



# LOQUAT: an open-source GPU-accelerated SPH solver for geotechnical modeling

Chong Peng<sup>1,2</sup> · Shun Wang<sup>2</sup> · Wei Wu<sup>2</sup> · Hai-sui Yu<sup>3</sup> · Chun Wang<sup>4</sup> · Jian-yu Chen<sup>5</sup>

Received: 29 November 2018 / Accepted: 28 May 2019 / Published online: 21 June 2019  
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

## Abstract

Smoothed particle hydrodynamics (SPH) is a meshless method gaining popularity recently in geotechnical modeling. It is suitable to solve problems involving large deformation, free-surface, cracking and fragmentation. To promote the research and application of SPH in geotechnical engineering, we present LOQUAT, an open-source three-dimensional GPU accelerated SPH solver. LOQUAT employs the standard SPH formulations for solids with two geomechanical constitutive models which are the Drucker–Prager model and a hypoplastic model. Three stabilization techniques, namely, artificial viscosity, artificial pressure and stress regularization are included. A generalized boundary particle method is presented to model static and moving boundaries with arbitrary geometry. LOQUAT employs GPU acceleration technique to greatly increase the computational efficiency. Numerical examples show that the solver is convergent, stable and highly efficient. With a mainstream GPU, it can simulate large scale problems with tens of millions of particles, and easily performs more than one thousand times faster than serial CPU code.

**Keywords** Geomaterial · GPU acceleration · LOQUAT · Smoothed particle hydrodynamics

## 1 Introduction

Numerical analysis plays a crucial role in modern geotechnical engineering. After approximately five decades of development, numerical analysis in geotechnical engineering has reached a certain degree of maturity. Various numerical methods were adopted in geotechnical modeling. Among them, grid-based methods such as finite element method (FEM), finite difference method (FDM) and finite

volume method (FVM) are well-established and successfully employed in various applications [64, 89]. Grid-based methods use mesh for the spatial discretization. Most of the advantages and disadvantages of grid-based methods come with the utilization of mesh. For instance, grid-based methods are generally considered more robust and efficient compared to meshless methods, and can be employed to a wide range of applications. Moreover, they have more solid mathematical theory on convergence and stability. On the other hand, grid-based methods have difficulties in modeling problems involving, among others, large deformation, free-surface, moving boundary, cracking and fragmentation. All of these problems have in common changing or discontinuous computational domains due to deforming, cracking and separating of material; thus, grid-based methods need additional efforts such as remeshing or interface tracking to handle these problems.

In recent years, Lagrangian particle-based meshless methods are developed and become more and more popular. These methods have advantages in modeling problems difficult for mesh-based methods because they completely or partly avoid the use of mesh. Popular methods include, for instance, smoothed particle hydrodynamics (SPH)

---

✉ Chong Peng  
chong.peng@essteyr.com; pengchong07@gmail.com

<sup>1</sup> ESS Engineering Software Steyr GmbH, Berggasse 35, 4400 Steyr, Austria

<sup>2</sup> Institut für Geotechnik, Universität für Bodenkultur, Feistmantelstrasse 4, 1180 Vienna, Austria

<sup>3</sup> School of Civil Engineering, Faculty of Engineering, University of Leeds, Leeds LS2 9JT, UK

<sup>4</sup> School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

<sup>5</sup> Department of Mechanical and Mechatronics Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada

[41, 47, 50], material point method (MPM) [4, 5, 80] and particle finite element method (PFEM) [56, 57]. Among these methods, SPH is well-established and has a long application history in astrophysics [70], fluid dynamics [23, 68, 73, 84] and solid mechanics [26, 39, 40]. SPH is also employed to geotechnical modeling in recent years. Reported applications include large deformation [7, 10, 59], granular flow [14, 16, 45, 54, 60, 72], soil-structure interaction [8, 12, 63, 76], landslide and debris flow [13, 36, 44, 58], fluid-soil mixture dynamics [7, 9, 35, 37, 43, 53, 77, 78, 86], cracking [6, 18, 85, 87] and underground explosion [15]. It is shown that SPH is able to solve the aforementioned problems which are otherwise difficult for grid-based methods. Therefore, SPH is promising and has the potential to be employed to a wider range of academic and engineering geotechnical applications.

Although SPH has a long application history in astrophysics, fluid dynamics and solid mechanics, its employment in geotechnics is relatively new. Maeda et al. [43] and Naili et al. [53] are among the first to use SPH to model real geotechnical problems such as seepage failure and sand liquefaction. Later, Bui et al. [10] developed a complete SPH framework for large deformation analysis in geomaterials. In their work, the hypoelastic treatment of large deformation is employed. Constitutive models based on infinitesimal strain assumption are augmented with objective stress rate to model large deformation. Following the SPH framework in [10], many researchers developed new formulations, numerical algorithms and implemented various constitutive models. Notable progresses of SPH research in geotechnical modeling include, among others, the soil-structure frictional contact [76], employment of new constitutive models [7, 59, 60], extension to simulate multiphase water-soil coupling problems [7, 9, 35, 37, 77, 86], new cracking treatment method [6, 85] and the latest development of multiplicative elastoplasticity for SPH [54]. Despite these achievements, it is generally observed that in the past ten years there lacks groundbreaking progresses in SPH theories and applications in geotechnical engineering since Bui et al. [10]. The well-known numerical instabilities such as tensile instability and hourglass mode are still not completely avoided. Among many researchers and engineers, SPH is still a quite unfamiliar approach. This somewhat awkward situation can only be improved if more researchers and engineers start working on and with SPH.

Yet there are several facts discouraging new users to try SPH. SPH is a meshless method, where the approximation of field quantities is based on the particles in the support domain. Usually, the number of particles in interaction is much larger than that in the grid-based methods, especially in three-dimensional cases, where dozens to hundreds of interacting particles can be used [20, 88]. The huge number

of interacting particles greatly increase the computational cost. Moreover, explicit time integration methods are used in SPH, where the time step is governed by the Courant–Friedrichs–Lewy (CFL) condition. Small time steps are resulted from the CFL condition due to the large value of the speed of sound. Therefore, the numerical efficiency of SPH is lower than grid-based methods. To the best of the authors' knowledge, there are no SPH applications in real-scale three-dimensional geotechnical problems. The main reason is that the computational time is prohibitive if a serial SPH computing is used. To this end, acceleration techniques such as multi-thread CPU computing and GPU acceleration are necessary for SPH to achieve reasonable computational time.

Another obstacle for new SPH users is the lack of resources. In recent years, several SPH open-source codes were released, e.g. GADGET [69, 71] and GIZMO [31, 32] for astrophysical modeling, SPHysics [24, 25] and DualSPHysics [17, 21] for fluid dynamics, and GPUSPH [28] for lava flow modeling. These open-source codes are designed for applications in which both the dominant physics and the material behaviors are far different from geotechnical problems. Therefore, considerable modifications of the codes are needed before they can be employed in geotechnical modeling. The SPH method is also included in commercial software LS-DYNA and ABAQUS; however, SPH module in these software focuses on fluid dynamics and impact simulation, which cannot be directly used in geotechnical simulations. Till now there is no open-source or commercial SPH solver designed for geotechnical applications.

Consequently, it is beneficial for the geotechnical researchers and engineers who have already been working with SPH, as well as the new users who want to start using SPH, to have an efficient open-source SPH solver designed for geotechnical modeling. To this end, in this work, we present LOQUAT, an open-source, GPU-accelerated SPH solver for geotechnical modeling. LOQUAT is developed using C++ and CUDA with object-oriented design. It is designed for three-dimensional large scale simulations with up to tens of millions of particles using single NVIDIA GPU device. LOQUAT solves the partial differential equations by means of SPH discretization and explicit time integration. Two geomechanical constitutive models, namely, the Drucker–Prager model [22] and the hypoplastic model of Wang and Wu [75, 83] are included to capture the mechanical responses of geomaterials. A general boundary condition able to model fixed and moving boundaries with arbitrary geometry is proposed and implemented in LOQUAT. With the current implementation, LOQUAT can be used to simulate a wide range of geotechnical problems involving large deformation, free-surface and moving boundary. For instance, the two-

dimensional version of the solver has already been recently applied to simulate granular flow [59, 60], slope stability [59], seepage and flow-porous media interaction [61, 62], and soil-structure interaction [63].

The main purpose of developing and releasing LOQUAT is to promote the research and use of SPH in geotechnical engineering. With this in mind, we try to keep the code as simple and readable as possible. Complex optimization techniques are not considered in the current version. Along with the solver kernel, input files for several numerical examples are also provided. The output files of LOQUAT is in .vtu format which can be visualized using the widely-used open-source visualization software ParaView [3]. The LOQUAT solver and the related documentation and test cases can be found at <https://github.com/ChongPengGeotech/LOQUAT/>.

In the remaining part of the paper, firstly, the theories of SPH, the implemented constitutive models and the stabilization techniques are detailed. Afterwards, the GPU acceleration, the structure of the code and brief documentation are described. Finally, several numerical examples are presented, where we demonstrate the accuracy, efficiency and stability of the solver.

## 2 SPH for geotechnical modeling

### 2.1 Governing equations

SPH is a Lagrangian particle-based methods. Therefore, in SPH the governing equations of geotechnical problems are written in the following Lagrangian form

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

$$\frac{d\mathbf{v}}{dt} = \frac{1}{\rho} \nabla \cdot \boldsymbol{\sigma} + \mathbf{g} \quad (2)$$

where  $\rho$  is the density;  $\mathbf{v}$  denotes the velocity,  $\boldsymbol{\sigma}$  is the Cauchy stress tensor and  $\mathbf{g}$  is the acceleration induced by body force, e.g. gravity in most geotechnical problems.  $d(\cdot)/dt$  represents the material time derivative and  $\nabla$  is the gradient operator. In this work, pore pressure is not considered, which means the material is either dry or can be modeled using total stress. In many simulations, the Eq. (1), i.e. the continuity equation can be removed from the simulation, because Lagrangian particles are used so the mass conservation is always satisfied [10, 59]. Eq. (1) is only needed when equation of state (EOS) based constitutive models are used [39].

### 2.2 SPH discretization of governing equations

In SPH, the computational domain is partitioned using particles which carry field variables and move with the material. The governing equations can be solved by tracking the motions of the particles and changes of the carried variables. To do this, one has to discretize the governing equations using the SPH interpolations.

A field function can be approximated using the following integral interpolation

$$\langle f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3)$$

where  $\langle \cdot \rangle$  represents the approximated value,  $\Omega$  is the influence domain of the kernel function  $W(\mathbf{x} - \mathbf{x}', h)$ , abbreviated as  $W$  hereafter.  $W$  depends on the distance  $\|\mathbf{x} - \mathbf{x}'\|$  and the parameter  $h$ , i.e. the smoothing length. In this work, the radius of the spherical support domain of  $W$  is  $2h$ . The kernel function  $W$  has to satisfy certain requirements such as normalization condition, delta function property and compact support condition [46]. Common choices of kernel functions include the Gaussian kernel [51], the cubic spine kernel [67] and the Wendland kernels [79]. In LOQUAT, only the C2 Wendland kernel is employed because of the benefit of preventing particle clumping [20, 66].

Similarly, the spatial derivatives of the field function can be approximated through [59]

$$\langle \nabla f(\mathbf{x}) \rangle = - \int_{\Omega} f(\mathbf{x}') \nabla_{\mathbf{x}'} W d\mathbf{x}' \quad (4)$$

where  $\nabla_{\mathbf{x}'}$  indicates that the derivatives are evaluated at  $\mathbf{x}'$ . The above equation is only valid when the support domain has no intersection with the boundary of the computational domain. However, this condition is generally not fulfilled for areas near the boundary. Therefore, SPH needs special treatment at boundaries to circumvent the so-called boundary deficiency. The treatment of boundary condition will be introduced in Sect. 2.4.

