Powder Technology 407 (2022) 117631

Contents lists available at ScienceDirect

Powder Technology

journal homepage: www.journals.elsevier.com/powder-technology

Bayesian calibration of GPU–based DEM meso-mechanics Part I: Parallelization of RVEs

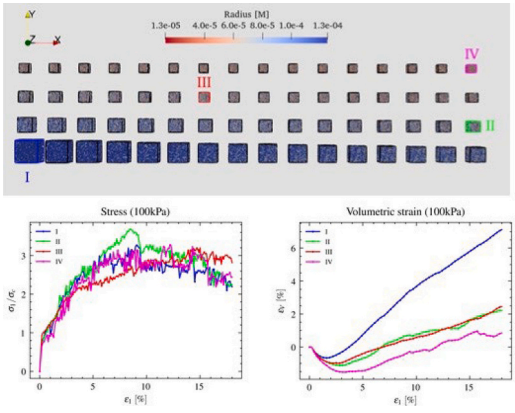
Retief Lubbe [a], Wen-Jie Xu [a,*], Qian Zhou [a], Hongyang Cheng [b]

[a] State Key Laboratory of Hydroscience and Department of Hydraulic Engineering, Tsinghua University, Beijing, China 100084
[b] Department of Civil Engineering, Faculty of Engineering, Technology, MESA+, University of Twente, P.O Box 217, 7500 AE Enschede, the Netherlands

HIGHLIGHTS

- A novel algorithm for the parallelization of GPU based DEM at the level of the RVE is presented.
- Simulation level parallelism of independent RVEs is provided.
- A low latency and memory efficient implementation of deformable PBC is performed.
- Modified UG and BVH contact detection algorithms is used to partition the simulation index into the hashing keys.
- Drained DEM triaxial test is used to validate the algorithm on dry graded quartz.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Discrete element method (DEM)
Graphical Processor Unit (GPU)
Periodic boundary conditions (PBC)
Parameter calibration
Representative Volume Element (RVE)

ABSTRACT

Calibration of Discrete Element Method (DEM) parameters is essential for modeling geotechnical applications. This task can, however, be extremely tedious or sometimes even impossible to undertake. This is largely due to two issues namely: (1) a large sample size of DEM simulations and number of sampling iterations are necessary to accurately infer the probability distribution of a model over a large parameter space and (2) DEM is computationally intractable compared to other numerical methods. In the scope of reducing the number of sampling iterations, automatic calibration techniques are available to extract and make use of the hidden contact meso-structure correlations through adaptive sampling. Coincidentally, to improve computational speed, significant advances toward Graphics Processor Unit (GPU) based DEM algorithms have been achieved over the past years on particle parallelism. Nevertheless, the problem remains that DEM simulations are serialized during the calibration processes. While the companion paper addresses parameter calibration, this study presents a novel algorithm to parallelize independent simulations within a sample set. The selected system is the Representative Volume Element (RVE) which is widely used in geotechnics for solving soil response in the static regime. The algorithm includes the following key features: (1) simulation level parallelism of non-interacting RVEs through highly efficient hierarchical memory groups and access patterns (2) a low latency and memory-efficient implementation of deformable periodic boundary conditions (PBC) which uses lookup tables and bitmasks (3)

* Corresponding author.
E-mail address: wenjiexu@tsinghua.edu.cn (W.-J. Xu).

https://doi.org/10.1016/j.powtec.2022.117631
Received 6 February 2022; Received in revised form 6 June 2022; Accepted 12 June 2022
Available online 17 June 2022

modified Uniform Grid and Bounding Volume Hierarchy (BVH) contact detection algorithms which partitions the RVE index into the hashing keys. The drained DEM triaxial compression is used to validate the algorithm on dry graded quartz. Three performance degrading factors for the calibration processes are considered: (1) the number of particles per RVE (2) calibration sample size and (3) sequential launch time per calibration step. This algorithm shows a factor of about 9.8 times speedup when parallelizing 100 DEM RVEs in one batch.

1. Introduction

Calibration of physical parameters for numerical models is essential for simulations to emulate real-world engineering problems. This process involves tuning model parameters such that the numerical response matches the experimental response. Well-established numerical models are generalizable enough to solve various problems but cumbersome to calibrate, especially for complex non-linear applications such as those modeling granular materials.

The Discrete Element Method (DEM) put forward by Cundall and Strack [1] is a fundamental numerical tool used to simulate the macroscopic behavior of granular material through mechanical interactions of discrete particles. Graphics Processing Unit (GPU) based DEM is becoming more relevant in large scale engineering problems [2,3,4,5] given the rise of high-performance parallel computing hardware and distributed nature of particles and their interactions.

The Representative Volume Element (RVE) is the smallest volume meso-structure used to constitute the macroscopic characteristics of soil [6,7]. Oftentimes, rigid walls are often used with a servo-control mechanism to impose some stress or strain conditions. However, the packing fraction at the wall is generally lower than the rest of the sample, and the stress response at the wall is different from that of the center [8]. Periodic boundary conditions (PBC) are used in RVEs to reduce the effects of rigid wall boundaries [9].

There is a growing need to simulate many calibrated independent RVEs. Among these methods is the hierarchical Finite Element Method - Discrete Element Method (FEM-DEM) multiscale simulations such as those from N. Guo, et al. [10] and others [11,12,13]. The hierarchical FEM-DEM simulations replace the constitutive model with the response stress deformation response of an RVE. This is achieved by mapping each RVE to a material point or Gaussian integration point in the FEM mesh. The parallelization of RVEs is significant in improving the performance of these simulations. Yade [14] can support independent RVEs as scenes but is developed on the CPU. S. Zhao et al. [15] developed a 2D parallel thread-block-wise GPU-based RVE code for DEM and Material Point Method (MPM) coupling. Finally, [13] performed multi-scale hierarchical FEM-DEM parallel simulations on RVEs for dry sand.

Another application to simulate many RVEs is the automatic calibration of DEM parameters such as genetic algorithms implemented by [16] and Bayesian inference algorithms developed [17,18]. These algorithms may require many RVEs for a large parameter set to be calibrated accurately and efficiently.

There are many challenges to achieving parallelism at an RVE level: (1) GPU algorithms are implemented in a thread-structured pattern which limits how memory is accessed between particles; (2) A particle may have up to 7 boundary particles that interact differently with other real and boundary particles during the contact detection and force calculation stages. For instance, if a real particle interacts with an another particle's boundary particle, force symmetry should ensure that the real particles register contact with each other while the contact is only made among the boundary and real particles; (3) Deformation of RVEs modifies the local coordinates and the velocities of the particles; (4) Parallel non-interacting RVEs may have a unique simulation state and initial configuration; (5) Contact detection algorithms are only able to partition one simulation domain at a time and do not consider the special rules of boundary particles; (6) Special techniques such as atomic functions or parallel reductions are necessary for threads to communicate to each other and perform volume averaging of state variables such as the stress.

This study addresses these issues by introducing a novel algorithm for GPU-based DEM to simulate many independent RVEs for the drained triaxial test. The code is developed into the existing CoSim-DEM [13] framework and used for generating a large statistical sample size and the fast calibration in the companion paper.

The rest of the paper is structured as follows: Section 2 provides an overview of the basic DEM contact model; Section 3 introduces the Moment Rotation Law (MRL) which is used to introduce grain roughness for triaxial simulations; Section 4 gives an overview of the DEM integration scheme; Section 5 discusses the RVE and presents the stress and strain invariants for the triaxial test; Section 6 introduces the parallel piped unit cell and transformation between local and global coordinate systems; Section 7 presents the implementation of the GPU algorithm; Section 8 shows the homogenization procedure; Section 9 presents a validation of the DEM simulation to the drained triaxial sand of dry sand quartz; Section 10 shows a potential application to the algorithm by studying the softening and hardening behavior of a designed material; Section 11 gives a performance comparison implement algorithm for isotropic strain-based compression. Section 12 concludes the paper and future recommendations are made.

2. Discrete element method

DEM is a force-displacement Lagrangian approach to solving Newton's equations of motion for an assembly of discrete particles. These particles are governed by force contact law which determines their interaction forces. The simplest law in DEM is based on a Hertzian spring model and called the Frictional Law (FL) [1]. The FL is used as an initial approximation in this study and modified to include the effects of the particle shape.

In FL, the resultant force vector $\overrightarrow{\mathbf{F}}^{t+\Delta t}$ of particle $\boldsymbol{A}$ of a pairwise contact with particle $\boldsymbol{B}$ is decomposed into the sum of the normal force $\overrightarrow{\mathbf{F}}_{\mathbf{N}}$ and the tangential (or shear) force $\overrightarrow{\mathbf{F}}_{\mathbf{T}}$:

$$\overrightarrow{\mathbf{F}}^{t+\Delta t} = \overrightarrow{\mathbf{F}}_{N}^{\,t+\Delta t} + \overrightarrow{\mathbf{F}}_{T}^{\,t+\Delta t} \tag{1a}$$

$$\overrightarrow{\mathbf{F}}_{N}^{\,t+\Delta t} = [\![K_N]\!]\lambda_P \overrightarrow{\boldsymbol{n}} \tag{1b}$$

$$\Delta\overrightarrow{\mathbf{F}}_{T}^{\,t+\Delta} = -[\![K_T]\!]\Delta\overrightarrow{\boldsymbol{\lambda}}_{T} \tag{1c}$$

where λ_P is the penetration depth, $\boldsymbol{\Delta}\overrightarrow{\boldsymbol{\lambda}}_{\boldsymbol{T}}$ is the incremental tangential displacement vector and $\overrightarrow{\boldsymbol{n}}$ is the normal direction of contact. The viscous dashpot term as in [19] is neglected since this study is restricted to systems in the quasi-static regime for dry sand quartz with very small relative velocities.

The normal stiffness $[\![K_N]\!]$ and tangential stiffness $[\![K_T]\!]$ are calculated using the radius of the two particles r_A and r_B:

$$[\![K_N]\!] = E\frac{2r_A r_B}{r_A + r_B} \tag{2a}$$

$$[\![K_T]\!] = \nu E\frac{2r_A r_B}{r_A + r_B} \tag{2b}$$

The tangential force $F_T^{t+\Delta t}$ is found by summing the history of the tangential increments $\Delta\overrightarrow{\mathbf{F}}_{T}^{\,t+\Delta}$ at the previous time steps of the same contact pair. A frictional Mohr-Coulomb criterion is applied which

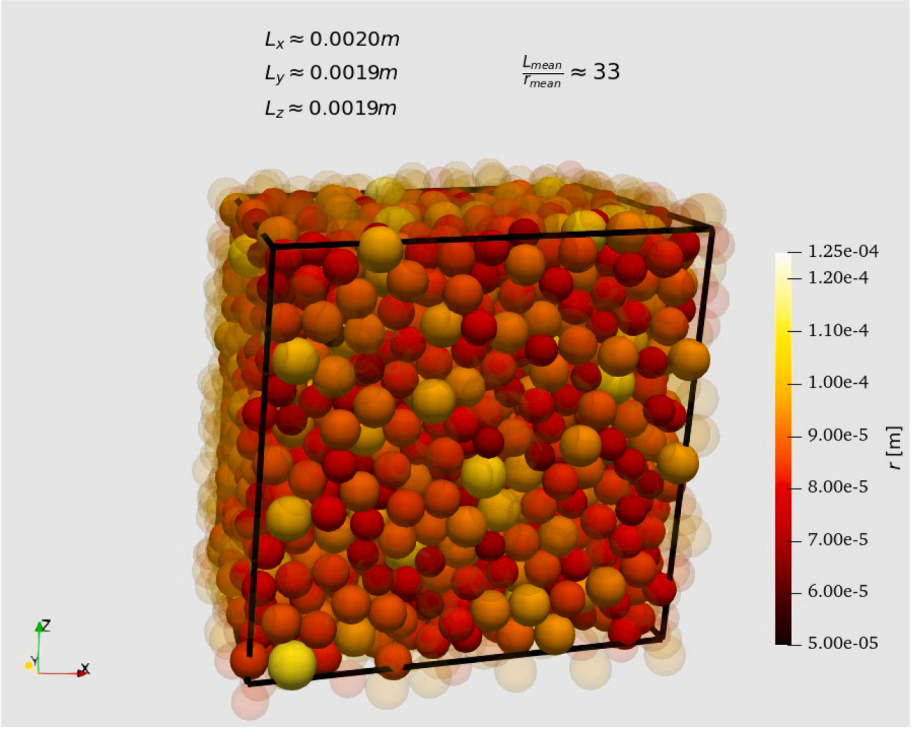


Fig. 1. RVE of 2000 particles with uniformly distributed radii between $5 \times 10^{-5}m$ and $1.25 \times 10^{-4}m$. The cell lengths are shown as L_x, L_y and L_z, and the mean cell length L_{mean} is about 33 times larger than the mean particle radius r_{mean}.

