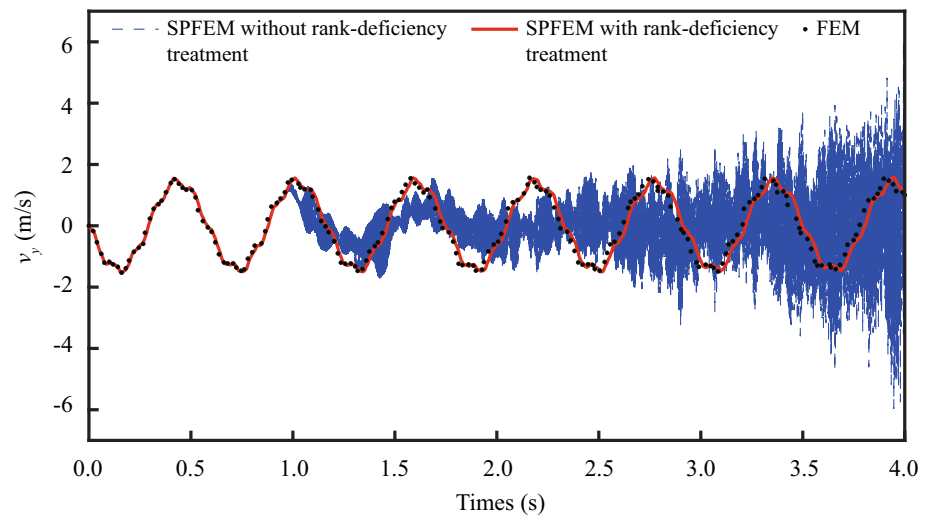


Fig. 11 Stress σ_{xx} history of point A (see Fig. 7)

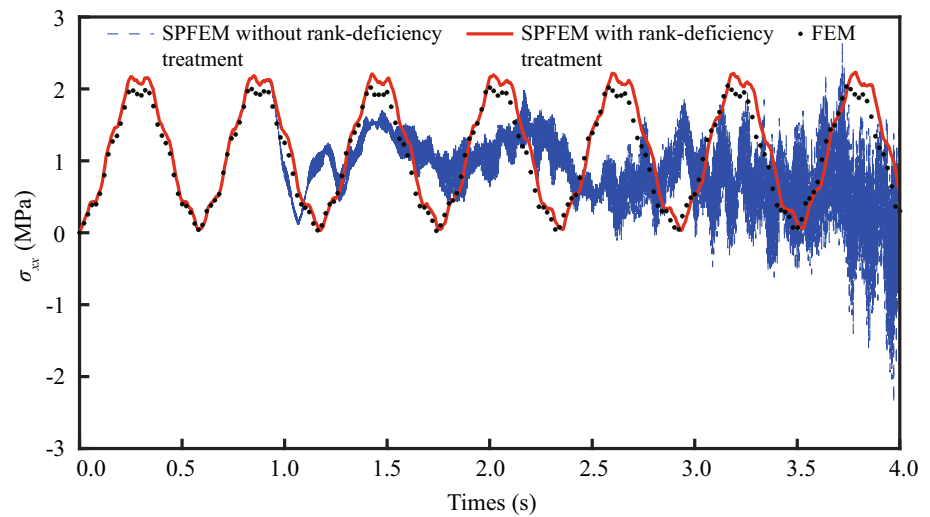
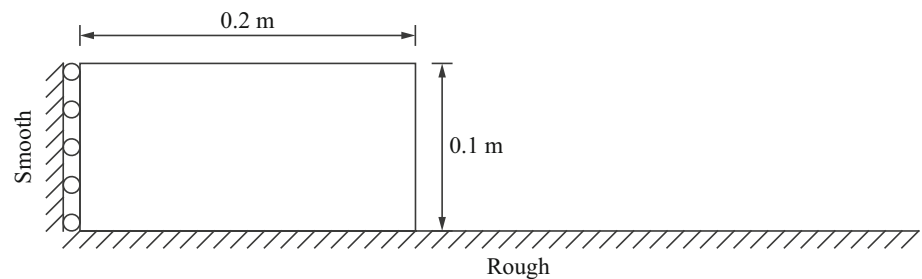


