Computers and Geotechnics 121 (2020) 103474

Contents lists available at ScienceDirect

Computers and Geotechnics

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

Research Paper

A scalable parallel computing SPH framework for predictions of geophysical granular flows

Edward Yang[a,b], Ha H. Bui[a,b,*], Hans De Sterck[d], Giang D. Nguyen[c], Abdelmalek Bouazza[b]

[a] Monash Computational Geomechanics (MCG) Lab, Monash University, Australia
[b] Department of Civil Engineering, Monash University, Australia
[c] School of Civil, Environmental & Mining Engineering, University of Adelaide, Australia
[d] Department of Applied Mathematics, University of Waterloo, Canada

ARTICLE INFO

Keywords:
Parallel computing
Message Passing Interface (MPI)
Smoothed Particle Hydrodynamics (SPH)
Granular flows
Geophysical flows

ABSTRACT

This paper presents a parallel computing Smoothed Particle Hydrodynamics (SPH) framework for geophysical granular flows scalable on large CPU clusters. The framework is accomplished by adopting a Message Passing Interface (MPI) approach with domain partitioning strategy. The Orthogonal Recursive Bisection (ORB) technique is utilised to subdivide the computational domain. The ORB algorithm is implemented such that any number of MPI processes can be used instead of being limited to powers of two. To avoid global communications in the particle distribution process, a diffusion-based distribution algorithm is implemented and demonstrated to be much faster than global communication approaches when distributing particles to non-neighbouring processes. The proposed parallel scheme achieves 95% weak scaling efficiency and up to 900 times strong scaling speedup on 1024 CPU cores. The parallel scheme enables previously unfeasible simulations to be carried out and here we apply it to the investigation of the granular column collapse experiment under full three-dimensional, axisymmetric conditions for aspect ratios up to 30, not attempted previously using numerical techniques in the literature. Enabled by the parallel scheme, the simulations use up to 11.7 million SPH particles. The investigation is conducted using two popular constitutive models commonly used in modelling of granular flows: the elasto-plastic model with Drucker-Prager yield criterion and the $\mu(I)$ rheological model. While very good agreement with experimental data has been reported for both models for small and intermediate aspect ratios, the large-scale simulations conducted for large aspect ratios show that the Drucker-Prager model tends to over-predict final deposit height, and the $\mu(I)$ model under-predicts it. Furthermore, due to the capability of the parallel scheme to model the 3D axisymmetric column collapse at higher resolutions, we demonstrate that the elasto-plastic approach is capable of capturing arching effects in the stress profile, whereas the $\mu(I)$ model cannot.

1. Introduction

Geophysical granular flows are of importance to geotechnical engineers due to the hazard of post-failure flows that can affect communities around the world. The 2008 Wenchuan earthquake that triggered thousands of landslides in Sichuan, China, which resulted in almost 20,000 deaths in the region is an example of such hazards [1]. Another example is the persistent high-frequency landslide events in New Zealand, which lead to an estimated annual cost of NZ$250–300 million in direct costs in addition to indirect costs like road closures and loss of productivity [2]. Because these types of post-failure events are large in scale, it is important to develop numerical tools that are capable of not only capturing the physics of failure flows related to these events, but also possessing computational power to do so.

Numerical modelling tools used in conjunction with traditional techniques have been popular in the study or prediction of flow behaviours to reduce damages or loss of life as a consequence of landslide events [3,4]. However, the computational cost of modelling Geophysical granular flows in three dimensions is very high. One choice to overcome this issue, is to model the problem in two spatial dimensions rather than three [4–11]. However, this can result in unrealistic predicted behaviour [12]. Another choice is to drastically reduce computation times by introducing parallel computing capabilities, thus enabling researchers to conduct full 3D simulations of more commonplace granular flows in geotechnical engineering using either mesh-based or mesh-less methods. A commonly used example of a mesh-based method

* Corresponding author at: Department of Civil Engineering, Monash University, Australia.
E-mail address: Ha.Bui@monash.edu (H.H. Bui).

https://doi.org/10.1016/j.compgeo.2020.103474
Received 6 September 2019; Received in revised form 10 November 2019; Accepted 29 January 2020
Available online 29 February 2020

in geotechnical engineering is the Finite Element Method (FEM). However, geological flows present a challenge to traditional FEM since large deformations can lead to severe mesh distortions which require the application of adaptive re-meshing techniques. Particle-based methods aim to address the shortcomings of mesh-based methods in the context of large deformation applications by using material points to store material information and using a background mesh to solve for the motion of the material. Prime examples of these methods are the Material Point method (MPM) and Particle Finite Method (PFM) [13–15]. The advantage of such methods are that they can overcome mesh-distortion issues that can occur in mesh-based methods, while retaining other useful features of mesh-based methods. These methods have been applied extensively to geotechnical engineering and granular flow applications [16–18].

Mesh-less methods, as the name suggests, do not rely on a mesh to perform computations and are free of challenges associated with mesh-based methods. Of the mesh-less numerical methods, the Discrete Element Method (DEM) is popular because it models material as discrete particles and then solves for the motion of particles using particle interaction forces relying on fewer constitutive behavioural assumptions [19–22]. However, DEM is a comparatively computationally expensive numerical method and using DEM to model runout events with sufficiently realistic particle diameters, even with the introduction of parallelism on large-scale computing systems, is not practicable to the best of our knowledge. In this sense, a truly continuum-based mesh-less method has clear advantages in terms of computational efficiency and can help overcome the above mentioned issues if enhanced with parallel algorithms. The Smoothed Particle Hydrodynamics (SPH) numerical method has demonstrated great potential in not only simulations of granular flows [11,23–28], but a wide range of other applications in geomechanics [22,29–31] and will be used as a basis for further development in this study.

SPH discretises a problem domain into a set of particles and allows for the determination of field quantities using a kernel approximation based on statistical interpolation techniques. These field quantity approximations can then be used to solve for the Lagrangian motion of the particles. The method was originally applied to astrophysical simulations [32,33] and found application in computational fluid dynamics [34]. SPH was also found capable of modelling solids [35] and granular materials [23]. SPH has been shown to be useful in related aspects of geotechnical modelling such as slope failure [36–38], soil-water coupling [29,39], and soil-structure interaction [24,25,28]. The versatility and suitability of SPH in geological hazard modelling can be demonstrated further by existing applications such as work done in [10] which uses depth-integrated equations to model saturated debris flow. Examples of full-3D modelling of geological hazards include application to lava flow modelling including thermal coupling and solidification behaviour [8]. SPH has also been shown to be compatible with visco-plastic constitutive models, representative of the fluid regime of granular material [26,40,41]. The key advantages of SPH over traditional techniques is the absence of any underlying mesh, its relative computational inexpensiveness, and versatility in being able to include various constitutive behaviours. However, a serial SPH code, despite relative inexpensiveness of the method, can only model flows at very low resolutions in practical time-frames, therefore introducing resolution based approximation error as well as sacrificing the influence of surface detail on geological flow behaviour. Therefore, parallel computing is required such that geological flows can be modelled at sufficient resolution.

While in the literature parallelism has already been introduced to geomaterial applications of SPH, existing schemes rely on single compute node approaches such as single Graphics Processing Units (GPUs) approaches or shared memory schemes such as OpenMP [42,43]. Currently, single high-end GPUs are capable of simulating ten or more millions of SPH particles. However, when simulating increasingly larger problems, single compute nodes eventually become constrained by available memory and/or compute power. One avenue to overcome this challenge, is to employ distributed-memory parallelism using the Message Passing Interface (MPI) communication standard, allowing for computations to be conducted in parallel on multiple compute nodes, therefore introducing more memory and computational power. MPI can be utilized either on its own or hybridised with OpenMP [44], or in multi-GPU approaches that use MPI to transfer information between compute nodes [45]. In this paper, we develop a pure scalable MPI framework implementation on CPU clusters. Many modern supercomputers feature powerful CPU nodes with high-end GPU back-ends on each node. These GPU back-ends are an attractive option for further speeding up the SPH code, and we defer extension of the MPI-based code to using GPU back-ends to future work. The most common approach to parallelising classical SPH on distributed memory computers is to separate the problem domain into distinct subdomains and allocate a subdomain and the contained particles in it to a compute core and associated MPI process. Halo layers of particles are then distributed between subdomains in each time-step so that field variables of particles contained within each subdomain can be computed in parallel. Because of the computational overhead and communications associated with distributing halo layers, it is generally desirable for the geometry of subdomains to be such that the interfacial surface area between them is minimised and does not grow for increased problem sizes or number of subdomains.

From a large-scale numerical modelling viewpoint, the axisymmetric granular column collapse experiment is a challenge, for which a parallelised numerical tool is needed. The experiment itself is a granular flow problem that can be related to the physics of gravity-driven geological hazards such as landsides, or to a cliff-face collapse and subsequent spreading of material over a horizontal surface [46,47]. While this experiment has been studied through numerical means, numerical modelling is often restricted to the two-dimensional case due to computational cost. And when the three-dimensional axisymmetric version of the experiment is modelled, only small to intermediate aspect ratios can be modelled for the same reason [28,47–50]. A shortcoming of being restricted to two-dimensional studies is that the spreading behaviour of the three-dimensional axisymmetric case is different when compared to the 2D case, and when considering realistic geophysical granular flows, they do not often flow under channelized or quasi-two-dimensional conditions, and typically spreading occurs in many directions. Furthermore, at large aspect ratios, the collapse of the granular column exhibits complex behaviours that can be difficult to characterise using analytical approaches and could have potential implications for hazard prediction. While large aspect ratios can be modelled by coarsening the spatial discretisation, a small ratio of problem size to Representative Volume Element (RVE) size (i.e. an SPH particle) may significantly reduce the accuracy. Furthermore, it can introduce error when predicting runout distances, as well as sacrificing important detail in the kinematics, such as small free-surface waves, and interfaces between quasi-static deposited material and dynamic flowing material [51].

In this paper, we present a parallelised SPH scheme targeting for scalability on large-scale compute clusters with more than 1,000 CPU cores as well as the first application of such a scheme to granular flows. The scheme uses the Orthogonal Recursive Bisection (ORB) domain partitioning technique for short-range interaction particle methods such as in [44,52], which is enhanced in several ways. The ORB technique is chosen as it offers a good trade-off in terms of simplicity and minimization of interfacial surface area between process subdomains, therefore facilitating scalability in the overall parallel scheme. Additionally, scalability is achieved by maintaining particle information exchange between neighbours only. To reduce global communications, a diffusion-based algorithm is introduced for distributing particle information between non-neighbouring processes [53]. While a similar ORB-based parallel scheme has been introduced for water flows in SPH in [44], this paper applies the scheme to granular flows, with two more complex constitutive models representative of granular flows combined with SPH. The first model is the $\mu(I)$ rheological model [54], a popular

example of the visco-plastic fluid approach of modelling granular flows, and the second model is the Drucker-Prager elastic-perfectly plastic model presented by Bui et al. [23] which is a popular example of an elasto-plastic approach to granular flows in SPH. The following three reasons motivate the work presented in this paper. First, while both approaches have been used in the literature [41,55–57], neither have been parallelised for large-scale compute clusters. Second, despite their popularity, a direct comparison has not been made between them. And finally, while both models have been applied to the granular column collapse experiment, simulations have yet to be conducted for very high aspect ratios under axisymmetric conditions. We present the parallelisation of both models to demonstrate the general applicability of the parallel scheme to granular flows modelled by SPH. From here on, the code variant that uses the $\mu(I)$ model [54] will be referred to as the $\mu(I)$ variant, and the code variant that uses the Drucker-Prager model will be referred to as the DP variant.

2. Numerical scheme

2.1. Governing equations

In continuum mechanics, the mass and momentum conservation equations are the governing equations describing the motion of a material. These can be applied to model granular flows and are given as:

$$\frac{D\rho}{Dt} = -\frac{1}{\rho}\frac{\partial v^{\alpha}}{\partial x^{\alpha}}, \tag{1}$$

$$\frac{Dv^{\alpha}}{Dt} = \frac{1}{\rho}\frac{\partial \sigma^{\alpha\beta}}{\partial x^{\beta}} + f^{\alpha}, \tag{2}$$

where Einstein summation is used for repeated indices; α and β correspond to the Cartesian x, y, z coordinates; ρ is the material density; v is the velocity; x is the Cartesian position; and $\sigma^{\alpha\beta}$ corresponds to the Cauchy stress tensor.

To close the above governing equations, a formulation for the stress tensor is required. The traditional approach to formulating the stress tensor is as follows:

$$\sigma^{\alpha\beta} = -P\delta^{\alpha\beta} + \tau^{\alpha\beta}, \tag{3}$$

where P is an isotropic pressure; $\tau^{\alpha\beta}$ is the shear stress tensor and $\delta^{\alpha\beta}$ is Kronecker's delta. In this paper, we adopt two different approaches to identify the stress tensor: the $\mu(I)$ rheology as described in [54] and the Drucker-Prager elastic-perfectly plastic model as implemented in SPH in [23]. This section will briefly outline each model.

2.2. Constitutive models

To formulate the stress tensor for granular flows, numerical modelling practitioners generally adopt either a visco-plastic or elasto-plastic approach. For the visco-plastic approach, we implement the $\mu(I)$ model [54], which can be thought of as an extension of the Bingham fluid model with pressure and friction dependent yield stress for application to dense granular flows. As for the elasto-plastic approach, the approach presented in [23] is adopted herein. This section will first describe the model and then the elasto-plastic approach.

2.2.1. $\mu(I)$ rheological constitutive model

The $\mu(I)$ rheological model was developed to capture the inertial effects of granular material in the dense regime. It models the granular material as a visco-plastic fluid, with a Drucker-Prager like yielding condition. When the stress state of the material is beneath the yielding stress, the material does not flow. Above the yielding stress, the material flows like a non-Newtonian fluid. To formulate the stress tensor in Eq. (3), the implementation of the $\mu(I)$ model adopts the following equation of state to represent the isotropic pressure:

$$P_i = c^2(\rho_i - \rho_0), \tag{4}$$

where c is the speed of sound, ρ_i is the density of particle i, and ρ_0 is the reference density of the material. The shear stress tensor is determined by,

$$\tau^{\alpha\beta} = 2\eta\dot{\varepsilon}^{\alpha\beta}, \tag{5}$$

$$\eta = \frac{\mu P}{\sqrt{\dot{\varepsilon}^{\alpha\beta}\dot{\varepsilon}^{\alpha\beta}}}, \tag{6}$$

$$\mu = \mu_s + \frac{\mu_p - \mu_s}{I_0/I_i + 1}, \tag{7}$$

where η is an apparent viscosity; μ is a frictional function dependent on the inertial number, $I_i = \frac{d\sqrt{\dot{\varepsilon}_i^{\alpha\beta}\dot{\varepsilon}_i^{\alpha\beta}}}{\sqrt{P_i/\rho_0}}$; $\dot{\varepsilon}_i^{\alpha\beta} = \frac{1}{2}\left(\frac{\partial v^{\alpha}}{\partial x^{\beta}} + \frac{\partial v^{\beta}}{\partial x^{\alpha}}\right)_i$ is the strain rate; d is the real grain diameter; and μ_s, μ_p, and I_0 are constants. The model includes a Drucker Prager-like yield criterion such that no flow occurs when $\sqrt{\frac{1}{2}\tau^{\alpha\beta}\tau^{\alpha\beta}} \leq \mu_s P$ [54]. To avoid undesirable or undefined behaviours, the shear component of the stress tensor is assumed to equal 0 when the pressure is negative, and the strain rate tensor is initialized as 10^{-7} as zero strain rates can result in mathematically undefined behaviour. It is also worth noting that the apparent viscosity does not require other regularizations that are typically implemented in non-Newtonian fluid models. This is because despite the local viscosity, η, diverging for a vanishing strain-rate, the shear stress is bounded by $\sqrt{\frac{1}{2}\tau^{\alpha\beta}\tau^{\alpha\beta}} = [\mu_s P, \mu_p P]\ \forall\ \sqrt{\dot{\varepsilon}^{\alpha\beta}\dot{\varepsilon}^{\alpha\beta}} \neq 0$.

2.2.2. Drucker-Prager elastic-perfectly plastic constitutive model

While the visco-plastic approach of modelling granular flows derive the stress tensor from instantaneous density and strain-rate, the elasto-plastic approach instead evolves the stress tensor over time using a stress-strain relationship that relates the stress-increment to the strain-increment. Plasticity theory dictates that for an elasto-plastic material, the total strain-rate tensor is decomposed into elastic, $\dot{\varepsilon}_e^{\alpha\beta}$, and plastic components, $\dot{\varepsilon}_p^{\alpha\beta}$:

$$\dot{\varepsilon}^{\alpha\beta} = \dot{\varepsilon}_e^{\alpha\beta} + \dot{\varepsilon}_p^{\alpha\beta}, \tag{8}$$

where the elastic part is determined by the generalized Hooke's Law:

$$\dot{\varepsilon}_e^{\alpha\beta} = \frac{\dot{s}^{\alpha\beta}}{2G} + \frac{1-2\nu}{3E}\dot{\sigma}^{\gamma\gamma}\delta^{\alpha\beta}, \tag{9}$$

where $\dot{s}^{\alpha\beta}$ is the deviatoric shear stress-rate tensor, E and G are Young's and shear modulus respectively, and ν is Poisson's ratio.