enables the particles to slip. That is if the tangential force exceeds the normal force times the friction angle $|F_T^{t+\Delta t}| > |F_N^{t+\Delta t}| \tan\varphi$, then the expression is:

$$\overrightarrow{\mathbf{F}}_T^{\,t+\Delta t} = tan\varphi \frac{\left|F_N^{t+\Delta t}\right|}{\left|F_T^{t+\Delta t}\right|} \overrightarrow{\mathbf{F}}_T^{\,t+\Delta t} \tag{3}$$

The FL introduces four material parameters: Young's moduli E which scales the magnitude of the normal and shear force; Poisson's ratio ν which scales the magnitude of the shear force; Intergranular frictional angle $\mu = \tan\varphi$ determines the strength limit of the tangential force. The torques of the particles are calculated by:

$$\overrightarrow{\boldsymbol{T}}^{t+\Delta t} = \lambda_T(-\overrightarrow{\boldsymbol{n}}) \times \overrightarrow{\mathbf{F}}^{t+\Delta t} \tag{4}$$

where $\lambda_T = (r_A - \lambda_P)$ is the mathematical complement of the radii overlap.

3. Moment Rotation Law

Particles may be represented by different geometries [20] which can influence the bulk properties such as shear strength and angle of repose [21]. However, resolving contact for particles with detailed geometry requires complex and computationally demanding algorithms [22,23]. This study uses spherical particles and artificially introduces a grain roughness by using the Moment Rotation Law (MRL).

MRL modifies the FL by introducing rolling $\overrightarrow{\mathbf{M}}_{roll}$ and twisting $\overrightarrow{\mathbf{M}}_{twist}$ moments. Artificially adding these moments was shown to produce desirable results under quasi-static triaxial compression [24,18,25,26,13].

$$\overrightarrow{\boldsymbol{M}}^{t+\Delta t} = \overrightarrow{\boldsymbol{T}}^{t+\Delta t} + \overrightarrow{\mathbf{M}}_{twist}^{\,t+\Delta t} + \overrightarrow{\mathbf{M}}_{roll}^{\,t+\Delta t} \tag{5}$$

The rolling and twisting moments are solved similarly to each other (denoted by the subscript). The incremental moments are found by:

$$\overrightarrow{\Delta \mathbf{M}_I}^{\,t+\Delta t} = -[\![K_I]\!]\,\Delta\overrightarrow{\lambda}_I \tag{6}$$

where $\boldsymbol{\Delta}\overrightarrow{\boldsymbol{\lambda}}_I$ is the incremental moment rotation vector. The total moment $\overrightarrow{\mathbf{M}}_I^{\,t+\Delta t}$ is obtained by the summation of the moment increments from the particle history from $\overrightarrow{\Delta \mathbf{M}}_I^{\,i+1}$. Then a stiffness is found by:

$$[\![K_I]\!] = \eta_I min(r_A, r_B)^2\,[\![K_T]\!] \tag{7}$$

where η_I is the stiffness coefficients.

After the moment is incremented, a threshold is applied if $\left|\overrightarrow{\mathbf{M}}_I^{\,t+\Delta t}\right| \rangle M_I^{eta}$ holds. The modified moment is given by:

$$\overrightarrow{\mathbf{M}}_I^{\,t+\Delta t} = M_I^{eta} \frac{\overrightarrow{\boldsymbol{\lambda}}_I}{\left|\overrightarrow{\boldsymbol{\lambda}}_I\right|} \tag{8a}$$

where

$$M_I^{eta} = \alpha_I \left|\overrightarrow{\mathbf{F}}_N^{\,t+\Delta t}\right| \tag{8b}$$

and α_I is a strength parameter. Eqs. (6) to (8) should be repeated for both the bending and twisting moments. MRL introduces 4 additional parameters: twisting and rolling stiffness coefficient η_{twist} and η_{roll}, and the twisting and rolling strength α_{twist} and α_{roll}.

4. Integration scheme

The translational acceleration $\overrightarrow{\boldsymbol{a}}^{t+\frac{\Delta t}{2}}$ and angular acceleration $\overrightarrow{\boldsymbol{\omega}}^{t+\frac{\Delta t}{2}}$ of a particle is found using the particle mass m_A and Newton's second law with the resultant force and moment. Then, a first-order Euler integration scheme is used to find the velocity $\overrightarrow{\boldsymbol{u}}^{t+\Delta t}$ and position $\overrightarrow{\boldsymbol{s}}^{t+\Delta t}$:

$$\overrightarrow{\boldsymbol{a}}^{t+\frac{\Delta t}{2}} = \frac{\overrightarrow{\mathbf{F}}^{t+\Delta t}}{m_A} \tag{9a}$$

$$a_{damp} = \vec{a}^{t+\frac{\Delta t}{2}}\left(\vec{u}^{t} + \frac{\Delta t}{2}\vec{a}^{t+\frac{\Delta t}{2}}\right) \tag{9b}$$

$$\vec{u}^{t+\Delta t} = \vec{u}^{t} + \vec{a}^{t+\frac{\Delta t}{2}}\left[1 - C_{damp} sign\left(\vec{a}^{t+\frac{\Delta t}{2}}\right)\right]\Delta t \tag{9c}$$

$$\vec{s}^{t+\Delta t} = \vec{s}^{t} + \vec{u}^{t+\Delta t} \tag{9d}$$

where C_{damp} is the normal damping coefficient. A similar integration scheme is performed for the moment for the particle. The time step Δt is set as half the critical time step [27]:

$$\Delta t_{cr} = r_{min}\sqrt{\frac{E}{\rho_{min}}} \tag{10}$$

where r_{min} and ρ_{min} are the minimum radius and density of the particles, respectively.

5. Upscaling and RVE

A large system of fine powders or soils often requires enormous amounts of computational resources to simulate to the exact scale. Instead, the material response can be found by studying a subscale of the system which is characterized by an RVE. The RVE solves the constitutive laws of the microscopic (particle) interactions to statistically find the macroscopic variables employing volume averaging [6]. The size of the RVE must be much smaller than the macroscopic system $L_{RVE} \ll L_{MAC}$, but large enough to be homogenized and eliminate the fluctuations (of the averaged field variables) $L_{SUB} \ll L_{RVE}$ which is due to microscopic effects [28]. An example of a homogenized RVE is shown in Fig. 1.

For a homogenized RVE, the effective stress is found by using the Cauchy stress [29]:

$$\sigma_{i,j} = \frac{1}{V_{RVE}}\sum_{N_c}\vec{d}^{c} \otimes \vec{f}^{c} \tag{11}$$

where V_{RVE} is the volume of the RVE. The tensor product $\otimes$ is calculated for the contact force $\vec{f}^{c}$ and the branch vectors $\vec{d}^{c}$ over all contact pairs N_c.

Using the effective stress, the mean principal stress p and deviatoric stress q for a 3D system is given by:

$$p = \frac{1}{3}\left(\sigma_{1,1} + \sigma_{2,2} + \sigma_{3,3}\right) \tag{12a}$$

$$q = \frac{1}{\sqrt{2}}\sqrt{\left(\sigma_{1,1} - \sigma_{2,2}\right)^2 + \left(\sigma_{1,1} - \sigma_{3,3}\right)^2 + \left(\sigma_{2,2} - \sigma_{3,3}\right)^2} \tag{12b}$$

The strain tensor $\varepsilon_{i,j}$ is the relative change in position of the RVE under deformation. The volumetric strain ε_V and deviatoric strain ε_q are given by:

$$\varepsilon_V = \varepsilon_{1,1} + \varepsilon_{2,2} + \varepsilon_{3,3} \tag{13a}$$

$$\varepsilon_q = \frac{1}{\sqrt{2}}\sqrt{\left(\varepsilon_{1,1} - \varepsilon_{2,2}\right)^2 + \left(\varepsilon_{1,1} - \varepsilon_{3,3}\right)^2 + \left(\varepsilon_{2,2} - \varepsilon_{3,3}\right)^2} \tag{13b}$$

The initial configuration of the RVE should approximately match the particle size distribution, porosity, and density of contacts of the experiment [28]. It is, however, not necessary to match the bulk density since the time step and the maximum deformation rate (defined in Section 6) are scaled relatively. Particle mass density may also be artificially increased to decrease the magnitude of the time step and lessen computational time [30,31]. However, this study chooses to match the porosity of the system but assumes a similar bulk density to the experiment. The effects of gravity within the packing are also ignored for mesoscale simulations. The simulation parameters used are defined in Appendix A.

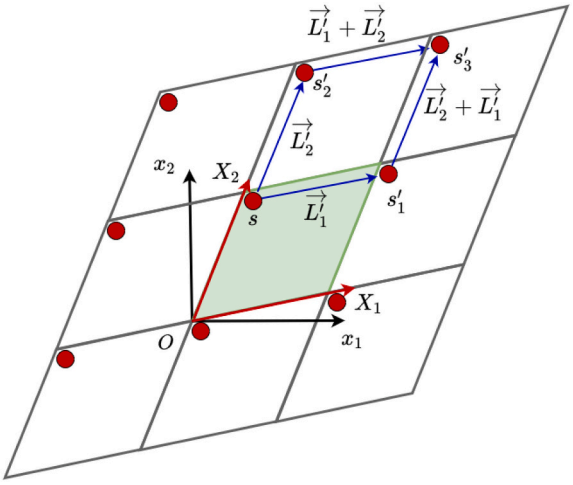


Fig. 2. 2D illustration of a parallel piped unit cell (green) in the global and local coordinates. The periodic images contain identical particles which are incremented by the cell length. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6. Deformable boundaries and periodicity

The parallel piped unit cell is used to incorporate different servo-controlled deformations and periodicity into an RVE. Fig. 2 shows a 2D illustration of this unit cell as part of an infinite system of images. As the centroid of a particle traverses an edge and leaves the unit cell, the position of the particle is translated across the opposite edge, conserving its direction of motion and energy. Particles in between the unit cell and its images lead to boundary particles. The number of boundary particles depends on how many edges are overlapped. The term real particle is used to distinguish between a particle in the main unit cell and its boundary particles in other images.

The periodic images contain identical particles which are incremented by the cell length.

For the parallel piped unit cell, two coordinate systems are used, namely, the global (cartesian) coordinate system and the local (cell) coordinate system. These coordinate systems are interpreted as matrices with the column entries as the basis of the system. The global coordinate system is denoted as $\boldsymbol{x_i}$ and the local coordinate system is denoted as $\boldsymbol{X_j}$. The coordinate transformation between these systems are:

$$\boldsymbol{x_i} = \boldsymbol{H_{ij}}\boldsymbol{X_j} \tag{14a}$$

$$\boldsymbol{X_j} = \boldsymbol{H_{jk}^{-1}}\boldsymbol{x_k} \tag{14b}$$

where $\boldsymbol{H_{ij}}$ is the transformation with its columns as basis vectors of the cell, and $\boldsymbol{H_{jk}^{-1}}$ is the inverse transformation.