Fig. 12 Non-cohesive soil collapse: setup and geometry of the analysis



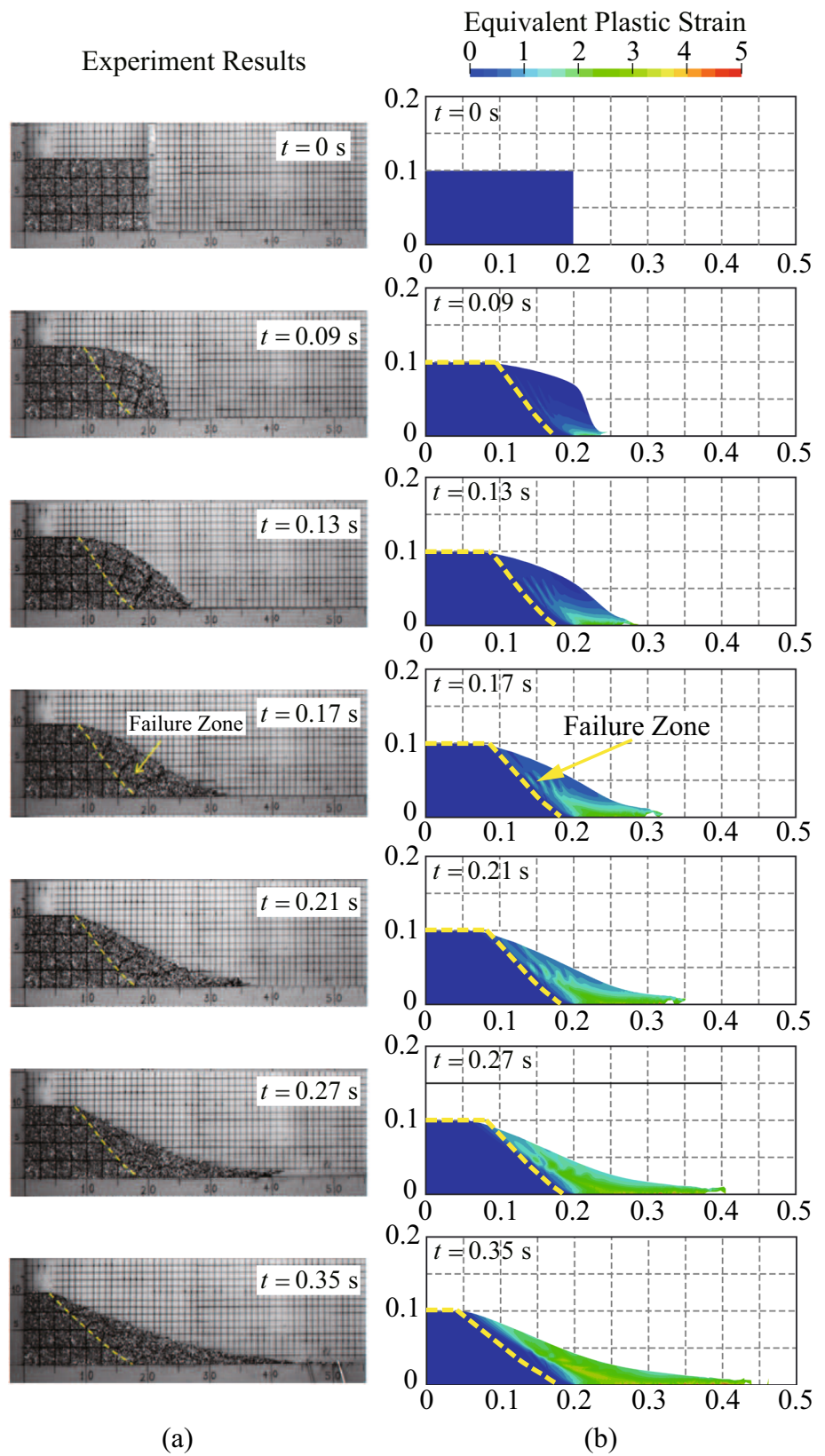
4.3 Cohesive soil collapse

The cohesive soil collapse problem was originally presented by Bui et al. [9] and has been previously investigated as a benchmark example by some researchers [63, 64]. The problem setup and geometric model are illustrated in Fig. 14. The rectangular domain has dimensions of height (H) = 2 m and width (L) = 4 m. The material parameters, as reported by Chalk et al. [63], are as follows: Young's modulus $E = 1.8$ MPa, Poisson's ratio $\nu = 0.2$, density $\rho = 1850$ kg/m³, fric-

tion angle $\varphi = 25^\circ$, dilation angle $\Phi = 0^\circ$, and cohesion $c = 5$ kPa. The boundary conditions include complete fixation at the left and lower surfaces, while the front and upper surfaces remain free. We investigate the problem using a total time of $t = 2.5$ s and a constant time step of $\Delta t = 5 \times 10^{-5}$ s. The domain is discretized as 23,256 particles.

