applied sciences

MDPI

Article

An Efficient Parallel Framework for the Discrete Element Method Using GPU

Youkou Dong [1], Dingtao Yan [1] and Lan Cui [2,*]

[1] College of Marine Science and Technology, China University of Geosciences, Wuhan 430074, China; dongyk@cug.edu.cn (Y.D.); yandingtao@cug.edu.cn (D.Y.)
[2] State Key Laboratory of Geomechanics and Geotechnical Engineering, Institute of Rock and Soil Mechanics, Chinese Academy of Sciences, Wuhan 430071, China
* Correspondence: lcui@whrsm.ac.cn

Abstract: The discrete element method (DEM), a discontinuum-based method to simulate the interaction between neighbouring particles of granular materials, suffers from intensive computational workload caused by massive particle numbers, irregular particle shapes, and complicated interaction modes from the meso-scale representation of the macro information. To promote the efficiency of the DEM and enlarge the modelling scales with a higher realism of the particle shapes, parallel computing on the graphics processing unit (GPU) is developed in this paper. The potential data race between the computing cores in the parallelisation is tackled by establishing the contact pair list with a hybrid technique. All the computations in the DEM are made on the GPU cores. Three benchmark cases, a triaxial test of a sand specimen, cone penetration test and granular flow due to a dam break, are used to evaluate the performance of the GPU parallel strategy. Acceleration of the GPU parallel simulations over the conventional CPU sequential counterparts is quantified in terms of speedup. The average speedups with the GPU parallelisation are 84, 73, and 60 for the benchmark cases.

Keywords: discrete element method; graphics processing unit; parallel computing; triaxial test; cone penetration test; dam break

Citation: Dong, Y.; Yan, D.; Cui, L. An Efficient Parallel Framework for the Discrete Element Method Using GPU. *Appl. Sci.* **2022**, *12*, 3107. https://doi.org/10.3390/app12063107

Academic Editor: Roberto Citarella

Received: 23 January 2022
Accepted: 2 March 2022
Published: 18 March 2022

1. Introduction

The discrete element method (DEM), a discontinuum-based method proposed by Cundall and Strack [1], is capable of simulating the interaction between neighbouring grains directly. The discontinuous nature allows the DEM to tackle large deformations of the material, avoiding the potential mesh entanglement and the necessary re-meshing procedures in the mesh-based methods [2–4]. Particular meso-scale phenomena (such as force network and coordinate number distribution), depending on the sizes, shapes, and cohesion of the grains, can be better captured with the DEM over the conventional RVE (representative volume element)-based methods and possibly also the experimental measurements [5–8]. Averaging techniques were developed to characterise the macro-scale information (such as density, velocity, strain, and stress) from the meso-scale representations, which make the DEM results comparable to those of the continuum-based models and the laboratory experiments [9–11]. In recent decades, the DEM has been applied to investigate a wide range of geotechnical problems, such as soil-structure interaction [12–17], particle breakage [18], shear localization [19–21], and the dense flow of sand [22–24], relating the micromechanics of particle interaction and the overall granular behaviour. A more comprehensive description of the DEM developments and their applications can be found in References [25–27], among many others.

The DEM has been increasingly used in scientific investigations, while its application in industry is much less commonplace. The reason for this is mainly that the number of particles, in reality, tends to overwhelm the computation power provided by the processing devices [28,29]. Therefore, the particle sizes of the prototype were exaggerated on the

basis of similarity theory, which was previously used in centrifuge tests [30–32]. However, the grading of realistic sand usually spans many orders of magnitude, which means the voids created by the large particles need to be filled by massive volumes of fine particles [33–35]. Then, in the DEM models, a relatively narrow particle distribution has to be adopted, especially for three-dimensional simulations. Another limitation in the existing DEM simulations is that the realistic particle shapes were over-simplified as analytical shapes, spheres [36], clumps [37] and polyhedra [38]. These simplifications seem to be acceptable for quartz sand, but apparently not for calcareous sand [39,40]. The more complex idealisation of particle shapes requires more computation efforts due to the complicated contact algorithm. Accurate representation of the particle size and shape is critical to the modelling results, and hence, needs to be calibrated prior to the simulations [41]. Unfortunately, a majority of the existing DEM simulations ignored this procedure and were based on assumptions of input parameters without justification.