In SPH, the continuous forms of integral interpolation Eqs. (3) and (4) are rewritten as the summation of contributions of all the particles in the support domain

$$f(\mathbf{x}_i) = \sum f(\mathbf{x}_j) W_{ij} \frac{m_j}{\rho_j} \quad (5)$$

$$\nabla f(\mathbf{x}_i) = \sum f(\mathbf{x}_j) \nabla_i W_{ij} \frac{m_j}{\rho_j} \quad (6)$$

where the bracket  $\langle \cdot \rangle$  is dropped for convenience.  $\mathbf{x}_i$  is the particle where the field function is approximated, and  $\mathbf{x}_j$  is the particle in the support domain.  $\nabla_i W_{ij}$  is the short form of  $\nabla_{\mathbf{x}_i} W(\mathbf{x}_i - \mathbf{x}_j, h)$ . Note that  $\nabla_i W_{ij} = -\nabla_j W_{ij}$  due to the symmetry of the kernel function.  $m_j$  is the mass of particle  $j$  and  $m_j/\rho_j$  denotes the volume of the particle.

With the discrete forms of SPH interpolations Eq. (5) and (6), the governing equations can be rewritten as

$$\frac{d\rho_i}{dt} = \sum m_j (\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla_i W_{ij} \quad (7)$$

$$\frac{d\mathbf{v}_i}{dt} = \sum m_j \left( \frac{\boldsymbol{\sigma}_i}{\rho_i^2} + \frac{\boldsymbol{\sigma}_j}{\rho_j^2} \right) \nabla_i W_{ij} + \mathbf{g}_i \quad (8)$$

The above formulations conserve the linear and angular momentum and are commonly employed, although there are different discretization forms of the governing equations in the literature [39, 76].

### 2.3 Stabilization techniques

In SPH, a dissipative term is usually added into the momentum equation to stabilize the numerical simulation and prevent large unphysical oscillations. The reason is that SPH is a dynamic method, where shock waves are always present, especially in problems with abrupt loading and change of boundary condition. If the shock waves are not damped out to make sure their wave lengths are larger than the characteristic length  $h$ , spurious mode and strong instability can occur. A common dissipative term is the artificial viscosity [48]

$$\Pi_{ij} = \begin{cases} \frac{\alpha c_s h}{\bar{\rho}_{ij}} \frac{\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{r_{ij}^2 + 0.01h^2}, & \text{if } \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where  $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ ,  $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ , and  $r = \|\mathbf{r}_{ij}\|$  is the distance between particle  $i$  and  $j$ .  $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$  is the average density.  $c_s$  denotes the speed of sound which will be introduced in Sect. 2.5.  $\alpha$  is a constant determining the magnitude of the artificial viscosity.

It is well-known that the SPH method suffers from the so-called tensile instability, i.e. when material is in tensile state, the particles tend to form small clumps which eventually lead to unphysical fractures. In geotechnical modeling, Bui et al. [10] demonstrated that for non-cohesive geomaterials the tensile instability is negligible. However, if the modeled material is with cohesion, the tensile instability develops and has to be removed using additional numerical treatments. One common technique to alleviate the tensile instability is the artificial stress method, which adds a short ranged repulsive force when the material is in tensile state. The additional stress term in the artificial stress method has the following form

$$S_{ij} = f_{ij}^n (\mathbf{R}_i + \mathbf{R}_j) \quad (10)$$

where  $f_{ij} = W_{ij}/W(\Delta p)$  and  $\Delta p$  is the initial particle spacing, the exponential factor  $n$  is taken as  $n = W(0)/W(\Delta p)$ . The term  $\mathbf{R}_i$  is usually obtained by first rotating the original stress tensor  $\boldsymbol{\sigma}_i$  to a pure principal stress state, then

multiplying a factor if the principal stress is in tension and finally rotating back to the original direction [10, 26]. This approach is not only computationally expensive but also very difficult to perform in three-dimensional condition. In LOQUAT, an alternative approach is employed, where

$$S_{ij} = f_{ij}^n (\mathbf{R}_i + \mathbf{R}_j) \quad (11)$$

where the term  $\mathbf{R}_i$  is obtained using

$$\mathbf{R}_i = \begin{cases} \frac{\epsilon |\mathbf{p}_i|}{\rho_i^2}, & \text{if } p_i < 0 \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

where  $p = -(\sigma_{xx} + \sigma_{yy} + \sigma_{zz})/3$  is the isotropic hydrostatic pressure.  $\epsilon$  is a constant controlling the magnitude of the artificial stress, usually taken as 0.2 in geotechnical modeling.  $\mathbf{R}_j$  can be computed in a similar way. This approach is very much like the original artificial stress method proposed in [49]. It is more convenient to implement in three-dimensional simulations.

With the artificial viscosity and artificial stress, the SPH momentum equation (8) is rewritten as

$$\frac{d\mathbf{v}_i}{dt} = \sum m_j \left[ \frac{\boldsymbol{\sigma}_i}{\rho_i^2} + \frac{\boldsymbol{\sigma}_j}{\rho_j^2} + (\Pi_{ij} - S_{ij}) \mathbf{I} \right] \nabla_i W_{ij} + \mathbf{g}_i \quad (13)$$

where  $\mathbf{I}$  is an identity matrix.

SPH also has the so-called short-length-scale-noise resulting in stress fluctuations in areas with large deformation [55, 60]. Stress regularization can be applied to improve the stress results, just like the Shepard filter in SPH for fluid dynamics [55]. The stress is re-initialized after a certain number of computational steps. The new stress can be written as

$$\boldsymbol{\sigma}_i^{new} = \frac{\sum \boldsymbol{\sigma}_j W_{ij}}{\sum W_{ij}} \quad (14)$$

The normalized SPH interpolation has linear consistency; thus, it can reproduce the linear variation of stress fields.

Among the aforementioned stabilization techniques, the artificial viscosity is suggested to be always activated. The artificial stress has to be used when simulating cohesive geomaterials. The stress regularization needs to be applied in caution, as its performance has not been thoroughly investigated.

### 2.4 Boundary conditions

Because SPH is a Lagrangian particle based method, the implementation of boundary condition is more difficult than in grid-based methods. The boundary treatments of SPH need to fulfill two requirements, the first is to prevent particle penetration, the second is to correct the SPH

interpolation in incomplete support domains due to the intersection with the boundary. In LOQUAT the non-slip rigid wall boundary is implemented using the boundary particle method.

In the boundary particle method, the solid boundary is represented by certain layers of SPH particles. The width of the boundary zone is determined in such a way that any material particles near the boundary can have full support domain. These boundary particles are either fixed in position or moved according to prescribed velocities. The boundary particles participate in SPH computations like normal material particles. However, the field variables on the boundary particles are not computed by evolving the governing equations, but extrapolated from the material domain. Before each computation of particle interaction, the stresses on boundary particles are interpolated as

$$\sigma_b = \frac{\sum \sigma_m W_{bm}}{\sum W_{bm}} \quad (15)$$

where the subscripts  $b$  and  $m$  denotes boundary and material particles, respectively. Furthermore, the stresses at boundary particles have to be corrected according to the body force to correctly approximate the stress gradient caused by the body force [1]

$$\sigma_b^l = \frac{\sum \sigma_m^l W_{bm} - g_b^l \sum \rho_m r_{bm}^l W_{bm}}{\sum W_{bm}} \quad (16)$$

where  $l = x, y, z$  is the direction of the spatial coordinates. Note that in the above equation the Einstein summation convention is not applied. It is observed that only the diagonal components of the stress tensor are subjected to the body force correction.

The proposed boundary treatment method has the benefit of being able to model fixed and moving boundaries with arbitrary geometry. However, it can not accurately simulate free-slip or frictional boundary conditions. The free-slip behavior can be roughly modeled by neglecting the effect of shear stress. However, this treatment is highly approximate. The accurate modeling of free-slip and frictional boundary conditions needs to consider the realistic contact between the material and boundary [63, 76].

## 2.5 Time integration

In the SPH method, the system of equations consists of Eqs. (7) and (13), which can be solved numerically using explicit integration methods, e.g. the Predictor-Corrector method [48], the leap-frog method [10] and the Verlet method [1]. In LOQUAT the original form of Predictor-Corrector scheme is adopted. Let  $X$  denote a variable (position, density, velocity, stress or strain) to be integrated

and  $F = dX/dt$  is the rate of change. With  $\Delta t$  the time step, the predictor predicts the variable at half time step as

$$X^{t+\Delta t/2} = X^t + F^t \Delta t/2 \quad (17)$$

where the superscript  $t$  is the time at the beginning of the computational step. The rate of change  $F^t$  is evaluated using the variables at time instance  $t$ . In the corrective step, the value of  $X$  at the end of the computational step is obtained using

$$X^{t+\Delta t} = X^t + F^{t+\Delta t/2} \Delta t \quad (18)$$

Note that the change of rate  $F^{t+\Delta t/2}$  is computed based on the variables at time instance  $t + \Delta t/2$ , which is obtained in the predictive step. It should be pointed out that the integration of stress should follow the constitutive model.

The time step of the Predictor-Corrector scheme is controlled by the CFL condition [23]:

$$\Delta t = \chi \min \left( \sqrt{\frac{h}{\max ||a_i||}}, \frac{h}{c_s} \right) \quad (19)$$

where  $a_i = dv_i/dt$  is the particle acceleration,  $\chi$  is the CFL coefficient usually in the range of 0.05–0.2,  $c_s$  is the speed of sound, which depends on the stiffness of the simulated material.

## 3 Constitutive models

The governing equations (7) and (13) should be closed by equations describing geometrical behaviors. In LOQUAT, two constitutive models are included, i.e. the elastoplastic Drucker–Prager model [22] and the hypoplastic model from Wang and Wu [75, 83]. These constitutive models are developed based on the small deformation and infinitesimal strain assumptions, whereas SPH modeling of geotechnical problems often has to simulate large deformation. As a result, additional treatments should be considered.

### 3.1 Hypoelastic approach for large deformation analysis

In computational geomechanics, there are two approaches for finite strain and large deformation analysis. One is to employ constitutive models based on hyperelasto-plasticity, employing the multiplicative decomposition of the deformation gradient [19, 54]. The other approach is the hypoelastic-based method which employs objective (frame invariant) stress rates to provide finite strain and large deformation extension to already existing infinitesimal models [19]. The hypoelastic approach is simple in formulation and implementation; thus, most of the existing SPH numerical simulations employ it. Furthermore, the



conventional SPH method (compared to total-Lagrangian SPH method) is based on the Eulerian kernel, which means that the computation is always based on the current configuration; therefore, employing the hypoelastic approach is natural and does not require modifying the SPH formulation. However, two conditions should hold to ensure the accuracy of the modeling using hypoelastic approach: (1) the elastic strain should be small compared to the plastic strain; (2) the time step should be small enough to keep the objectivity [38]. These two conditions are naturally satisfied in most SPH geotechnical modelings.

In LOQUAT, the Jaumann stress rate is adopted, which reads

$$\dot{\sigma} = \dot{\sigma} + \sigma \cdot \omega - \omega \cdot \sigma = \mathbf{H}(\dot{\epsilon}, \sigma, \dots) \quad (20)$$

where  $\dot{\sigma}$  is the Jaumann stress rate,  $\dot{\sigma}$  is the rate of Cauchy stress.  $\omega$  denotes the spin tensor, and  $\dot{\epsilon}$  is the strain rate tensor.  $\omega$  and  $\dot{\epsilon}$  are defined as  $\omega = (\mathbf{L} - \mathbf{L}^T)/2$  and  $\dot{\epsilon} = (\mathbf{L} + \mathbf{L}^T)/2$ , where  $\mathbf{L} = \partial \mathbf{v} / \partial \mathbf{x}$  is the velocity gradient. In the SPH context, the velocity gradient at a particle is obtained as

$$\mathbf{L}_i = \sum (\mathbf{v}_j - \mathbf{v}_i) \otimes \nabla_i W_{ij} \frac{m_j}{\rho_j} \quad (21)$$

In Eq. (20),  $\mathbf{H}$  denotes the rate form infinitesimal constitutive model which depends on strain rate, Cauchy stress and possibly other internal variables. Finally, the rate of Cauchy stress is written as

$$\dot{\sigma} = \omega \cdot \sigma - \sigma \cdot \omega + \mathbf{H}(\dot{\epsilon}, \sigma, \dots) \quad (22)$$

Integrating the above equation we can obtain the stress in large deformation analysis.