The simulation results are compared with the SPH simulation results from the literature [63]. Figure 15 illustrates the distribution of equivalent plastic strains during the progressive destruction process of cohesive soil at different

Fig. 13 Comparison of equivalent plastic strain between the experiment results (left) and the simulation results obtained from ESPFEM2D (right)



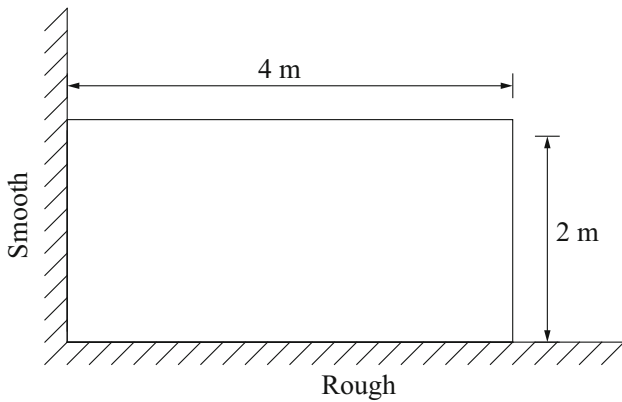


Fig. 14 Cohesive soil collapse: setup and geometry of the analysis

time steps. Throughout the progressive damage process, the first appearance of equivalent plastic strain is observed at the lower right corner of the granular column, and the destruction of the granular column gradually unfolds as the irreversible deformation region propagates through the spec-

imen, forming a shear zone. The damage initiates from the upper right corner and progressively develops, resulting in the formation of a triangular damage region. From Fig. 15, it can be observed that the simulation results obtained from ESPFEM2D generally agree with those reported by Chalk et al. [63].

4.4 Failure of a Mohr–Coulomb soil slope

The problem setup and geometric model for the failure of a Mohr–Coulomb soil slope are presented in Fig. 16. The soil material properties are as follows: Young’s modulus $E = 100$ MPa, Poisson’s ratio $\nu = 0.3$, density $\rho = 2000$ kg/m³, friction angle $\varphi = 20^\circ$, dilation angle $\Phi = 0^\circ$, and cohesion $c = 10$ kPa. The discretization of the domain consists of 7,912 particles.

ESPFEM2D is utilized to simulate the whole failure process of the slope, including both pre-failure and post-failure stages. The strength reduction method is used to obtain the safety factor of the slope, in which the shear strength parameters are decreases gradually with increasing strength

Fig. 15 Comparison of equivalent plastic strain between the SPH simulation results (left) and the simulation results obtained from ESPFEM2D (right)