The images of the particles in the local coordinates can be obtained by periodically shifting their positions $s_i \rightarrow s'_i$ by units of local cell length L_i', as shown in Fig. 2.

$$s'_i = s_i \pm L_i' \tag{15}$$

The motion due to the homogeneous deformation of the boundaries is considered by differentiating Eq. (16) and decomposing it to the affine mean-field velocity $\vec{u}_{mf}'$ and fluctuating velocity $\vec{u}_{fl}'$.

$$\dot{\boldsymbol{x}}_i = \dot{\boldsymbol{H}}_{ij}\boldsymbol{X}_j + \boldsymbol{H}_{ij}\dot{\boldsymbol{X}}_j = \vec{u}_{mf}' + \vec{u}_{fl}' \tag{16a}$$

$$\vec{u}_{mf}' = \vec{u}_{mf} \pm \dot{\boldsymbol{H}}_{ij}L_i' \tag{16b}$$

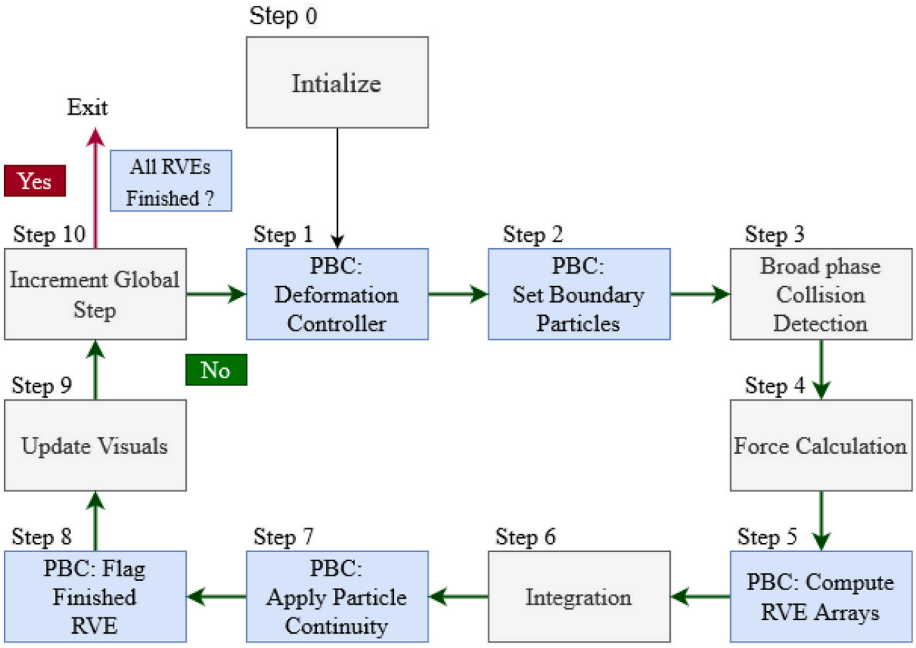


Fig. 3. Flow chart of the GPU DEM simulation loop. The modified GPU DEM processes are shown in grey and the extended PBC sub-class processes are shown in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\vec{\boldsymbol{u}}_{fl}{}' = \vec{\boldsymbol{u}}_{fl} \tag{16c}$$

The affine mean-field velocity is attributed to the macroscopic homogeneous deformation of the RVE cell, and the fluctuating velocity or velocity is driven by a particle force and is not affine.

The velocity due to the homogeneous deformation of the RVE is found in the global coordinate system by:

$$u_{hi} = U_{ij}x_j \tag{17a}$$

$$U_{ij} = \dot{H}_{ik}H_{kj}^{-1} \tag{17b}$$

where U_{ij} the velocity gradient (tensor) which is used to represent the gradient of homogeneous deformation. The induced acceleration is found by differentiating Eqs. (17)

$$\dot{u}_{hi} = \dot{U}_{ik}x_k + U_{ik}\dot{x}_k \tag{18}$$

The boundary-induced velocity and acceleration are considered in the motion integration scheme. The velocity gradient or strain rate allows for scaling and rotation and is applied incrementally by the servo-controller to deform the cell.

$$C_{ij}^{t+\Delta t} = H_{ij}^{t}C_{ij}^{t} \tag{19a}$$

$$H_{ij}^{t} = \delta_{i,j} + U_{ij}^{t}\Delta t \tag{19b}$$

$$C_{ij}^{0} = \begin{bmatrix} L_1 & 0 & 0 \\ 0 & L_2 & 0 \\ 0 & 0 & L_3 \end{bmatrix} \tag{19c}$$

where $\delta_{i,j}$ is the identity matrix, $C_{i,j}$ is the cell matrix with wall lengths L_1, L_2 and L_3.

The shear rate $\dot{\gamma}$ is defined as the second invariant of the velocity gradient (strain rate tensor) in Eq. (19). From this, the following a rheological relation is given [32]:

$$\dot{\gamma} = \frac{I'}{2r_{mean}\sqrt{\frac{\sigma_c}{\rho_{mean}}}} \tag{20}$$

where ρ_{mean} and r_{mean} are the mean radii and densities of the particles, respectively. σ_c is the confining pressure. The dimensionless inertia number I' corresponds to the flow regime which is set to 10^{-3} for the quasi-static regime in this study.

This study applies two constraints to correctly simulate a quasi-static response in DEM. The first constraint is that the time step should be below the critical time step in Eq. (10) to have numerical stability. The second constraint is that the maximum incremental strain for the servo-controller should be sufficiently small.

$$d\varepsilon_{max} = \Delta t \bullet \dot{\gamma} \tag{21}$$

As $\dot{\gamma}$ and Δt becomes smaller, more simulation iterations are necessary to reach a final simulation time or servo-controlled goal (strain or stress). Therefore, to ensure reasonable computational time, $\dot{\gamma}$ and Δt are automatically calculated for the RVE to the mean radii, mean particle density, and Young's modulus. In the case of multi-RVE simulations with different Young's moduli or particle densities, the time steps may vary. The servo-controller also checks if the potential energy (calculated from particles' stiffness) is greater than their kinetic energy (calculated from the particles' mass and velocity) before a strain increment is applied to the boundaries. The DEM solver uses the global coordinate system but temporarily deforms the RVE to its local coordinate system to find the respective boundary particles.

7. GPU algorithm

7.1. GPU architecture

The implementation is done in C++ and the NVIDIA CUDA toolkit [33] for General-Purpose Graphics Processing Unit (GPGPU) programming. Depending on the GPU device used, the implementation in the code may differ, but the procedure and principles would remain the same.

The CUDA programming model is separated into host-side and device-side (GPU) processes which execute on different memory and processing spaces. To efficiently utilize the GPU the following principles should be followed [34,35,36]: (1) global memory access (read and write) should be reduced and performed in a coalesced (grouped) manner; (2) the total amount memory requirements should be reduced;

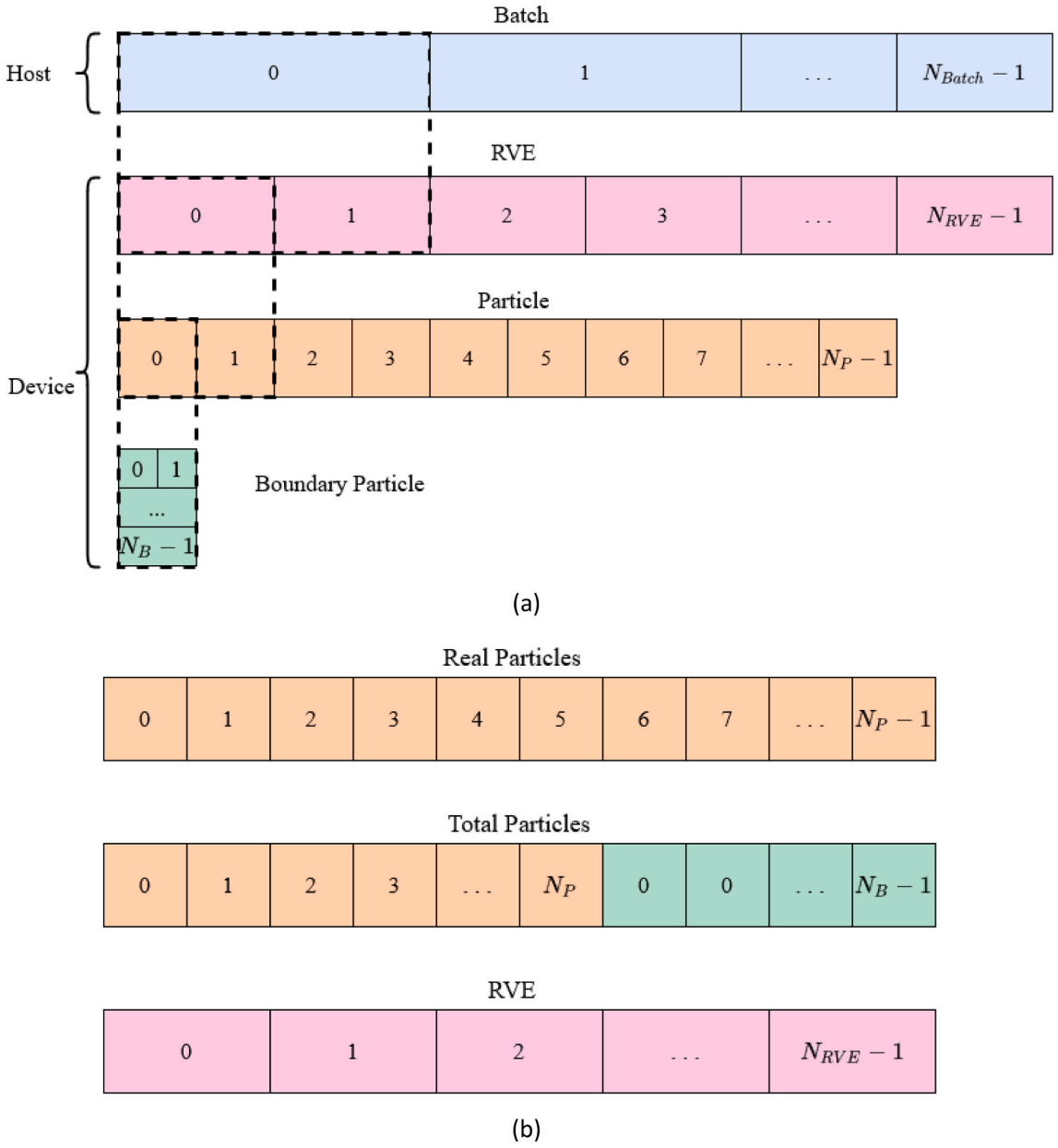


Fig. 4. Schematic diagrams of the (a) hierarchical memory groups and (b) collective thread groups. The numbers indicate the indices for N_{Batch} number of batches, N_P number of particles, N_B number of boundary particles and N_{RVE} number of RVEs. The dashed line in (a) indicates the class membership.

(3) thread occupancy should be increased by using enough threads to keep the processor busy; (4) bank conflicts (overlapping memory address) should be minimized; (6) branch divergence or forking logical statements must be reduced.

The present GPU algorithm for solving multiple non-interacting RVEs concurrently attempts to adhere to these optimization principles. Some procedures utilize existing optimized algorithms such as sorting, inclusive scans, and reductions made available by libraries CUB (D. [37]) and thrust [38].

7.2. Concurrent RVE simulation scheme

The developed framework follows a modified simulation loop compared to traditional DEM codes [39,2,36]. The modified simulation loop includes (1) servo-controlled deformable boundaries and periodic boundary conditions for the RVE; (2) particle and boundary particle interaction at the RVE walls; (3) many non-interactable RVEs are solved concurrently. The rest of this section presents the overarching simulation loop while the next sections explain the detailed implementation.

The simulation loop of the framework is shown in Fig. 3. The steps shown in grey are adapted from the general DEM simulation loop, such as the modified collision detection and integration. While the steps shown in blue are newly introduced features implemented into a module called PBC. The PBC module introduces some additional steps: the servo-controller applies a deformation to each RVE (Step 1); boundary particles are assigned (Step 2); volume averaging for each RVE is performed (Step 5); periodicity is applied to the particles crossing the boundary (Step 7); RVEs are flagged as finished after they reach user-specified stress or strain goal, and the total number of thread is reduced to improve overall efficiency (Step 8).