To promote the efficiency of the DEM and enlarge the modelling scales with higher realism of the particle shapes, parallel computing on the central processing unit (CPU) or graphics processing unit (GPU) has been explored by a limited number of researchers. References [42–44] developed single-CPU parallel algorithms using a loop-based parallel library OpenMP with a shared memory. Yan and Regueiro [45] presented a multiple-CPU parallel framework with a message passing interface on a distributed memory platform. In comparison with most commercially available CPUs, GPUs feature more dedicated cores in the processor, lower thread-scheduling cost and higher memory bandwidth; as such, they are ideal for arithmetically intensive and highly parallel computations. Currently, most GPU parallelisation is implemented using the 'compute unified device architecture' (CUDA) [46], which provides a friendly platform for coding in languages, such as C++ and Fortran [47–50]. Govender et al. [51] performed an industrial-scale analysis of mill charges with more than eight million polyhedron particles using a GPU-parallelised DEM code. Due to the differences in architectures between the CPU and GPU, DEM codes developed for the conventional CPU calculations need to be specially tuned to be efficiently parallelised on GPUs.

The main bottleneck in parallelising the DEM is the potential data race between the computing cores writing to the common memory address concurrently for the creation of a contact pair list of particles. Gao et al. [17] utilised atomic operations to update the particle list in the material point method by taking advantage of modern graphics hardware features, whose complication excludes most developers from their implementation. Govender et al. [51] established a cell-based hash list for the neighbouring particles, in which extra overhead is taken on sorting the particles by their distances to the origin. Nishiura et al. [52] placed each individual particle into a specific cell with very fine mesh, which often means a heavy memory requirement for a large-scale model along with its simplicity. The technique to update the particle list is vital to the acceleration effect of the parallel framework. In this paper, a new hybrid technique to establish the contact pair list is presented, which combines the advantages of the methods developed by References [51,52]. A GPU-parallel framework of the DEM is presented with all the computations taken on the GPU cores. Three benchmark cases, the triaxial test of a sand specimen, the cone penetration test and sand flow due to dam break, are used to evaluate the performance of the GPU parallel strategy. Acceleration of the GPU parallel simulations over the CPU sequential counterparts is quantified in terms of speedup.

2. Parallelisation of DEM

2.1. DEM Algorithm

The GPU-parallelised computing program DEM-GeoFluidFlow, developed in this study, was coded in C++ on the Windows operating system. To focus on the parallel algorithm itself, soft balls were considered for modelling the granular materials with the novel aspects of the DEM algorithm, such as rolling resistance, mass scaling effect, novel contact strategy, and irregular particle shapes, being ignored.

Consider an assembly of N particles. The inherent properties carried by Particle i are radius R_i, mass m_i, moments of inertia I_i, positions X_i, translational velocities v_i, translational accelerations a_i, angular velocities ω_i and angular accelerations $\dot{\omega}_i$. The dynamic behaviour of the granular materials is obtained by solving the governing equations, i.e., Newton's second law of motion, for individual particles. In each incremental step, the following essential procedures are performed.

2.1.1. Calculate Contact Forces

A contact pair list between connected particles (also particle-wall) is updated prior to the essential calculation of the contact forces, which will be detailed in the following section. Assume Particle i is in contact with Particle j with a distance d_{ij}. The overlap between the two particles will be (Figure 1)

$$U_{ij} = (R_i + R_j - d_{ij})/2 \tag{1}$$

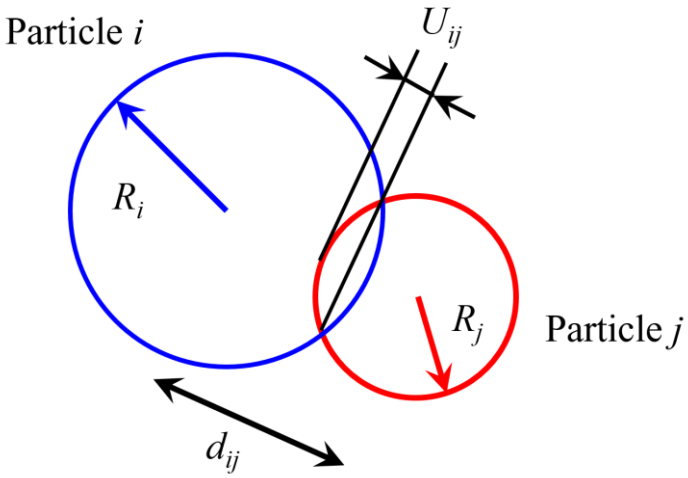


Figure 1. Overlap between particles.