The plastic component of the strain-rate tensor can be defined using the plastic flow rule:

$$\dot{\varepsilon}_p^{\alpha\beta} = \dot{\lambda}\frac{\partial g}{\partial \sigma^{\alpha\beta}}, \tag{10}$$

where $\dot{\lambda}$ is the rate of the change of the plastic-multiplier, λ; and g is the plastic potential function. Plastic deformation only occurs when the yield criterion is satisfied where the Drucker-Prager criterion is adopted:

$$f(I_1, J_2) = \alpha_{\phi} I_1 + \sqrt{J_2} - k_c = 0, \tag{11}$$

where I_1 and J_2 are the first and second invariants of the stress tensor; α_{ϕ} and k_c are Drucker-Prager constants related to the Coulomb internal friction angle (ϕ) and cohesion (c) respectively; α_{ϕ} and k_c are respectively calculated in three dimensions as:

$$\alpha_{\phi} = \frac{2\sin\phi}{\sqrt{3}(3 - \sin\phi)}, \tag{12}$$

$$k_c = \frac{6c \times \cos\phi}{\sqrt{3}(3 - \sin\phi)}, \tag{13}$$

The non-associated plastic flow rule is used to define the plastic flow rule:

$$g = \alpha_\psi I_1 + \sqrt{J_2} - k_c, \tag{14}$$

where α_ψ is a Drucker-Prager constant and is determined the same as Eq. (12) where ϕ is substituted for the dilation angle, ψ.

Eq. (8) can be closed after substituting Eq. (14) into Eq. (10), and then Eqs. (9) and (10) into (8). By rearranging Eq. (8) and adopting the Jaumann stress-rate tensor for the large deformation treatment, the final stress-strain relationship becomes:

$$\frac{D\sigma^{\alpha\beta}}{Dt} = \sigma^{\alpha\gamma}\dot{\omega}^{\beta\gamma} + \sigma^{\gamma\beta}\dot{\omega}^{\alpha\gamma} + 2G\dot{e}^{\alpha\beta} + K\dot{\varepsilon}^{\gamma\gamma}\delta^{\alpha\beta} - \dot{\lambda}\left[3K\alpha_\psi\delta^{\alpha\beta} + \frac{G}{\sqrt{J_2}}s^{\alpha\beta}\right], \tag{15}$$

where $\dot{\omega} = \frac{1}{2}\left(\frac{\partial v^\alpha}{\partial x^\beta} - \frac{\partial v^\beta}{\partial x^\alpha}\right)$ is the spin rate tensor, $\dot{e}^{\alpha\beta} = \dot{\varepsilon}^{\alpha\beta} - \frac{1}{3}\dot{\varepsilon}^{\gamma\gamma}\delta^{\alpha\beta}$ is the deviatoric strain-rate tensor, and K is the bulk modulus. For non-associated flow, the rate of change of the elastic multiplier, $\dot{\lambda}$, is:

$$\dot{\lambda} = \frac{3\alpha_\phi K\dot{\varepsilon}^{\gamma\gamma} + (G/\sqrt{J_2})s^{\alpha\beta}\dot{\varepsilon}^{\alpha\beta}}{9\alpha_\phi\alpha_\psi K + G}. \tag{16}$$

The numerical issues of tensile cracking and imperfect plasticity are overcome by the same stress-return algorithm described in [23]. For finite deformations, the elasto-plastic constitutive relations adopted in this paper lose incremental objectivity in the strain and stress measures, despite using the objective Jaumann stress rate. However owing to the relatively small time-step adopted in this paper, deformations are sufficiently small to minimize inaccuracies [58].

2.3. SPH Formulations

To solve the governing equations Eqs. (1) and (2) using SPH, the problem domain is first discretised into a finite number of particles. These particles are associated with field variables such as mass, velocity, stress etc. Then, field quantities can be determined by using the kernel approximation of a field function:

$$\langle f(\mathbf{x})\rangle = \int_\Omega f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)d\mathbf{x}', \tag{17}$$

where $W(\mathbf{x}-\mathbf{x}', h)$ is the kernel function, and h is the smoothing length. For values of $|\mathbf{x}-\mathbf{x}'| > \kappa h$, $W = 0$, where κ is a constant which defines the support domain for the kernel function at $\mathbf{x}$. There are many choices of kernel function in the literature, and the kernel function chosen in this paper is the cubic spline [59]. The cubic spline possesses a good balance of computational efficiency, accuracy, and stability, and is taken as the following:

$$W_{ij} = \alpha_d \times \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & , 0 \le q < 1 \\ \frac{1}{6}(2-q)^3 & , 1 \le q < 2 \\ 0 & , q \ge 2 \end{cases} \tag{18}$$

where $\alpha_d = \frac{1}{\pi h^3}$ in three dimensions and $q = \frac{r_{ij}}{h}$. For the cubic spline, $\kappa = 2$, and $h = 1.2dx$, where dx is the initial inter-particle spacing, resulting in a kernel radius of $2.4dx$. Once the kernel has been defined, Equation (17) as well as its spatial gradient can be respectively approximated by

$$f(\mathbf{x}_i) = \sum_j f(\mathbf{x}_j)W_{ij}\frac{m_j}{\rho_j}, \tag{19}$$

$$\frac{\partial f(\mathbf{x}_i)}{\partial \mathbf{x}} = \sum_j f(\mathbf{x}_j)\cdot\frac{\partial W_{ij}}{\partial \mathbf{x}_i}\frac{m_j}{\rho_j}, \tag{20}$$

where i corresponds to particle indices; j corresponds to neighbouring particles of i located within i's support domain; $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$; and $\frac{\partial W_{ij}}{\partial \mathbf{x}_i} = \left(\frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}\right)\frac{\partial W_{ij}}{\partial r}$. The governing equations can then be discretised in SPH as follows:

$$\left(\frac{D\rho}{Dt}\right)_i = \rho_i\sum v_{ij}^\alpha\cdot\frac{\partial W_{ij}}{\partial x_i^\alpha}\frac{m_j}{\rho_j}, \tag{21}$$

$$\left(\frac{Dv^\alpha}{Dt}\right)_i = \sum_j\left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x_i^\alpha}\frac{m_j}{\rho_j} + \alpha_\pi hc_0\rho_0\sum_j\pi_{ij}\frac{\partial W_{ij}}{\partial x_i^\alpha}\frac{m_j}{\rho_i\rho_j}, \tag{22}$$

where α_π is a tunable parameter, h is the kernel smoothing length, c_0 is a reference speed of sound, ρ_0 is the reference material density, and $v_{ij}^\alpha = v_i^\alpha - v_j^\alpha$. In the $\mu(I)$ variant, $\pi_{ij} = \frac{v_{ji}^\alpha\cdot r_{ji}^\alpha}{|r_{ij}|^2}$, and α_π is a tunable parameter, chosen to equal 0.01 [60]. In the DP variant, $\pi_{ij} = \frac{\langle v_{ji}^\alpha\cdot r_{ji}^\alpha\rangle}{|r_{ij}|^2}$ where $\langle\cdot\rangle$ denotes the Macaulay bracket, and α_π is taken as 0.1 [23]. The use of the Macaulay bracket, as recommended by Monaghan [34], is to ensure the artificial viscosity stabilizes the scheme rather than introducing viscosity.

The traditional discretisation of the governing equations, when coupled with a state equation, results in the well documented problem of noise in the pressure profile due to short length-length-scale noise [61]. As the pressure term in the $\mu(I)$ rheological model is determined by the state equation (4), the model's kinematics suffer significantly from the pressure noise. To resolve this pressure noise problem, Molteni and Colagrossi [60] introduced a density diffusion term into the mass conservation equation to smoothen the pressure profile. Accordingly, the continuity equation used in the $\mu(I)$ code variant then becomes

$$\left(\frac{D\rho}{Dt}\right)_i = \rho_i\sum v_{ij}^\alpha\cdot\frac{\partial W_{ij}}{\partial x_i^\alpha}\frac{m_j}{\rho_j} + \delta hc_0\sum_j\psi_{ij}^\alpha\cdot\frac{\partial W_{ij}}{\partial x_i^\alpha}\frac{m_j}{\rho_j}, \tag{23}$$

where

$$\psi_{ij}^\alpha = 2\rho_{ji}\frac{r_{ji}^\alpha}{|r_{ji}|^2} - [\langle\nabla\rho\rangle_i^L + \langle\nabla\rho\rangle_j^L], \tag{24}$$

$$\langle\nabla\rho\rangle_a^L = \sum_b\rho_{ba}L_a^{\alpha\beta}\frac{\partial W_{ab}}{\partial x_i^\alpha}\frac{m_b}{\rho_b}, \tag{25}$$

$$\mathbf{L}_a = \left[\sum_b\mathbf{x}_{ba}\otimes\nabla W_{ab}\frac{m_b}{\rho_b}\right]^{-1}. \tag{26}$$

Here c_0 is the reference speed of sound, h is the kernel smoothing length, δ is a tunable parameter taken to equal 0.1 [60], $\mathbf{L}$ is the kernel gradient correction matrix [62], and $\langle\nabla\rho\rangle_a^L$ is the renormalized density gradient. The authors' investigation revealed that the density diffusion term does not improve the kinematics or dynamics in the DP code variant and is thus only implemented in the $\mu(I)$ variant.

2.4. Boundary conditions

There exist a few different means of modelling solid boundaries in SPH, such as virtual boundaries [34], and the combined ghost and virtual boundary adopted in [23]. In this paper, we adopt the approach described in [63], which uses multiple layers of virtual SPH particles to represent the solid boundary. The first layer of particles is placed $0.5dx$ away in the normal direction of the boundary, where dx is the initial particle spacing of the physical SPH particles. Subsequent layers are placed a distance dx further away. The virtual SPH particles serve the purpose of ensuring physical SPH particles do not suffer from the particle deficiency problem near solid boundaries [64]. Because the kernel radius adopted in this paper is $\kappa h = 2.4dx$, three layers of virtual particles are used to complete the kernel interpolation of the physical particles. The virtual particles will have their properties determined by a corrected kernel interpolation [65] centred at their position and involving only the physical particles. The corrected kernel approximation equation calculates the kernel weighting function as

$$W_{ij,i} = W_{ij} / \left(\sum_j W_{ij} m_j / \rho_j \right). \tag{27}$$

The mass is made equal to that of the physical SPH particles, and the density is determined by interpolating the density of surrounding physical particles:

$$\rho_i = \sum_j W_{ij,i} m_j. \tag{28}$$

For no-slip boundaries we impose:

$$v_i^\alpha = - \sum_j v_j^\alpha W_{ij,i} m_j / \rho_j, \tag{29}$$

$$\sigma_i^{\alpha\beta} = \sum_j \sigma_j^{\alpha\beta} W_{ij,i} m_j / \rho_j. \tag{30}$$

For free-slip horizontal and vertical boundaries in this paper we impose:

$$v_i^\alpha = \sum_j (v_j^\alpha - 2 v_{j,n}^\alpha) W_{ij,i} m_j / \rho_j, \tag{31}$$

$$\sigma_i^{\alpha\beta} = \begin{cases} \sum_j \sigma_j^{\alpha\beta} W_{ij,i} m_j / \rho_j & \alpha = \beta \\ - \sum_j \sigma_j^{\alpha\beta} W_{ij,i} m_j / \rho_j & \alpha \neq \beta \end{cases}, \tag{32}$$

where $v_{j,n}^\alpha$ is the component of the velocity of particle j normal with respect to the boundary surface.

2.5. Time-integration scheme

To temporally discretise equations Eqs. (21)–(23), the second-order accurate Leap Frog (LF) time-integration scheme is adopted in this study as it is sufficiently stable and accurate and only requires a single force computation in a given time-step and is thus highly efficient [66]. The LF integration scheme also requires small memory footprint and is computationally efficient and straightforward to implement in parallel. In the LF scheme, the density, velocity, and stress are incremented at mid-time-steps, while the position is incremented at full time-steps (where the evolution of stress applies only to the DP variant and not the $\mu(I)$ variant). At the initialisation of the problem, the state variables must be incremented to the mid-time-step:

$$\rho_{\frac{1}{2}} = \rho_0 + \frac{\Delta t}{2} \left(\frac{D\rho}{Dt} \right)_0, \tag{33}$$

$$v_{\frac{1}{2}}^\alpha = v_0^\alpha + \frac{\Delta t}{2} \left(\frac{Dv^\alpha}{Dt} \right)_0, \tag{34}$$

$$\sigma_{\frac{1}{2}}^{\alpha\beta} = \sigma_0^{\alpha\beta} + \frac{\Delta t}{2} \left(\frac{D\sigma^{\alpha\beta}}{Dt} \right)_0. \tag{35}$$

Thereafter at a given time-step, state variables and position are incremented at full time-steps according to:

$$\rho_{n+\frac{1}{2}} = \rho_{n-\frac{1}{2}} + \Delta t \left(\frac{D\rho}{Dt} \right)_n, \tag{36}$$

$$v_{n+\frac{1}{2}}^\alpha = v_{n-\frac{1}{2}}^\alpha + \Delta t \left(\frac{Dv^\alpha}{Dt} \right)_n, \tag{37}$$

$$\sigma_{n+\frac{1}{2}}^{\alpha\beta} = \sigma_{n-\frac{1}{2}}^{\alpha\beta} + \Delta t \left(\frac{D\sigma^{\alpha\beta}}{Dt} \right)_n, \tag{38}$$

$$x_{n+1}^\alpha = x_n^\alpha + \Delta t \times v_{n+\frac{1}{2}}^\alpha. \tag{39}$$

At each increment of stress, the stress state of the material is maintained such that it does not violate the yield surface by the return mapping algorithm described in [23]. Note that Eq. (38) does not apply to the $\mu(I)$ variant as the stress tensor is dependent only on the instantaneous strain rate and pressure (which is evaluated by Eq. (4)).

The stability of the LF time integration scheme is maintained by enforcing the Courant-Friedrichs-Lewy (CFL) condition. This restricts the size of the time-step to be proportional to the smoothing length:

$$\Delta t \leq C_{CFL} \frac{h}{c}, \tag{40}$$

where c is the speed of sound in the material, which is also sufficiently restrictive such that errors that occur due to finite deformations are minimized [58]. In the classical SPH, the physical speed of sound (i.e. speed of sound in water) is avoided as it is overly restrictive on time-steps size, resulting in longer computation times. Instead, the speed of sound is typically chosen such that the density variation does not exceed 1% to reflect the compressibility of water. However, as the speed of sound in granular material is not so restrictive, throughout this paper we adopt the physical speed of sound for 3D solids:

$$c = \sqrt{\frac{K + \frac{4}{3} G}{\rho_0}} = \sqrt{\frac{E(1-\nu)}{(1+\nu)(1-2\nu)\rho_0}}, \tag{41}$$

where ρ_0 is the reference density of the material, K is the bulk modulus, and G is the shear modulus of the material. A suitable value for C_{CFL} was found to be 0.2 which is used throughout this paper.

3. Parallel scheme

The parallel scheme is implemented in the in-house developed code [23] written in FORTRAN. The strategy adopted to parallelise the in-house developed granular flow code is to partition the domain into subdomains and assign each of these subdomains to an MPI process such that each process treats its subdomain independently. This approach offers the advantage of minimal impact on the structure and contents of the serial code. The parallel scheme is implemented in both code variants and does not differ structurally, although there is a slight difference in the information exchange due to the difference in continuity equation used. The main priority of the parallel scheme is to achieve scalability on large computing systems and for large problem sizes.

To achieve scalability, the choice in algorithm to determine the subdomain boundaries is important because the geometry of the subdomains affects the scalability of the scheme. Here we implement the Orthogonal Recursive Bisection (ORB) algorithm which has been shown to lead to scalable results at large processor numbers [44]. The ORB algorithm subdivides the domain into coaxial bounding boxes, which aims to group nearby particles on the same process, while balancing the load between processes and minimising the communication between processes when performing simulation steps in time. The direction of each recursive subdivision of a block in two parts (one of the x, y, or z directions) is chosen based on the face of the sub-section with the smallest surface area, and the location of the subdivision is chosen such that the number of particles is approximately equal in the two resulting sub-blocks. An advantage of this domain partitioning technique is that the data structures used to store subdomain boundary information are simple. The algorithm can easily be parallelised and the inter-process message sizes can be shown to remain constant with increasing core count in weak scaling test cases (where the number of particles per core is kept constant), and grow more slowly for strong scaling test cases (where the total number of particles is kept constant), compared to simpler techniques such as the partitioning by slices [45,67].