7.3. Coalescence of memory and threads

The memory structure and data access patterns are aligned with the

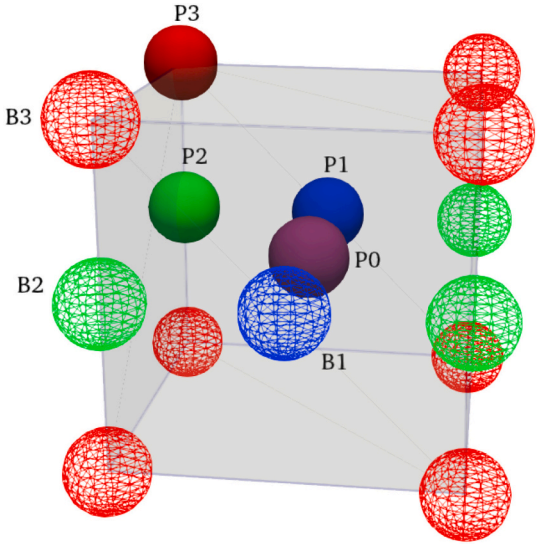


Fig. 5. Four possibilities of particle-wall intersections of a periodic cell, and the respective boundary particles.

GPU optimization principles mentioned in Section 7.1. This is achieved through hierarchical (top-down) memory groups and collective thread groups.

The memory groups illustrated in Fig. 4 (a) may be either located on the host-side (CPU) or device-side (GPU). The class memberships are indicated through a dashed line. For instance, a batch has a collection of member RVEs, an RVE has a collection of member particles, and so on. The batch memory group is processed serially using asynchronous host to device and device to host copies using CUDA streams. Batch processing is used when the memory limit on the GPU card is reached. RVEs are solved concurrently in the same simulation loop. This is achieved by defining an RVE index (Section 7.4). Particles that belong to the same RVE have the same RVE index and may not interact with particles that have different RVE indices. Each member in a group has their state variables. For instance, particles contain information on their positions, velocities, angular velocities, forces, moments, radii, etc.

Fig. 4 (b). shows the layout of the thread groups which are dispatched to read and write to the respective memory indices. In general GPU DEM codes [2,3,4,5], the memory layout usually follows a set of bytes per particle, and a thread per particle is assigned as the global thread group. Similarly, this algorithm allows highly efficient memory and data access patterns. Boundary particles can retrieve information about their parents (real particles) such as radii or contact history.

Volume averaging is achieved by first summing all forces and contacts of the particle in the force calculation kernel. Then, a segmented reduce function using CUB (D. N.-L. [40]) is utilized (based on the RVE index) to calculate the RVE state variables. The hierarchical memory layout and collective thread groups remove the restriction of using shared memory to compute the RVE state variables such as [15].

7.4. Handling boundary particles

If a real particle intersects two or more walls within a parallel piped unit cell then boundary particles are created at the adjacent walls as shown in Fig. 5. There are four possible configurations: particle (P0) has no boundary particles; particle (P1) intercepts one wall and has one boundary particle (B1) at the adjacent cell wall; particle (P2) intercepts two walls and has three boundary particles (B2) at the three adjacent walls; particle (P3) intercepts three walls and has seven boundary particles (B3) at the adjacent walls.

A bitmask is used to store the possible combinations of boundary particles. An example of a bitmask in Fig. 5 is (P0) 000000 (no boundary particles), (P1) 000100, (P2) 010100, and (P3) 010101. The first three

```
1   procedure SetBoundaryParticles(DryRun) on Device (GPU)
2       Run parallel for a thread per real particle
3         ‖C_ij‖_2 , L_1', L_2', L_3' ← L2-norm(C_ij) compute norm of cell box and global cell lengths
4         H_ij ← C_ij / ‖C_ij‖_2  Normalize cell box
5         s⃗' ← H_jk^-1 s⃗  Map particle global coordinates into local cell coordinates
6         if (DryRun is True) then
7             for every i, L_i', s_i', C_ii do
8                 if C_ii < r then
9                     bitmask ← (bitmask ≪ i)&1)
10                    n_B +=1
11                end if
12                if C_ii + r > L_i' then
13                    bitmask ← (bitmask ≪ i + 3)&1)
14                    n_B +=1
15                end if
16            end for
17        else
18            for every k, mask in bitmask do (get boundary particles mask with lookup table)
19                B_ref ← S_ref  set pointer to real particles
20            end for
21        end if
22  end procedure
23
24  procedure MallocBoundaryParticles( on Host (CPU)
25      SetBoundaryParticles(DryRun=True)
26      N_B, B_ref ← PrefixSum(n_B)
27      Dynamically Allocate Memory(B_ref ) of size N_B
28      SetBoundaryParticles(DryRun=False)
29  end Procedure
```

```
1   procedure TransformBoundaryParticle on Device (GPU)
2       Run parallel for a thread per total particles (real and boundary)
3       for every (thread_index >= N_P) do
4           L_i', r, H_jk^-1, s_i' ← B_ref  get real particle state and RVE values
5           for every i, L_i', do
6               w_i' ←LookupTable(mask)
7               s_B' ←w_i' L_i' (translate boundary particle)
8               u_B' ← w_i' U_ii (apply velocity gradient to boundary particle velocity)
9           end for
10          s_B ← H_jk s_B'  transform to global coordinates
11      end for
12  end procedure
```

digits are set for intersecting a wall near the origin of the cell. The second three digits are set when overlapping adjacent cells. The bitmask of each particle has a permutation that describes the boundary particle bitmask. For instance, (P2) permutes into the three boundary particles 000000; 010001; 000100. Therefore, after finding the real particle bitmask, the boundary particle bitmask can be calculated.

Algorithm 1 is used to find the boundary particles of a real particle that intersect the RVE walls. The algorithm contains two main procedures. The first procedure is a device-side kernel that is launched for a thread per real particle, and the second is a host-side function used to dynamically allocate memory. The SetBoundaryParticles procedure is called twice (lines 25 and 28). The first call performs a dry run that calculates the real particle bitmasks and counts the number of boundary particles to be created. Particles are temporarily transformed to the deformed coordinates of their respective RVE (lines 2–5) and tested for the intersection of the RVE walls (lines 7–15). The TheMallocBoundaryParticles procedure is responsible for dynamically allocating a reference array for the boundary particles using a prefix sum (line 26). After the number of boundary particles per real particle is found n_B, a boundary particle reference array ($\boldsymbol{B}_{ref}$ of size N_B) is dynamically allocated which points a boundary particle to its real particle (line 27). After the boundary particle reference array has been allocated, each boundary particle is permutated using a lookup table and then assigned a pointer to its real particle (lines 18–20).

Algorithm 1. Bitmask Boundary Particles.

Algorithm 3 is a procedure called to obtain the boundary particle positions and velocities. As shown in line 4, the values to a respective boundary particle are obtained by referring to its set pointer $\boldsymbol{B}_{ref}$ (line 4). The vector for the wall of the boundary particle $\vec{w}_i'$ is found using a lookup table and the calculated mask of the particles (line 6). It is memory-efficient to store only the bitmask of the shifted particle images. However, this comes at a performance trade-off since matrix multiplication is necessary to move the particles' coordinates from local to global coordinates. This study instead stores the boundary particle positions and velocities while referring to the real particle for other variables such as the particle radii.

The force calculation of particles must also consider contact between real and boundary particles. It is common for the contact between particles to be performed in a thread per particle kernel [39,2]. In this case, the contact between particles is performed also performed similarly but a real particle also sequentially loops over its boundary particles. The force that is computed for a boundary particle should be mapped back to the real particle following Newton's third law. The allowed binary contacts are real-real and real-boundary. The boundary-boundary contacts are avoided by excluding contacts during the collision detection.

Algorithm 2. Get boundary particles.

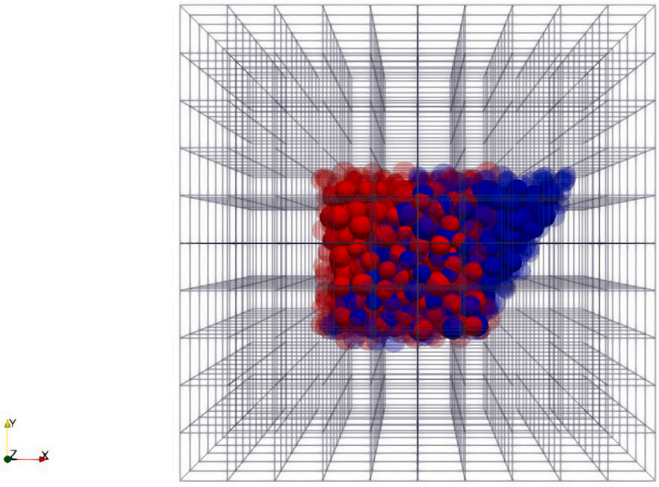


Fig. 6. Two RVEs (red and blue particles) share the same global coordinates and Uniform Grid. Particle contacts of the same RVE index are allowed and those of different indices are ignored. The domain and grid cell sizes are enlarged for visual purposes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7.5. Modified collision detection

As discussed in Section 7.2, the time evolution of all RVEs is solved within the same simulation loop to improve efficiency. To solve many non-interacting systems (simulations), an RVE index is used which can be interpreted to add a non-spatial dimension. Particles that have the same RVE index are allowed to interact, while particles of different RVE indices do not interact. This is achieved by modifying existing collision detection algorithms and by ensuring all particles are within the same reference frame (global coordinates). This study modifies the GPU-based Uniform Grid [41,2] and BVH [42,23] collision detection algorithms. The Uniform Grid is more efficient for monodisperse simulations and compact simulation domains while the BVH is more efficient for highly polydisperse simulations and sparse domains [23].

The conventional Uniform Grid partitions the spatial domain into a 3D grid of $M = M_x M_y M_z$ number of grid cells. The particles are binned into the grid cells through a spatial hashing scheme. The neighboring cells of the selected particles are inspected for their nearest neighbors. The Uniform Grid resolution is determined by the cell size and a cell can contain more than one particle. However, computational performance will be degraded if there are too many particles within a cell.

The modified Uniform Grid (Fig. 6) partitions the particles and their boundary particles into the grid. Then a modified hash scheme is used to partition the particles based on their spatial coordinates and RVE index:

$$Hash = w + bin_x N_{RVE} + bin_y N_{RVE} M_x + bin_z N_{RVE} M_x M_y \tag{22}$$

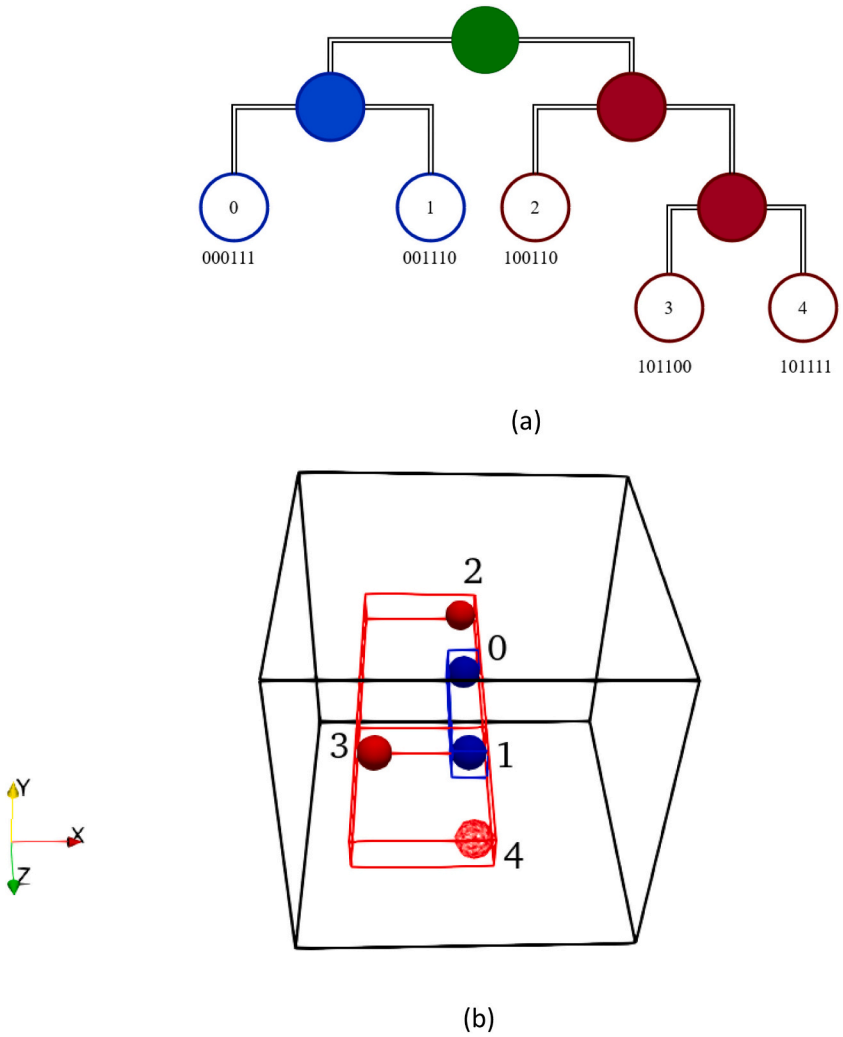


Fig. 7. The BVH for two non-interacting RVEs with two particles each. In (a), the constructed binary tree on the RVE index (red and blue) and the root node (green). The corresponding scene is shown in (b) in the same reference frame. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where w is the RVE index; N_{RVE} is the number of RVEs; M_x, M_y, M_z are the number of grid cells; bin_x, bin_y, bin_z is the number of bins in the *ith* dimension. The entire domain is generally defined to ensure all possible deformations of the RVE are contained within the grid. A possible downside to this method is that the number of grid cells can become quite large as N_{RVE} increases which may lead to increased memory usage.