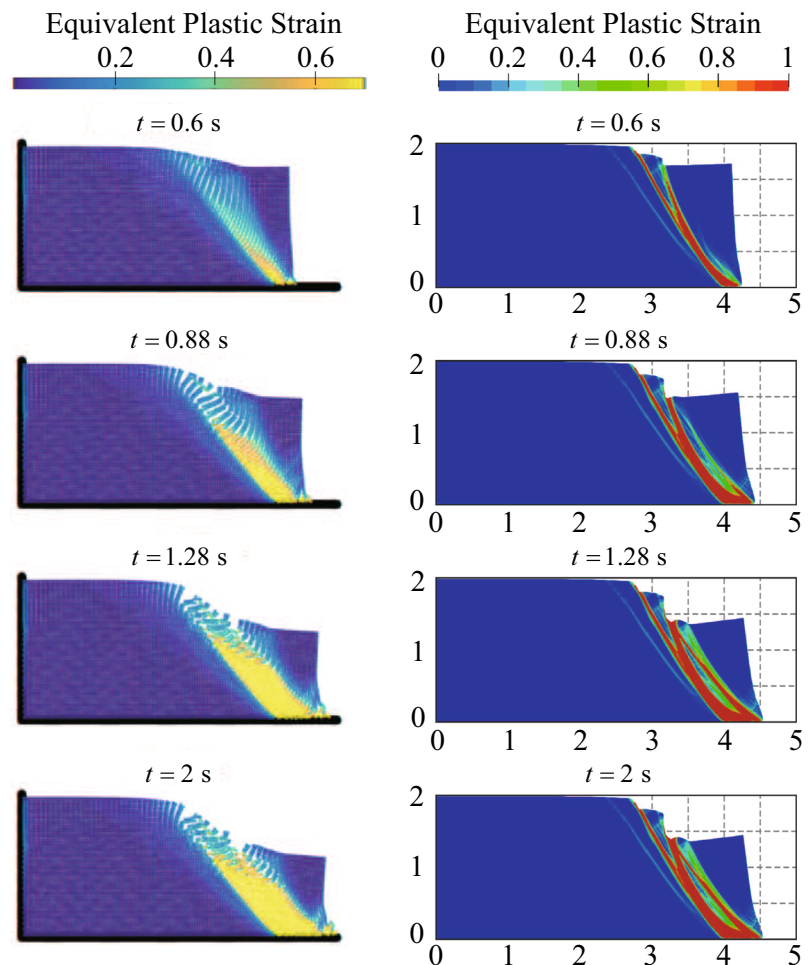


Fig. 16 Failure of a Mohr–Coulomb soil slope: setup and geometry of the analysis

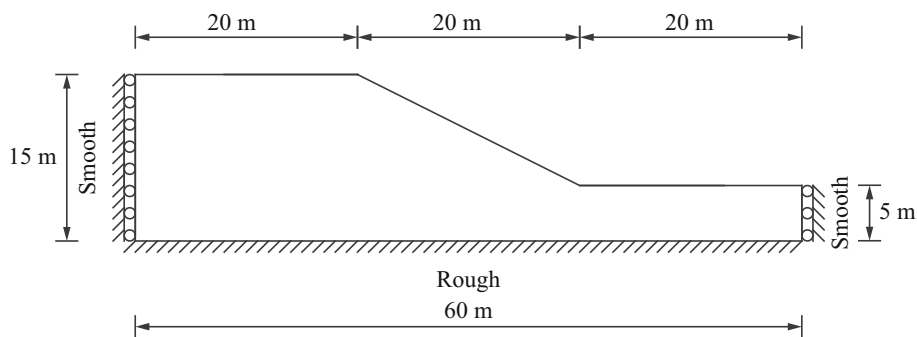


Fig. 17 Maximum displacement with different values of SRF

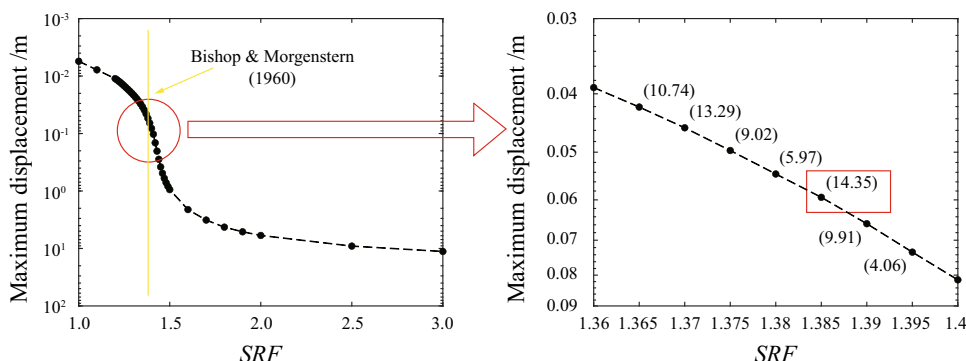
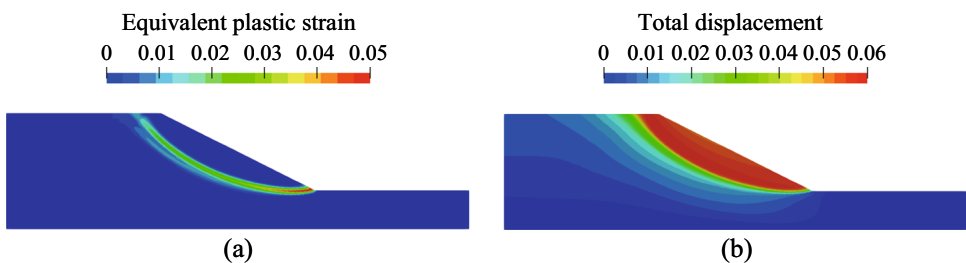


Fig. 18 Mohr–Coulomb slope with SRF=1.385: **a** Equivalent plastic strain distribution; **b** Total displacement distribution



reduction factor (SRF) as follows:

$$\begin{aligned} \varphi_f &= \arctan(\tan\varphi/\text{SRF}) \\ c_f &= c/\text{SRF} \end{aligned} \tag{23}$$