### 3.2 Drucker–Parger model

The Drucker–Prager (DP) model is a commonly-used constitutive model in geomechanics. It is a elasto-perfectly plastic model with a smooth yield surface

$$f^s = \sqrt{J_2} - k_\phi p - k_c \quad (23)$$

where  $p = -(\sigma_{xx} + \sigma_{yy} + \sigma_{zz})/3$  is the isotropic hydrostatic pressure;  $J_2 = \mathbf{s} : \mathbf{s} / 2$  is the second invariant of the deviatoric stress tensor defined as  $\mathbf{s} = \sigma - p\mathbf{I}$ .  $k_\phi$  and  $k_c$  are constitutive parameters related to parameters from the Mohr–Coulomb model

$$k_\phi = \frac{3 \tan \phi}{\sqrt{9 + 12 \tan^2 \phi}}, k_c = \frac{3c}{\sqrt{9 + 12 \tan^2 \phi}}, \quad (24)$$

where  $\phi$  and  $c$  are the friction angle and cohesion, respectively.

Under the non-associated plastic flow assumption, the plastic potential function is written as

$$g^s = \sqrt{J_2} - k_\psi p \quad (25)$$

Similarly,  $k_\psi$  is defined as

$$k_\psi = \frac{3 \tan \psi}{\sqrt{9 + 12 \tan^2 \psi}} \quad (26)$$

where  $\psi$  is the dilatancy angle.

At each computational step, an elasto-predicted stress  $\sigma^*$  is first obtained using linear elastic relation. Taking the corrective step as an example, we have

$$\sigma^* = \sigma^t + (\omega \cdot \sigma^t - \sigma^t \cdot \omega + 2G\dot{\epsilon} + K\epsilon_v \mathbf{I})\Delta t \quad (27)$$

where  $\epsilon_v = (\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy} + \dot{\epsilon}_{zz})/3$ , and  $\dot{\epsilon} = \dot{\epsilon} - \epsilon_v \mathbf{I}$  is the deviatoric part of the strain rate tensor.  $K$  and  $G$  are the elastic bulk modulus and shear modulus, respectively.

If the predicted stress  $\sigma^*$  is outside of the yield surface, plastic failure occurs. The corresponding plastic strain increment is computed based on the plastic flow rule

$$\Delta \epsilon^p = \Delta \lambda^s \frac{\partial g^s}{\partial \sigma} \quad (28)$$

In the DP model, the increment of the plastic multiplier  $\Delta \lambda^s$  reads

$$\Delta \lambda^s = \frac{f^s(\sigma^*)}{G + Kk_\phi k_\psi} \quad (29)$$

Finally, the corrected stress is obtained as

$$\begin{aligned} \mathbf{s}^{t+\Delta t} &= \frac{\sqrt{J_2^*} - G\Delta \lambda^s}{\sqrt{J_2^*}} \mathbf{s}^*, p^{t+\Delta t} = p^* + Kk_\psi \Delta \lambda^s, \sigma^{t+\Delta t} \\ &= \mathbf{s}^{t+\Delta t} + p^{t+\Delta t} \mathbf{I} \end{aligned} \quad (30)$$

where  $\mathbf{s}^*$ ,  $J_2^*$  and  $p^*$  are the corresponding variables of the predicted stress tensor  $\sigma^*$ .

### 3.3 Hypoplastic constitutive model

In LOQUAT, the hypoplastic constitutive model of Wang and Wu [75, 83] is also included. This model has the following rate form

$$\begin{aligned} \dot{\sigma} &= \omega \cdot \sigma - \sigma \cdot \omega + c_1(\text{tr} \sigma) \dot{\epsilon} + c_2(\text{tr} \dot{\epsilon}) \sigma \\ &+ c_3 \frac{\text{tr}(\sigma \dot{\epsilon})}{\text{tr} \sigma} \sigma + c_4(\sigma + s) \|\dot{\epsilon}\| \end{aligned} \quad (31)$$

where  $\|\dot{\epsilon}\| = \sqrt{\dot{\epsilon} : \dot{\epsilon}}$  is the Euclidean norm of the strain rate tensor,  $\mathbf{s}$  is the deviatoric stress tensor.  $c_1$ – $c_4$  are the four constitutive parameters that can be related to the conventional geomaterial parameters such as Young's modulus, Poisson's ratio, friction angle and dilatancy angle [81–83]. In the above equation, the first two terms are from the Jaumann stress rate to keep the rotation objectivity. The terms associated with parameters  $c_1$  to  $c_3$  are linear part of

the model. It can be observed that constitutive equation is homogeneous of the first degree in stress, which means the predicted behaviors, e.g. stiffness and shear strength depend on the stress level, which is realistic for geomaterials. The last term in the model is the nonlinear term. The hypoplastic model does not split the strain into elastic and plastic parts. The nonlinear property of geomaterials is automatically accounted for using the model. For details of the implementation of this model in SPH, the readers please refer to [59, 60]. The Cauchy stress rate is computed from the hypoplastic model directly. Therefore, no plastic correction of the final stress is required.

The hypoplastic model Eq. (31) is mainly used for granular soils. But it can also be extended to model geomaterials with moderate cohesion by simply replacing the Cauchy stress  $\sigma$  with a translated stress [59, 75]

$$\eta = \sigma - cI \quad (32)$$

where  $c$  is the cohesion. With the translated stress, the hypoplastic model is rewritten as

$$\dot{\eta} = \omega \cdot \eta - \eta \cdot \omega + c_1(\text{tr}\eta)\dot{\epsilon} + c_2(\text{tr}\dot{\epsilon})\eta + c_3 \frac{\text{tr}(\eta\dot{\epsilon})}{\text{tr}\eta} \eta + c_4(\eta + s_\eta)||\dot{\epsilon}|| \quad (33)$$

where  $s_\eta$  is the deviatoric part of the translated stress.

## 4 GPU acceleration and numerical implementation

### 4.1 GPU acceleration for SPH

SPH suffers from low efficiency in serial computation because of the high computational cost. However, SPH is very attractive for parallel computing due to its embarrassingly parallel nature. Therefore, parallel computation has been used in SPH for a long time, mostly using multi-core CPUs or CPU clusters in early time [52, 74]. Recently, the high-performance computing of SPH mainly focuses on GPUs [17, 21, 28], which are superior in terms of price and energy consumption compared to traditional CPUs. The modern GPUs have hundreds or even thousands of processing cores which can execute computing kernels concurrently. A computing kernel can be viewed as a function, the same version of the function runs on all the GPU cores and shares the same global memory. Therefore, the computations of a large number of SPH particles can be distributed over many threads in a massively parallel fashion.

We implemented LOQUAT based on the Compute Unified Device Architecture (CUDA), a toolkit developed by NVIDIA to allow using their graphic cards as high-performance computing devices. The solver has a CPU part

and a GPU part, as shown in Fig. 1. The CPU part works as a process controller. It is used for loading and saving data, initializing the settings of simulation, and linking the GPU computing kernels to finish computations in a designed order. The GPU part executes all the actual computations including neighbor search, boundary extrapolation, particle interaction and time integration. As memory bandwidth is a bottleneck in GPU computing, we reduce memory transfer, especially transfer between GPU and CPU as much as possible. All the data is stored in the GPU memory, with only one transfer of data from CPU to GPU at the beginning and some occasional transfers from GPU to CPU for saving results.

The neighbor search consists of several computational operations on GPU and will be detailed in Sect. 4.2. The remaining three computations, i.e. boundary extrapolation, particle interaction and time integration are performed using four computing kernels: `AdamiBoundary_cuk`, `ParticleInteraction_cuk`, `Predictor_cuk` and `Corrector_cuk`. During simulation, each computing kernel is run by a large number of GPU cores simultaneously; thus, high numerical efficiency can be achieved. Note that the computing kernels for stress integration, i.e. `MatModDP_cuk` and `MatModHypo_cuk` are called by the computing kernels for time integration.

### 4.2 Neighbor search

The basic and most time consuming operation in SPH is to compute the particle interaction. In three dimensions, the support domain of the SPH kernel function is a sphere with a radius of  $2h$ . Only particles with distance less than  $2h$  have interactions with the centering particle. Therefore, the computation of particle interaction needs to first identify the particles within the support domain for each particle. A naive method is to loop over all the total  $N$  particles to compute the distances and store the interacting particles. However, this method is prohibitively expensive in terms of both computational time and memory consumption; thus nobody really uses this method in code aiming to simulate large-scale problems.

Hence, most SPH solvers perform the neighbor search based on search algorithms such as linked-list [30] and tree [29]. Not all of these methods are suitable for GPU computing. In LOQUAT, we adopt a modified version of the linked-list method described in [27], which is used by most of the GPU-based SPH simulations [17, 21, 28]. Here we briefly describe the search algorithm.

The modified linked-list search algorithm is based on a virtual search grid which occupies the whole computational domain, as shown in Fig. 2. The grid consists of cubic cells with an edge length of  $2h$ . With the virtual grid, a particle only needs to search possible interaction particles in its

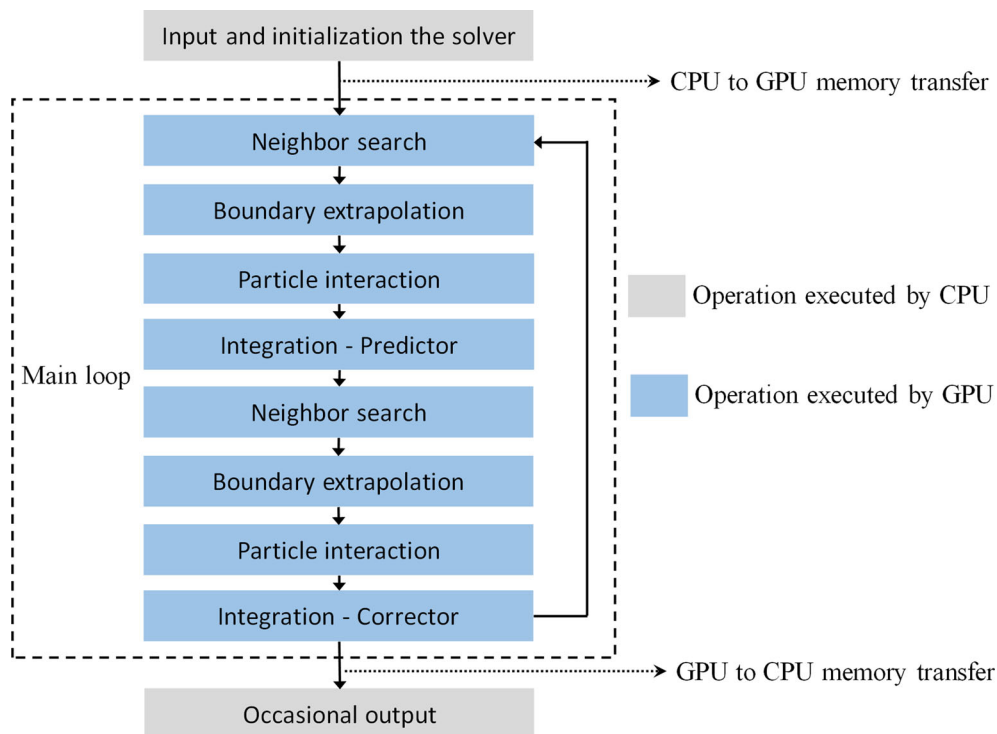


Fig. 1 Flowchart of the simulation process in LOQUAT

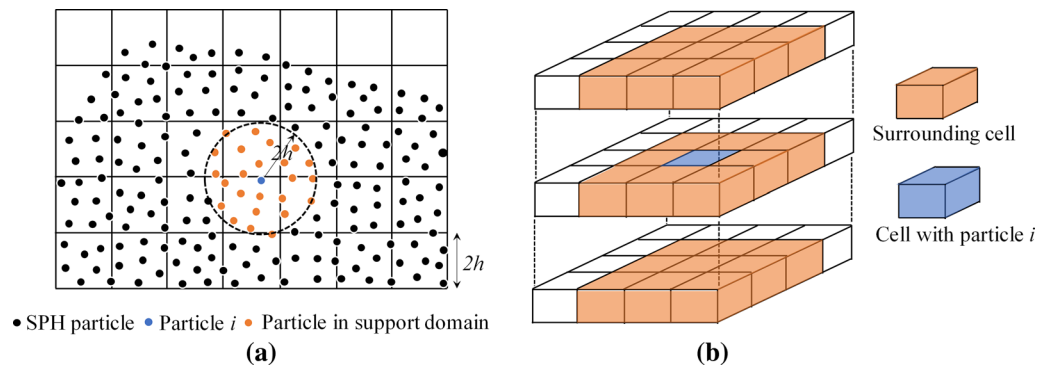


Fig. 2 Sketch of the virtual search grid: **a** virtual search grid and support domain in two-dimensions; **b** virtual search grid in three-dimensions