The conventional BVH partitions the spatial domain into a 3D Morton key. The particles are assigned indices which are then sorted with the Morton keys such that their localities are preserved. The tree is constructed using a binary search on the keys and the leading bits of the Morton key are used to construct the hierarchy. The internal nodes (numbered one less than the number of particles) are assigned two children nodes which can either be another internal node or a leaf node. In this case, the particles are assigned as the leaf nodes. An upwards agglomeration is performed, from the particles to the root node to assign the tree axis-aligned bounding boxes (AABB). Broad phase collision detection is performed by traversing the tree top down.

The modification to the BVH is made by adding a bit value at the start of the Morton hashing scheme, as shown in Fig. 7. The first level of internal nodes is partitioned by the RVE index and then by the spatial hash of the particles. The traversal is initiated at the top-most internal node of the RVE index instead of the root node.

8. RVE homogenization

The RVE must have enough contacting particles at about even orientations to transition from a mesoscopic constitutive model to a macroscopic one. Multiple factors may influence homogenization such as particle size distribution, porosity, particle geometry, and the simulation dimensions (1D, 2D, or 3D). As some of these parameters are varied, the number of particles must be increased and therefore require more computational resources. For instance, the number of possible contact orientations in 1D simulations is significantly less than in 2D simulations. Therefore, 2D simulations require more particles to homogenize an RVE than 1D simulations.

To determine the number of particles necessary to homogenize the RVE, this study follows a similar approach to others [18,10,7,28,15] along with an additional statistical step presented bellow.

RVEs with different numbers of particles are generated. They are then slightly pre-consolidated to 50 Pa with a high Youngs modulus and low friction angle to ensure a jammed low-stress packing. Samples are consolidated to 100 kPa after pre-consolidation. The parameters used are shown in (Appendix A, Table 1).

An example of three RVEs with an increasing number of particles (400, 2000, 2800) is shown in Fig. 8(a). The rose diagrams show that the contact orientations of the RVE with 400 particles are non-uniform. On the contrary, the rose diagrams for 2000 and 2800 are more uniform.

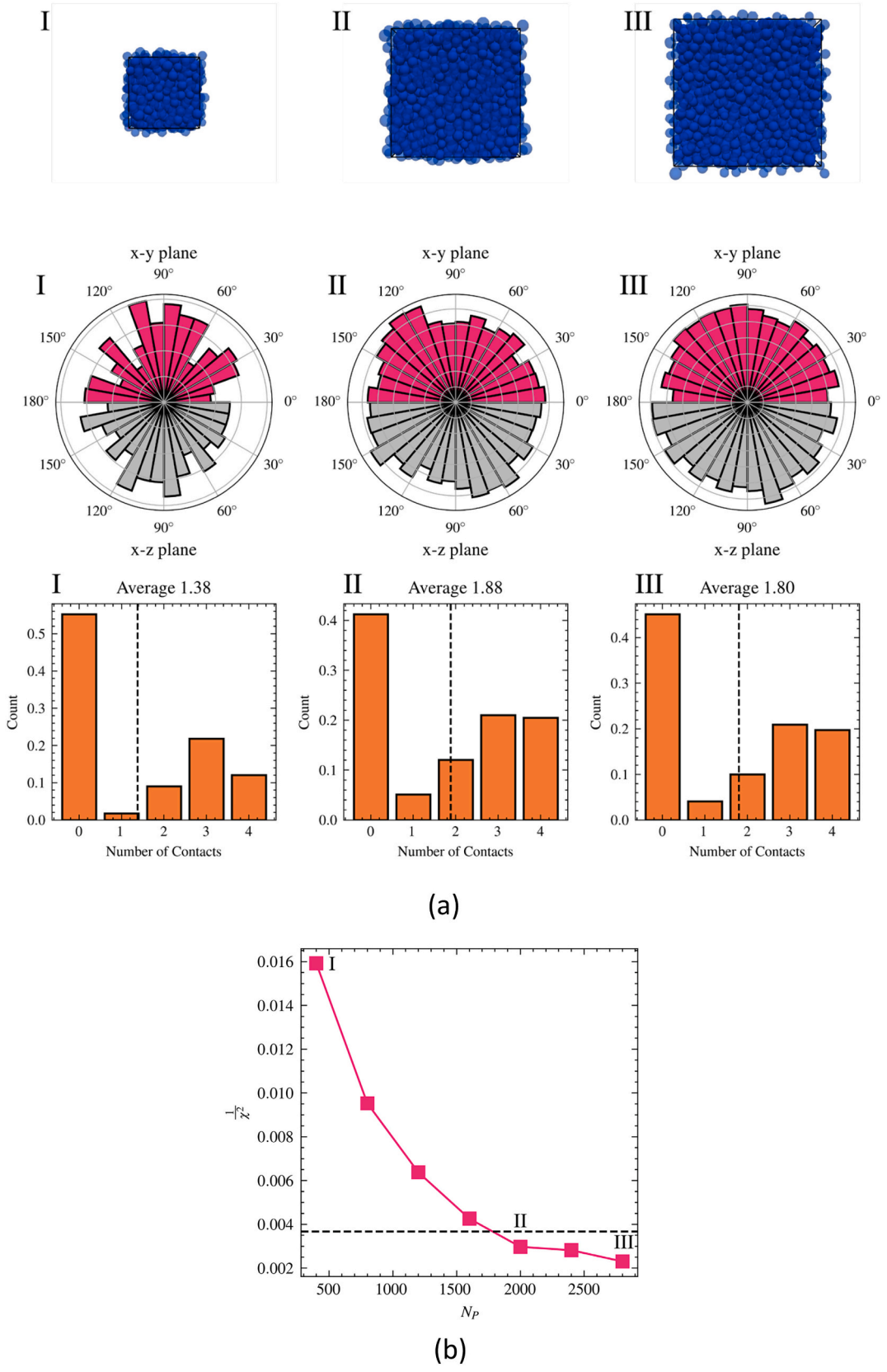


Fig. 8. The number of particles necessary for an RVE to be homogenized. Three example RVEs are shown as I, II, and II. In (a) the spatial configuration, contact orientations, and histogram of contacts are shown. The χ^2 similarity test (b) is shown for the contact orientations, along with a cut-off range (dashed line) were $1/\chi^2$ plateau

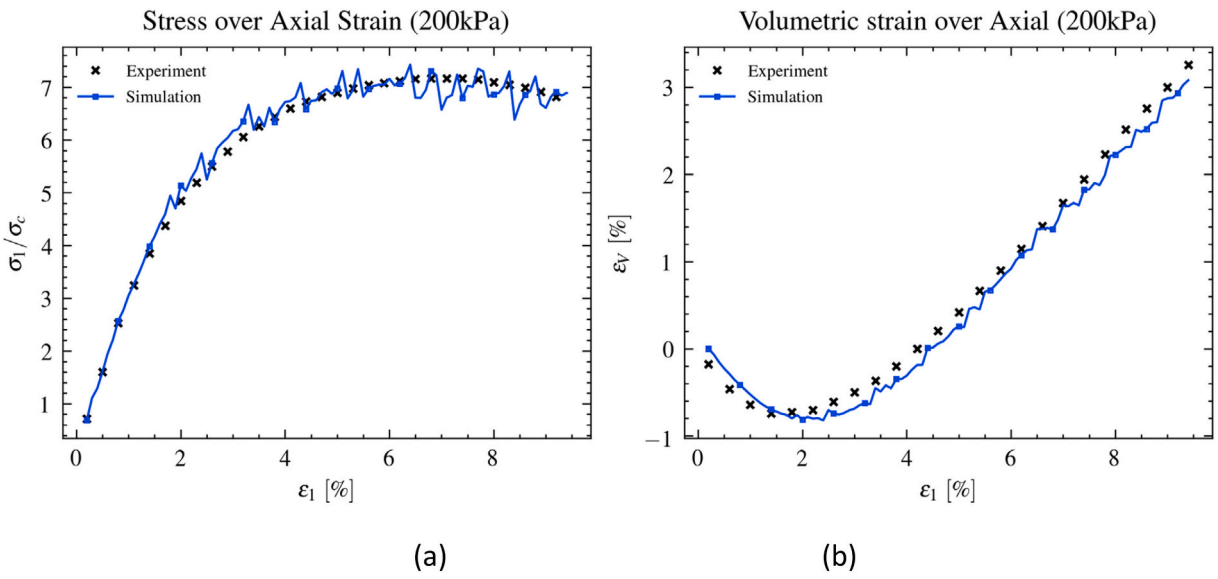


Fig. 9. The simulated and experimental drained triaxial response for graded dry quartz under 200 kPa: (a)The stress (as a fraction over confining pressure); (b) the volumetric strain.

The histogram and average for the number of contacts per RVE with 400 particles are much different than those of 2000 and 2800.

The additional statistical step involves using the Chi-squared (χ^2) test (Appendix B) on the contact orientations as shown in Fig. 8(b). The contact orientations are binned into a frequency table and have an equally likely chance to occur in any direction. The χ^2 the test indicates how likely the observed frequencies are to be equal to the expected frequencies which, in this case, is a uniform distribution. Thus, if values of $1/\chi^2$ plateaus then the RVE is more likely to be homogenized.

The RVE must also be small enough to avoid localization and to reduce computational costs. There is a large decrease in $1/\chi^2$ at 400,800 to 1600 number of particles. At 1600 the $1/\chi^2$ plateaus at 1600 to 2000 number of particles. Therefore, a number of particles of 2000 are sufficient to be homogenized and computationally feasible.

9. Triaxial test of dry quartz sand

Dry sand is often used to understand the behavior of geomaterials under various loading conditions. It is common for the triaxial test to be used to simulate the drained triaxial response of dry sand [24,43,44,25,26]. This section shows that our algorithm can reproduce experimental results if the *meso* parameters are calibrated. The detailed procedure of the experiment and calibration process can be found in the companion paper.

The experiment involves a drained triaxial experiment on graded dry quartz sand. The particle size distribution of the sample is graded to be about uniformly distributed between 0.01 mm and 0.025 mm. The experiment is performed for a confining pressure of 200 kPa. Other confining pressures are presented in the companion paper. The following DEM configurations were used: a uniformly distributed particle diameters of 0.01 mm to 0.025 mm; the DEM initial void ratio matches that of the experiment at about $e_0 = 0.65$; the time step is about $1.72 \times 10^{-8} s$ and the particle density of about $2600 kg/m^{-3}$ is used and the bulk density is assumed to match. The material parameters used are shown in (Appendix A, Table 2).

Fig. 9 shows the stress over axial strain (a) and volumetric strain over axial strain (b) of the simulated and experimental drained triaxial test under a confining pressure of 200 kPa. The simulated response matches closely the experimental response.