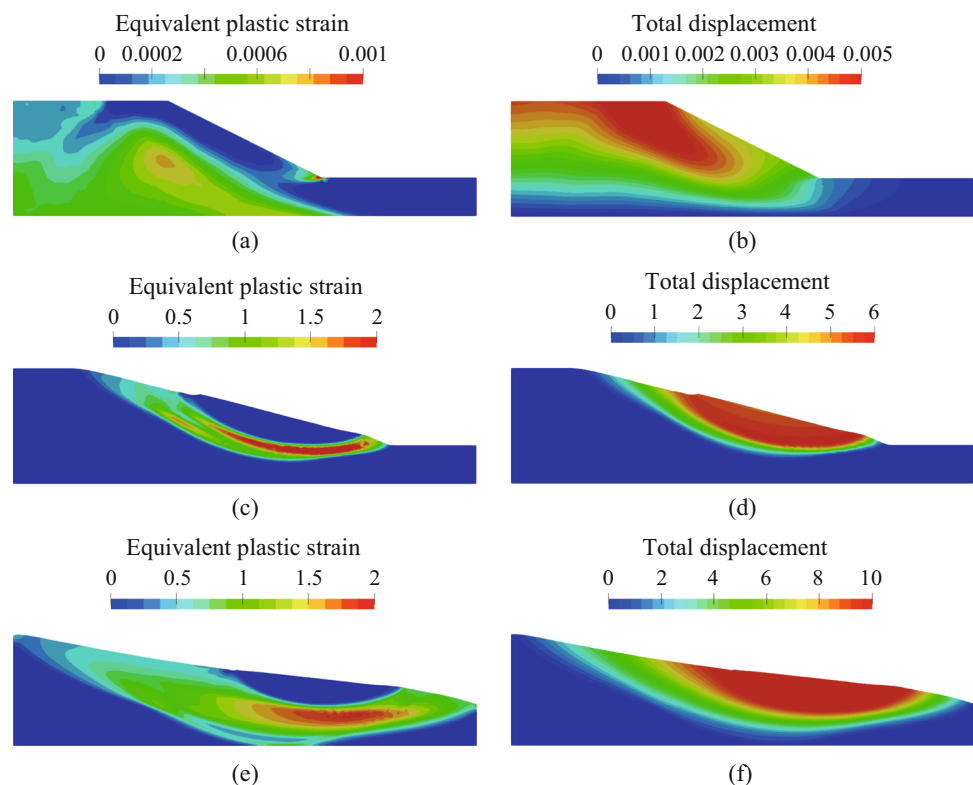
where φ_f and c_f are the friction angle and cohesion used in the simulation while φ and c are the soil friction angle and cohesion. The factor of safety (FOS) is defined as the minimum SRF value inducing slope failure. The whole simulation process is divided into two phases: In the first phase, the initial stress field is generated by applying gravity load. At the end of this phase, the displacements of all the particles are discarded with the stress state preserved. Since the ESPFEM2D uses an explicit time integration scheme, the dynamic relaxation method is employed in this phase to obtain the quasi-static solution with a local damping factor of 0.7. In the second phase, the slope failure process is simulated using the strength reduction technique, with a total time of $t = 5.0$ s and a constant time step $\Delta t = 1 \times 10^{-4}$ s. The

total time of 5.0 s is found to be large enough to engender large deformation if the slope is unstable.

The maximum nodal displacements with different SRFs are depicted in Fig. 17. By examining the criterion of slope factor of safety based on the abrupt increase in nodal displacements (the maximum curvature radius are used here to identify the abrupt increase and all the curvature radius with different SRFs are indicated in Fig. 17b by numbers in brackets), it is evident that the FOS of the slope is 1.385. Figure 18a illustrates that the plastic zone has fully penetrated the slope when $\text{SRF} = 1.385$, and the displacement distribution in Fig. 18b reveals the sliding zone associated with slope failure. The obtained factor of safety of 1.385 agrees with Bishop and Morgenstern’s definition of 1.38 with the limit equilibrium method in 1960 [65].

It should be pointed out that, thanks to the large deformation simulation capability of the ESPFEM2D, problems with a large range of SRF values can be successfully solved, including both stable and unstable cases. Figure 19 illustrates the final plastic zone and displacement distributions

Fig. 19 Mohr–Coulomb slope: **a, c and e** Equivalent plastic strain distribution with SRF=1.0, 2.0 and 3.0; **b, d and f** Total displacement distribution with SRF=1.0, 2.0 and 3.0



for the simulation with $SRF = 1.0, 2.0$ and 3.0 . Note that for the traditional strength reduction method for slope stability analysis [66], no convergent result can be obtained when the SRF value exceeds the safety factor of the slope. As shown in Fig. 19e, a plastic zone that fully penetrates the slope with $SRF = 3.0$ is successfully captured with the ESPFEM2D. Meanwhile, an intuitive sliding can be observed in Fig. 19f. It is clear that by combining the present ESPFEM2D with the strength reduction method, an in-depth slope stability analysis can be performed.