own cell and the immediately surrounding cells, because all particles outside of these cells have distances larger than  $2h$ ; thus are outside its support domain. The goal of the search algorithm is, for each cell, to search and store all the particles inside it. With this information, the particle interaction can be simply performed using the algorithm described in Algorithm 1. Note that when computing particle interaction the actual interacting particles need to be determined based on the distance. With the employed virtual search grid, the number of operations reduces from  $O(N^2)$  to  $O(MN)$ , where  $N$  is the total particle number and  $M$  is the particle number in all the surrounding cells, which is far less than  $N$ .

---

**Algorithm 1:** Compute interaction for particle  $i$ .

---

```

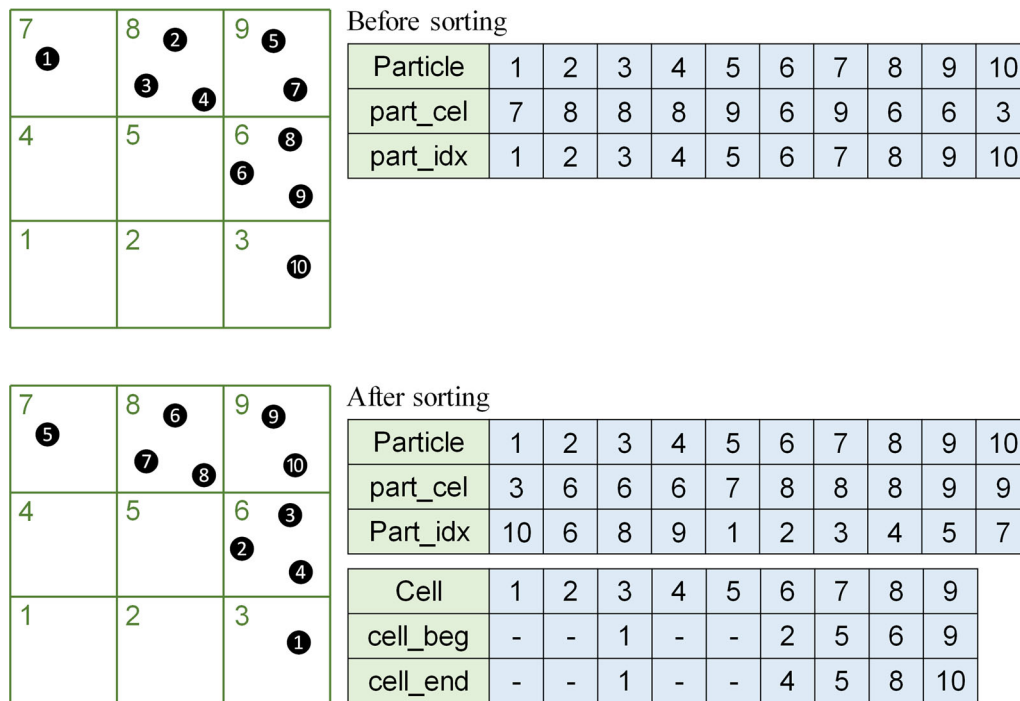
Determine the cell in which particle  $i$  resides;
for all surrounding cells do
    for particle  $j$  in the considered cell do
        if  $r_{ij} < 2h$  then
            Compute the particle interaction between  $i$  and  $j$ ;
        end
    end
end

```

---

The modified linked-list search algorithm is optimized to use GPU acceleration and save memory. Its main concept is shown in Fig. 3. First, all the cells in the virtual search grid is given an index following a predefined order.





**Fig. 3** A simple example to show the work flow of the modified linked-list search algorithm

Then, an array `part_cel` is created to store the index of the cell in which a particle resides. For example, `part_cel[i]` stores the index of the cell containing particle  $i$ . Afterwards, all the particles, including the associated variables, are sorted according to their cell index. This sorting is to ensure that all particles in one cell locate in the same piece of memory. Therefore, we do not need to store all the particle index for each cell. Instead, we only compute the index of the beginning and ending particles in the cell, which are stored in `cell_beg` and `cell_end`, respectively. This approach does not only save much memory, but also speeds up the memory access because it ensures that the data from the memory is accessed sequentially. The whole neighbor search algorithm is performed using four CUDA computing kernels: `GetParticleCellIndex_cuk`, `SortParticleByCellIndex`, `FindBegEndInCell_cuk` and `SortArray_cuk`. An auxiliary array `part_idx` is used to save the old indexes of particles before sorting. This auxiliary array is important in reordering the variables carried by particles. All the computing kernels are developed by the authors except the sorting, for which we use the highly-efficient CUDA library CUB [34].

### 4.3 Optimization strategies

Although GPU devices have high computational power, proper design of the GPU algorithm is also very important for achieving high efficiency. Principally, there are three

basic strategies for optimization: (1) achieve high occupation of GPU devices; (2) optimize memory usage and transfer; (3) optimize instruction usage. Following these principals, the following optimization strategies are employed in LOQUAT.

First, particles are properly grouped in memory to avoid code branching and warp divergence. In SPH, soil and boundary particles have different computations. If particles of different types are put in the same kernel, code branching and warp divergence is inevitable, leading to low occupancy of the device. In LOQUAT, soil and boundary particles are grouped in memory sequentially. This can ensure that in one kernel only the computation of one type of particles is performed. Consequently, each thread has the same code flows and similar computational loads.

Second, the optimal block size of each kernel is chosen automatically using the CUDA configurator API `cudaOccupancyPotentialBlockSize`. This API returns the optimal block size for a kernel based on the usage of registers and constant memory in the considered kernel, the properties of the device, and the number of particles computed using the kernel. With this API, there is no need to tune the block size for given computing kernels.

Third, the memory transfer between CPU and GPU is minimized by storing all the data in GPU memory. Consequently, there is no memory transfer between CPU and GPU in the whole computation except simulation initialization and results saving.

Fourth, only single-precision float numbers are used in LOQUAT. GPUs can process single-precision float numbers and the corresponding operations much faster than double-precision ones. Furthermore, to speed-up the memory access, all particle variables are stored in GPU using the built-in type `float4`, which fulfills the size and alignment requirement of GPU devices.

## 5 Brief solver documentation

### 5.1 Overview of the solver

The LOQUAT solver consists of a series of C++ and CUDA source files. It also uses two open-source libraries in the form of source codes, i.e. CUB for particle sorting using GPU and TINYXML2 [33] for parsing XML file using CPU. All the sources files developed by the authors and their functionality are listed in Table 1. In LOQUAT, the CUDA computing kernels performs most of the computational work in parallel. Table 2 lists all the computing kernels and their functionality. Note that CUDA computing kernels can be called by both host (CPU) and device (GPU). If a computing kernel is called by device, it means that it is called by another computing kernel, not by the CPU code.

The users need to give controlling parameters to run simulations. The controlling parameters include SPH parameters (e.g. particle size, smoothing length/particle size), material constants (e.g. density, constitutive parameters), boundary conditions such as body force and other parameters related to the simulation (e.g. maximal simulation time, CFL number). Other derived parameters important for the simulation, e.g. mass of particle and normalization coefficient of SPH kernel, can be computed from the controlling parameters. All the parameters are stored in a data type `Parameters`. These parameters are transferred to the constant memory of the GPU device to

ensure faster access speed and reduce branching in the GPU computing kernels.

Six kinds of field variables are the primary variables that should be given to initialize the simulation and saved in the results files, i.e. position, velocity, particle type, equivalent plastic strain, stress and strain. The particle type is used to separate different groups of particle. There are other auxiliary variables necessary for the simulation. For instance, we need to store the results of acceleration and velocity gradient obtained from the particle interaction. However, these variables are only used in the simulation and are not saved in results files.

### 5.2 Input and output format

The LOQUAT solver is designed in such a way that output files can be used as input files for subsequent simulations. To this end, the input files consist of two files, one XML file with all controlling parameters, and one plane data file (DAT) with all the field variables. For each time of output, the solver also saves a XML file and a DAT file, containing all the information needed to restart the simulation.

The LOQUAT solver also provides an option to save results as VTU files, which can be visualized using the widely-used free visualization tool ParaView. The conversion between DAT and VTU file formats would be very easy by writing a small program. However, the current version of LOQUAT solver does not provide this functionality.

### 5.3 Compilation and execution

The solver is compiled by using the provided Makefile. To compile the solver, a C++ compiler and the NVIDIA CUDA compiler NVCC need to be installed. Furthermore, the solver uses several libraries from the BOOST package, so BOOST needs to be installed. We have tested the compilation on operation systems Ubuntu (versions 14.04

**Table 1** The source files and their functionality

File name	Functionality
types.h	Define new data types and associated operations and functions
main.cpp	Main file of the LOQUAT solver that executes the simulation
loquat.h/cu	Declare the class <code>LoquatSph</code> that executes the whole simulation process
loquat_io.h/cu	Declare the class <code>LoquatIo</code> that manages all input and output
loquat_grid.h/cu	Declare the class <code>LoquatGrid</code> that performs neighbor search
adami_boundary.h/cu	Extrapolate variables for the boundary particles
particle_interaction.h/cu	Perform computation of particle interaction
predictor_corrector.h/cu	Perform time integration, including stress integration using constitutive models.
Makefile	File for compilation

**Table 2** The CUDA computing kernels and their functionality

Kernel name	Location	Called by	Functionality
GetParticleCellIndex_cuk	loquat_grid.cu	Host	Compute cell index for each particle
FindBegEndInCell_cuk	loquat_grid.cu	Host	Find starting and ending particles for cells
SortArrayFloat4_cuk	loquat_grid.cu	Host	Sort an array of float4 based on new particle order
SortArrayTensor2d_cuk	loquat_grid.cu	Host	Sort an array of tensor2d based on new particle order
AdamiBoundary_cuk	adami_boundary.cu	Host	Loop over cells to extrapolate variables on boundary
AdamiBoundaryCalculation_cuk	adami_boundary.cu	Device	Perform the particle pair interaction for boundary extrapolation
ParticleInteraction_cuk	particle_interaction.cu	host	Loop over cells to compute particle interaction
ParticleInteractionCalculation_cuk	particle_interaction.cu	device	Perform the particle pair interaction
Predictor_cuk	predictor_corrector.cu	host	Perform predictive step of the time integration
Corrector_cuk	predictor_corrector.cu	host	Perform corrective step of the time integration
MatModDP_cuk	predictor_corrector.cu	device	Perform stress integration based on the Drucker–Prager model
MatModHypo_cuk	predictor_corrector.cu	device	Perform stress integration based on the hypoplastic model

LTS and 16.04 LTS) and CentOS 6.0, CUDA versions from 7.0 to 9.2, and BOOST library versions from 1.62 to 1.68. The compilation generates an executable named LOQUAT.