The normal direction from Particle i to j is:

$$n_{ij} = (X_j - X_i)/d_{ij} \tag{2}$$

The normal force imposed on Particle i from Particle j is:

$$F_{ij}^{\mathrm{n}} = -K^{\mathrm{n}} U_{ij} n_{ij} \tag{3}$$

where K^{n} is the normal stiffness at contact. Calculation of the shear force is more complicated. The relative velocity of Particle j to Particle i is:

$$\Delta V_{ij} = \left(V_j - \omega_j \times n_{ij}\right) - \left(V_i + \omega_i \times n_{ij}\right) \tag{4}$$

Then the shear component of the relative velocity can be derived as:

$$\Delta V_{ij}^{\mathrm{s}} = \Delta V_{ij} - \left(\Delta V_{ij} \cdot n_{ij}\right) n_{ij} \tag{5}$$

The corresponding shear force occurring over an incremental step of Δt is:

$$\Delta F_{ij}^{\mathrm{s}} = K^{\mathrm{s}} \Delta V_{ij}^{\mathrm{s}} \Delta t \tag{6}$$

where K^s is the shear stiffness at contact. The new shear contact force is calculated by summing the old shear force $F^s_{ij,old}$ at the start of the timestep by considering the transition of the contact plane [53,54].

$$F^s_{ij} = \left\{F^s_{ij}\right\}_{rot,2} + \Delta F^s_{ij} \tag{7}$$

$$\left\{F^s_{ij}\right\}_{rot,1} = F^s_{ij,old} - F^s_{ij,old} \times n^{old}_{ij} \times n_{ij} \tag{8}$$

$$\left\{F^s_{ij}\right\}_{rot,2} = \left\{F^s_{ij}\right\}_{rot,1} - \left\{F^s_{ij}\right\}_{rot,1} \times \left[\frac{1}{2}(\omega_i + \omega_j)\cdot n_{ij}\right] n_{ij}\Delta t \tag{9}$$

where n^{old}_{ij} is the old normal direction from Particle i to Particle j at the previous incremental step. The widely used slip model is adopted to allow for mutual slip between particles by limiting the shear force values. The frictional force is adjusted by:

$$F^s_{ij} = F^s_{ij}\frac{\min\left(\mu\left|F^n_{ij}\right|, \left|F^s_{ij}\right|\right)}{\left|F^s_{ij}\right|} \tag{10}$$

Then the total force at Particle i from Particle j is:

$$F_{ij} = F^n_{ij} + F^s_{ij} \tag{11}$$

and the moment is:

$$M_{ij} = n_{ij} \times F^s_{ij} \tag{12}$$

2.1.2. Update Kinematic State of Particles

The kinematic state of each particle, i.e., translational and rotational motions, is determined by the resultant force and moment from the surrounding particles. The translational motion of a particle is described by the position X_i, velocity V_i, and acceleration a_i. The rotational motion is expressed as angular velocity ω_i, and angular acceleration $\dot{\omega}_i$. The translational and angular accelerations can be calculated by:

$$a_i = \sum F_{ij}/m_i + g - \alpha\left|\sum F_{ij} + g\right|\mathrm{sign}(V_i)/m_i \tag{13}$$

$$\dot{\omega}_i = \sum M_{ij}/I_i - \alpha\left|\sum M_{ij}\right|\mathrm{sign}(\omega_i)/I_i \tag{14}$$

where Σ represents a summation over all neighbour particles that needs a sophisticated contact detection algorithm and will be detailed in the sequel; the second term is to consider the local damping effect with coefficient α [1]; g is the gravitational acceleration. Then the velocities can be updated as:

$$V^{new}_i = V_i + a_i\Delta t \tag{15}$$

$$\omega^{new}_i = \omega_i + \dot{\omega}_i\Delta t \tag{16}$$

The timestep Δt for a linear contact model can be determined by:

$$\Delta t = \beta\sqrt{m_{min}/K^n} \tag{17}$$

where β is a coefficient, m_{min} is the minimum mass of the particles, and the position of the particle is updated as:

$$X^{new}_i = X_i + V_i\Delta t \tag{18}$$

2.2. GPU Parallelisation of "Calculate Contact Forces"

2.2.1. Naïve Algorithm

The connection network (Figure 2a) between the neighbour particles must be sorted out before the essential calculation of contact forces. There have been various contact

detection schemes for specific application purposes, of which the most popular ones are adopted in this study. The descriptions of other schemes, such as the nearest-neighbour scheme and the sweep and prune scheme, can be found in References [55–58]. A simple but "naïve" algorithm is used to check each particle against every other in a nested loop, as in Algorithm 1, which takes the total number of manipulations of $O(N^2)$. The naïve algorithm would be very time-consuming, even with a small number of particles [59].

Algorithm 1 "naïve" algorithm.

```
For i = 0 to N-1
For j = i + 1 to N
If distance(i, j) < Ri + Rj
Particles i and j in contact
end
end
end
```

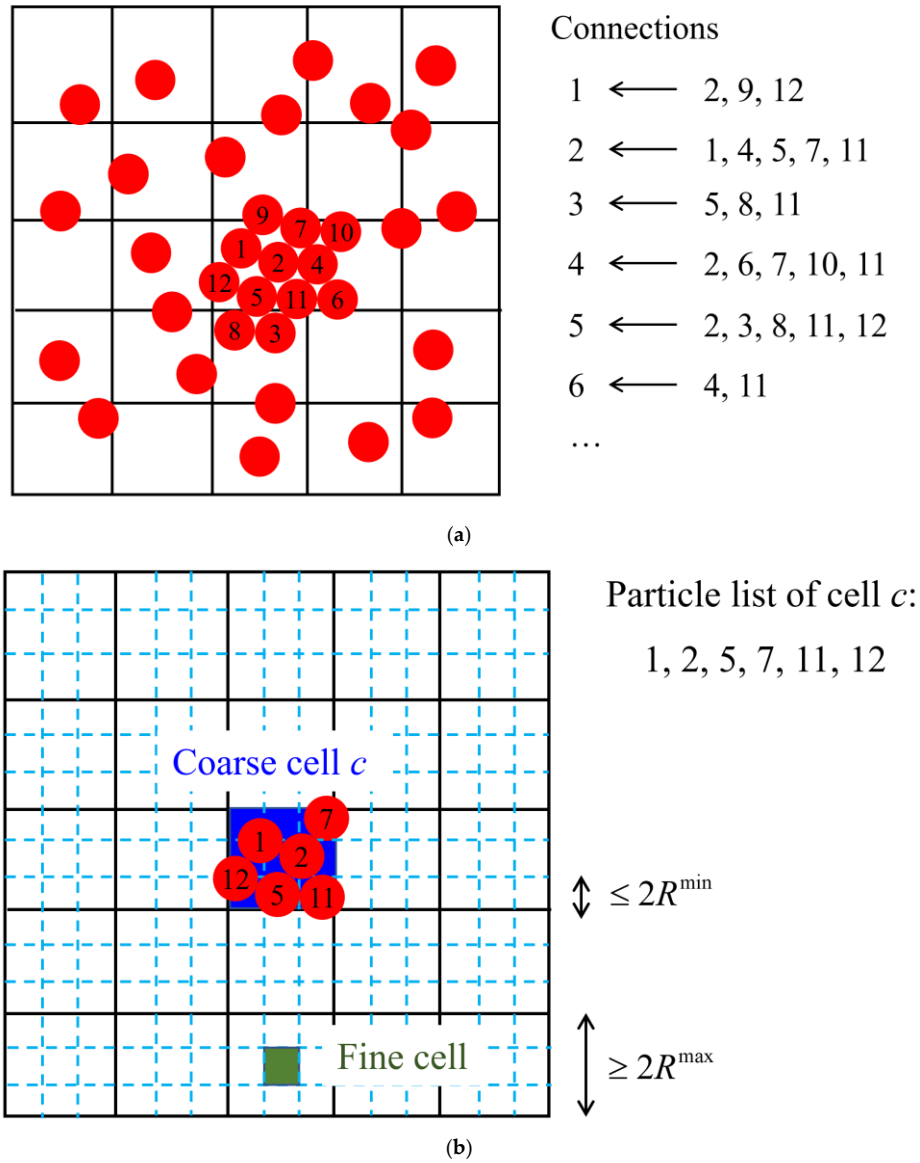


Figure 2. Connection network between particles: (**a**) One set of cells (**b**) hierarchical cells.