5 Conclusions

We propose ESPFEM2D, an open-source solver that implements a two-dimensional explicit Smooth Particle Finite Element Method (SPFEM), aiming to facilitate the understanding and application of SPFEM in geotechnical engineering. The solver incorporates all the state-of-art techniques for explicit SPFEM, addressing challenges related to mesh distortion, state variable mapping, volume locking and rank-instability.

ESPFEM2D utilizes explicit time integration and the Drucker-Prager constitutive model to describe soil behavior for demonstration purposes. The solver's performance is validated through a series of numerical examples, ensuring its accuracy, validity, and stability. ESPFEM2D, being an

open-source solver based on MATLAB programming, offers advantages such as generality, simplicity, and accessibility, making it convenient for researchers to grasp the methodology and promote the application of SPFEM in geotechnical engineering.

Acknowledgements The research is supported by the National Natural Science Foundation of China (Nos. 52379101 and 41807223), the Key Projects of the National Natural Science Foundation of China (No. 52239008), the Provincial Major Scientific Research Project of General Universities of Guangdong Province (No. 2022KTSCX013), the Youth Innovation Promotion Association CAS (No. 2022379).

References

1. Benson DJ (1989) An efficient, accurate and simple ALE method for nonlinear finite element programs. *Comput Methods Appl Mech Engrg* 72:305–350
2. Ghosh S, Kikuchi N (1991) An arbitrary Lagrangian-Eulerian finite element method for large deformation analysis of elastic-viscoplastic solids. *Comput Methods Appl Mech Engrg* 86:27–188
3. Bao YD, Sun XH, Zhou X, Zhang YS, Liu YW (2021) Some numerical approaches for landslide river blocking: introduction, simulation, and discussion. *Landslides* 18(12):3907–3922
4. Yang ZX, Gao YY, Jardine RJ, Guo WB, Wang D (2020) Large deformation finite-element simulation of displacement-pile installation experiments in sand. *J Geotech Geoenviron Eng* 146(6):04020044
5. Ren GF, Wang YX, Tang YQ, Zhao QX, Qiu ZG, Luo WH, Ye ZL (2022) Research on lateral bearing behavior of spliced helical piles with the SPH method. *Appl Sci* 12(16):8215