To facilitate the ORB algorithm, a *grid* is defined and used solely for the purpose of the ORB algorithm. The grid is discarded after the subdomain boundaries are determined. This grid is a regular grid defined by the *domain boundary*, which is the smallest bounding box that contains all physical particles at the time the domain partitioning algorithm is called. The size of the grid cells is user-defined, but the author's choice is to define the grid cell size to be a single smoothing length, h (this is used throughout the paper unless otherwise stated). Generally larger grid cells reduce partitioning quality, but also reduces memory

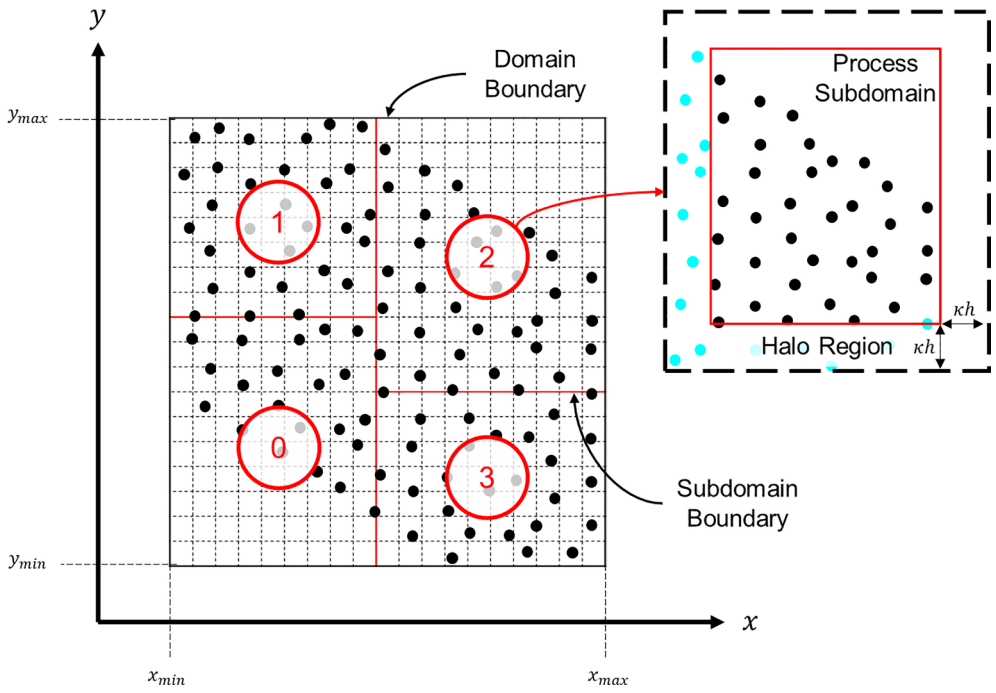

Fig. 1. Illustration of Domain Subdivision using grid.

required to store the grid. The positions of all particles are stored in a *particle-in-cell matrix*, **P**, where each entry of the matrix corresponds to the number of particles within the corresponding grid cell. As grid cell sizes are kept fixed throughout a simulation, the number of grid cells may grow as the particles spread out throughout the simulation. On average, each grid cell contains 1.44 particles initially if the particles are initialised on a square lattice. For example, if a simulation is initialised in a square array with $1000 \times 1000 = 1,000,000$ particles with initial spacing, Δx, between particles, the smoothing length, h, of the cubic spline is taken as $1.2\Delta x$, so we have a grid with 833×833 grid cells and 1.44 particles per grid cell on average. The grid is partitioned among the p available cores (with one MPI process per core) using the ORB algorithm. The algorithm recursively subdivides the grid and assigns each process a rectangular part of the grid that determines the process' *subdomain boundaries*. Local subdomain boundaries coincide with boundaries between grid cells. Once the subdomain boundaries of each process are known, the grid and particle-in-cell matrix are discarded. Each process' local subdomain is extended by a halo region (Fig. 1) of width equal to the kernel radius ($\kappa h = 2.4\Delta x$). Each process can update the state variables of the physical particles in its local subdomain in parallel, using halo particles (which are copies of physical particles sent by neighbouring processes in a communication step between particle updates) in its halo region. Processes with domain adjacent to or containing a physical boundary also create virtual particles.

The subdomain boundaries are determined in the first time-step, and as the simulation progresses, and particles move from one process' subdomain to another, the algorithm detects load imbalances and shifts the location (but not the direction) of cuts made in the ORB algorithm to rebalance the load. Also, when particles move and the particle cloud changes its shape or alignment in a substantial manner, detected by a change in aspect ratio in the bounding box of the particle cloud, we perform a comprehensive repartitioning that also reassigns cut direction in each cut of the ORB algorithm (see example in Fig. 2). At the mid-increment of each time-step and after any necessary calls to the ORB algorithm, *physical particles* are redistributed so that each process possesses the physical particles contained within its own subdomain boundaries, and then *halo particle* information is distributed so physical particles near subdomain boundaries have correct neighbours with which to perform interactions.

In shared memory and GPU parallelisation approaches, modifications are required to the serial subroutines for the code to be efficient [68]. Here, however, because subdomains are treated as smaller subproblems, most of the serial code remains applicable for updating the sub-domains, and the domain partitioning can be contained within a single subroutine. At the mid-increment of the time-stepping procedure, the domain partitioning algorithm is called where domain boundaries are updated if necessary. Then physical particles are distributed as required, and halo particles are distributed. Halo particle state variables exchanged are velocity, density, position, and a particle identification number. Next, a particle interaction pair-list is created using a cell-linked list algorithm, although this can be replaced with any other generic interaction pair-searching algorithm. The strain rate of local physical particles is calculated on each process and then the stress tensor is updated. Both the Drucker-Prager and the $\mu(I)$ code variants require updating of halo particles' stress values and thus an additional communication is required where stress variables for halo particles are exchanged. Fig. 3 shows the structure of the code after parallelisation and Algorithm 0 shows the container subroutine that is called at the mid-increment of every time-step, prior to any force calculations for that time-step.

Algorithm 0 (*Domain partitioning and parallel communication container algorithm*).

Require: *particle properties* $\mathbf{x}_{loc}$, $\mathbf{v}_{loc}$, ρ_{loc} *etc. and other state variables to be exchanged between processes*

1. ***If*** *number of physical particles held by any process changes by* ±5%
2. ***Call*** *Algorithm* 1*a* (*Particle-In-Cell Matrix Construction*)
3. ***Call*** *Algorithm* 1*b* (*Orthogonal Recursive Bisection*)
4. *Determine process neighbours*
5. *Generate Virtual Particles within subdomain boundaries*
6. ***End if***
7. ***Call*** *Algorithm* 2*a* (*Physical particle distribution*)
8. ***Call*** *Algorithm* 2*b* (*Halo Particle Distribution*)
9. *Wait for halo particles to complete distribution*
10. ***End*** *Algorithm* 0

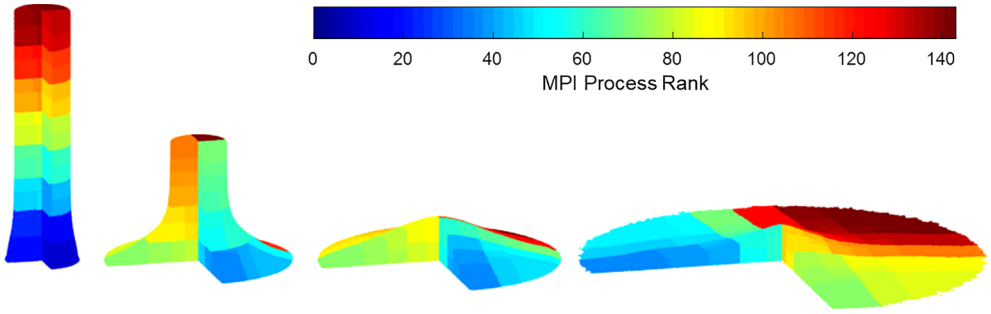

Fig. 2. Demonstration of the adaptation of the subdomain configuration through the collapse of the axisymmetric granular column collapse. Simulated using the DP variant with 3,630,000 SPH particles on 144 CPU cores.

3.1. Domain boundaries determined by ORB (Algorithms 1a-1b)

Algorithms 1a-b is where the ORB algorithm is invoked and subdomain boundaries are determined by each process. It is called at the first time-step of the simulation, and in subsequent time-steps when the number of physical particles has changed by at least ± 5% on any process since the previous boundary update. This check is conducted every 50 time-steps. Every time boundaries are updated, process neighbours are determined and virtual particles located adjacent to or within any subdomain are generated as necessary. The input physical to the algorithm is the number of particles currently owned by a process and each particles' position. At the end of the algorithm, each process will be aware of its own and all other process' boundaries as well as which processes are neighbouring processes.

Algorithm 1a (*Particle-In-Cell Matrix Construction*).

Require: *Position of locally held particles,* $\mathbf{x}_{loc}$*, on this process*

1. *Determine bounding box of locally held physical particles*
2. *To define global grid, find global bounding box using MPI reduction*
   Call *MPI_ALLREDUCE*(…)
3. *Populate* **P** *with locally held physical particles on this process*
4. *From the locally populated* **P***, create a list of cells occupied and the number of particles in each cell,* $\mathbf{G}_{loc}$
5. *Share number of cells occupied by local particles among all processes*
   Call *MPI_ALLGATHER*(…)
6. *exchange each process'* $\mathbf{G}_{loc}$ *with all processes so all processes own a copy of cells occupied by all processes' particles and the number of particles within each cell,* **G**
   Call *MPI_ALLGATHER*(…)
7. *Populate* **P** *with***G**
8. ***End*** *Algorithm* 1*a*

The ORB algorithm relies on a particle-in-cell matrix, **P** to perform the domain partitioning. **P** is a matrix whose entries correspond to the number of particles belonging to a cell of an underlying grid used only for the purposes of the domain partitioning. Use of **P** as the basis of partitioning reduces computational cost and memory requirements compared to direct particle counting. The particle-in-cell construction algorithm is shown in Algorithm 1a.

The ORB algorithm relies on a binary tree data structure to subdivide **P** to determine each process' subdomain boundaries. At the first node of the tree, each process begins with a copy of **P**, and the first cut is made along one of the x, y, or z axes at the node and the sub-sections are passed onto the children nodes. This process is repeated recursively on each sub-section of **P** until an MPI process reaches its leaf node, so each process traces its own path through the tree. At each node of the tree, the direction of the cut is kept fixed to reduce the frequency of exchanging large portions of process' subdomains, except when the geometry of the entire domain changes significantly. Here, this is measured by the aspect ratio of the bounding box of the entire domain. Besides **P**, the information stored as part of the subdomain boundary procedure is a list maintaining each node's cutting axis direction, a list of leaf-node and process pairings, and a list of boundary extents associated with each process.

Algorithm 1b (*Orthogonal Recursive Bisection*).

Require: *the current node,* k*; portion of the particle-in-cell matrix,* $\mathbf{P}_k$*; number of particles entering node,* n_k*; number of processes,* p_k*, and range of processes entering node,* k*; and the current process' rank,* a

1. ***If*** *aspect ratio of* **P** *changed significantly from previous cutting axes reorientation*
2. *Determine face of* $\mathbf{P}_k$ *with smallest area and save normal vector as cutting axis*
3. ***End If***
4. *Determine location of cut in* $\mathbf{P}_k$ *along cutting axis, balancing particles in the two subblocks*
5. *Determine destination child node,* k_{child} *i.e.* $k_{child} = 2k$ *or* $k_{child} = 2k + 1$
6. *Define boundary locations of sub-blocks* $\mathbf{P}_{2k}$ *and* $\mathbf{P}_{2k+1}$
7. ***If*** k_{child} *is leaf node*
8. *Save sub-section boundaries as local subdomain boundaries for process* a*,* $\mathbf{B}_{loc}$
9. ***If*** *local process is an edge process, extend local boundaries* $\mathbf{B}_{loc} = \mathbf{B}_{loc} \pm 2.5kh$
10. *Share* $\mathbf{B}_{loc}$ *such that all processes have a copy of all other processes' boundaries*
    Call *MPI_ALLGATHER*(…)
11. ***Else***
12. ***Recurse*** *Algorithm* 1b (k_{child}, $\mathbf{P}_{k_{child}}$, $n_{k_{child}}$, $p_{k_{child}}$, a)
13. ***End if***
14. ***End*** *Algorithm* 1*b*

The ORB algorithm relies on a binary tree that is constructed as follows. Each process, a, begins at the root node $k = 1$, and individually travels a unique path down the tree such that each process reaches a unique leaf node. This is done without any communication, except once all processes reach their respective leaf. At any given node, k, and given the number of processes travelling to it, p_k:

The children nodes of k, i.e. k_{child}, will have node IDs of $2k$ and $2k + 1$
The lowest p_{2k} ranks travel to node $2k$ where $p_{2k} = \text{ceiling}\left(\frac{p_k}{2}\right)$. The remaining processes travel to node $2k + 1$
If $p_{child} = 1$ then k_{child} is a leaf node

This results in a tree with the following features: The number of nodes that results from this algorithm is equal to $2p - 1$, where p is the total number of processes, and the number of levels on the tree is equal to ceiling($\log_2 p$) not including the root node as a level. Each process travels to a unique leaf node. For p that are not powers of two, leaf nodes will be either on level ceiling($\log_2 p$) or ceiling($\log_2 p$) − 1.

When travelling down the tree, a process with rank a enters a given node knowing the range of grid indices of the sub-section $\mathbf{P}_k$; the number of particles, n_k, contained within $\mathbf{P}_k$; the total number of processes, p_k, entering the node and their respective process ranks; and its own process rank, a. At k, process a first determines the cut axis and location, and then determines which processes travel to the same

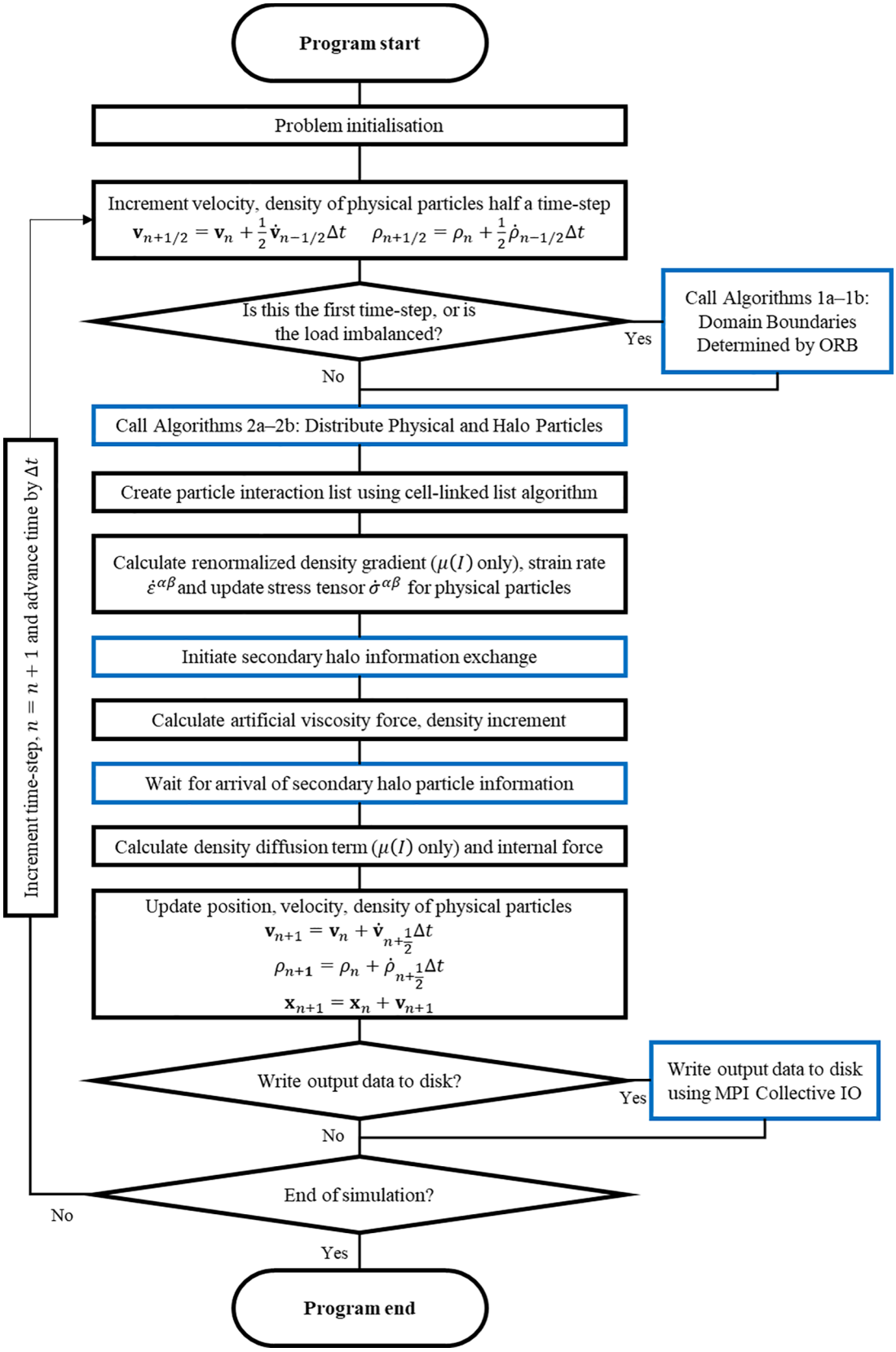

Fig. 3. Program flow-chart for parallel scheme used for both code variants. Blue boxes represent MPI related steps. The equivalent serial program would exclude the blue boxes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