2.2.2. Advanced Scheme

The advanced scheme can be divided into two phases: neighbour searching and exact contact detection. The neighbour searching phase roughly identifies the potential neighbours of each particle and constructs a potential contact pair list, while the exact contact detection phase accurately calculates the overlaps between particles through naïve detection. The efficiency of the contact detection schemes is mainly dependent on the neighbour searching phase by excluding the unconnected particles from the potential contact pair list. Particularly, the computational domain is divided into a grid of cells (Figure 2a), each of which accommodates a number of particles. An individual particle may be in contact with another one sharing a common cell or in neighbouring cells. Then the computational cost of contact detection is proportional to the average number of particles in the neighbour cells n_c. In Algorithm 2, the total number of manipulations is $O(Nn_c)$, which is much less than in the naïve algorithm.

Algorithm 2 Neighbour cell scheme.

```
For i = 0 to N
Check Xi
Add Particle i into particle list of Cell c
End
For i = 0 to N
Check Xi, then know:
Neighbour cells of Particle i
For particles in neighbour cells
"Naïve" detection
 End
End
```

To evaluate the distribution of the workload across the procedures, a sequential CPU simulation of a triaxial test was performed with a total of 3920 particles (the first scenario in Table 1, which will be detailed later). As the naïve algorithm is used, more than 99.6% of the computational effort is allocated to the contact detection operations. With the neighbour cell contact detection scheme adopted, the computational efficiency is promoted around 44 times. The establishment of the contact pair list takes about 11.6% of the total runtime, while the essential contact force calculation accounts for 69.2%. The "Update kinematic state of particles" operations occupy 19.2% of the total computational efforts. Trivial runtime is allocated to other manipulations, such as the movement of wall and save old state variables.

Table 1. Speedups of CPU and GPU parallel simulations in triaxial test cases (10,000 incremental steps, d_{50} = 3.2 mm).

Number of Particles	D/d_{50}	CPU Sequential	CPU Parallel (10 Threads)		GPU Parallel (Double-Precision)	
		Runtime (s)	Runtime (s)	Speedup	Runtime (s)	Speedup
3920	25	618	144	4.3	19	33
7840	31.5	2248	500	4.5	49	46
15,680	40	6072	1265	4.8	76	80
31,360	50	13,047	2558	5.1	150	87
100,880	72	40,375	7764	5.2	475	85
250,880	100	115,593	21,810	5.3	1409	82
501,760	126	333,600	60,655	5.5	3879	86
1,003,520	158	666,298	118,980	5.6	8328	80
2,007,040	200	1,399,226	249,861	5.6	16,461	85
4,997,600	316	3,464,750	607,850	5.7	40,287	86

2.2.3. Hybrid Scheme

A hybrid scheme to detect the contact pair list is developed here, which follows the two-step framework of the advanced algorithm (Figure 3). In the first phase of the contact detection operations, the computational domain is divided into a grid of coarse cells. The sorting of particles into each cell can be parallelised across the GPU threads over the particles. However, a data race may be induced at the common cell that is shared by neighbouring particles: the result of writing different particle IDs (i.e., $1 \ldots N$) concurrently into a common memory address is unexpected. Therefore, an expanded particle list of each cell is adopted by further dividing the coarse cell into a number of finer ones (Figure 2b). Each fine cell corresponds to a specific memory address of the particle list of a coarse cell and accommodates only one particle or less. That means the data race can be avoided by writing the particle IDs into different memory addresses of the particle list, which is determined by the coordinates X_i of the particles (as in Algorithm 3, where N_c is the total number of the coarse cells). The size of the fine cells should be smaller than $2R^{min}$ to avoid accommodating more than two particles in one fine cell, where R^{min} is the minimum radius of the particles. The expanded particle list is sparse due to a large number of empty spaces, which means a heavy memory requirement. An additional compression step is then performed to obtain a dense particle list. The potential contact pair list will be updated in a number of incremental steps, in between which the particles would not cross different coarse cells. The number of incremental steps to regenerate the contact pair list can be determined by trial calculations and empirical predictions.