To run the simulation, the user has to prepare two input files `name-of-project.xml` and `name-of-project.dat`. These files can be either generated by the user or from previous simulations. The simulation can be started using the command line

```
./LOQUAT name-of-project i
```

`i` is an optional input parameter, which allows the user choose which GPU device should be used to perform the simulation. If it is not given, the solver runs on the first GPU device. The solver creates a folder with the same name as the project name and saves all the results files there. There are several project examples delivered along with the solver to help the users to get familiar with it.

## 6 Performance of the solver

The performance of the solver is checked using a Nvidia GTX 1080Ti graphic card installed on a desktop computer. The card has 11 GB memory and 3584 CUDA cores, each with a core clock of 1556 MHz.

The axisymmetric collapse of dry granular materials on flat surface was experimentally studied by Lube et al [42]. It is widely-used as a benchmark for numerical methods including SPH. In this section, a series of simulations are performed with varying particle size and computational parameters to demonstrate the accuracy, convergence, efficiency and memory usage of LOQUAT. Based on the experiments, Lube et al proposed the following empirical relation linking the final runout  $r_\infty$  and the initial configuration

$$\frac{r_\infty}{r_0} = \begin{cases} 1 + 1.24a, & a < 1.7 \\ 1 + 1.6^{0.5}, & a \geq 1.7 \end{cases} \quad (34)$$

where  $r_0$  is the initial radius of the column,  $a = h_0/r_0$  is the aspect ratio with  $h_0$  the initial height.

All the simulation focus on a single configuration of the experiment with an aspect ratio of  $a = 0.5$ . The initial radius and height of the granular column are  $r_0 = 0.2$  m and  $h_0 = 0.1$  m, respectively. According to the empirical relation, the final runout in this case is  $r_\infty = 0.324$  m, which is used as the experimental reference value. In all the simulations, the Drucker–Prager model is employed. Although Lube et al concluded that the final runout is independent of frictional angle, many studies reported that a frictional angle of  $\phi = 37^\circ$  gives more accurate results, a lower frictional angle leads to overestimated runout distance [54, 59, 72]. Therefore, we choose to use  $\phi = 37^\circ$  in the simulations. The dilatancy angle  $\psi$  is taken as zero. The coefficient for artificial viscosity is chosen as  $\alpha = 0.25$ . The simulated physical time is one second.

Simulations with different resolutions and support domain size are performed. The radius of the support domain is  $2h$ , where  $h = \kappa\Delta p$ ,  $\Delta p$  is the particle size and  $\kappa$  is the parameter controlling the size of the support domain. Table 3 summaries the configurations for the simulations. In total 45 simulations are carried out, the number of particles in the simulations range from 23,772 to upto 15.7 millions. The 45 simulations are with five different particle sizes. To investigate the convergence behaviors, except  $\Delta p = 1$  mm, for each particle size eleven simulations are carried out with varying  $\kappa$  from 1.1 to 2.1. For brevity, some results are not shown in Table 3, but will be discussed in the remaining part of this section. The simulation with the finest resolution  $\Delta p = 1$  mm has an unusually large support domain of  $\kappa = 2.8$ , resulting in a huge

**Table 3** Summary of the simulations for the sand column collapse test with aspect ratio  $a = 0.5$ . FPS: frame per second, which is the number of computational steps the solver can execute in one second

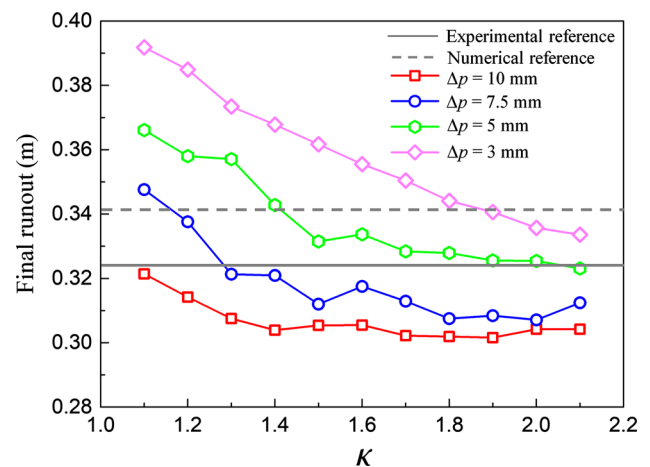
$\Delta p$ (mm)	$\kappa$	Number of particles	$r_\infty$ (m)	FPS ( $s^{-1}$ )	$T$ (hh:mm:ss)
10	1.1	23,772	0.3214	748.20	00:00:27
	1.2	23,826	0.3142	718.62	00:00:27
	...	...	...	...	...
7.5	2.1	37,116	0.3042	267.10	00:00:42
	1.1	50,668	0.3476	548.65	00:00:52
	1.2	50,750	0.3376	461.36	00:00:57
5	...	...	...	...	...
	2.1	72,926	0.3124	181.25	00:01:23
	1.1	143,044	0.3661	290.00	00:02:28
3	1.2	143,162	0.3580	244.58	00:02:42
	...	...	...	...	...
	2.1	189,692	0.3231	85.65	00:04:23
1	1.1	590,228	0.3918	76.31	00:15:42
	1.2	590,444	0.3849	60.42	00:18:11
	...	...	...	...	...
	2.1	712,542	0.3366	18.66	00:33:38
	2.8	15,666,190	0.3412	0.41	55:06:48

number of particles of around 15.7 million. The results from this simulation serve as the numerical reference solution.

### 6.1 Accuracy and convergence

The theoretical analysis of accuracy and convergence of SPH is more complex than traditional methods, because SPH has two processes of approximation: the continuous SPH integral interpolation and the discrete particle approximation. The convergence of the continuous integral integration requires  $h \rightarrow 0$ , which is natural as  $h$  denotes the numerical resolution of SPH. On the other hand, the discrete particle approximation is more accurate when there are more particles in the support domain, which means that its convergence requires  $\kappa = h/\Delta p \rightarrow \infty$  [88]. Therefore, the SPH method converges when  $h \rightarrow 0$  and  $\kappa \rightarrow \infty$ . Obviously, it is very difficult to satisfy these two conditions at the same time, as it leads to prohibitively large number of particles. Therefore, in most SPH simulations, convergence analysis is performed with fixed  $\kappa$  and varying smoothing length  $h$ , i.e.,  $h$  and  $\Delta p$  change proportionally. The common range of  $\kappa$  is from 1.2 to 1.6. This practice is sufficient in most numerical modeling, but not enough for fully investigating the behaviors of the method and our LOQUAT solver. To this end, we performed simulations of different resolutions with a wide range of  $\kappa$ .

The results of the final runout from all the simulations are shown in Fig. 4. Generally, for a particular resolution, a larger support domain (larger  $\kappa$ ) gives rise to a smaller runout. SPH is a continuous numerical method, where a larger support domain results in more accurate particle approximation hence better results in numerical simulation. However, as it can be observed in Fig. 4, the runouts reach stable values in simulations with  $\Delta p = 10, 7.5$ , and 5 mm if  $\kappa$  are larger than certain values. This change of trend happens at  $\kappa = 1.4, 1.5$  and 1.8 for the three resolutions. For  $\Delta p = 3$  mm, the final runout decreases monotonically with increasing  $\kappa$ . The observed phenomenon can be

**Fig. 4** The runout results for all the simulations

explained by checking the sources of errors. As aforementioned, the approximation error of SPH consists of a integral interpolation error related to  $h$ , and a discrete particle approximation error linked to  $\kappa$ . In simulations with a fixed  $h$ , increasing  $\kappa$  only reduces the particle approximation error, while the integral interpolation error is unchanged. Therefore, it can be observed that in simulations with  $h = 10, 7.5$  and  $5$  mm, when  $\kappa$  is respectively larger than  $1.4, 1.5$  and  $1.8$ , further increasing  $\kappa$  can not give more accurate results, as the numerical error is already dominated by the integral interpolation error, which only depends on  $h$ . For  $\Delta p = 3$  mm, even  $\kappa = 2.1$  cannot fully eliminate the particle approximation error. But the effect of increasing  $\kappa$  is close to “saturated”, as the curve show clear trend of converging.

The results show that in SPH simulation the resolution ( $h$ ) and support domain size ( $\kappa$ ) have to be carefully chosen. Only reducing  $h$  or increasing  $\kappa$  does not necessarily result in more accurate results. These two parameters need to be consistent to make sure that both the integral interpolation and particle approximation errors are minimized simultaneously.

From the above observation and analysis, we can tell that the simulations with  $h = 1$  mm and  $\kappa = 2.8$  give more accurate results than all the other simulations; therefore, it is used as the numerical reference case. It can be seen that the SPH method converges to this reference solution with decreasing  $h$  and increasing  $\kappa$ . However, the numerical reference value of the final runout is  $0.3412$  m, larger than that obtained in the experiments. This means that our numerical simulations do not converges to the experimental data. This discrepancy may be due to many reasons such as numerical formulations, implementation and the constitutive model. However, we believe that the most dominant reason of this discrepancy is that the quasi-static Drucker–Prager model is not sufficient for the dynamic process of granular collapse. In the Drucker–Prager model, the mechanism of energy dissipation is friction; however, in dynamic process of granular collapse, particle collision plays a significant role, it has big impact on the energy dissipation and the mechanical behavior of the material. Therefore, the Drucker–Prager model naturally overestimates the final runout. This overestimation was also noticed in other studies [54, 72], the reasons are, however, not discussed there. Consequently, for granular collapse modeling, constitutive models considering both the quasi-static and dynamic regimes should be more appropriate [2, 60, 65].

It is worth noting that some of our simulations give runout values quite close to the experimental one. e.g. when  $\Delta p = 10$  mm and  $\kappa = 1.1$ , or when  $\Delta p = 7.5$  mm and  $\kappa = 1.3$ , as observed in Fig. 4. However, this agreement is purely coincidence, because the numerical results

are very different from the converged solution and are inaccurate. Probably this type of agreement is also obtained in other studies, and it may happen that the corresponding results are considered as accurate solutions. This misinterpretation can lead to questionable conclusion, e.g. neglecting of the incapability of the constitutive model.

Using the numerical reference solution  $r_\infty^r$ , we can obtain the convergence rate of the simulations. The error is defined as  $e_r = |r_\infty^r - r_\infty|/|r_\infty^r|$ . Note that only results from simulations with  $\kappa \geq 1.8$  are used in the convergence analysis. For other values of  $\kappa$ , the numerical results from four resolutions are on both the upper and lower sides of the numerical reference solution, which complicates the analysis and interpretation. The results of the convergence analysis are given in Fig. 5. The rate of convergence is  $1.482$ , between linear and quadratic convergence. This is not surprising as the SPH scheme does not have the first-order consistency.

In all the simulations, the collapse process and the final profile of the sand body are similar, and are well-collaborated with experiments and other numerical studies [16, 42, 54, 72]. The final profiles of the simulation with the finest resolution of  $\Delta p = 1$  mm and  $\kappa = 2.8$  are shown in Fig. 6. It is observed that the sand column collapses axisymmetrically. A undisturbed region with the initial height can be clearly observed. The final profile shows good agreement with the experiments.