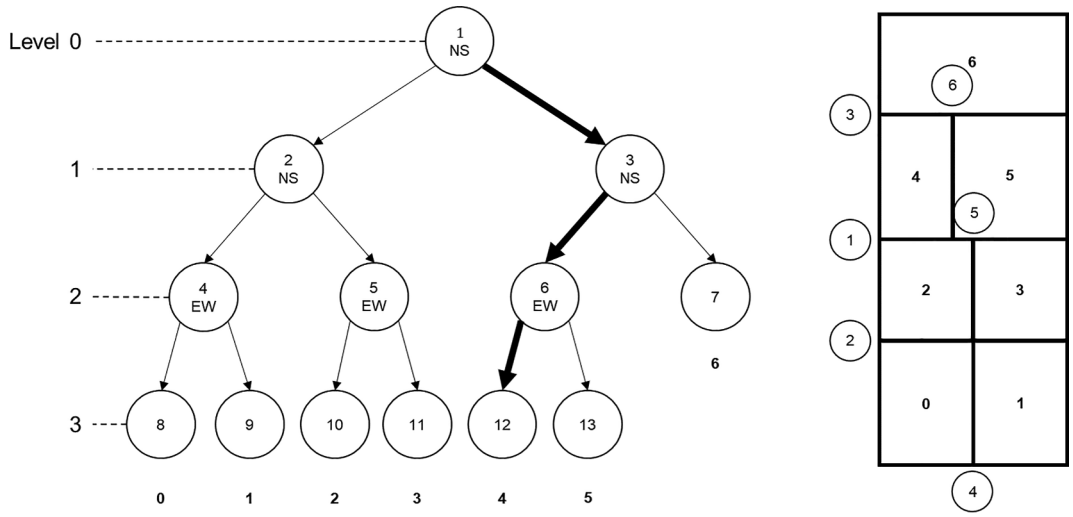

Fig. 4. Illustration of path followed by process 4 along binary tree and the corresponding assignment of processes to subdomains. Cuts are labelled by numbers within circles and correspond to a node on the tree. Emboldened numbers correspond to process IDs. The letters in each node give information on the direction a sub-section is separated into i.e. a vertical cut made at a node has letters EW (East-West), and a horizontal cut has letters NS (North-South).

destination node. Firstly the cutting axis is chosen, which is in the direction x, y, or z that possesses the smallest surface. These cut axes are mostly kept fixed between subsequent boundary updates, except when significant changes in the global problem domain aspect ratio, in which case cutting axes at all nodes in the tree are re-selected. The cut location is chosen such that the number of particles belonging to $\mathbf{P}_{2k}$ is approximately equal to $n_{2k} \approx \frac{p_{2k}}{p_k} n_k$ and $n_{2k+1} \approx \left(1 - \frac{p_{2k}}{p_k}\right) n_k$. Then the destination node for process a, k_{child}, and related information ($\mathbf{P}_{k_{child}}$, $n_{k_{child}}$, and $p_{k_{child}}$) are determined as per the rules above. If $p_{k_{child}} = 1$, then k_{child} is a leaf node and the bounding box of $\boldsymbol{P}_{k_{child}}$ is saved as $\mathbf{B}_{loc}$ which defines the subdomain boundaries of process a. Finally, boundary information is shared among all processes so that the information of all processes' boundaries is contained in $\mathbf{B}$, which all processes hold a copy of. All other information is discarded, with the exception of the cutting axes, which are mostly kept fixed between subsequent boundary updates. Fig. 4 shows an illustration of an example with 7 MPI processes.

3.2. Physical Particle Distribution (Algorithm 2a)

As a simulation is run, the Lagrangian movement of particles causes them to move from one process' subdomain to another. Therefore at every time-step, algorithm 2a is called to redistribute physical particles so that at the end of the algorithm, each process contains only information of physical particles within its own subdomain boundaries. This is accomplished by process a searching over its currently held physical particles and packaging and sending particles that have left a's subdomain and entered another process' subdomain. The communication of the particle information between processes is facilitated by a series of non-blocking point-to-point communications. These non-blocking communications are partially overlapped with algorithm 2b, to reduce communication overhead.

Due to the relatively restrictive time-step size, particles do not move beyond the boundaries of its owner process' neighbours. In this case, physical particle distribution is straightforward as communication can be maintained amongst neighbours. However, in cases where process subdomain boundaries are updated (either changing position or orientation of cutting axes), or when the problem is initialized at the start of the simulation, particles may require relocation to a non-neighbouring process. Instead of using a global communication which can be very costly, a diffusion-based approach is used [53] where the neighbour-based physical particle relocation algorithm is called multiple times in succession until all processes no longer own any particles not contained within their own boundaries. If any processes find any particles that do not belong within their own or their neighbours' subdomain boundaries, they note that diffusion is required. After the first exchange of physical particles, if any processes have noted that diffusion is required, then the search, package, and send-to-neighbour part of the algorithm is repeated until each process possesses particles located within its own subdomain boundaries. Diffusion is only required when subdomain boundaries are updated as displacement of particles between time-steps is very small with respect to subdomain size. This diffusion algorithm is considerably faster than a single global communication call. For example, considering a configuration of particles in a rectangular prism with dimensions 0.2 m × 0.05 m × 0.1 m with 512 million particles arranged on a cubic lattice and allocated to 1024 processes sequentially in slices, to completely relocate particles as per the boundaries determined by the ORB algorithm, using global communications (i.e. MPI_ALLTOALLV) takes 21.84 s, while the diffusion algorithm takes 8 iterations and a total of 7.36 s to complete the relocation. Besides cases such as in the example above, in the authors' experience, process' subdomain boundary updates are not so extreme and, therefore, usually require no more than 3 iterations of the diffusion algorithm and often do not require more than 1 iteration. Details of the algorithm are shown in Algorithm 2a.

Algorithm 2a (*Physical Particle Distribution*).

Require: *Particle properties to be sent and the subdomain boundaries of adjacent processes*

1. ***For*** *all currently held particles, i*
2. ***If*** *i is outside of local boundaries*
3. ***If*** *i belongs to neighbour processes, package for sending*
4. ***Else***
5. *Determine closest neighbour process and package for sending*
6. *Note that diffusion is required*
7. ***End If***
8. ***End If***
9. ***End For***
10. *Notify neighbours of how many particles to be exchanged*
Call *MPI_ISEND*(…)
Call *MPI_IRECV*(…)
11. *Shift remaining particle information in array to remove sent particles and to ensure remaining particle information is contiguous in memory.*
12. *Wait for* 10. *To complete*

Call *MPI_WAITALL*(…)
13. *Post non-blocking communications to distribute particles' information to neighbours*
Call *MPI_ISEND*(…)
Call *MPI_IRECV*(…)
14. *Check if any processes have noted whether diffusion is required*
Call *MPI_ALLREDUCE*(…)
15. ***If*** *diffusion is required*
16. *Wait for* 13. *To complete*
Call *MPI_WAITALL*(…)
17. ***Go to*** 1. *Repeat for newly received physical particles from neighbours*
18. ***End If***
19. ***End*** *Algorithm* 2*a*

3.3. Halo distribution (Algorithm 2b)

Similar to algorithm 2a, this algorithm is called every time-step. However only neighbour communications are used. All processes enter this algorithm with the exchange of physical particle information, in Algorithm 2a. line 13, still awaiting completion. At the beginning of this algorithm, processes search the remaining particles located within their subdomain boundaries, and if any particle is located within κh of its own subdomain boundaries (i.e. the kernel radius), then the process searches through its neighbours' subdomain boundaries which have been extended by κh. If the particle falls within the extended subdomain boundaries of a neighbour process, then the particle information is copied and packaged to be sent to that neighbour process as halo information. The process is then repeated for the newly received physical particles once the communication in algorithm 2a line 13 has completed. Like the physical particle relocation algorithm, the processes first communicates how many particles are to be exchanged, before exchanging the actual halo particle information. Details are shown in Algorithm 2b.

Both code variants require calculation of stress in the current time-step, and therefore require an additional halo communication after the first. In the first communication, position, velocity, and density of halo particles are communicated. Then the stress tensor can be calculated for each process' own particles and be sent to neighbours to update the halo particles. Because it is already known which of a process' particles' information needs to be sent to neighbour processes as halo information, this step takes very little overhead computation time, but still requires additional communication time. To minimize the impact of this second communication, it is overlapped with other computation. It must also be noted that the $\mu(I)$ rheology code variant requires the calculation of the renormalized density gradient, $\langle\nabla\rho^L\rangle$, for equation (24) after the first distribution of physical and halo information; so the $\mu(I)$ variant also communicates this information in the second halo exchange.

Algorithm 2b (*Halo Particle Distribution*).

Require: *Particle properties to be sent and the subdomain boundaries of adjacent processes*

For *all currently held physical particles, i*
  If *i is within κh of local boundaries*
    For *all neighbour processes, j*
      If *i is in halo region of j*
        Package i to be sent to j as a halo particle
      End If
    End For
  End If
End For
Wait for 13. *In Algorithm* 2*a to complete*
Call *MPI_WAITALL*(…)
Goto 1. *Repeat only for newly received physical particles from neighbours*
Communicate how many halo particles are being sent to neighbour processes
Call *MPI_ISEND*(…)
Call *MPI_IRECV*(…)
Wait for 12. *To complete*
Call *MPI_WAITALL*(…)
Distribute halo particles to neighbours using non-blocking communication
Call *MPI_ISEND*(…)
Call *MPI_IRECV*(…)
Wait for 14. *To complete*
Call *MPI_WAITALL*(…)
End *Algorithm* 2*b*

In our scheme, virtual particles are also generated by a process if they belong to the process' halo region. But because the virtual particles' properties are determined using a corrected kernel interpolation [65] centred at the virtual particles' positions, an extended halo region is defined, so that additional halo particles are sent to that process. This procedure is illustrated in Fig. 5.

3.4. Parallel Scheme Scaling Performance

The purpose of the parallel algorithm is to enable simulation of

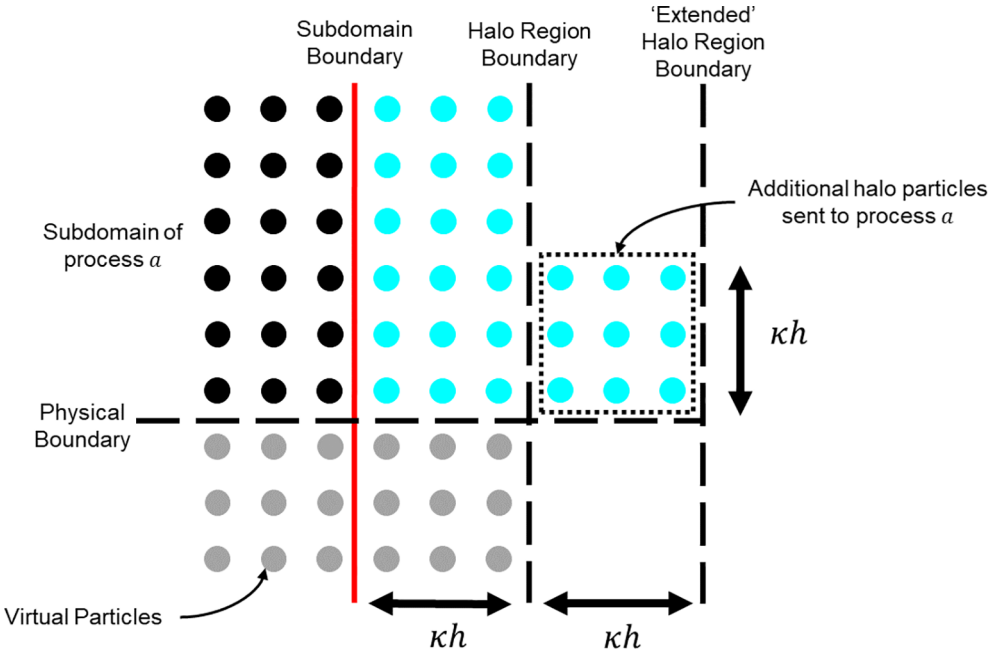

Fig. 5. Illustration of the extra halo particles required for the completion of virtual particles' support domain. Maroon coloured particles are the physical SPH particles, the grey particles are the virtual boundary particles, and the cyan particles are the halo particles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

larger problems and faster simulation of existing problems. To evaluate the ability of the parallel scheme to satisfy these purposes, weak and strong scaling tests must be conducted. A weak scaling test corresponds to multiple simulations of the same problem at increasing particle resolution (i.e. problem size and shape remains the same, but increasing particle density) proportional to increasing number of cores being used to run the simulation, keeping the number of particles per core constant. For the weak scaling tests, we consider a reference simulation with $n = 32$ cores, taking time t_n. For increasing core counts $p \geq n$, we measure the times t_p, which ideally should be equal to t_n, but in practice grow due to increasing communication overhead. We also define the weak scaling efficiency e_p^w, given by

$$e_p^w = \frac{t_n}{t_p}, \tag{42}$$

Ideally e_p^w is maintained close to 1 for increasing p, but due to additional communication and computation overhead, t_p increases therefore reducing e_p^w to less than 1.

Strong scaling studies are conducted to measure the ability of the parallel scheme to run simulations faster when introducing additional cores, keeping the total number of particles constant. These involve a fixed problem size and particle resolution, but increasing the number of cores, and measuring the equivalent efficiency and overall speedup, respectively:

$$e_p^S = \frac{n \times t_n}{p \times t_p}, \tag{43}$$

$$s_p^S = \frac{n \times t_n}{t_p}. \tag{44}$$

Three sets of weak scaling studies are conducted with 125, 250, and 500 thousand particles per core, and 3 sets of strong scaling studies are conducted using 8, 16, and 32 million particles in total. The studies are run from 32 to 1,024 cores and using the 32 core simulation as reference ($n = 32$). All tests use the same test geometry with varying particle densities. The test case examined is the qausi-2D granular column collapse shown in Fig. 6. Each individual simulation is run for 1000 time-steps. Between simulations, variability of simulation times occur due to cluster load fluctuations and differences in the configuration of nodes. Therefore, each simulation is run 3 times and the average is taken as t_p. The first time-step of each test will involve a call to the ORB algorithm and multiple iterations of the physical particle diffusion algorithm. For the weak scaling tests, the ORB algorithm is called only at the beginning since time-step sizes decrease as the number of processes increase due to increasing particle density, so load balances do not occur. The ORB algorithm is called increasingly for higher core counts during the strong scaling tests as load imbalances occur due to the decreasing number of particles per process. The maximum number of calls to the ORB algorithm is 5 times when using 1,024 cores during the strong scaling tests. Unformatted binary output files are written using the collective MPI-IO subroutines included in the MPI standard. Time taken to write data to files is inconsequential with respect to overall simulation time and is not taken into consideration in these tests.

The tests are conducted on the Australian National Computing Infrastructure (NCI) Raijin supercomputer. NCI-Raijin consists of 84,656 cores on 4,416 compute nodes with full fat tree interconnect between nodes. The tests are conducted on 16 core CPU nodes each consisting of 2 Intel Xeon E5-2670 CPUs and 32 GB of RAM. Compilers used are Intel FORTRAN and Open MPI. All tests assign one MPI process to a core.

3.5. Weak Scaling Study

Sets of weak scaling studies are conducted with 125, 250, and 500 thousand particles per core and are run from 32 cores to 1,024 cores. Although the total number of particles grows, the problem size and geometry is maintained, so the particle spacing reduces and particle density increases. For the 125 and 250 thousand particles per core tests, the grid cell size is a single smoothing length, but for the 500 thousand particles per core set, the grid cell size is chosen as two smoothing lengths for all tests. Each test makes the call to the ORB algorithm once throughout the test, which occurs at the first time-step.

The results are shown in Fig. 7 in terms of both total simulation time, t_p, and efficiency, e_p^w. Both code variants maintain above 95% efficiency. The DP variant maintains efficiency between 95.9% and 97.6% at 1,024 cores and similarly the $\mu(I)$ variant maintains between 94.2% and 98.3%. While weak-scaling results are good, further improvement could be achieved by further overlapping communication time with computation time. Between the three sets of studies, there is little variation and all three sets maintain efficiencies of at least 90% for the core counts tested. The DP variant demonstrates improved scaling over the $\mu(I)$ variant, and this can be attributed to the $\mu(I)$ variant requiring the exchange of renormalised density gradient information in the second halo communication.

The Weak Scaling studies reveal a scalability issue for large core counts above 1,024 that is related to memory usage of the ORB algorithm. Firstly, due to the size of the grid cells used to construct **P** usually being a multiple of the smoothing length of the kernel, the size of **P** is proportional to the total number of particles simulated. Secondly, the algorithm becomes memory intensive because each process retains a copy of the **P** matrix, which stores binned location information of all simulated particles, and consumes a significant amount of memory. The issue can be mitigated by coarsening the grid used to create **P**, but sufficiently coarse grids may require additional refinement to ensure load balancing is not adversely affected. Another solution would be to hybridise MPI with shared memory schemes such as OpenMP since this would mean a single copy of **P** could be shared amongst multiple computing cores in a node. Also, more complicated approaches may be developed that avoid the need for a process to require the entirety of **P** to perform the subdomain partitioning. Investigation is left for future work.