10. Softening and hardening behavior

In industry, it may be useful to know the shear strength of sand for different meso parameters or meso structures. This section shows a potential application to the presented GPU-based algorithm by modeling the softening and hardening response of the same material. The algorithm may simulate many different material parameter sets or particle configurations to efficiently perform a parametric sweep.

For this study, samples are generated using a sphere packing algorithm [45] which is modified in the companion paper. The samples have the same material parameters and particle size distributions as that in Section 9 (Appendix A, Table 2). The initial void is varied to control the dilatancy of the sample.

The drained triaxial response of 5 RVEs is performed for 100 kPa, and the material response is shown in Fig. 10. The stress over axial strain is shown in (a) and volumetric strain and axial strain is shown in (b). The dense sample shows a stress peak and a softening response afterward. The invariants q-p in Eq. (12) is shown in (c). Since there is no back-pressure the total and effective stress paths are linear with a slope of M = 3, as expected. The specific volume and logarithm of the mean effective stress invariant are shown in (d). The hardening response is shown by a downward trend of the specific volume and a softening is shown by the upward trend of the specific volume. The simulations are performed for 5 triaxial tests but can easily be performed for 400 or more triaxial tests in parallel which are not included for visual purposes. Using many triaxial tests, useful statistics may be extracted from the numerical simulations.

11. Performance comparison

This section shows the significant benefit of the present GPU algorithm to simulate many non-interacting RVEs (in parallel). First, the size of the sequential batches is increased. One batch solves many RVEs in parallel, and each batch is solved sequentially (see Section 7.2). Next, the number of RVEs is increased to study the influence of the sample size on the performance. Lastly, the number of particles is increased to study how the algorithm scales with the number of particles.

The simulations are performed under a constant isotropic strain-based compression for a 3D RVE. The particles are monodisperse with a diameter of 0.015 mm. The time step is about 1.51×10^{-8} s, the strain rate is about 2.86×7 m.s^{-1} and the RVEs are compressed isotropically

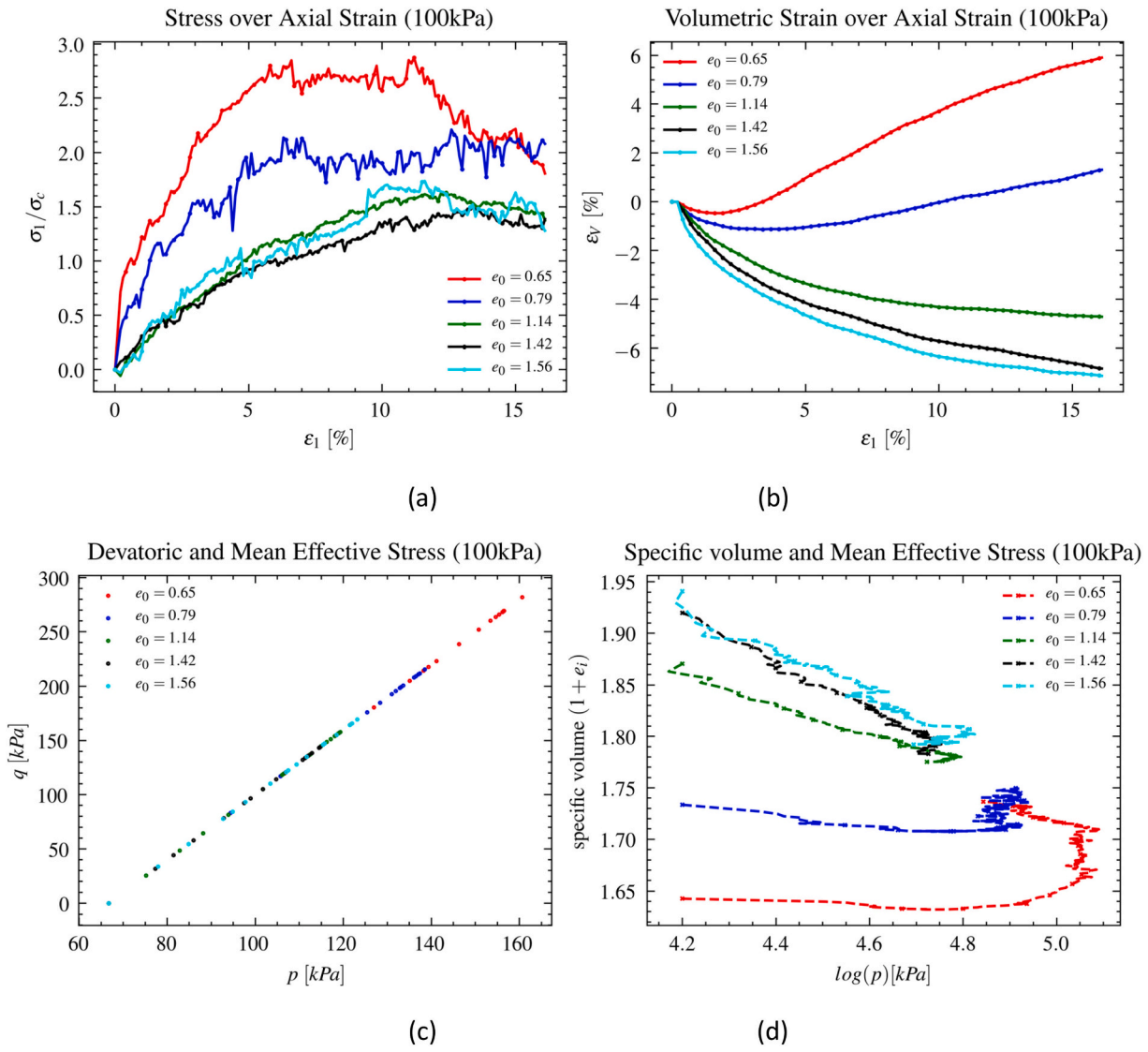


Fig. 10. Simulated drained triaxial response for the same material with different initial void ratios and under different confining pressures. Shown are the (a) stress and axial strain curve, (b) volumetric strain and axial strain curve, (c) deviatoric and mean effective stress with a slope of M = 3 (d) specific volume and log mean effective stress, showing softening.

to a volume fraction of about 12%. The contact law and parameters are the same as those from Section 9 (Appendix A, Table 1). The simulations are performed on a Tesla V100 NVIDIA GPU graphics card.

11.1. Parallel versus sequential simulations

The number of concurrent RVE simulations is increased by changing the number of batches N_{Batch} and keeping the number of particles and RVEs constant ($N_P = 2000$ and $N_{RVE} = 100$). The runtime and memory is plotted in Fig. 11. The fully sequential case is shown at ($N_{RVE}/N_{Batches} = 1$) and the fully parallel case is shown at $N_{RVE}/N_{Batches} = 100$.

A significant improvement is shown for solving even 10 RVEs at once, with a factor of about 5.4 runtime speedup. The fully parallel case shows a speedup with a factor of about 9.8 at the trade-off of about 100 times the memory requirement. Interestingly, the fraction of memory over the number of RVEs per batch increases linearly, while the fraction of runtime decreases in an exponential manner.

11.2. Scaling the number of particles

The number of particles N_P per RVE is increased and the number of RVEs is kept constant for a fully parallel simulation ($N_{RVE} = 1$ and $N_{batches} = 1$). The runtime and memory for the Uniform Grid and BVH are plotted in Fig. 12. Two details are shown, namely, how the performance and memory scale with the number of particles per RVE, and how the collision detection algorithms scale with the number of particles per RVE.

The overall runtime and memory increase linearly with the number of particles. The runtime for the Uniform Grid is better than that of the BVH since the simulation uses monodisperse particles. The memory for the BVH scales similarly to that of the Uniform Grid.

The percentage execution time of a kernel function within the simulation loop (in Section 7.2), with an increase in the number of particles, is shown in Fig. 13. The implemented algorithm (B) Bitmasked Boundary Particles shows an overall decrease in percentage performance. For RVEs with many particles, the presented GPU-based algorithm works quite well. Additionally, the percentage runtime of the following processes also decreases with an increase in the number of particles: (A) servo-controller, (C) Uniform Grid, and (E) Volume averaging. (F) integration and periodicity and (G) Memory copy. The (D) Force calculation algorithm increases significantly with an increase in percentage runtime. The increase in the number of particles leads to

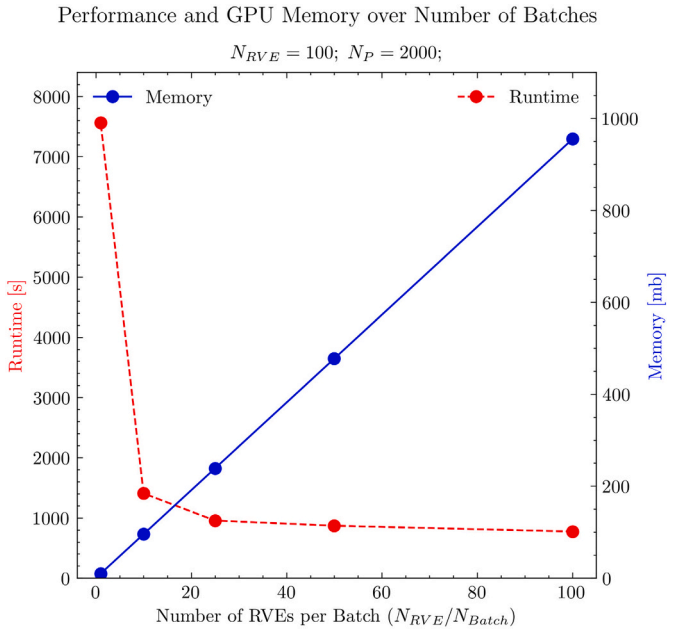


Fig. 11. Performance of the isotropic strain-based compression for 100 RVEs. The number of batches is increased while the number of particles and RVEs are kept constant. A fully serial simulation is shown on the left ($N_{RVE}/N_{Batches} = 1$) and fully parallel simulation is shown on the right ($N_{RVE}/N_{Batches} = 100$).

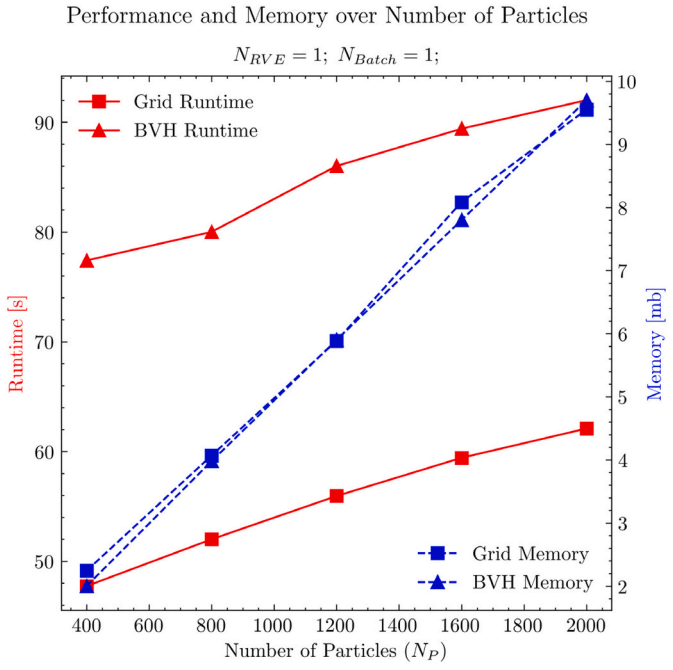


Fig. 12. Performance of the isotropic strain-based compression for one RVE. The number of particles is increased while the number of RVEs and batches is kept constant.

additional boundary particles which require additional computational work. Furthermore, the collision detection may have false-positive (non-overlapping) contacts due to matching a single precision sphere intersection test, while the force calculation matches a double precision intersection test.

11.3. Scaling the number of RVEs

The number of RVEs N_{RVE} is increased and the number of particles is kept constant for a fully parallel simulation ($N_P = 2000$ and $N_{batches} = 1$). The runtime and memory for the Uniform Grid and BVH are plotted in Fig. 14. Two details are shown, namely, how the performance and memory scale with the number of RVEs, and how the collision detection algorithms scale with the number of RVEs.

The overall runtime and memory increase linearly with the number of RVEs. The runtime for the Uniform Grid is again better than that of the BVH since the simulation uses monodisperse particles. The memory for the BVH scales similarly to that of the Uniform Grid.