## 6.2 Efficiency

The efficiency of the simulation is measured using the frame per second (FPS), which is the number of computational steps executed in one second of wall-clock time. The FPS does only depend on total particle number, particle distribution, particle size  $\Delta p$  and  $\kappa$ , not affected by

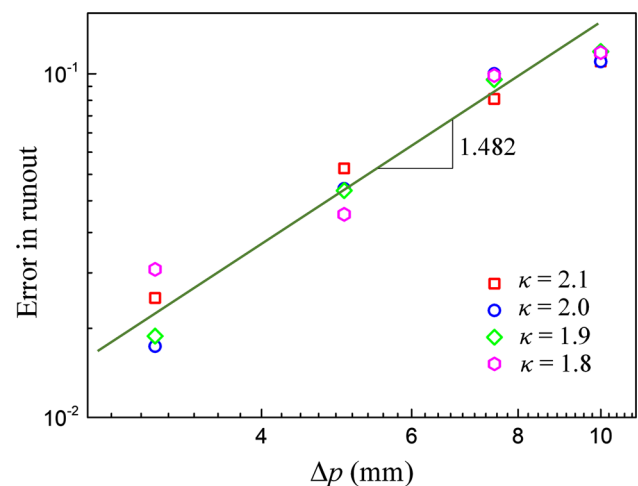
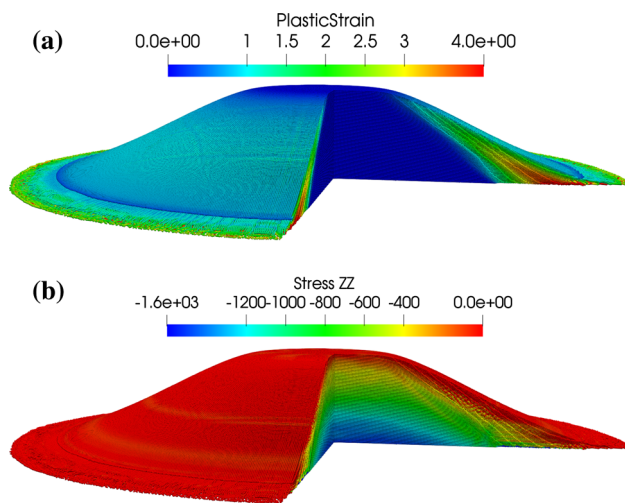


Fig. 5 The rate of convergence of the sand collapse simulation

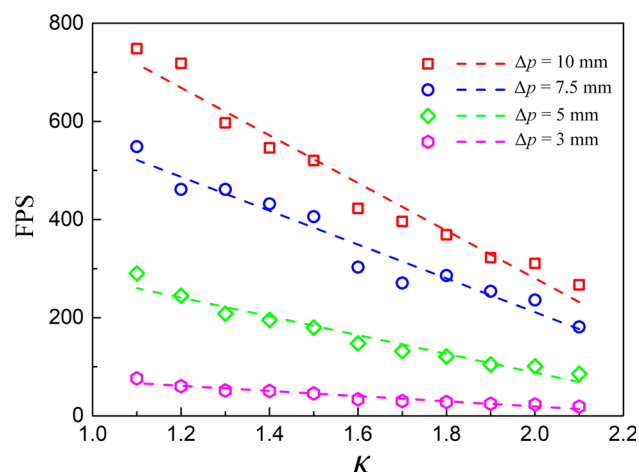




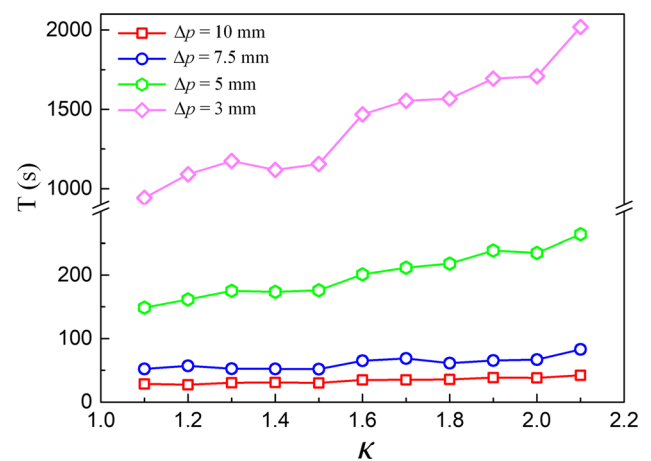
**Fig. 6** The final profile of the simulation of  $\Delta p = 1$  mm and  $\kappa = 2.8$  with a total particle number of 15.7 million: **a** equivalent plastic strain; **b** vertical stress

material parameters or the time step; thus, it is a proper measurement. The FPS and total simulation time for all the 45 simulations are listed in Table 3. The FPS and total simulation time of all the simulations except the finest one are also shown in Figs. 7 and 8. Note that the results of FPS and simulation time do not consider the time used for saving output files.

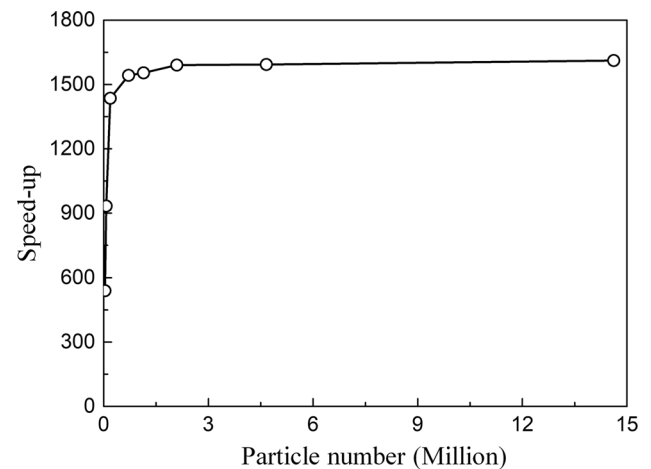
From Fig. 7, it is observed that the simulation is very fast using LOQUAT. Several hundreds of computational steps can be executed in one second of wall-clock time if the particle number is less than half a million. The FPS decreases with decreasing particle size, it also decreases if a higher  $\kappa$  is employed. This is expectable because larger number of particles and bigger support domain lead to higher computational cost in one step. For a certain particle size, the FPS drops linearly with growing  $\kappa$ . For instance, in the simulations of four particle sizes (10–3 mm), the



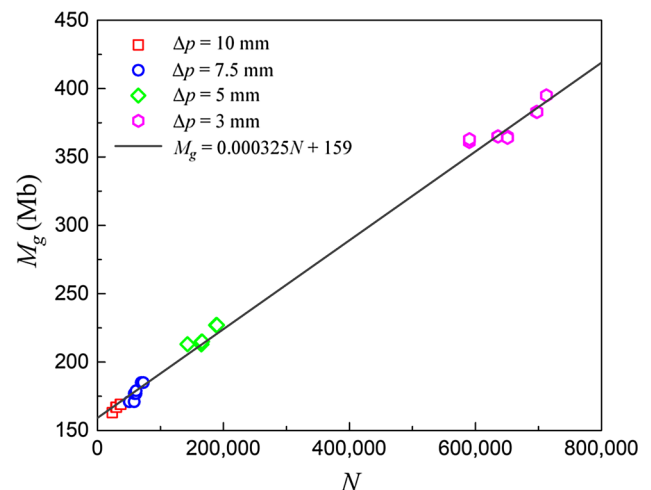
**Fig. 7** The FPS results from the 44 simulations



**Fig. 8** The simulation time used for the simulations



**Fig. 9** The speed-up of the GPU simulations over the single-core CPU simulations



**Fig. 10** The relation between GPU memory usage and total particle number

FPS at  $\kappa = 2.1$  is only 36%, 33%, 30% and 25% of that at  $\kappa = 1.1$ , indicating a fast grow in the computational cost in one step. However, this does not mean that the total simulation will be respectively 2.8, 3.0, 3.4 and 4.0 times longer. A larger  $\kappa$  gives rise to a larger smoothing length  $h$ , hence a bigger time step  $\Delta t$ . As a result, for the four resolutions, the simulation time is respectively 1.47, 1.58, 1.77 and 2.14 times longer for the simulations with the largest support domain compared to those with the smallest support domain. Therefore, contrary to the common belief, using high values of  $\kappa$  does not slow down the simulation much. Considering the additional gain on accuracy, large values of  $\kappa$  are always recommended.

As shown in Table 3 and Fig. 8, one second of simulation can be finished in around half an hour even with 712,542 particles and very large  $\kappa$ . The finest simulation with 15,666,190 particles and extremely large  $\kappa$  can be finished in around two days. It is demonstrated that with the GPU acceleration, LOQUAT is very efficient, and can be used in large scale three-dimensional geotechnical simulation.

To further check the acceleration of LOQUAT, we carry out simulations of 8 different resolutions with fixed  $\kappa = 2.1$ . The particle resolutions are  $\Delta p = 0.01, 0.0075, 0.005, 0.003, 0.002, 0.0015$ , and  $0.001$  m, resulting in numbers of particle ranging from 37,116 to 14,619,748. Both GPU-accelerated LOQUAT and a single-core CPU code are used for the simulations. The implementation of the single-core CPU code follows the same algorithms described in Sect. 4 without any parallelization. The CPU simulations are performed using an Intel Core i7-7700k CPU, which has a core clock of 4.2 GHz. Because the single-core CPU code is very slow in three-dimensional simulations with large number of particles, we only execute the first 100 computational steps and compute the FPS based on the measured time. Time used for input and output are not considered. Once obtaining the FPS of GPU and CPU simulations, the speed-up is computed as

$$s_u = \frac{\text{FPS}_{\text{inGPUsimulation}}}{\text{FPS}_{\text{inCPUsimulation}}} \quad (35)$$

All the parameters are the same in the GPU and CPU simulations.

The speed-up of LOQUAT is shown in Fig. 9. Hundreds to more than one thousand times of speed-up can be achieved with LOQUAT. The speed-up increases as more particles are used in the simulations, growing rapidly when the particle number is less than one million. The speed-up converges to around 1600 when the simulation scale is large enough. Traditionally, this high order of speed-up can only be realized using super-computer. However, with LOQUAT, a desktop with a mainstream graphic card is the only required hardware.

### 6.3 GPU memory usage

Memory usage is always an important concern in GPU acceleration, because the GPU memory is not easily extendable as conventional memory. We plot the GPU memory usage in all the simulations except the finest one in Fig. 10. It is found that the memory usage and the particle number have the following relation

$$M_g = 0.000325N + 159 \quad (36)$$

where  $M_g$  is the GPU memory usage, its unit is MB. With this relation, we predict that the finest simulation uses 4858.86 MB of GPU memory, which is quite close to the actual memory consumption (4905.02 MB). This indicates that the relation is applicable regardless of the simulation scale.

Recent mainstream graphic cards usually have 3–11 GB of GPU memory on board. As a result, LOQUAT can easily handle problems of dozens of millions of particles, making large scale three-dimensional simulation an easy task. It should be pointed out that in the current implementation, the Predictor-Corrector time integrator uses a lot of memory. If a time integrator requiring less memory is employed, even more particles can be simulated.