3.6. Strong Scaling Study

Ultimately, the parallel scheme has been devised to save time when simulating large-scale problems. Therefore, strong scaling studies are of

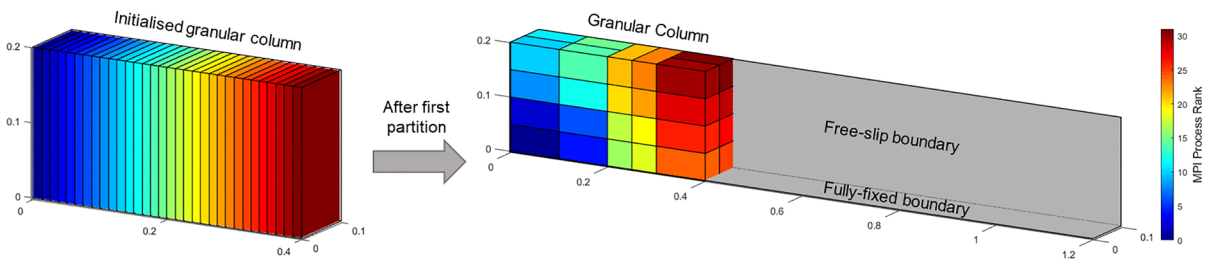

Fig. 6. Simulation setup for scaling tests. Left shows the particle-process configuration at initialization and the subsequent configuration after the partition in the first time-step, for 32 cores.

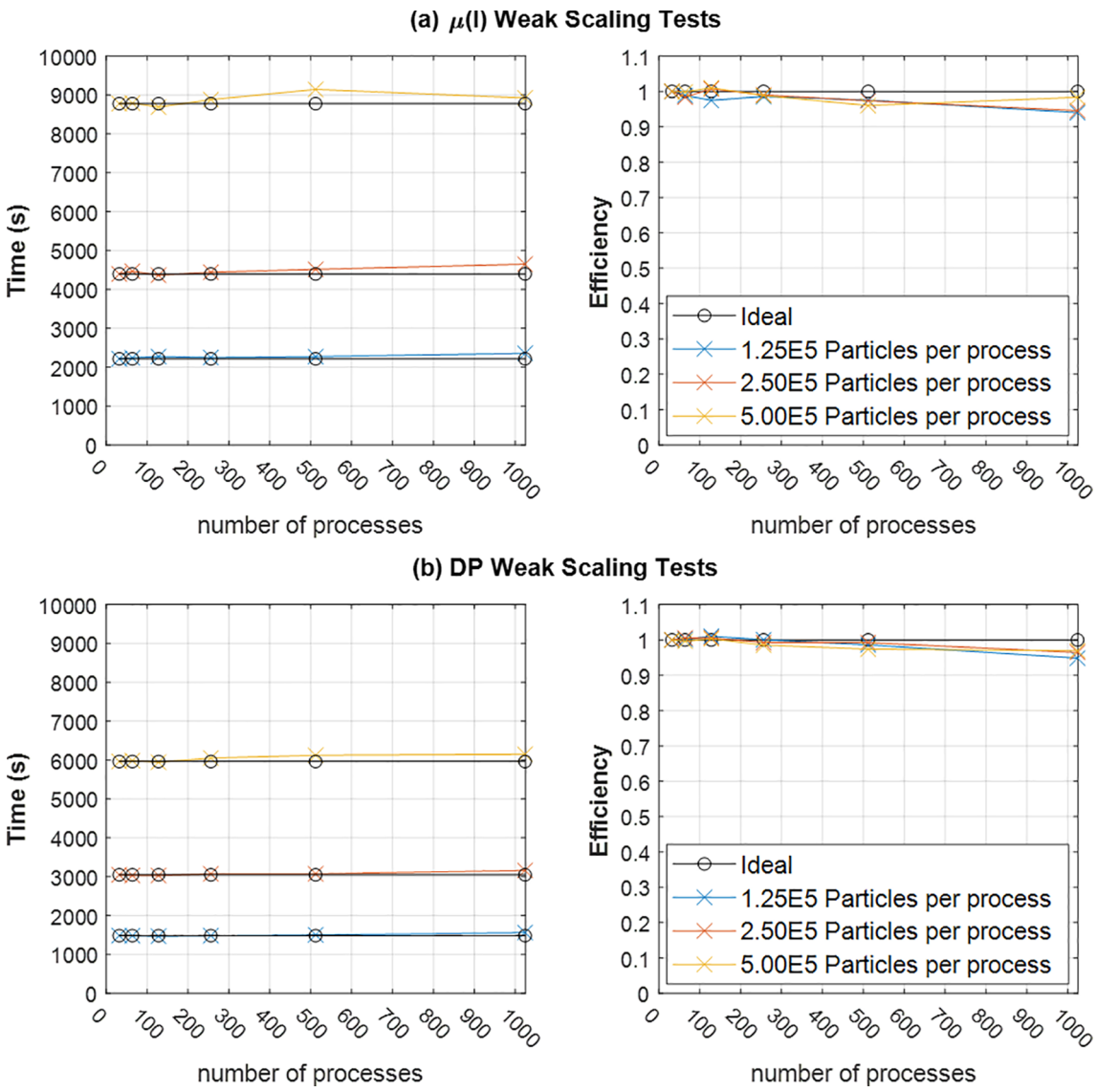

Fig. 7. Weak scaling simulation time and efficiency for various particles per process for (a) $\mu(I)$ and (b) Drucker-Prager code variants.

interest because they represent the potential time savings that can be attained by introducing additional computational resources working on a problem. In ideal scenarios, the speedup relative to 1 core equals the number of computing cores working on the problem. However, in practice, this ideal is nearly impossible to achieve as some time is required for overhead computation and for communication of information. Three sets of strong scaling studies are conducted with a total problem size of 8, 16, and 32 million particles in total and run from 32 cores to 1,024 cores for both code variants. Results are shown in Fig. 8 in terms of both speedup and efficiency. The tests on the $\mu(I)$ and DP variants show good maximum efficiencies at 1,024 cores of 88% and 90%, respectively, for the largest problem size. Minimum efficiencies are 74% and 79% respectively, for the smallest problem size. Differences between the strong scaling performances of the two codes are not substantial, as the large number of particles per process allows the additional communication to be almost entirely overlapped with meaningful computation.

Both code variants show similar efficiency decreases with smaller problem size and increasing core count. The composition of the computation time in the $\mu(I)$ variant shown in Fig. 9 shows the increase in overhead computation and communication time for larger core counts and smaller problems. Firstly, in smaller problems, the ratio of physical to halo particles is lower, meaning that halo communication and computation overhead times with respect to meaningful physical computation times increase. Secondly, as the number of processes working on a problem increases, the load balance performance of the ORB algorithm is reduced. The poorer load balance performance results in some processes waiting for others that have yet to finish computation. The reduced load balancing performance is a consequence of the grid cell size in **P** scaling with initial particle spacing and not process count. It must be noted that in the serial code, the symmetry of interactions allow for contributions of particles i and j to each other's motion to be computed only once. However, when considering interactions between particle i and j owned by processes a and b, respectively, where i is a halo particle of process b, and j is a halo particle of process a, both processes a and b have to perform the calculations involved with the interaction between particles i and j, effectively doubling the computational cost of the interaction between i and j, compared to the serial code. This "double computation" for halo particles also contributes to the reduction in efficiency as the ratio of halo to physical particles increases, but is not represented in Fig. 9.

The main bottlenecks for increasing core counts under strong scaling can be summarised as the exacerbation of load imbalances due to grid cell size with respect to subdomain size, and the domination of

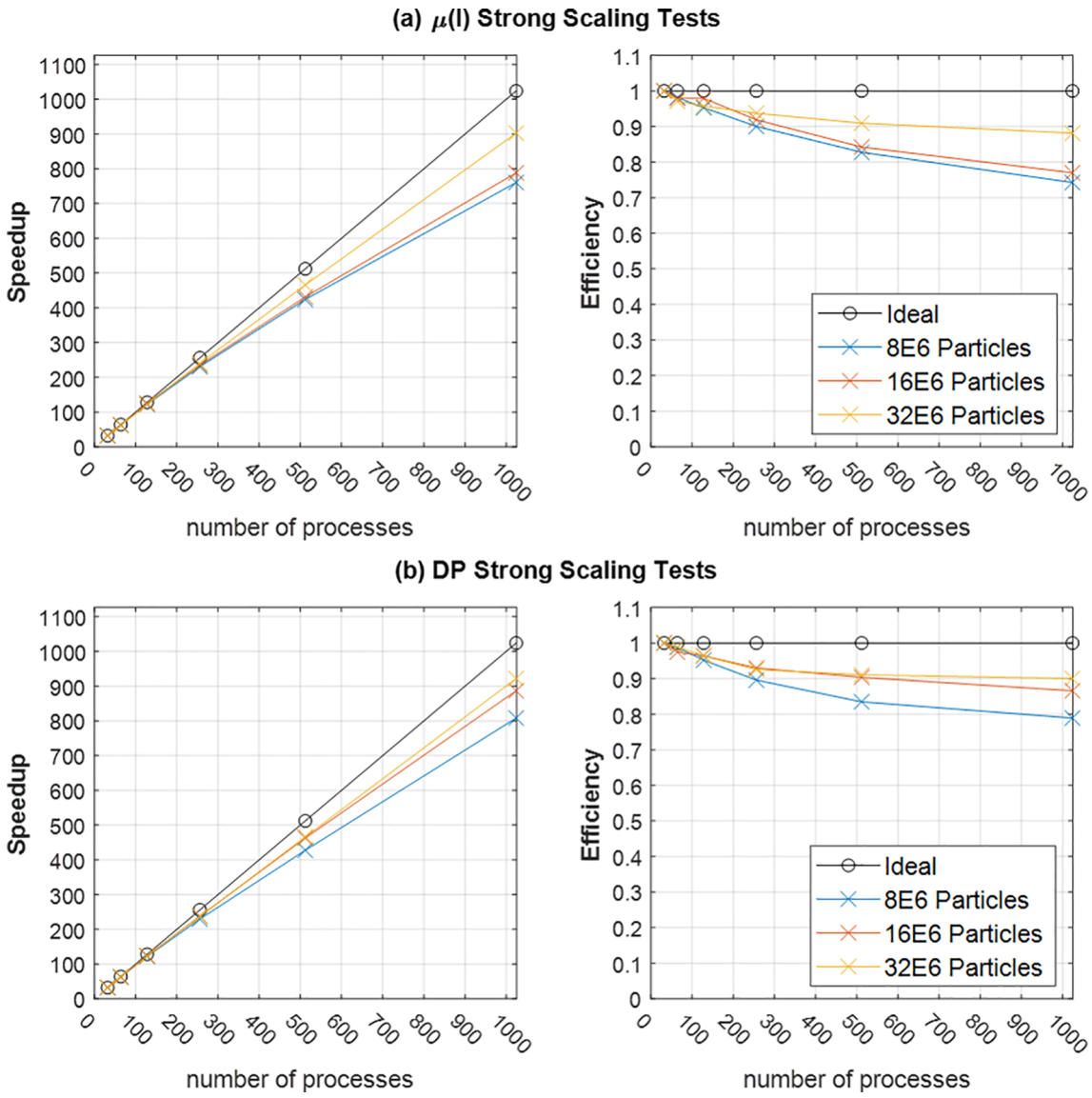

Fig. 8. Strong scaling speedup and efficiency for various fixed problem sizes of 8, 16, and 32 million particles for both the (a) $\mu(I)$ and (b) Drucker-Prager code variants.

computational overhead and communication time due to high halo to physical particle ratios. In regard to the former bottleneck, the geometric restrictions on the subdomain shape result in some unavoidable level of load imbalance when particles are initially configured in an orderly way such as on a square or cubic lattice. However, as particles move and become more disordered, load balance becomes dependent on the coarseness of the grid as the subdomain boundaries coincide with grid cell boundaries. To achieve improved load balancing, once a cut location is chosen, it may be improved by shifting the cut slightly so that it does not necessarily align with the grid cells, providing an improved load balance. The latter bottleneck could be improved by using more complex algorithms to determine subdomain boundaries to reduce the surface area to volume ratio such as generic graph partitioning libraries [69], although this likely comes at the cost of simplicity and ease of implementation. Another approach would be to further increase overlap of communication with computation. This is accomplished to some degree in our parallel scheme, but further improvement could be made such as in [44].

3.7. Application to Axisymmetric Granular Column Collapse

The axisymmetric granular column collapse experiment begins with an initially confined cylindrical column of granular material. Collapse is initialised by swiftly lifting the confining structure to allow the column to flow freely. The experiment has been investigated using a variety of granular materials such as rice and sand [51]. Attention is often paid to the relationship between the final runout (r_∞) and initial aspect ratio (a), as well as final deposit height (h_∞) and a. In [51], the following empirical relationships were found relating non-dimensionalised final runout distance and deposit height as:

$$\bar{r}_\infty = \frac{r_\infty - r_0}{r_0} \cong \begin{cases} 1.24a & a < 1.7 \\ 1.6a^{1/2} & a \geq 1.7 \end{cases}, \tag{45}$$

$$\bar{h}_\infty = \frac{h_\infty}{r_0} \cong \begin{cases} a & a < 1.7 \\ 0.88a^{\frac{1}{6}} & a \geq 1.7 \end{cases}, \tag{46}$$

where the first regime is described by a linear scaling law and the second described by a power law. The same kind of relationships have been observed in other experiments in the literature, but with varying coefficients and regime change aspect ratio [46,70,71]. For very large

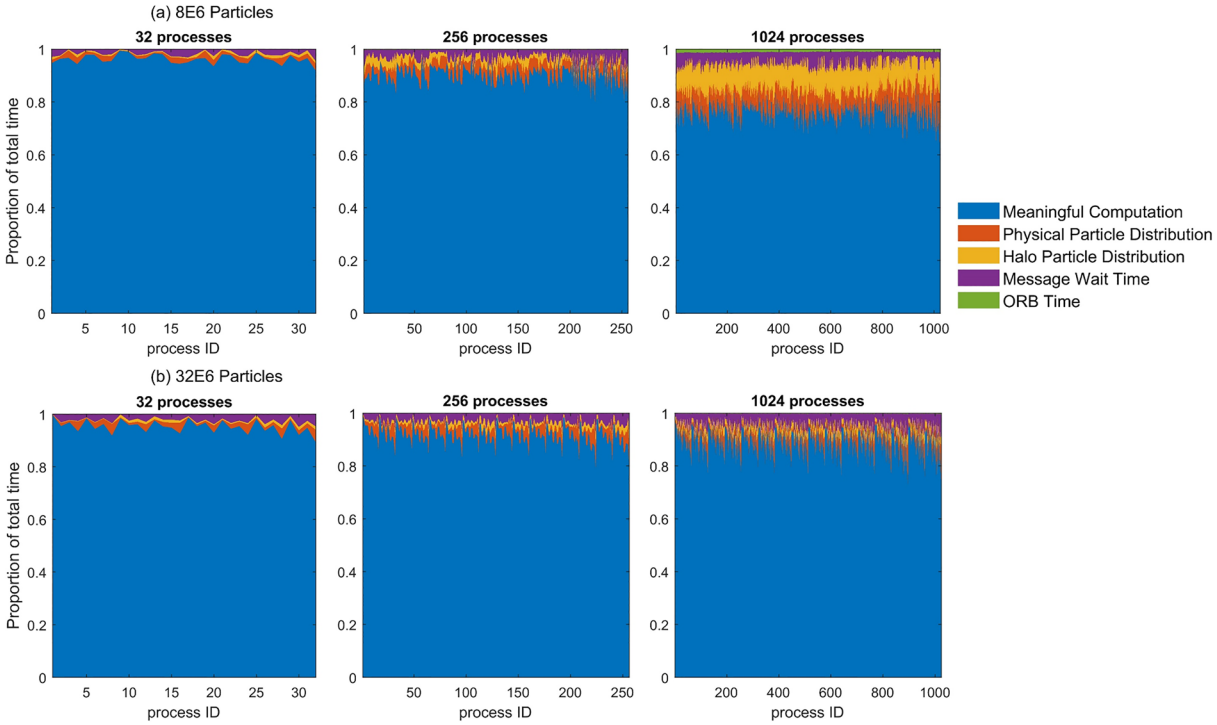

Fig. 9. Comparison of growth in computational and communication overhead time in strong scaling tests for the $\mu(I)$ code variant for: (a) 8 million particles and (b) 32 million particles.

aspect ratios, a third regime is visible in the $\bar{h}_\infty$-a relationship where the relationship changes from a power law to a monotonically decreasing relation. In addition to the use of the non-dimensionalised runout distance and final deposit height, time is non-dimensionalised by

$$T = \frac{t}{\sqrt{h_0/g}} \tag{47}$$

where g is the gravitational acceleration. Here, we simulate the fully three-dimensional axisymmetric granular column collapse using the parallelised SPH solver combined with the $\mu(I)$ and DP constitutive relations described in previous sections. Aspect ratios of $0.2 \leq a \leq 30$ are simulated with constant radius as per Fig. 10. SPH particles are initially placed on a square lattice configuration spaced at a constant 2 mm apart from each other. Due to the fixed particle spacing, the number of SPH particles being simulated increases with aspect ratio. The largest simulation conducted was at $a = 30$ where 11.7 million particles were simulated on 512 CPU cores. Generally, the number of cores chosen to simulate a test was based on ensuring a physical to halo SPH particle ratio of approximately 3:1 to maintain a balance of resource usage efficiency and simulation time. Simulations are conducted on NCI-Raijin in addition to another Australian supercomputer, Pawsey Supercomputer Centre's Magnus, a Cray XC-40 supercomputer.