The percentage execution time of a kernel function within the simulation loop (in Section 7.2), with an increase in the number of RVEs, is shown in Fig. 15. The percentage runtime of the (A) servo-controller decreases with the number of RVEs compared to the other computationally demanding processes. The implemented algorithm (B) Bit-masked Boundary Particles shows a decrease in percentage performance which again shows the presented GPU-based algorithm works quite well for many RVEs. Additionally, the percentage runtime of the (E) Volume averaging, and (F) integration and periodicity also decreases with an increase in the number of RVEs.The collective increase in the number of particles, boundary particles, and RVE indices leads to an increase in the (C) Uniform Grid and the (D) Force calculation algorithm. Interestingly, the percentage kernel execution times of the cases with 100, 200, 300, and 400 RVEs are similar with slight variances.

11.4. Discussion on performance

An RVE must be homogenized to transition from a *meso* to a macro constitutive response. Homogenization occurs if there are enough statistics on the meso scale such that few fluctuations are present when performing volume averaging of the state variables [28]. The distribution of contacts and the contact orientations must be consistent when increasing the number of particles. Some factors that influence this is the particle size polydispersity, void ratio, and simulation dimensions (1D, 2D, or 3D). For 2D simulations, a fewer number of contacts are needed to account for a uniformly distributed orientation. Therefore, a 2D RVE may require fewer particles to be homogenized than a 3D RVE. A GPU-based RVE parallelization algorithm exists for the 2D case [10]. However, such an algorithm utilizes the shared memory (typically of size 64kB or 96kB) between thread blocks (up to 1024 threads) to perform volume averaging. The algorithm is therefore restricted to 1024 parallelized particles per RVE. The benefit of the present GPU algorithm in this paper is that it parallelizes all particles and performs volume averaging by using a segmented reduce (D. [37]). This approach may be more beneficial for RVEs with a large number of particles.

The speed improvement of parallelizing 100 RVEs under isotropic strain-based compression is about 9.8 times faster than the sequential case. The present GPU algorithm exploits thread occupancy (concurrency of many tasks) and keeps the hardware busy to hide latencies [33]. However, the speedup comes at the cost of about 100 times the GPU memory. Depending on the memory limitation of the GPU card, one must consider a balance between the number of particles, the number of RVEs, and the number of batches (serial simulations).

The performance and memory are found to increase linearly with both the number of particles and the number of RVEs. There is a sharper increase in memory for an increase in the number of particles than for an increase in the number of RVEs. This shows the importance of minimizing the number of particles in an RVE. That is, it is advantageous to homogenize an RVEs as in Section 8 to optimize the performance.

It is shown that BVH scales overall worse in terms of performance compared to the Uniform grid. The performance simulations are however limited to monodisperse particles and not highly polydisperse particles which the BVH may perform better. A details study of the BVH performance with particle size polydispersity and geometry is presented

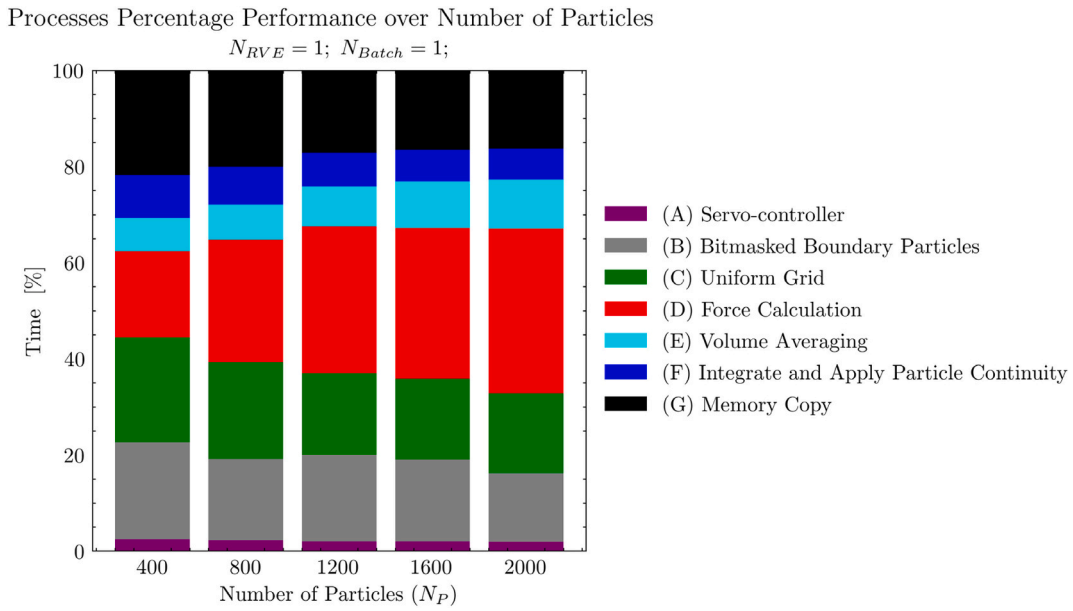


Fig. 13. Percentage runtime of the GPU functions for the isotropic strain-based compression for one RVE. The number of particles is increased while the number of RVEs and batches is kept constant.

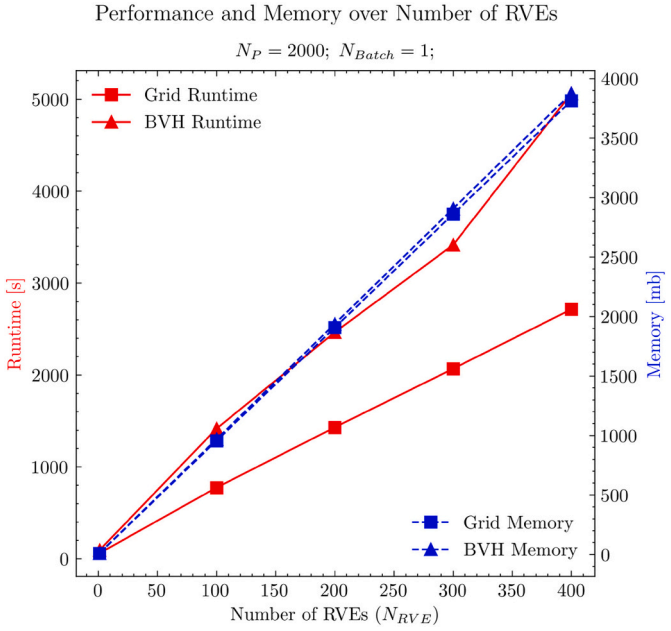


Fig. 14. Performance of the isotropic strain-based compression for many RVEs. The number of RVEs is increased while the particles and batches are kept constant.

in [23].

12. Conclusion

In this paper, a novel algorithm for the parallelization of RVEs in GPU-based DEM is implemented. GPU-based parallelization of the RVEs may be used to generate statistics of material responses by simulating many non-interacting RVEs efficiently. The algorithm features the following key components: (1) GPU parallelism of non-interacting RVEs within the same simulation loop; (2) A member hierarchy of RVE, particle, and boundary memory groups that are aligned with the GPU optimization principles (3) handling of boundary particle and periodicity of deformable walls efficiently by assigning particle bitmasks and a referencing lookup tables (4) modification to the contact detection algorithms to partition non-interacting RVEs. The DEM simulations share a simulation domain on the global cartesian basis which removes the complexities from the contact detection process. Volume averaging of the RVEs is performed by utilizing a segmented summation (D. [37]) of the hierarchical memory groups.

The algorithm in this paper is validated using the drained triaxial experiment of dry quartz sand. Then, a potential application for the algorithm is presented to study the softening and hardening behavior of a material under drained triaxial compression (or incremental loading). Drained triaxial tests are simulated for the same material parameters but with different initial void ratios. The algorithm showed to reproduce expected results for a typical drained triaxial test of dens and loose sand. A large sample set of statistics may be generated by increasing the sample size from 5 drained triaxial tests to 400 or even 1200 drained triaxial tests.

The RVE homogenization as a performance impacting factor is discussed. For instance, the number of particles needed for a system to be homogenized in 3D may be far greater than that of 2D. The number of particles necessary for an RVE to be homogenized is selected if the inverse Chi-squared $1/\chi^2$ plateaus.

A series of performance tests were done under isotropic strain-based compression. The speed improvement of parallelizing 100 RVEs is about 9.8 times faster than the sequential case. This comes at the cost of 100 times the GPU memory usage. It is also shown that handling boundary particles have a fast runtime compared to other processes in the same simulation loop. The total GPU memory for both the BVH and Uniform Grid scales linearly with an increase in the number of particles and RVEs. The Uniform Grid method shows the overall best performance. To improve memory usage, methods can be explored to partition the grid over the deformed RVEs instead of over an extended domain.

The disadvantages of the present GPU algorithm are as follows: (1) The algorithm has a high memory usage, especially for the collision detection algorithm. The high memory usage may also influence the overall performance; (2) Non-interacting RVEs are restricted to have the same number of particles; (3) The force calculation algorithm dominates the percentage performance and should be optimized. The current implementation performs the contact for a thread per real particle and sequentially loops over their respective boundary particles. One

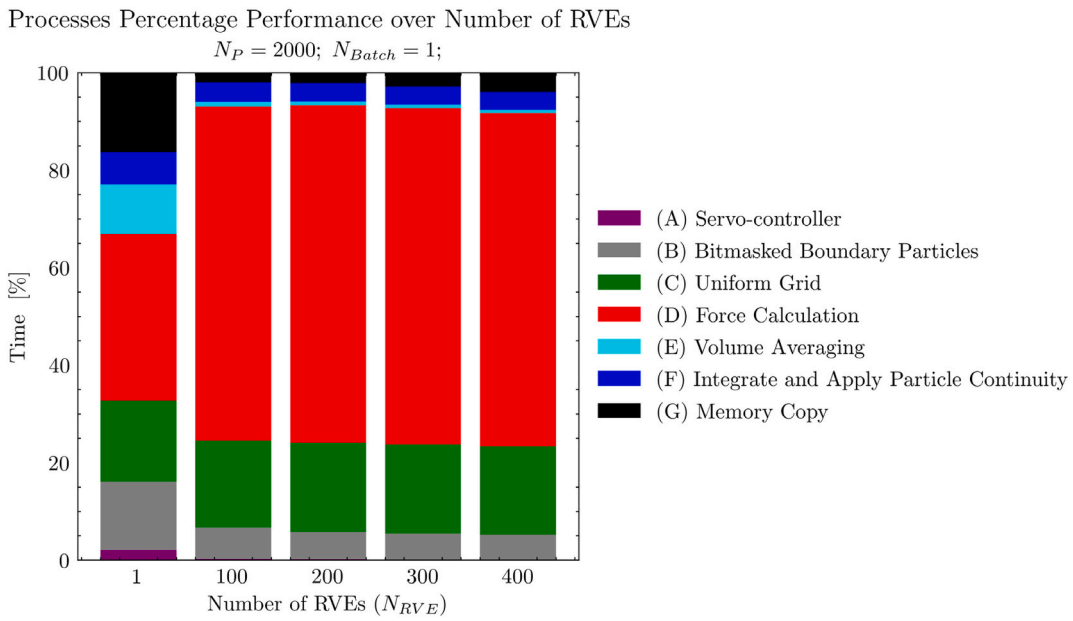


Fig. 15. Percentage runtime of the GPU functions for the isotropic strain-based compression for one RVE. The number of RVEs is increased while the number of particles and batches is kept constant.

approach would be to launch the kernel for a thread per real and boundary particle; (4) Currently, only spherical particles can be parallelized, but the algorithm can be extended to simulate polyhedral or spherical clumps.

In the future, concurrent simulations of a sparse domain with large deformation would be advantageous. The DEM parameters of a landslide simulation may be easily calibrated, and sufficient statistics can be extracted [46]. This may be especially useful in disaster control software, where fast numerical models are critical. Furthermore, simulation level parallelism may be implemented to generate a large statistical sample size or solve a multi-optima calibration problem (i.e., triaxial compression and angle of repose). In these cases, the use of global coordinates and volume averaging by segmented reduction may be well suited since the simulations may have different configurations (number of particles, sparsity of domain size, etc.).