6. Hu Y, Randolph MF (1998) A practical numerical approach for large deformation problems in soil. *Int J Numer Anal Methods Geomech* 22:327–350
7. Gingold RA, Monaghan JJ (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon Not R Astron Soc* 181(3):375–389
8. Blanc T, Pastor M (2012) A stabilized fractional step, Runge–Kutta Taylor SPH algorithm for coupled problems in geomechanics. *Comput Methods Appl Mech Engrg* 221:41–53
9. Bui HH, Fukagawa R, Sako K, Ohno S (2008) Lagrangian meshfree particles method (SPH) for large deformation and failure flows of geomaterial using elastic-plastic soil constitutive mode. *Int J Numer Anal Methods Geomech* 32(12):1537–1570
10. Peng C, Wu W, Yu HS, Wang C (2015) A SPH approach for large deformation analysis with hypoplastic constitutive model. *Acta Geotech* 10(6):703–717
11. Soga K, Alonso E, Yerro A, Kumar K, Bandara S (2016) Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. *Géotechnique* 66(3):248–273
12. Sulsky D, Chen Z, Schreyer HL (1994) A particle method for history-dependent materials. *Comput Methods Appl Mech Engrg* 118:179–196
13. Yuan WH, Zhen HG, Zheng XC, Wang B, Zhang W (2023) An improved semi-implicit material point method for simulating large deformation problems in saturated geomaterials. *Comput Geotech* 161:105614
14. Zhang W, Wu ZZ, Peng C, Li S, Dong YK, Yuan WH (2023) Modelling large-scale landslide using a GPU-accelerated 3D MPM with an efficient terrain contact algorithm. *Comput Geotech* 158:105411
15. González Acosta JL, Vardon PJ, Remmerswaal G, Hicks MA (2020) An investigation of stress inaccuracies and proposed solution in the material point method. *Comput Mech* 65(2):555–581
16. Idelsohn SR, Oñate E, Del Pin F (2004) The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *Int J Numer Methods Eng* 61(7):964–989
17. Idelsohn SR, Oñate E, Del Pin F, Calvo N (2006) Fluid-structure interaction using the particle finite element method. *Comput Methods Appl Mech Engrg* 195(17–18):2100–2123
18. 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
19. Monforte L, Arroyo M, Carbonell JM, Gens A (2017) Numerical simulation of undrained insertion problems in geotechnical engineering with the particle finite element method (PFEM). *Comput Geotech* 82:144–156
20. Rodríguez JM, Carbonell JM, Cante JC, Oliver J (2017) Continuous chip formation in metal cutting processes using the particle finite element method (PFEM). *Int J Solids Struct* 120:81–102
21. Zhang X, Krabbenhoft K, Pedrosa DM, Lyamin AV, Sheng D, da Silva MV, Wang D (2013) Particle finite element analysis of large deformation and granular flow problems. *Comput Geotech* 54:133–142
22. Yuan WH, Zhang W, Dai BB, Wang Y (2019) Application of the particle finite element method for large deformation consolidation analysis. *Eng Comput* 36(9):3138–3163
23. Yuan WH, Liu M, Guo N, Dai BB, Zhang W, Wang Y (2023) A temporal stable smoothed particle finite element method for large deformation problems in geomechanics. *Comput Geotech* 156:105298
24. Yuan WH, Liu K, Zhang W, Dai BB, Wang Y (2020) Dynamic modeling of large deformation slope failure using smoothed particle finite element method. *Landslides* 17(7):1591–1603
25. Yuan WH, Wang B, Zhang W, Jiang J, Feng XT (2019) Development of an explicit smoothed particle finite element method for geotechnical applications. *Comput Geotech* 106:42–51
26. Zhang W, Yuan WH, Dai BB (2018) Smoothed particle finite-element method for large-deformation problems in geomechanics. *Int J Geomech* 18(4):04018010
27. 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
28. Springel V (2005) The cosmological simulation code GADGET-2. *Mon Not R Astron Soc* 364(4):1105–1134
29. Springel V, Yoshida N, White SDM (2001) GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astron* 6:79–117
30. Hopkins PF (2015) A new class of accurate, mesh-free hydrodynamic simulation methods. *Mon Not R Astron Soc* 450(1):53–110
31. Hopkins PF (2017) A new public release of the GIZMO code. arXiv preprint: [arXiv:1712.01294](https://arxiv.org/abs/1712.01294)
32. Gomez-Gesteira M, Crespo AJC, Rogers BD, Dalrymple RA, Dominguez JM, Barreiro A (2012) SPHysics-development of a free-surface fluid solver-part 2: efficiency and test cases. *Comput Geosci* 48:300–307
33. Gomez-Gesteira M, Rogers BD, Crespo AJC, Narayanaswamy M, Dominguez JM (2012) SPHysics-development of a free-surface fluid solver-part 1: theory and formulations. *Comput Geosci* 48:289–299
34. Crespo AJC, Domínguez JM, Rogers BD, Gomez-Gesteira M, Longshaw S, Canelas R, Vacondio R, Barreiro A, Garcia-Feal O (2015) DualSPHysics: open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Comput Phys Commun* 187:204–216
35. Domínguez JM, Crespo AJC, Valdez-Balderas D, Rogers BD (2013) New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Comput Phys Commun* 184(8):1848–1860
36. Hérault A, Bilotta G, Vicari A, Rustico E, Negro CD (2011) Numerical simulation of lava flow using a GPU SPH model. *Ann Geophys* 54(5):600–620
37. Peng C, Wang S, Wu W, Yu HS, Wang C, Chen JY (2019) LOQUAT: an open-source GPU-accelerated SPH solver for geotechnical modeling. *Acta Geotech* 14(5):1269–1287
38. Guo N, Yang ZX (2021) NSPFEM2D: a lightweight 2D node-based smoothed particle finite element method code for modeling large deformation. *Comput Geotech* 140:104484
39. de St. Germain JD, McCorquodale J, Parker SG, Johnson CR (2000) Uintah: A massively parallel problem solving environment. In: Proceedings of the 9th IEEE international symposium on high performance distributed computing. IEEE Computer Society, USA, pp 33–41
40. Dadvand P, Rossi R, Oñate E (2010) An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Arch Comput Methods Eng* 17(3):253–297
41. Zhang X, Krabbenhoft K, Sheng DC (2014) Particle finite element analysis of the granular column collapse problem. *Granul Matter* 16(4):609–619
42. Chen JS, Wu CT, Yoon S, You Y (2001) A stabilized conforming nodal integration for Galerkin mesh-free methods. *Int J Numer Methods Eng* 50(2):435–466
43. Liu GR, Nguyen-Thoi T, Nguyen-Xuan H, Lam KY (2009) A node-based smoothed finite element method (NS-FEM) for upper bound solutions to solid mechanics problems. *Comput Struct* 87(1–2):14–26
44. Jarušek J (1983) Contact problems with bounded friction coercive case. *Czechoslov Math J* 33(2):237–261
45. Cremonesi M, Frangi A, Perego U (2010) A Lagrangian finite element approach for the analysis of fluid-structure interaction problems. *Int J Numer Methods Eng* 84(5):610–630