The material properties used in the simulations are based on those reported in [51] for sand and the friction constants are chosen based on the reported angle of repose. The $\mu(I)$ specific model parameters are

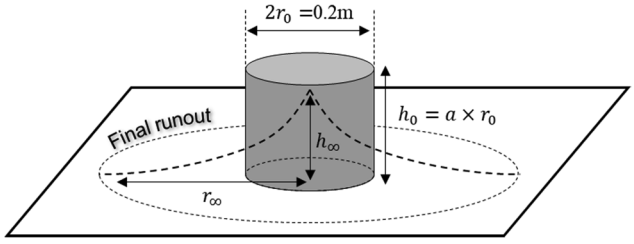

Fig. 10. Granular column simulation setup geometry.

Table 1
Input material properties and model parameters for both code variants.

Material Properties		$\mu(I)$ parameters		Drucker-Prager parameters	
Density (ρ_0)	2600kg/m^3	μ_s	tan30°	Friction angle (ϕ)	30°
Young's Modulus (E)	5.98MPa	μ_p	tan50°	Dilation Angle (ψ)	0°
Poisson's ratio (ν)	0.3	I_0	2.65		
		d	0.32mm		

based on those reported by Goldhirsch [72], which are based on dry sand with similar material properties as in [51] and are listed in Table 1. It is common practice to initialize the column under hydrostatic stress conditions as it would be representative of the stress state within the soil when it is being confined. However, at large aspect ratios, most of the column is in free-fall and in a state of near-zero stress and, therefore, a K_0 stress profile is likely to induce errors. In fact, we find that using a K_0 initial stress condition induces severe numerical issues at the beginning of the simulation leading to unrealistic behaviour such as cavitation or extreme dilation in the column. Instead, particles are initialised with zero pressure and/or stress and the density is made equal to the material reference density. In the authors' experience, for granular flow problems, zero stress initialization does not significantly influence either the kinematics or the dynamics of the flows. The floor is modelled using virtual particles that enforce a fully-fixed boundary condition.

3.8. Comparison of scaling laws predicted by $\mu(I)$ and DP variants

At final arrest, the leading edge of the column as modelled using the DP model, often contains some SPH particles that separate from the main body of the flows and travel some distance further than where the bulk material rests. While this kind of behaviour is expected in real granular material, based on the authors' experience, this is a numerical artefact possibly from boundary conditions, or numerical error due to large deformations. As shown in Fig. 11, this discontinuous-like

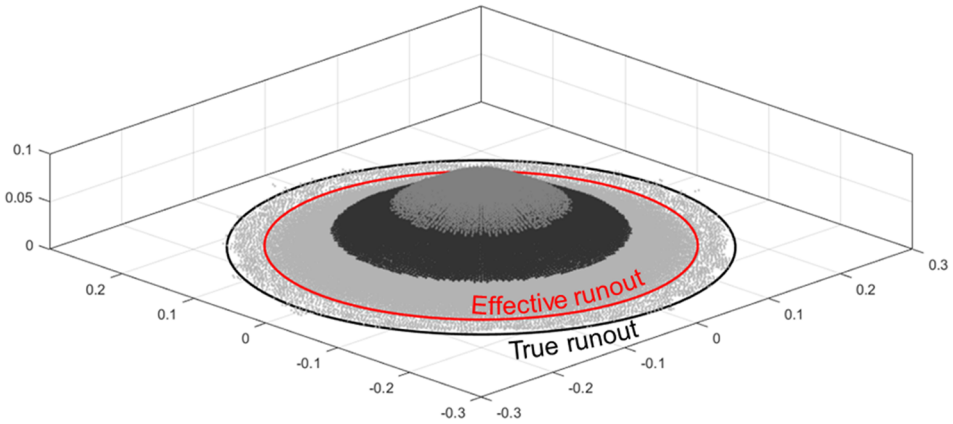

Fig. 11. Illustration of the difference in true versus effective runout distance.

behaviour creates some ambiguity in measurement of the final runout distance. Therefore, the 'effective' runout distance is chosen by the distance travelled by the continuous domain, as opposed to the 'true' runout distance travelled by stray SPH particles. The stray SPH particle region is identified by a single layer of particles, with large spacing between particles, and particles possessing relatively low material densities. From hereon, the runout distance reported for the DP model is the effective runout distance. The $\mu(I)$ model does not produce this kind of behaviour due to the regularizing effect the δ-SPH scheme has on kinematics.

Fig. 12 shows a comparison between the experimental results and SPH simulations for the non-dimensionalised final runout distance and final deposit height relationships with respect to the initial aspect ratio. Results produced by both models are in very good agreement with respect to the entire $\bar{r}_\infty$-a relationship. Additionally, both match very well with the experimental data provided in [51] and are able to capture both the linear scaling behaviour at small aspect ratios, and the power law relationship at intermediate and large aspect ratios. Additionally, the change in regime occurs close to $a = 2$, matching well with experimental data. Considering the $\bar{h}_\infty$-a relationship, both code variants maintain the $h_0 = h_\infty$ relationship for small aspect ratios, and at intermediate aspect ratios, good qualitative agreement is achieved between each other and experimental results, although the DP variant results match better with the experimental data. Reasons for the deviation between the two variants will be discussed in the following sections. In the third regime where the height decreases monotonically, the relationships followed by the DP and $\mu(I)$ variants respectively are

$$\bar{h}_\infty = 2.11a^{-1/5}, \tag{48}$$

and

$$\bar{h}_\infty = 1.18a^{-1/6}. \tag{49}$$

The results here appear to match relatively well with the comparison experimental data, but the DP model over-predicts the transition point at which the third regime begins. In the literature, for this regime, the reported coefficients, exponents, and point of regime change can vary [46].

3.9. Flows during the first regime

The first regime corresponds to $a < 2$ where $\bar{r}_\infty$ is linearly related to a and $h_\infty = h_0$. Fig. 13 shows the development of flows for $a = 0.5$. For simulations conducted using both models, at the initialization of collapse, shear planes develop originating from the outer perimeter of the base of the column, and propagate upward at an inclination 45° with the horizontal and toward the column centre. This is consistent with Roscoe's slip line theory, where it is proposed that failure planes (Fig. 12) occur at angles of inclination of $(45 + \psi/2)^\circ$ with respect to the minor principal stress axis, but inconsistent with Mohr-Coulomb theory, where inclinations are proposed to be $(45 + \phi/2)^\circ$ with respect to the minor principal stress axis [73]. Material flows above these planes leaving an un-deformed region which forms a truncated cone. The shear

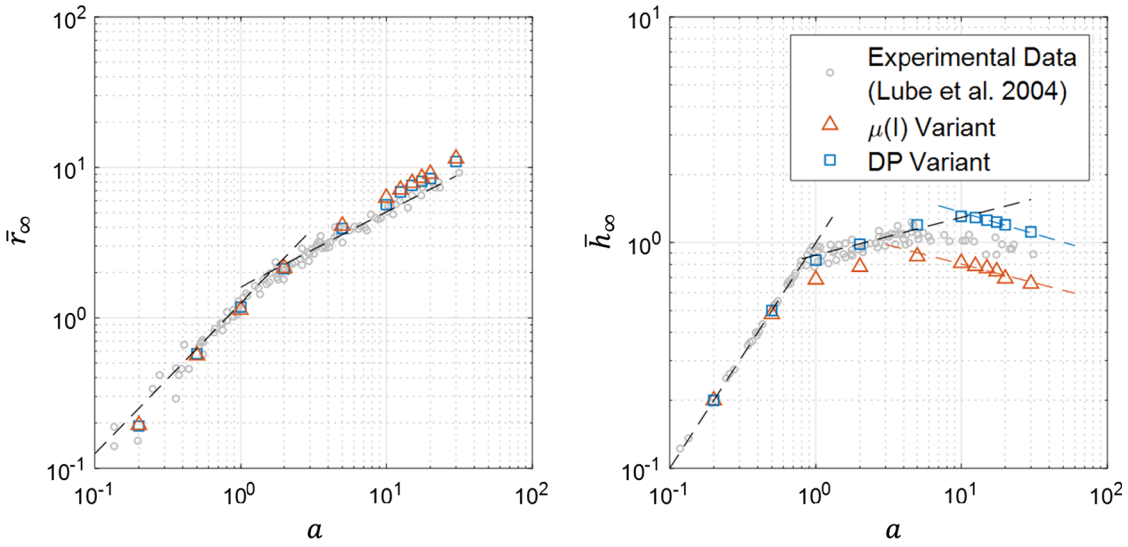

Fig. 12. $\bar{r}_\infty$-a and $\bar{h}_\infty$-a relationships obtained from column collapse simulations.

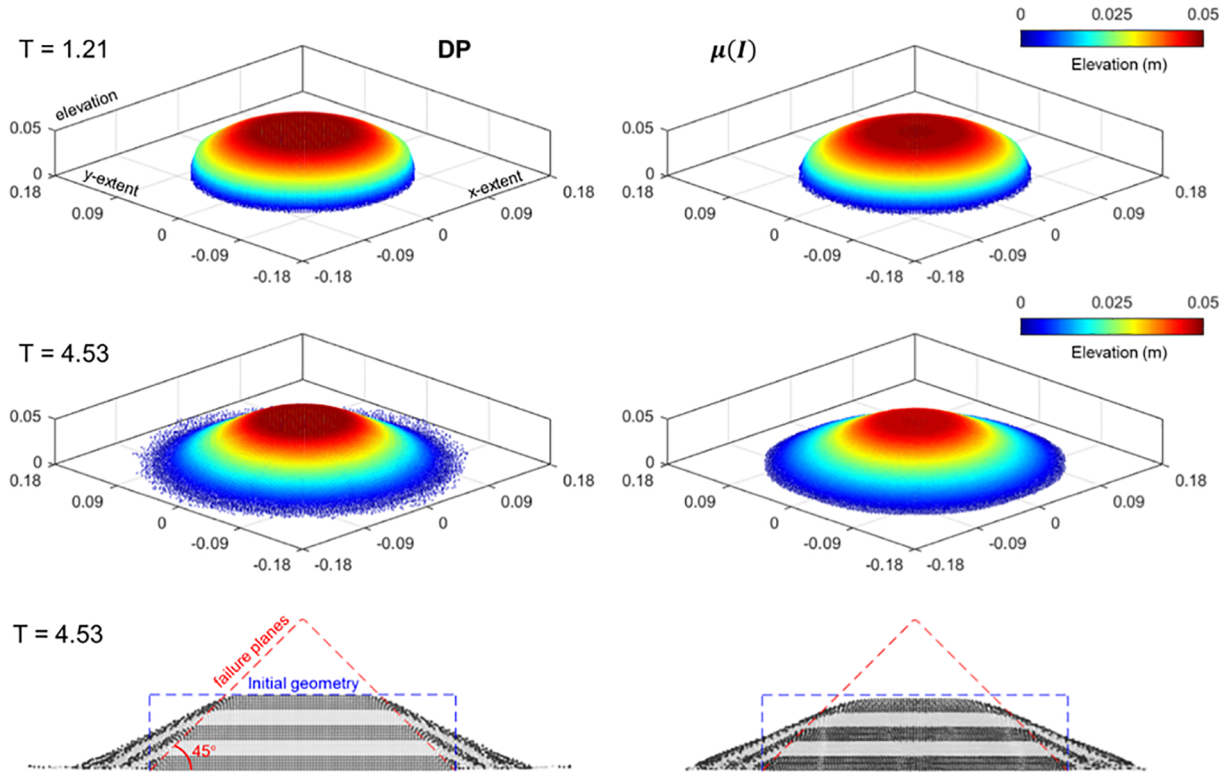

Fig. 13. Collapse progression for column with $a = 0.5$.

plane geometry is the same for both models and is independent of aspect ratio. However, initial parametric studies show that shear planes are dependent on the ϕ and ψ angles chosen for the DP model, and the μ_s chosen for the $\mu(I)$ model. When marking the column into horizontal layers, the free surface of the deformed region of the deposit is covered by the top-most layer and layers remain separated with respect to the initial configuration as lower layers of material are deposited first and subsequent layers travel over it.

Noticeable in the $\mu(I)$ final deposit is a small downward displacement of the top of the column. This downward displacement occurs at an order of 10^{-2} m/s and continues even after flow arrest. This creeping motion has been observed in other examples of implementation of the $\mu(I)$ model into fluid-based schemes [48]. In [48], it was suggested that that this is caused by regularization of the model where the effective viscosity is restricted to a finite value. However, this cannot be the case here as such regularisations are not implemented. Noticing that the creeping motion is concentrated at the free surface, we suppose that the creeping occurs as a consequence of low-pressure conditions and ill-posedness of the $\mu(I)$ model in the quasi-static regime which can introduce perturbations in the stress profile [74].

3.10. Flow kinematics in the second regime

In the numerical results presented here, the second regime begins around $a \cong 2$ and extends to $a = 5$ for the $\mu(I)$ model, and 10 for the DP model results. Fig. 14 shows the progression of the collapse for a column with $a = 2$. For both models, flows are initialized at the base of the column along 45° inclined failure planes (the same as small aspect ratios), except the entirety of the top face is involved in the collapse. Curvature in the top face is visible very soon after collapse begins. The final deposit shape in both models is cone-like in shape with curved vertex and non-linear slopes. A curved vertex instead of a pointed vertex, as observed in experiments, is probably a consequence of the continuous nature of the constitutive relations. The un-deformed region is still defined by 45° inclined planes, but the vertex is eroded so that it is also curved. More severe erosion of the vertex occurs in the $\mu(I)$ model so that the un-deformed region is more dome than cone-shaped. When viewing the collapse of the column as marked by vertical layers (Fig. 15), initial spreading consists mostly of the first and second outermost layers, and the innermost layer begins to spread after the top part of the column has fallen significantly. Lube et al. [51] reports the presence of free surface waves during flow. But here, the numerical results do not show any wave propagation at the free surface as the collapse progresses.

3.11. Flow kinematics in the third regime

The third regime is characterised by the negative power law relationship in the $\bar{h}_\infty$-a profile. As shown in Fig. 16, the entire final deposit can no longer be considered conical due to the presence of free-surface inflection points or ridges in addition to non-linear slopes. However, the central region is always a conical shape with a curved vertex or dome-like shape. Lateral spreading and consequential shortening of the basal layer of the column predicted by the $\mu(I)$ model leads to the under-prediction of the final deposit height. In general, the overall thickness of the final deposit predicted by the $\mu(I)$ model is much thinner compared to the DP model, which is consequential of the different compressibility assumptions used in the two models.

Fig. 17 shows the collapse progression for $a = 30$ for both code variants. Initialisation of the flows follows the same pattern as observed for intermediate aspect ratios where material near the base flows outward. Here, any part of the column that is approximately above $3r_0$ shows no disturbance in the free surface, indicating that it is in a free fall flow regime. As falling material reaches the base, it is ejected outward radially. Under these circumstances, frictional behaviours of the granular material alone no longer dominate flow trajectories. Instead, motion of the freefalling material is governed predominantly by the accumulated kinetic energy from the time spent in free-fall, which is a behaviour shown in both models. Just before the entire height of the column has collapsed, outward propagating material forms a wave at

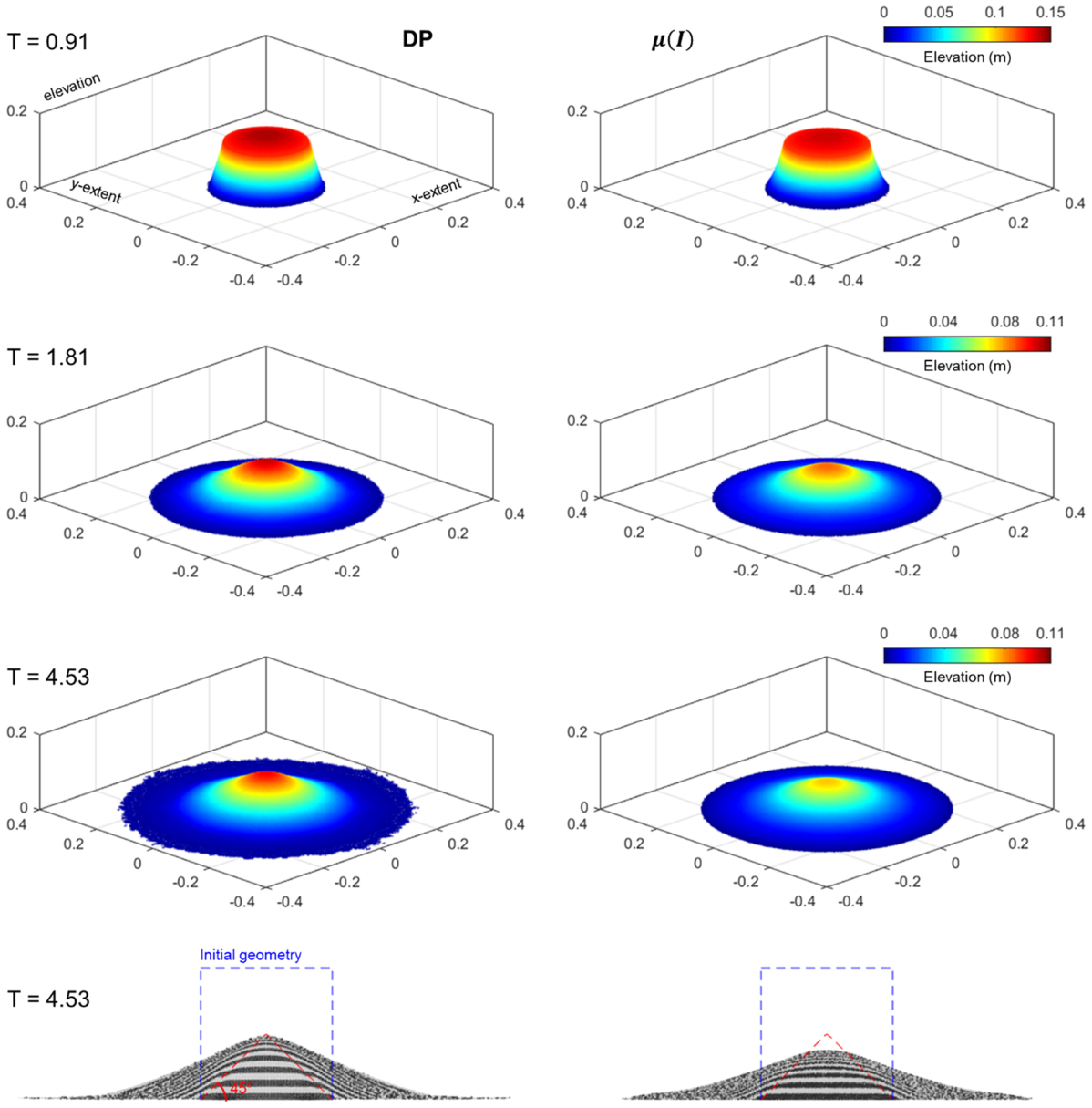

Fig. 14. Collapse progression for column with $a = 2.0$.