## 7 Applications

In this section, LOQUAT is employed to analysis the safety factor and the failure process of a exemplary slope. The geometry of the slope is shown in Fig. 11. Because LOQUAT is a three-dimensional solver, the slope is modeled as a 3D object with a thickness of 1.0 m. The particle motion in the  $y$  direction is restricted to mimic a plain strain condition. The material parameters of the slope are: Young's modulus  $E = 30.0$  MPa, Poisson ratio  $\nu = 0.2$ , friction angle  $\phi = 30.0^\circ$ , dilatancy angle  $\psi = 0^\circ$ , cohesion  $c = 5.0$  kPa and density  $\rho = 1850.0$  kg/m<sup>3</sup>. Both the Drucker–Prager and hypoplastic models are used in the simulation. The hypoplastic constitutive parameters are

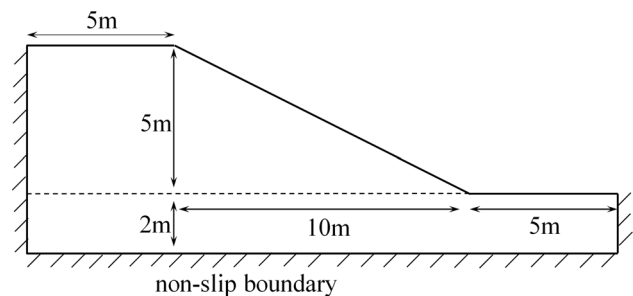
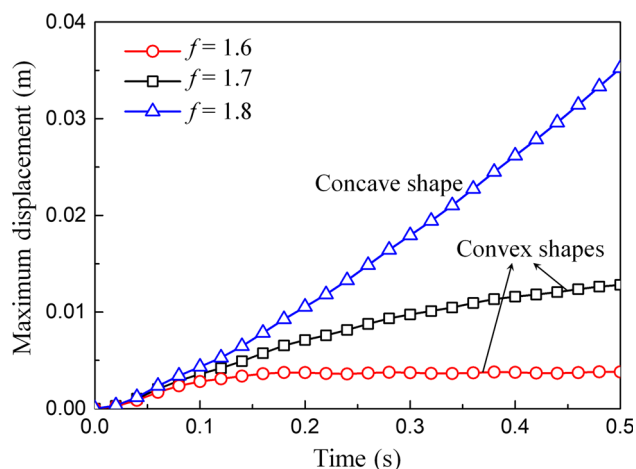


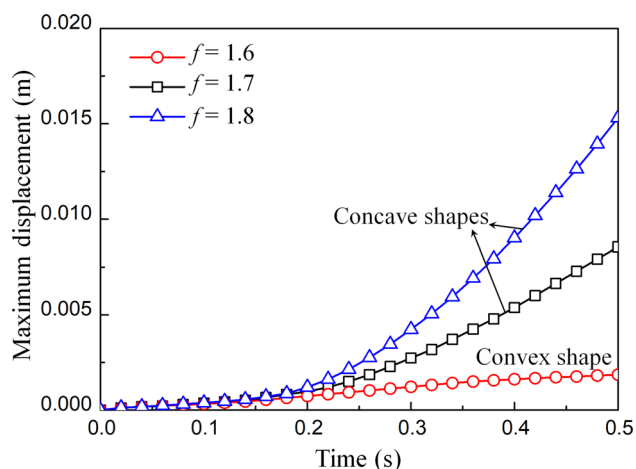
Fig. 11 Geometry and boundary conditions of the exemplary slope

obtained from the Drucker–Prager parameters following the procedure introduced in [81, 82].

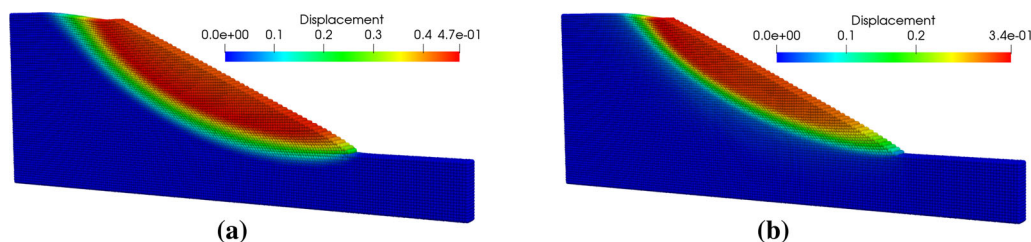
The slope with the original strength parameters is stable. A strength reduction method is used to evaluate the safety of the slope. The two strength parameters, i.e., friction angle  $\phi$  and cohesion  $c$  are reduced by a factor of  $f$ . The safety factor  $f_s$  is obtained by gradually increasing the reduction factor  $f$  until a convergence solution cannot be



**Fig. 12** The maximum displacement–time relation under different reduction factor with the Drucker–Prager model



**Fig. 13** The maximum displacement–time relation under different reduction factor with the hypoplastic model



**Fig. 14** The displacements under reduction factor 1.8 (in meters): **a** Drucker–Prager model, **b** hypoplastic model

achieved in a specific time (0.5 s in this work). During this specific time, if the time–maximum displacement curve remains convex, the simulation is considered convergence; otherwise, a concave curve represents an divergence simulation [11, 59].

All the simulations with the Drucker–Prager and the hypoplastic models employ the same numerical model. The particle size in the model is  $\Delta p = 0.1$  m and the smoothing length is  $h = 1.5\Delta p$ , leading to a total particle number of 60,624. The slope material is cohesive; therefore, artificial stress is applied with  $\epsilon = 0.2$ . Each simulation consists of two stages. The first stage is to obtain the initial stresses, in which the original unreduced strength parameters are used. A numerical damping scheme described in [9] is used to damp out the oscillation induced by the sudden imposition of the gravity. After an equilibrium state is reached, the stresses in the slope are kept unchanged but the strains and displacements are reset to zero. At the beginning of the second stage, the strength parameters are reduced according to the reduction factor  $f$ . Then the slope deforms under the gravity without numerical damping. The total physical time for the simulations is 6 s, including 0.5 s for obtaining the initial state and 5.5 s for slope failure computation.

Figures 12 and 13 show the results of the strength reduction analysis using the Drucker–Prager model and the hypoplastic model. The change of the maximum displacement in the first 0.5 s of the failure stage under different reduction factors are plotted. Three reduction factors are employed, i.e.,  $f = 1.6$ , 1.7 and 1.8. For the stability analysis using the Drucker–Prager model, The curves for reduction factor 1.6 and 1.7 are in convex shapes, indicating converged solutions according to the failure definition. The simulation with  $f = 1.8$  shows a concave shape in the displacement–time curve, which means that the simulation is divergent and the slope is unstable. Therefore, the safety factor should be between 1.7 and 1.8 according. On the other hand, from Fig. 13 it is observed that the simulations using the hypoplastic model show that the reduction factor 1.6 gives rise to converged solution, while 1.7 and 1.8 lead to failure. Therefore, the safety factor obtained using the hypoplastic model is between 1.6 and 1.7, lower than that obtained using the Drucker–Prager model. This

discrepancy is also observed in [59], where the safety factor obtained from the hypoplastic model is lower.

The final displacements of the slope under reduction factor 1.8 are shown in Fig. 14. Complete shear bands develop from the toe to the crest of the slope in both the simulations. Displacements concentrate in the sliding body above the shear bands. The simulation using the Drucker–Prager model gives a sliding surface deeper than that from the simulation using the hypoplastic model. This observation is well agreed with the findings in [59]. It should be pointed out that SPH is a continuum-based approach, and both the two constitutive models do not have any regularization regarding internal length of the material. Therefore, the SPH simulations suffer from the same mesh-dependency commonly found in FEM analysis. This means that the results of safety factor, the location and width of shear band are affected by the SPH numerical resolution, i.e., smoothing length  $h$ . The accurate modeling of strain localization is a hot topic in FEM, we believe that many approaches applicable in FEM can also be employed in SPH to overcome the mesh-dependency. However, this is beyond the scope of this work.

The physical time and the numerical model are the same for all the simulations using different constitutive models and reduction factors. As a result, the efficiency performance of LOQUAT in all the simulations are identical. For the 6 s physical time simulations with around one million particles, the used wall-clock time is around 100 s using the same GTX 1080Ti graphic card. The average FPS is 402, which means in one second of wall-clock time 402 computational steps can be executed.

## 8 Summary and future work

We present LOQUAT, an open-source 3D SPH solver for geotechnical modeling hoping to encourage the study and usage of SPH in geotechnical engineering. The solver uses standard SPH formulation with three stabilization options: artificial viscosity, artificial stress and stress normalization. Two soil constitutive models, i.e. the Drucker–Prager and the hypoplastic model from [75] are included in the solver. A convenient generalized boundary condition is present, which can handle static and moving solid boundaries with arbitrary geometry.

The performance of the presented solver is demonstrated using a series of numerical examples. The accuracy, convergence, efficiency and memory consumption of the solver are discussed. Specifically, LOQUAT can easily achieve more than one thousand times of speed-up compared with SPH code running on single CPU core, and is able to perform large-scale simulations with dozens of million of particles using modern GPU devices.

Future development of LOQUAT will include total-Lagrangian and updated-Lagrangian formulations, more sophisticated constitutive models, inconsistency treatment and soil-fluid interaction.

**Acknowledgements** The work carried out in this paper is partly funded by the project “MUMOLADE” within the ITN programme of the European Commission Seventh Framework Programme under Grant Number 289911, the Project “GEORAMP” within the RISE programme of the European Commission HORIZON 2020 programme under the Grant Number 645665, and Project 51709230 from National Natural Science Foundation of China. C. Peng and S. Wang are grateful for the financial support from the Otto Pregl Foundation for Fundamental Geotechnical Research in Vienna.

## References

- Adami S, Hu XY, Adams NA (2012) A generalized wall boundary condition for smoothed particle hydrodynamics. *J Comput Phys* 231(21):7057–7075
- Andrade JE, Chen QS, Le PH, Avila CF, Evans TM (2012) On the rheology of dilative granular media: bridging solid-and fluid-like behavior. *J Mech Phys Solids* 60(6):1122–1136
- Ayachit U (2015) The paraview guide: a parallel visualization application. Kitware, Clifton Park, USA
- Bardenhagen SG, Kober EM (2004) The generalized interpolation material point method. *Comput Model Eng Sci* 5(6):477–496
- Bardenhagen SG, Brackbill JU, Sulsky D (2000) The material-point method for granular materials. *Comput Methods Appl Mech Eng* 187(3–4):529–541
- Bi J, Zhou XP, Xu XM (2016) Numerical simulation of failure process of rock-like materials subjected to impact loads. *Int J Geomech* 17(3):04016073
- Blanc T, Pastor M (2012) A stabilized fractional step, Runge–Kutta Taylor SPH algorithm for coupled problems in geomechanics. *Comput Methods Appl Mech Eng* 221:41–53
- Bojanowski C (2014) Numerical modeling of large deformations in soil structure interaction problems using FE, EFG, SPH, and MM-ALE formulations. *Arch Appl Mech* 84(5):743–755
- Bui HH, Fukagawa R (2013) An improved sph method for saturated soils and its application to investigate the mechanisms of embankment failure: Case of hydrostatic pore-water pressure. *Int J Numer Anal Methods Geomech* 37(1):31–50
- 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 model. *Int J Numer Anal Methods Geomech* 32(12):1537–1570
- Bui HH, Fukagawa R, Sako K, Wells JC (2011) Slope stability analysis and discontinuous slope failure simulation by elastoplastic smoothed particle hydrodynamics (sph). *Geotechnique* 61(7):565–574
- Bui HH, Kodikara JK, Bouazza A, Haque A, Ranjith PG (2014) A novel computational approach for large deformation and post-failure analyses of segmental retaining wall systems. *Int J Numer Anal Methods Geomech* 38(13):1321–1340
- Cascini L, Cuomo S, Pastor M, Sorbino G, Piciullo L (2014) SPH run-out modelling of channelised landslides of the flow type. *Geomorphology* 214:502–513
- Chambon G, Bouvarel R, Laigle D, Naaim M (2011) Numerical simulations of granular free-surface flows using smoothed particle hydrodynamics. *J Non-Newton Fluid Mech* 166(12–13):698–712