46. Field DA (1988) Laplacian smoothing and Delaunay triangulations. *Commun Appl Numer Methods* 4(6):709–712
47. Freitag LA, Ollivier-Gooch C (1996) A comparison of tetrahedral mesh improvement techniques. United States
48. Meduri S, Cremonesi M, Perego U (2019) An efficient runtime mesh smoothing technique for 3D explicit Lagrangian free-surface fluid flow simulations. *Int J Numer Methods Eng* 117(4):430–452
49. Vartziotis D, Wipper J, Schwald B (2009) The geometric element transformation method for tetrahedral mesh smoothing. *Comput Methods Appl Mech Engrg* 199(1–4):169–182
50. Beissel S, Belytschko T (1996) Nodal integration of the element-free Galerkin method. *Comput Methods Appl Mech Engrg* 139:49–74
51. Belytschko T, Guo Y, Kam Liu W, Xiao SP (2000) A unified stability analysis of meshless particle methods. *Int J Numer Methods Eng* 48:1359–1400
52. Hillman M, Chen JS (2016) An accelerated, convergent, and stable nodal integration in Galerkin meshfree methods for linear and nonlinear mechanics. *Int J Numer Methods Eng* 107(7):603–630
53. Huang TH, Wei HY, Chen JS, Hillman MC (2020) RKPM2D: an open-source implementation of nodally integrated reproducing kernel particle method for solving partial differential equations. *Comput Part Mech* 7(2):393–433
54. Puso MA, Chen JS, Zywickz E, Elmer W (2008) Meshfree and finite element nodal integration methods. *Int J Numer Methods Eng* 74(3):416–446
55. Silva-Valenzuela R, Ortiz-Bernardin A, Sukumar N, Artioli E, Hitschfeld-Kahler N (2020) A nodal integration scheme for meshfree Galerkin methods using the virtual element decomposition. *Int J Numer Methods Eng* 121(10):2174–2205
56. Wei HY, Chen JS, Beckwith F, Baek J (2020) A naturally stabilized semi-Lagrangian meshfree formulation for multiphase porous media with application to landslide modeling. *J Eng Mech* 146(4):04020012
57. Ganzenmüller GC (2015) An hourglass control algorithm for lagrangian smooth particle hydrodynamics. *Comput Methods Appl Mech Engrg* 286:87–106
58. Yuan WH, Liu M, Dai BB, Wang Y, Chan A, Zhang W (2023) Stabilizing nodal integration in dynamic smoothed particle finite element method: a simple and efficient algorithm. submitted
59. 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 Geotech* 16(8):2449–2462
60. 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
61. Chen D, Huang WX, Sloan SW (2019) An alternative updated Lagrangian formulation for finite particle method. *Comput Methods Appl Mech Engrg* 343:490–505
62. Nguyen CT, Bui HH, Fukagawa R (2015) Failure mechanism of true 2D granular flows. *J Chem Eng Japan* 48(6):395–402
63. Chalk CM, Pastor M, Peakall J, Borman DJ, Sleigh PA, Murphy W, Fuentes R (2020) Stress-particle smoothed particle hydrodynamics: an application to the failure and post-failure behaviour of slopes. *Comput Methods Appl Mech Engrg* 366:113034
64. Wang L, Zhang X, Lei QH, Panayides S, Tinti S (2022) A three-dimensional particle finite element model for simulating soil flow with elastoplasticity. *Acta Geotech* 17(12):5639–5653
65. Bishop AW, Morgenstern N (1960) Stability coefficients for earth slopes. *Géotechnique* 10(4):129–153
66. Griffiths DV, Lane PA (1999) Slope stability analysis by finite elements. *Géotechnique* 49(3):387–403

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.