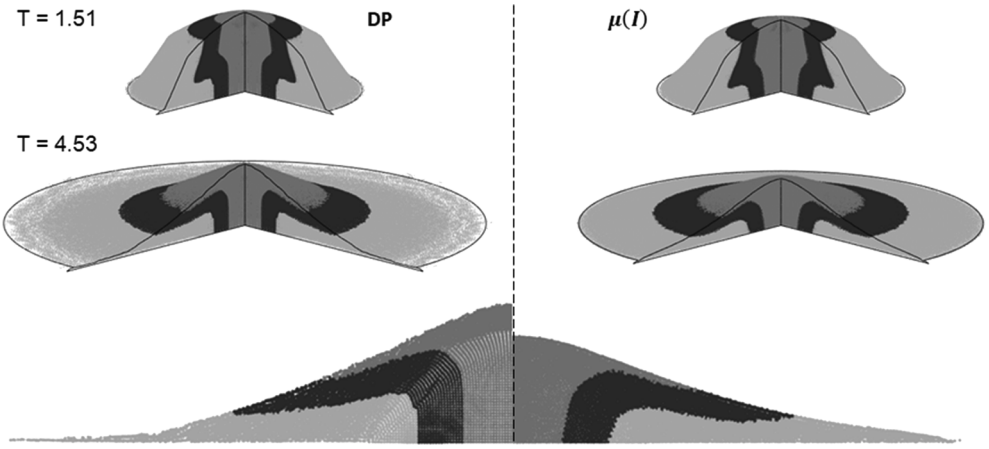

Fig. 15. Collapse progression for column with a = 2.0 initially marked into three concentric vertical layers.

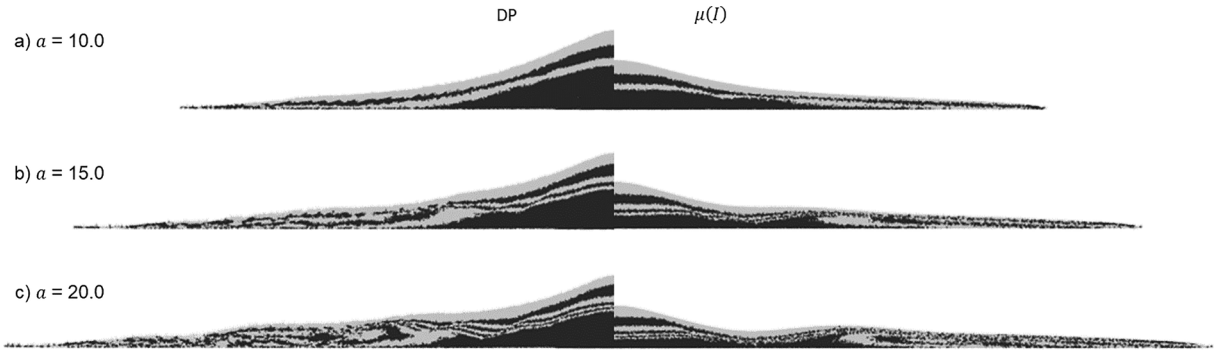

Fig. 16. Comparison of final deposits for columns with large aspect ratios. Non-linear slopes are present for (a), (b), and (c); as well as inflection points and free-surface ridges in c).

the free surface that is tallest just after the entire column has reached the base, but then reduces in height as it eventually halts. Some material from the wave continues to the flow front, thickening the front and causing a short acceleration. The final deposit left is what has been referred to as a 'mexican-hat' geometry by Lajeunesse et al. [70].

An advantage of numerical methods is the ability to study the internal structure of the flow. Fig. 18 shows the collapse of the column when initially marked by horizontal layers 0.25 m thick for $a = 30$. At flow initialisation, a quasi-static region at the base of the column is present. Both code variants predict deformation of this quasi-static region, but in different manners leading to slightly different internal flow kinematics. The DP model predicts slow erosion and minor vertical compression of the quasi-static region as it is eventually flattened into a dome by falling material. This dome alters the path of the falling material forcing it to deviate outward. In the results produced using the DP model, the falling mass is observed to cause the already deposited layers to be pushed downward and outward, eventually creating a slow moving sub-surface wave. The $\mu(I)$ model result differs in the region near the base where the quasi-static regime is not maintained, and instead is spread outward such that the behaviour resembles a water jet impacting on a wall. In either case, it is clear that the energy dissipation processes at these large aspect ratios are complex and difficult to characterise.

3.12. Stress in the column

Fig. 19 shows the vertical stress profiles produced by both SPH code variants at the beginning of collapse for $a = 1.0$, 2.0, and 20.0. At the onset of collapse, both models show a concentration of stress beneath the initial failure planes as shear stresses from the collapse are transferred to the base. For $a = 1.0$, the $\mu(I)$ model vertical stress approximates a hydrostatic solution at the centre of the column, whereas the DP model predicts a stress dip at the column centre. For $a = 2.0$, the DP model predicts a non-linear increase in stress with depth along the column centreline, except near the base, where the stress decreases slightly to $\sigma^{zz}/\rho g = 0.1$ m. At this aspect ratio, along the centreline, the $\mu(I)$ model predicts the vertical stress to increase non-linearly and monotonically also to $\sigma^{zz}/\rho g = 0.1$ m at the base. In both sets of results, both models show minimal change in vertical stress profile as the aspect ratio is increased beyond $a = 2.0$. Also clear in the figure is the stress noise in the profile produced by the $\mu(I)$ model. The noise is due to the $\mu(I)$ model being ill-posed when in the quasi-static regime. We note that solutions to this problem exist [75,76], but investigation is left for future work.

For collapsing columns with aspect ratios less than 10, the final deposit approximates a cone shape. Previous work [77–79] has noted the presence of arching in conical sand piles that leads to unintuitive non-hydrostatic stress profiles. Often noted is the presence of a stress dip at the centre of the pile. To investigate whether the models adopted in this work capture this phenomenon, basal stress of the collapsed column is measured. To avoid any numerical noise due to boundary effects, stresses are measured at $3h$ above the base (0.0072 m) using the corrected SPH approximation [65]. These measured basal stresses are shown in Fig. 20 for both models for $a = 0.5$ and 2.0, alongside the equivalent hydrostatic stress profile and experimental data from [79] where the pressure is measured at the base of a perfectly conical sand pile. The SPH results and experimental data are normalized with respect to maximum measured stress and deposit radius for the purposes of qualitative trend comparison. The $\mu(I)$ model predicts basal stresses that are close to the hydrostatic solution. On the other hand, we notice that for conical final deposits obtained with the SPH code variant using the DP model, there is a stress dip at the centre of the deposit that increases with radial position until $r_0/2$, where thereafter the stress approximately follows the hydrostatic solution. A stress dip is still observed for $a = 0.5$ despite having a truncated cone final deposit geometry instead of a conical one. This could still be considered realistic, considering findings by Hummel & Finnan [77], who found that after deposition of a conical sand pile, the removal of material from the top of the pile did not influence the stress dip. Another observation is the significant deviation between the $a = 2.0$ stress results and the experimental results in the decreasing part of the stress profile. This can be attributed to the non-linear slopes in the final deposit of the collapsing column simulated by SPH. Having observed the central stress dip from the results using the DP model and not the $\mu(I)$ model suggests that the elasto-plasticity approach is necessary to capture arching effects in granular material.

4. Conclusions

A scalable parallel SPH numerical tool for granular flows, parallelized using an MPI-based parallel scheme targeting scalability on large-scale CPU clusters using the ORB domain partitioning algorithm has been presented. The applicability of the scheme is demonstrated by implementing two different stress–strain relations into two separate code variants: the $\mu(I)$ rheological model [54] and an elasto-plastic approach [23]. Scaling tests were conducted on two supercomputers demonstrating weak scaling efficiency of up to 95% and strong scaling speedup of up to 900 times at 1,024 CPU cores. Scaling tests have also revealed bottlenecks that will need to be addressed if scalability to large core counts is desired. Firstly, strong scalability was shown to be affected by growth in message communication times associated with physical and halo particle distribution. A possible solution is to introduce greater overlap of these communications with meaningful computation. The second bottleneck identified is the memory required to construct a global particle-in-cell matrix. While this can be mitigated by increasing the grid cell sizes, to maintain the load-balancing performance of the ORB algorithm, a subdivision refinement algorithm would need to be introduced. Another possible solution would be to

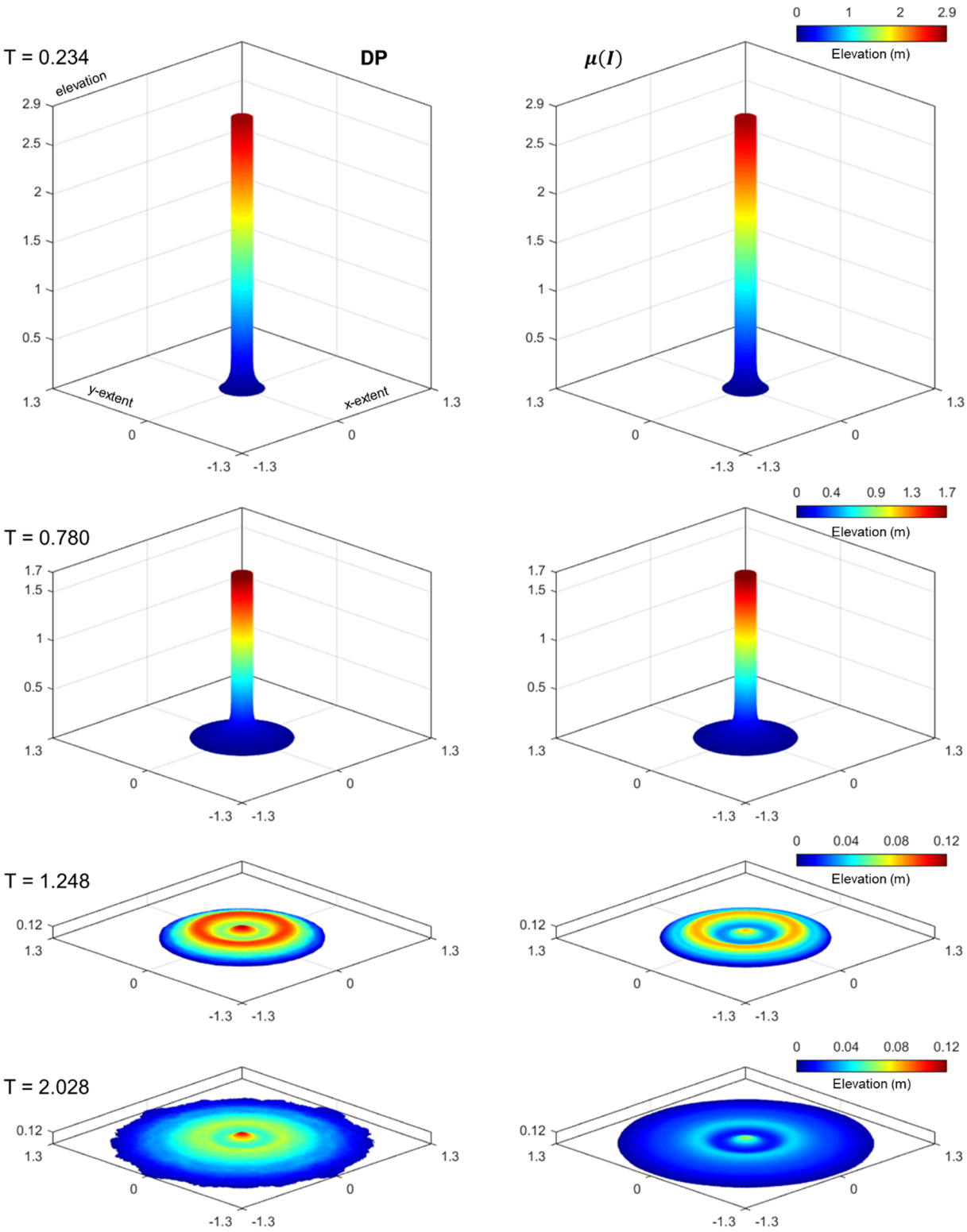

Fig. 17. Collapse progression for column with $a = 30.0$.

hybridize the parallel scheme with OpenMP as this would allow for a single copy of the particle-in-cell matrix to be shared amongst multiple CPU cores. More complex solutions that avoid requiring the entire particle-in-cell matrix to be available to each process can also be considered.

In addition to demonstrating the scalability of the scheme at core counts of up to 1,024, we model the granular column collapse experiment at aspect ratios up to 30 under full 3D axisymmetric conditions at relatively high resolution. The tallest simulation consisted of a column of aspect ratio equal to 30 modelled by 11.7 million physical SPH particles and took approximately 12–24 h on 512 CPU cores to simulate 140,000 time-steps, depending on the hardware and compiler performance. The numerical results reproduce very well the scaling relationships obtained between runout distance and initial height, and aspect ratios during all regimes reported in the literature. Due to the high-resolution modelling of the granular column collapse, at large

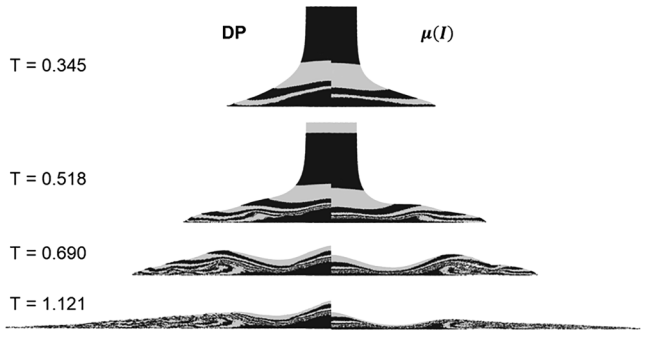

Fig. 18. Collapse progression for column with $a = 30.0$ when initially marked into horizontal layers 0.25 m thick.

aspect ratios both code variants were capable of capturing free-surface artefacts such as inflection points and free-surface wave propagations. We observed that these artefacts were a consequence of an accumulation of material into a slow-moving heap beneath the surface that acts as an obstacle. Differences in the code variants were most obvious in the final deposit height, overall volume change, and development of free-surface waves, which was mostly a consequence of differences in compressibility and modelling of the static-fluid regime change. The DP model more accurately reproduces the $\bar{h}_\infty$-a relationship up to an aspect ratio of 5 when compared to the $\mu(I)$ model, suggesting that the dilative response of an elasto-plastic model is more physically realistic than a weakly-compressible or incompressible assumption for the collapse of granular columns. When examining the vertical stress profile of the two code variants for columns with small aspect ratios, we found that the $\mu(I)$ variant generally predicted hydrostatic stress conditions throughout the entire collapse as well as final deposit, whereas the DP variant predicted a stress dip at the centre of the column, which was noted to be similar to static sand pile experiment reported in the literature. Future work will consist of applying the parallel scheme to the modelling of large-scale geophysical granular flows, such as landslides and debris flow.

CRediT authorship contribution statement

Edward Yang: Conceptualization, Methodology, Writing - original draft, Software, Visualization, Investigation, Validation, Data curation, Resources. **Ha H. Bui:** Software, Supervision, Conceptualization, Methodology, Writing - review & editing, Data curation, Project administration, Funding acquisition, Resources. **Hans De Sterck:** Methodology, Conceptualization, Writing - review & editing, Funding acquisition, Resources. **Giang D. Nguyen:** Conceptualization, Methodology, Software, Writing - review & editing, Funding acquisition. **Abdelmalek Bouazza:** Conceptualization, Methodology, Writing - review & editing.