15. Chen JY, Lien FS (2018) Simulations for soil explosion and its effects on structures using SPH method. *Int J Impact Eng* 112:41–51
16. Chen W, Qiu T (2011) Numerical simulations for large deformation of granular materials using smoothed particle hydrodynamics method. *Int J Geomech* 12(2):127–135
17. Crespo AJC, Domínguez JM, Rogers BD, Gómez-Gesteira M, Longshaw S, Canelas R, Vacondio R, Barreiro A, García-Feal O (2015) DualSPHysics: open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Comput Phys Commun* 187:204–216
18. Das R, Cleary PW (2010) Effect of rock shapes on brittle fracture using smoothed particle hydrodynamics. *Theor Appl Fract Mech* 53(1):47–60
19. de Souza Neto EA, Peric D, Owen DRJ (2011) Computational methods for plasticity: theory and applications. Wiley, Hoboken
20. Dehnen W, Aly H (2012) Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Mon Not R Astron Soc* 425(2):1068–1082
21. Domínguez JM, Crespo AJC, Valdez-Balderas D, Rogers BD, Gómez-Gesteira M (2013) New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Comput Phys Commun* 184(8):1848–1860
22. Drucker DC, Prager W (1952) Soil mechanics and plastic analysis or limit design. *Q Appl Math* 10(2):157–165
23. Gomez-Gesteira M, Rogers BD, Dalrymple RA, Crespo AJC (2010) State-of-the-art of classical SPH for free-surface flows. *J Hydraul Res* 48(S1):6–27
24. Gómez-Gesteira M, Crespo AJC, Rogers BD, Dalrymple RA, Domínguez JM, Barreiro A (2012) SPHysics-development of a free-surface fluid solver-Part 2: efficiency and test cases. *Comput Geosci* 48:300–307
25. Gomez-Gesteira M, Rogers BD, Crespo AJC, Dalrymple RA, Narayanaswamy M, Domínguez JM (2012) SPHysics-development of a free-surface fluid solver-Part 1: theory and formulations. *Comput Geosci* 48:289–299
26. Gray JP, Monaghan JJ, Swift RP (2001) SPH elastic dynamics. *Comput Methods Appl Mech Eng* 190(49–50):6641–6662
27. Green S (2013) Particle simulation using CUDA. NVIDIA, Santa Clara
28. Hérault A, Bilotta G, Vicari A, Rustico E, Del Negro C (2011) Numerical simulation of lava flow using a GPU SPH model. *Ann Geophys* 54(5):600–620
29. Hernquist L, Katz N (1989) TREESPH-A unification of SPH with the hierarchical tree method. *Astrophys J Suppl Ser* 70:419–446
30. Hockney RW, Eastwood JW (1988) Computer simulation using particles. CRC Press, Boca Raton
31. Hopkins PF (2017) A new public release of the GIZMO code. arXiv preprint [arXiv:1712.01294](https://arxiv.org/abs/1712.01294)
32. Hopkins PF (2015) A new class of accurate, mesh-free hydrodynamic simulation methods. *Mon Not R Astron Soc* 450(1):53–110
33. <http://www.grinninglizard.com/tinyxml/>. Accessed: 21 Aug 2018
34. <https://nvlabs.github.io/cub/>. Accessed 21 Aug 2018
35. Huang Y, Zhang WJ, Mao WW, Jin C (2011) Flow analysis of liquefied soils based on smoothed particle hydrodynamics. *Nat Hazards* 59(3):1547–1560
36. Huang Y, Zhang WJ, Xu Q, Xie P, Hao L (2012) Run-out analysis of flow-like landslides triggered by the Ms 8.0 2008 Wenchuan earthquake using smoothed particle hydrodynamics. *Landslides* 9(2):275–283
37. Korzani MG, Galindo-Torres SA, Scheuermann A, Williams DJ (2018) Sph approach for simulating hydro-mechanical processes with large deformations and variable permeabilities. *Acta Geotech* 13(2):303–316
38. Lewis RW, Schrefler BA (1998) The finite element method in the static and dynamic deformation and consolidation of porous media. Wiley, Chichester
39. Libersky LD, Petschek AG, Carney TC, Hipp JR, Allahdadi FA (1993) High strain lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response. *J Comput Phys* 109(1):67–75
40. Libersky LD, Randles PW, Carney TC, Dickinson DL (1997) Recent improvements in SPH modeling of hypervelocity impact. *Int J Impact Eng* 20(6–10):525–532
41. Liu GR, Liu MB (2003) Smoothed particle hydrodynamics: a meshfree particle method. World Scientific, Singapore
42. Lube G, Huppert HE, Sparks RSJ, Hallworth MA (2004) Axisymmetric collapses of granular columns. *J Fluid Mech* 508:175–199
43. Maeda K, Sakai H, Sakai M (2006) Development of seepage failure analysis method of ground with smoothed particle hydrodynamics. *Struct Eng/Earthq Eng* 23(2):307s–319s
44. McDougall S, Hungr O (2004) A model for the analysis of rapid landslide motion across three-dimensional terrain. *Can Geotech J* 41(6):1084–1097
45. Minatti L, Paris E (2015) A SPH model for the simulation of free surface granular flows in a dense regime. *Appl Math Model* 39(1):363–382
46. Monaghan JJ (1988) An introduction to SPH. *Comput Phys Commun* 48(1):89–96
47. Monaghan JJ (1992) Smoothed particle hydrodynamics. *Ann Rev Astron Astrophys* 30(1):543–574
48. Monaghan JJ (1994) Simulating free surface flows with SPH. *J Comput Phys* 110(2):399–406
49. Monaghan JJ (2000) SPH without a tensile instability. *J Comput Phys* 159(2):290–311
50. Monaghan JJ (2005) Smoothed particle hydrodynamics. *Rep Prog Phys* 68(8):1703
51. Morris JP, Fox PJ, Zhu Y (1997) Modeling low Reynolds number incompressible flows using SPH. *J Comput Phys* 136(1):214–226
52. Morris JP, Zhu Y, Fox PJ (1999) Parallel simulations of pore-scale flow through porous media. *Comput Geotech* 25(4):227–246
53. Naili M, Matsushima T, Yamada Y (2005) A 2D smoothed particle hydrodynamics method for liquefaction induced lateral spreading analysis. *J Appl Mech* 8:591–599
54. Neto AHF, Borja RI (2018) Continuum hydrodynamics of dry granular flows employing multiplicative elastoplasticity. *Acta Geotechn* 13:1–14
55. Nguyen CT, Nguyen CT, Bui HH, Nguyen GD, Fukagawa R (2017) A new SPH-based approach to simulation of granular flows using viscous damping and stress regularisation. *Landslides* 14(1):69–81
56. Oñate E, Idelsohn SR, Del Pin F, Aubry R (2004) The particle finite element method-an overview. *Int J Comput Methods* 1(02):267–307
57. Oñate E, Celigueta MA, Idelsohn SR, Salazar F, Suárez B (2011) Possibilities of the particle finite element method for fluid-soil-structure interaction problems. *Comput Mech* 48(3):307
58. Pastor M, Haddad B, Sorbino G, Cuomo S, Drempetic V (2009) A depth-integrated, coupled sph model for flow-like landslides and related phenomena. *Int J Numer Anal Methods Geomech* 33(2):143–172
59. 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
60. Peng C, Guo XG, Wu W, Wang YQ (2016) Unified modelling of granular media with smoothed particle hydrodynamics. *Acta Geotech* 11(6):1231–1247



61. Peng C, Xu GF, Wu W, Yu HS, Wang C (2017) Multiphase SPH modeling of free surface flow in porous media with variable porosity. *Comput Geotech* 81:239–248
62. Peng C, Wu W, Yu HS, Wang C (2015) Numerical simulation of free surface seepage in saturated soil using smoothed particle hydrodynamics. In: *Proceedings of IV international conference on particle-based methods, fundamentals and applications*
63. Peng C, Zhou MZ, Wu W (2017) Large deformation modeling of soil-machine interaction in clay. In: *International workshop on bifurcation and degradation in geomaterials*, pp 249–257. Springer
64. Potts DM, Zdravkovic L (2001) *Finite element analysis in geotechnical engineering: application*. Thomas Telford, London
65. Prime N, Dufour F, Darve F (2013) Unified model for geomaterial solid/fluid states and the transition in between. *J Eng Mech* 140(6):04014031
66. Robinson M (2009) *Turbulence and viscous mixing using smoothed particle hydrodynamics*. PhD thesis, Monash University
67. Schoenberg IJ (1946) Contributions to the problem of approximation of equidistant data by analytic functions, Part A. On the problem of smoothing or graduation, a first class of analytic approximation formulas. *Q Appl Math* 4(2):112–141
68. Shadloo MS, Oger G, Le Touzé D (2016) Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: motivations, current state, and challenges. *Comput Fluids* 136:11–34
69. Springel V (2005) The cosmological simulation code GADGET-2. *Mon Not R Astron Soc* 364(4):1105–1134
70. Springel V (2010) Smoothed particle hydrodynamics in astrophysics. *Ann Rev Astron Astrophys* 48:391–430
71. Springel V, Yoshida N, White SDM (2001) GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astron* 6(2):79–117
72. Szwec K (2017) Smoothed particle hydrodynamics modeling of granular column collapse. *Granul Matter* 19(1):3
73. Violeau D, Rogers BD (2016) Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future. *J Hydraul Res* 54(1):1–26
74. Wadsley JW, Stadel J, Quinn T (2004) Gasoline: a flexible, parallel implementation of treesph. *New Astron* 9(2):137–158
75. Wang XT, Wu W (2011) An updated hypoplastic constitutive model, its implementation and application. In: Wan R, Alsaleh M, Labuz J (eds) *Bifurcations, instabilities and degradations in geomaterials*, pp 133–143. Springer, Berlin, Heidelberg
76. Wang J, Chan D (2014) Frictional contact algorithms in SPH for the simulation of soil-structure interaction. *Int J Numer Anal Methods Geomech* 38(7):747–770
77. Wang C, Wang YQ, Peng C, Meng XN (2016) Smoothed particle hydrodynamics simulation of water-soil mixture flows. *J Hydraul Eng* 142(10):04016032
78. Wang C, Wang YQ, Peng C, Meng XN (2017) Dilatancy and compaction effects on the submerged granular column collapse. *Phys Fluids* 29(10):103307
79. Wendland H (1995) Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv Comput Math* 4(1):389–396
80. Wiecekowski Z (2004) The material point method in large strain engineering problems. *Comput Methods Appl Mech Eng* 193(39–41):4417–4438
81. Wu W, Bauer E (1994) A simple hypoplastic constitutive model for sand. *Int J Numer Anal Methods Geomech* 18(12):833–862
82. Wu W, Bauer E, Kolymbas D (1996) Hypoplastic constitutive model with critical state for granular materials. *Mech Mater* 23(1):45–69
83. Wu W, Lin J, Wang XT (2017) A basic hypoplastic constitutive model for sand. *Acta Geotech* 12(6):1373–1382
84. Zhan L, Peng C, Zhang BY, Wu W (2019) A stabilized TL-WC SPH approach with GPU acceleration for three-dimensional fluid-structure interaction. *J Fluids Struct* 86:329–353
85. Zhang XQ, Zhou XP (2018) Analysis of the numerical stability of soil slope using virtual-bond general particle dynamics. *Eng Geol* 243:101–110
86. Zhang WJ, Maeda K, Saito H, Li ZQ, Huang Y (2016) Numerical analysis on seepage failures of dike due to water level-up and rainfall using a water-soil-coupled smoothed particle hydrodynamics model. *Acta Geotech* 11(6):1401–1418
87. Zhou XP, Bi J, Qian QH (2015) Numerical simulation of crack growth and coalescence in rock-like materials containing multiple pre-existing flaws. *Rock Mech Rock Eng* 48(3):1097–1114
88. Zhu QR, Hernquist L, Li YX (2015) Numerical convergence in smoothed particle hydrodynamics. *Astrophys J* 800(1):6
89. Zienkiewicz OC, Chan AHC, Pastor M, Schrefler BA, Shiomi T (1999) *Computational geomechanics*. Citeseer, New York

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