Declaration of Competing Interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgment

The authors would like to acknowledge the project of “Natural Science Foundation of China (52079067, 51879142)”, “Research Fund Program of the State Key Laboratory of Hydroscience and Engineering (2020-KY-04)” and “South African Department of Higher Education and Training (DHET)” for contributing funds and supporting this research.

Appendix A. Simulation parameters

Table 1
The DEM parameters used in this study for the RVE homogenization and performance study. See companion paper for details on the Moment Rotation Law parameters.

E [GPa]	ν	μ_s [°]	η_{twist}	η_{roll}	α_{twist}	α_{roll}
8.00	0.50	25.00	2.00	2.00	0.20	0.20

Table 2
The calibrated DEM parameters for the drained triaxial compression of dry quartz sand under 200 kPa. See the companion paper for details on the calibration processes and Moment Rotation Law parameters.

E [GPa]	ν	μ_s [°]	η_{twist}	η_{roll}	α_{twist}	α_{roll}
4.3	0.50	33.4	1.70	1.80	1.30	1.9

Appendix B. Chi-squared test

The Chi-squared test [47] compares the observed frequencies with the expected frequencies for a table with frequencies of categorical data (contingency table). This method can be used to quantify the similarity between two distributions by binning the observation distribution and simulated distribution on a histogram. The Chi-squared similarity test is defined by

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{23}$$

where O_{ij} is the observational frequencies and E_{ij} is the expected frequencies. The Chi-squared test has a requirement that there should be at least 5 entries in a bin and 13 samples. This study found this condition to be met even for 400 particles and 36 bins of x-y and x-z orientations. The calculated p-values of all RVEs are also well below 0.05 which indices that we can reject the null hypothesis.

Credit author statement

Retief Lubbe, develops the algorithms and write the paper.
Wen-Jie Xu, gives the ideas. structure, and improvement on the contents of the study.
Qian Zhou, together with the first author for the development of the algorithms.
Hongyang Cheng, gives suggestion on the algorithm of Bayesian calibration.

References

[1] P.A. Cundall, O.D.L. Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1979) 47–65, https://doi.org/10.1680/geot.1979.29.1.47.
[2] N. Govender, D.N. Wilke, S. Kok, Blaze-DEMGPU: modular high performance DEM framework for the GPU architecture, SoftwareX 5 (2015) 62–66, https://doi.org/10.1016/j.softx.2016.04.004.
[3] J.-P. Longmore, P. Marais, M.M. Kuttel, Towards realistic and interactive sand simulation: a GPU-based framework, Powder Technol. 235 (2013) 983–1000.
[4] C.A. Radeke, B.J. Glasser, J.G. Khinast, Large-scale powder mixer simulations using massively parallel GPUarchitectures, Chem. Eng. Sci. 65 (2010) 6435–6442.
[5] L. Zhang, S. Quigley, A. Chan, A fast scalable implementation of the two-dimensional triangular Discrete Element Method on a GPU platform, Adv. Eng. Softw. 60 (2013) 70–80.
[6] M.G.D. Geers, V. Kouznetsova, W.A.M. Brekelmans, Multi-scale computational homogenization: trends and challenges, J. Comput. Appl. Math. 234 (2010) 2175–2182, https://doi.org/10.1016/j.cam.2009.08.077.
[7] H.A. Meier, P. Steinmann, E. Kuhl, Towards multiscale computation of confined granular media–contact forces, stresses and tangent operators, Tech. Mech. 28 (2008) 32–42.
[8] C. Thornton, L. Zhang, A DEM comparison of different shear testing devices, in: Powders and Grains 2001, CRC Press, 2020, pp. 183–190.
[9] A. Thabet, A.G. Straatman, The development and numerical modelling of a representative elemental volume for packed sand, Chem. Eng. Sci. 187 (2018) 117–126.
[10] N. Guo, J. Zhao, A coupled FEM/DEM approach for hierarchical multiscale modelling of granular media, Int. J. Numer. Methods Eng. 99 (2014) 789–818, https://doi.org/10.1002/nme.4702.
[11] Q.X. Meng, H.L. Wang, W.Y. Xu, M. Cai, J. Xu, Q. Zhang, Multiscale strength reduction method for heterogeneous slope using hierarchical FEM/DEM modeling, Comput. Geotech. 115 (2019), 103164, https://doi.org/10.1016/j.compgeo.2019.103164.
[12] K. Wang, W. Sun, An updated Lagrangian LBM–DEM–FEM coupling model for dual-permeability fissured porous media with embedded discontinuities, Comput. Methods Appl. Mech. Eng. 344 (2019) 276–305, https://doi.org/10.1016/j.cma.2018.09.034.
[13] Q. Zhou, W.-J. Xu, R. Lubbe, Multi-scale mechanics of sand based on FEM-DEM coupling method, Powder Technol. (2020), https://doi.org/10.1016/j.powtec.2020.11.006.
[14] V. Šmilauer, E. Catalano, B. Chareyre, S. Dorofeenko, J. Duriez, A. Gladky, J. Kozicki, C. Modenese, L. Scholtès, L. Sibille, J. Stránský, K. Thoeni, Yade reference documentation, in: The Yade Project, 2010.
[15] S. Zhao, J. Zhao, W. Lianga, A thread-block-wise computational framework for large-scale hierarchical continuum-discrete modeling of granular media, Int. J. Numer. Methods Eng. 122 (2021) 579–608.
[16] H. Do, A.M. Aragón, D.L. Schott, Automated Discrete Element Method Calibration Using Genetic and Optimization Algorithms, 2017.
[17] H. Cheng, T. Shuku, K. Thoeni, P. Tempone, S. Luding, V. Magnanimo, An iterative Bayesian filtering framework for fast and automated calibration of DEM models, Comput. Methods Appl. Mech. Eng. 350 (2019) 268–294, https://doi.org/10.1016/j.cma.2019.01.027.
[18] H. Cheng, T. Shuku, K. Thoeni, H. Yamamoto, Probabilistic calibration of discrete element simulations using the sequential quasi-Monte Carlo filter, Granul. Matter (2018), https://doi.org/10.1007/s10035-017-0781-y.
[19] Y. Tsuji, T. Kawaguchi, T. Tanaka, Discrete particle simulation of two-dimensional fluidized bed, Powder Technol. 77 (1993) 79–87, https://doi.org/10.1016/0032-5910(93)85010-7.
[20] W.R. Ketterhagen, M.T. am Ende, B.C. Hancock, Process modeling in the pharmaceutical industry using the discrete element method, J. Pharm. Sci. 98 (2009) 442–470, https://doi.org/10.1002/jps.21466.
[21] L. Rothenburg, R.J. Bathurst, Effects of particle shape on micromechanical behavior of granular materials, in: Studies in Applied Mechanics, 1992, https://doi.org/10.1016/B978-0-444-89213-3.50041-9.
[22] N. Govender, D.N. Wilke, S. Kok, R. Els, Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs, J. Comput. Appl. Math. 270 (2014) 386–400, https://doi.org/10.1016/j.cam.2013.12.032.
[23] R. Lubbe, W. Xu, D. Wilke, P. Pizette, N. Govender, Analysis of parallel spatial partitioning algorithms for GPU based DEM, Comput. Geotech. 125 (2020), https://doi.org/10.1016/j.compgeo.2020.103708.
[24] N. Belheine, J.-P. Plassiard, F.V. Donze, F. Darve, A. Séridi, Numerical simulation of drained triaxial test using 3D discrete element modeling, Comput. Geotech. 36 (2009) 320–331.
[25] J. Kozicki, J. Tejchman, Numerical simulations of sand behavior using DEM with two different descriptions of grain roughness, Particle-Based Methods II - Fundament. Appl. (2011) 62–71.
[26] J. Kozicki, J. Tejchman, et al., Numerical simulations of triaxial test with sand using DEM, Arch. Hydro-Eng. Environ. Mech. 56 (2009) 149–172.
[27] C. O'Sullivan, J.D. Bray, Selecting a suitable time step for discrete element simulations that use the central difference time integration scheme, in: Engineering Computations (Swansea, Wales), 2004, https://doi.org/10.1108/02644400410519794.
[28] J. Rojek, G.F. Karlis, L.J. Malinowski, G. Beer, Setting up virgin stress conditions in discrete element models, Comput. Geotech. 48 (2013) 228–248, https://doi.org/10.1016/j.compgeo.2012.07.009.
[29] J. Christoffersen, M. Mehrabadi, S. Nemat-Nasser, A micromechanical description of granular material behavior, J. Appl. Mech. Trans. Asme - J APPL MECH 48 (1981), https://doi.org/10.1115/1.3157619.
[30] Y. Sheng, C.J. Lawrence, B.J. Briscoe, C. Thornton, Numerical studies of uniaxial powder compaction process by 3D DEM, in: Engineering Computations (Swansea, Wales), 2004, https://doi.org/10.1108/02644400410519802.
[31] S.C. Thakur, J.Y. Ooi, H. Ahmadian, Scaling of discrete element model parameters for cohesionless and cohesive solid, Powder Technol. (2016), https://doi.org/10.1016/j.powtec.2015.05.051.
[32] P. Jop, Y. Forterre, O. Pouliquen, A constitutive law for dense granular flows, Nature 441 (2006), https://doi.org/10.1038/nature04801.
[33] NVIDIA, Cuda C Programming Guide, Programm. Guides (2015) 1–261.
[34] P. Goorts, S. Rogmans, S. vanden Eynde, P. Bekaert, Practical examples of GPU computing optimization principles, in: 2010 International Conference on Signal Processing and Multimedia Applications (SIGMAP), 2010, pp. 46–49.
[35] J. Siegel, J. Ributzka, X. Li, CUDA memory optimizations for large data-structures in the gravit simulator, J. Algorith. Comput. Technol. 5 (2011) 341–362.
[36] V. Skorych, M. Dosta, Parallel CPU–GPU computing technique for discrete element method, Concurrency Comput. Practice Experience 34 (2022), e6839, https://doi.org/10.1002/cpe.6839.
[37] D. Merrill, Cub. NVIDIA Research, 2015.
[38] N. Bell, J. Hoberock, Thrust: a productivity-oriented library for CUDA, in: GPU Computing Gems Jade Edition, Elsevier, 2012, pp. 359–371.
[39] J.Q. Gan, Z.Y. Zhou, A.B. Yu, A GPU-based DEM approach for modelling of particulate systems, Powder Technol. 301 (2016), https://doi.org/10.1016/j.powtec.2016.07.072.
[40] D.N.-L. Merrill, CUDA UnBound (CUB) Library [WWW Document], 2015.
[41] J.A. Anderson, C.D. Lorenz, A. Travesset, General purpose molecular dynamics simulations fully implemented on graphics processing units, J. Comput. Phys. 227 (2008) 5342–5359, https://doi.org/10.1016/j.jcp.2008.01.047.
[42] T. Karras, Maximizing parallelism in the construction of bvhs, octrees, and k-d trees, in: High-Performance Graphics 2012, HPG 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings, 2012, pp. 33–37, https://doi.org/10.2312/EGGH/HPG12/033-037.
[43] J.P. de Bono, G.R. McDowell, DEM of triaxial tests on crushable sand, Granul. Matter 16 (2014) 551–562.
[44] R.A. Hosn, L. Sibille, N. Benahmed, B. Chareyre, Discrete numerical modeling of loose soil with spherical particles and interparticle rolling friction, Granul. Matter 19 (2017) 1–12.
[45] V. Baranau, U. Tallarek, Random-close packing limits for monodisperse and polydisperse hard spheres, Soft Matter 10 (2014) 3826–3841, https://doi.org/10.1039/c3sm52959b.
[46] W.-J. Xu, Q. Xu, G.-Y. Liu, H.-Y. Xu, A novel parameter inversion method for an improved DEM simulation of a river damming process by a large-scale landslide, Eng. Geol. 293 (2021), 106282, https://doi.org/10.1016/j.enggeo.2021.106282.
[47] R.L. Plackett, Karl Pearson and the chi-squared test, Int. Stat. Rev./Revue Internationale de Statistique (1983) 59–72.