Declaration of Competing Interest

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

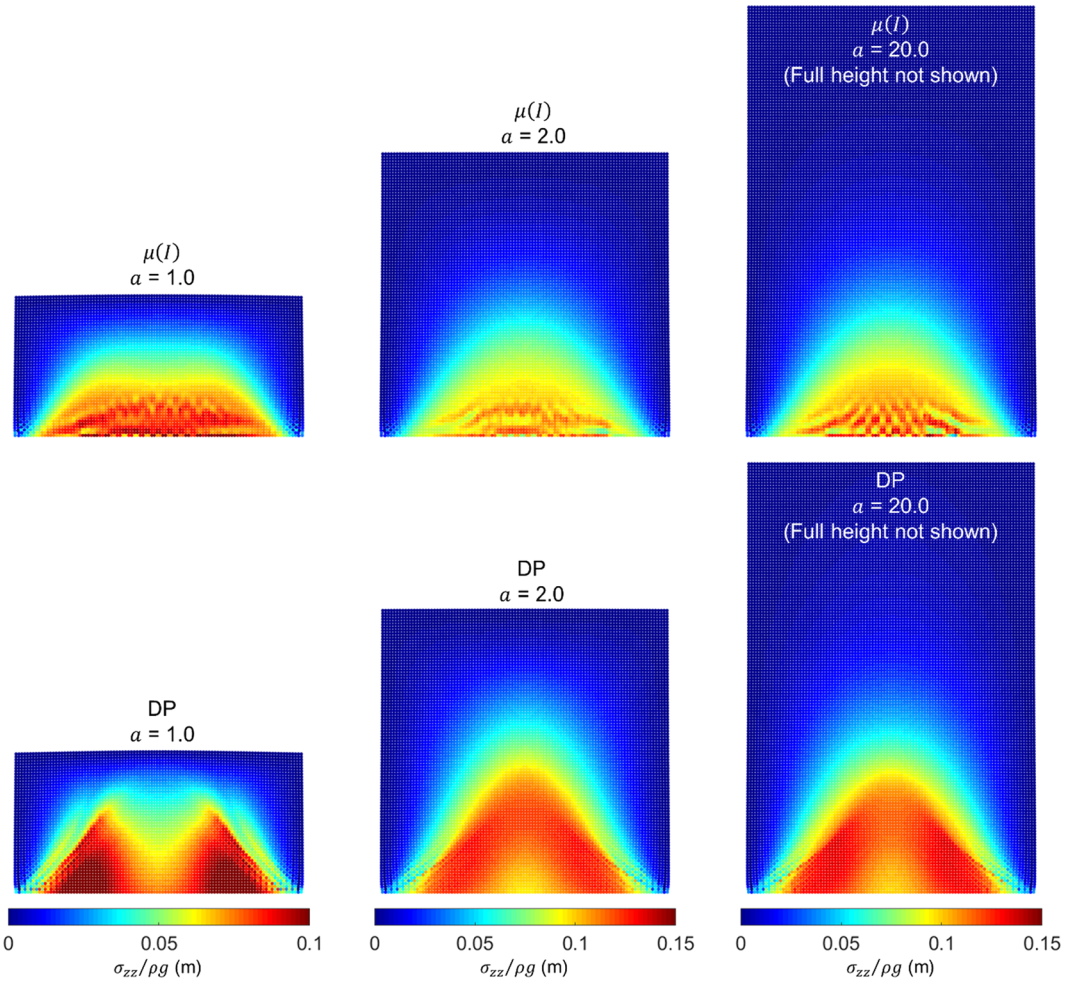

Fig. 19. Vertical stress profiles for $a = 1.0$, 2.0, and 20.0 just after collapse initiates (at $t = 0.025$s).

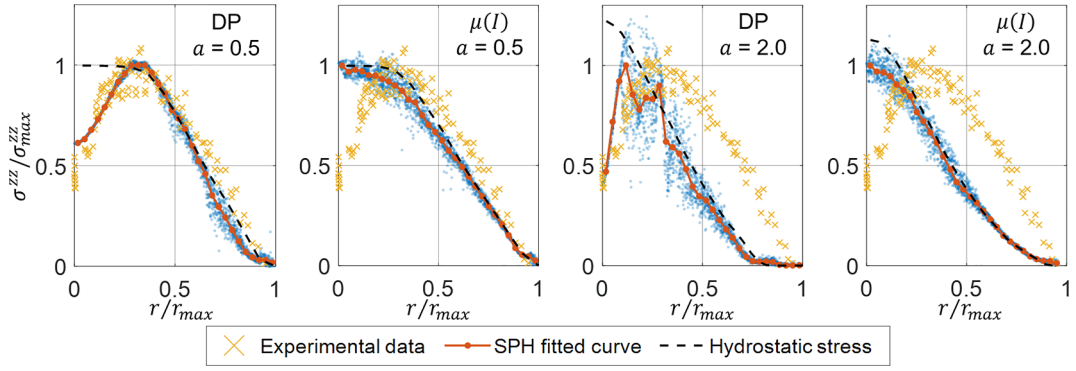

Fig. 20. Comparison of basal vertical stresses obtained from SPH simulations at final collapse, with experimental data obtained from [63], and a theoretical hydrostatic solution. Stress results are normalized with respect to maximum measured stress, and radial position is normalized with respect to maximum effective runout.

Acknowledgement

Funding support from the Australian Research Council via projects DP160100775, DP170103793 and DP190102779, and from NSERC Canada via project RGPIN-2019-04155, is gratefully acknowledged. This research was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

References

[1] Yin Y, Wang F, Sun P. Landslide hazards triggered by the 2008 Wenchuan earthquake, Sichuan, China. Landslides 2009;6(2):139–52.
[2] Rosser B, et al. New Zealand's national landslide database. Landslides 2017;14(6):1949–59.
[3] Del Negro C, et al. Lava flow hazards at Mount Etna: constraints imposed by eruptive history and numerical simulations. Sci Rep 2013;3:3493.
[4] Iverson RM, Schilling SP, Vallance JW. Objective delineation of lahar-inundation hazard zones. Geol Soc Am Bull 1998;110(8):972–84.
[5] Assier-Rzadkieaicz S, et al. Numerical modelling of a landslide-generated tsunami: the 1979 Nice event. Pure Appl Geophys 2000;157(10):1707–27.
[6] Christen M, et al. Integral hazard management using a unified software environment. 12th congress interpraevent. 2012.
[7] Favalli M, et al. Forecasting lava flow paths by a stochastic approach. Geophys Res Lett 2005;32(3).
[8] Herault A, et al. Forecasting lava flow hazards during the 2006 Etna eruption: using the MAGFLOW cellular automata model. Comput Geosci 2009;35(5):1050–60.
[9] McEwen AS, Malin MC. Dynamics of Mount St. Helens' 1980 pyroclastic flows, rockslide-avalanche, lahars, and blast. J Volcanol Geoth Res 1989;37(3–4):205–31.
[10] Pastor M, et al. A depth-integrated, coupled SPH model for flow-like landslides and related phenomena. Int J Numer Anal Meth Geomech 2009;33(2):143–72.
[11] Cascini L, et al. SPH run-out modelling of channelised landslides of the flow type. Geomorphology 2014;214:502–13.
[12] Griffiths D, Marquez R. Three-dimensional slope stability analysis by elasto-plastic finite elements. Geotechnique 2007;57(6):537–46.
[13] Sulsky D, Chen Z, Schreyer HL. A particle method for history-dependent materials. Comput Methods Appl Mech Eng 1994;118(1):179–96.
[14] Idelsohn SR, Oñate E, Pin FD. The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. Int J Numer Meth Eng 2004;61(7):964–89.
[15] Bardenhagen SG, Kober EM. The generalized interpolation material point method. Comput Model Eng Sci 2004;5(6):477–96.
[16] Zhang X, et al. Particle finite element analysis of large deformation and granular flow problems. Comput Geotech 2013;54:133–42.
[17] Abe K, et al. Numerical study on dynamic behavior of slope models including weak layers from deformation to failure using material point method. Soils Found 2017;57(2):155–75.
[18] Bandara S, Soga K. Coupling of soil deformation and pore fluid flow using material point method. Comput Geotech 2015;63:199–214.
[19] Nguyen NH, et al. Discrete element method investigation of particle size distribution effects on the flexural properties of cement-treated base. Comput Geotech 2019;113:103096.
[20] Nguyen NHT, et al. A discrete element modelling approach for fatigue damage growth in cemented materials. Int J Plast 2019;112:68–88.
[21] Tong C-X, et al. A stochastic particle breakage model for granular soils subjected to one-dimensional compression with emphasis on the evolution of coordination number. Comput Geotech 2019;112:72–80.
[22] Tran H, Wang Y, Nguyen GD, Kodikara J, Sanchez M, Bui HH. Modelling 3D desiccation cracking in clayey soils using a size-dependent SPH computational approach. Comput Geotech 2019;116:103209https://doi.org/10.1016/j.compgeo.2019.103209. (in press).
[23] Bui HH, et al. Lagrangian meshfree particles method (SPH) for large deformation and failure flows of geomaterial using elastic–plastic soil constitutive model. Int J Numer Anal Meth Geomech 2008;32(12):1537–70.
[24] Bui HH, et al. A novel computational approach for large deformation and post-failure analyses of segmental retaining wall systems. Int J Numer Anal Meth Geomech 2014;38(13):1321–40.
[25] Bui, H.H., et al. SPH-based numerical simulations for large deformation of geomaterial considering soil-structure interaction. In: The 12th international conference of international association for computer methods and advances in geomechanics (IACMAG); 2008.
[26] Ghaïtanellis A, et al. A SPH elastic-viscoplastic model for granular flows and bed-load transport. Adv Water Resour 2018;111:156–73.
[27] Nguyen CT, et al. A new SPH-based approach to simulation of granular flows using viscous damping and stress regularisation. Landslides 2017;14(1):69–81.
[28] Neto AHF, Borja RI. Continuum hydrodynamics of dry granular flows employing multiplicative elastoplasticity. Acta Geotech 2018;13(5):1027–40.
[29] Bui HH, Nguyen GD. A coupled fluid-solid SPH approach to modelling flow through deformable porous media. Int J Solids Struct 2017;125:244–64.
[30] Wang Y, et al. A new SPH-based continuum framework with an embedded fracture process zone for modelling rock fracture. Int J Solids Struct 2019;159:40–57.
[31] Zhao S, et al. A generic approach to modelling flexible confined boundary conditions in SPH and its application. Int J Numer Anal Meth Geomech 2019;43(5):1005–31.
[32] Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. MNRAS 1977;181(3):375–89.
[33] Lucy LB. A numerical approach to the testing of the fission hypothesis. The astronomical journal 1977;82:1013–24.
[34] Monaghan JJ. Simulating Free Surface Flows with SPH. J Comput Phys 1994;110(2):399–406.
[35] Libersky LD, Petschek AG. Smooth particle hydrodynamics with strength of materials, in Advances in the free-Lagrange method including contributions on adaptive gridding and the smooth particle hydrodynamics method. Springer; 1991. p. 248–57.
[36] Bui HH, et al. Slope stability analysis and discontinuous slope failure simulation by elasto-plastic smoothed particle hydrodynamics (SPH). Geotechnique 2011;61(7):565–74.
[37] Bui HH, Fukagawa R. 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 Meth Geomech 2013;37(1):31–50.
[38] Peng C, et al. Unified modelling of granular media with Smoothed Particle Hydrodynamics. Acta Geotech 2016;11(6):1231–47.
[39] Bui HH, Sako K, Fukagawa R. Numerical simulation of soil–water interaction using smoothed particle hydrodynamics (SPH) method. J Terramech 2007;44(5):339–46.
[40] Huang Y, et al. Run-out analysis of flow-like landslides triggered by the Ms 8.0 2008 Wenchuan earthquake using smoothed particle hydrodynamics. Landslides 2012;9(2):275–83.
[41] Minatti L, Paris E. A SPH model for the simulation of free surface granular flows in a dense regime. Appl Math Model 2015;39(1):363–82.
[42] Peng X, et al. Parallel computing of three-dimensional discontinuous deformation analysis based on OpenMP. Comput Geotech 2019;106:304–13.
[43] Zhan L, et al. Three-dimensional modeling of granular flow impact on rigid and deformable structures. Comput Geotech 2019;112:257–71.
[44] Oger G, et al. On distributed memory MPI-based parallelization of SPH codes in massive HPC context. Comput Phys Commun 2016;200:1–14.
[45] Valdez-Balderas D, et al. Towards accelerating smoothed particle hydrodynamics simulations for free-surface flows on multi-GPU clusters. J Parallel Distrib Comput 2013;73(11):1483–93.
[46] Roche O, et al. On the run-out distance of geophysical gravitational flows: Insight from fluidized granular collapse experiments. Earth Planet Sci Lett 2011;311(3):375–85.
[47] Staron L, Hinch E. Study of the collapse of granular columns using two-dimensional

discrete-grain simulation. J Fluid Mech 2005;545:1–27.
[48] Lagrée P-Y, Staron L, Popinet S. The granular column collapse as a continuum: validity of a two-dimensional Navier-Stokes model with a μ (I)-rheology. J Fluid Mech 2011;686:378–408.
[49] Mast CM, et al. Simulating granular column collapse using the Material Point Method. Acta Geotech 2015;10(1):101–16.
[50] Zhang X, Krabbenhoft K, Sheng D. Particle finite element analysis of the granular column collapse problem. Granular Matter 2014;16(4):609–19.
[51] Lube G, et al. Axisymmetric collapses of granular columns. J Fluid Mech 2004;508:175–99.
[52] Fleissner F, Eberhard P. Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection. Int J Numer Meth Eng 2008;74(4):531–53.
[53] Schornbaum F, Rüde U. Massively parallel algorithms for the lattice Boltzmann method on nonuniform grids. SIAM J Sci Comput 2016;38(2):C96–126.
[54] Jop P, Forterre Y, Pouliquen O. A constitutive law for dense granular flows. Nature 2006;441:727.
[55] Eslamian A, Khayat M. A novel hybrid solid-like fluid-like (SLFL) method for the simulation of dry granular flows. Particuology 2017;31:200–19.
[56] Chambon G, et al. Numerical simulations of granular free-surface flows using smoothed particle hydrodynamics. J Nonnewton Fluid Mech 2011;166(12):698–712.
[57] Chen W, Qiu T. Simulation of earthquake-induced slope deformation using SPH method. Int J Numer Anal Meth Geomech 2014;38(3):297–330.
[58] Chen WF, Mizuno E. Nonlinear Analysis in Soil Mechanics: Theory and Implementation. Elsevier; 1990.
[59] Monaghan JJ, Lattanzio JC. A refined particle method for astrophysical problems. A & A 1985;149:135–43.
[60] Molteni D, Colagrossi A. A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. Comput Phys Commun 2009;180(6):861–72.
[61] Monaghan JJ. Smoothed particle hydrodynamics and its diverse applications. Annu Rev Fluid Mech 2012;44:323–46.
[62] Oger G, et al. An improved SPH method: Towards higher order convergence. J Comput Phys 2007;225(2):1472–92.
[63] Bui HH. Lagrangian mesh-free particle method (SPH) for large deformation and post-failure of geomaterial using elasto-plastic constitutive models [PhD thesis]. Japan: Ritsumeikan University; 2007.
[64] Liu M, Shao J, Chang J. On the treatment of solid boundary in smoothed particle hydrodynamics. Sci China Technol Sci 2012;55(1):244–54.
[65] Chen JK, Beraun JE. A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems. Comput Methods Appl Mech Eng 2000;190(1):225–39.
[66] Gui-rong L. Smoothed particle hydrodynamics: a meshfree particle method. World Scientific; 2003.
[67] Cherfils JM, Pinon G, Rivoalen E. JOSEPHINE: A parallel SPH code for free-surface flows. Comput Phys Commun 2012;183(7):1468–80.
[68] Domínguez JM, et al. Neighbour lists in smoothed particle hydrodynamics. Int J Numer Meth Fluids 2011;67(12):2026–42.
[69] Schloegel K, Karypis G, Kumar V. Parallel static and dynamic multi-constraint graph partitioning. Concurr Comput: Pract Exp 2002;14(3):219–40.
[70] Lajeunesse E, Mangeney-Castelnau A, Vilotte JP. Spreading of a granular mass on a horizontal plane. Phys Fluids 2004;16(7):2371–81.
[71] Lajeunesse E, Monnier JB, Homsy GM. Granular slumping on a horizontal surface. Phys Fluids 2005;17(10):103302.
[72] Goldhirsch I. Rapid granular flows. Annu Rev Fluid Mech 2003;35(1):267–93.
[73] Roscoe KH. The influence of strains in soil mechanics. Geotechnique 1970;20(2):129–70.
[74] Barker T, et al. Well-posed and ill-posed behaviour of the ${\it\mu}(I) $-rheology for granular flow. J Fluid Mech 2015;779:794–818.
[75] Barker T, Gray J. Partial regularisation of the incompressible ${\it\mu}(I) $-rheology for granular flow. J Fluid Mech 2017;828:5–32.
[76] Schaeffer D, et al. Constitutive relations for compressible granular flow in the inertial regime. J Fluid Mech 2019;874:926–51.
[77] Hummel F, Finnan E. The distribution of pressure on surfaces supporting a mass of granular material. Minutes of the Proceedings of the Institution of Civil Engineers. Thomas Telford-ICE Virtual Library; 1921.
[78] Trollope D, Burman B. Physical and numerical experiments with granular wedges. Geotechnique 1980;30(2):137–57.
[79] Vanel L, et al. Memories in sand: Experimental tests of construction history on stress distributions under sandpiles. Phys Rev E 1999;60(5):R5040.