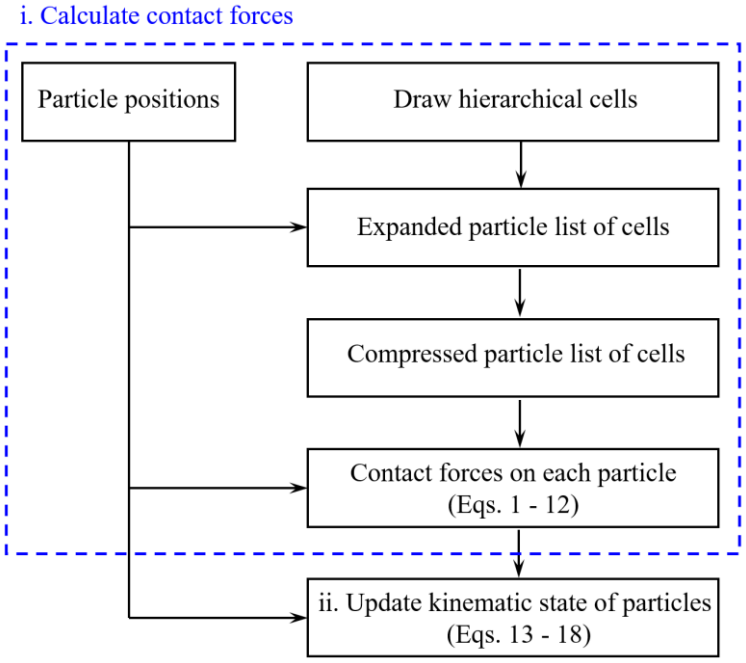


Figure 3. Flowchart of parallelisation operations in each incremental step.

At the second phase, the essential contact force on each particle is calculated by checking all the particles in its neighbour coarse cells, which is parallelised across the GPU threads over the particles. For a specific particle i, it may be in contact with all the particles in its range of $R_i + R^{max}$, where R^{max} is the maximum radius of the particles. Therefore, we construct the coarse cells with a uniform size of $2R^{max}$ and all the particles in 27 neighbouring cells for three-dimensional analysis are involved in the calculations. For each contact pair of particles, Equations (1)–(12) are used to calculate the contact forces between the particles.

Algorithm 3 GPU parallelisation of 'Calculate contact forces'.

```
For i = 0 to N (divided by threads)
Check Xi, based on configuration of fine cells, then:
Put Particle i to its position in the expanded particle list of Cell c
End
For c = 1 to Nc (divided by threads)
Compress the expanded particle list
End
For i = 1 to N (divided by threads)
Check Xi, then know:
  27 Neighbour cells of Particle i
  For particles j in neighbour cells
  Equations (1)–(12)
  End
End
```

2.3. GPU Parallelisation of "Update Kinematic State of Particles"

The procedure "Update kinematic state of particles" in Equations (13)–(18) is straightforward; to be parallelised across the GPU threads over the particles (Figure 3), i.e., the update of the velocities and accelerations of each particle is scheduled into a thread (as in Algorithm 4). The required information, such as the contact forces and old velocities, is in terms of the particular particle in each thread and is not dependent on any variable on other threads. Therefore, there is no risk of a data race as in the procedure "Calculate contact forces".

All of the calculations in Equations (1)–(18) are expected to be parallelised on the GPU to maximise the speedup of the GPU parallelisation. The variables were copied from the CPU memory to the GPU counterpart prior to the calculations. The GPU memory spaces include the global, register, and local memories. The variables transferred between the most functions, such as the masses, velocities, and momenta at the particles, were allocated to the global memory; therefore, they can be accessed directly by the executed GPU threads. The temporary variables, such as the local coordinates of particles in the cells, were stored on the register memory of each thread. The local memory serves as the backup of the register memory; once the register memory is fully occupied, the temporary variables are automatically allocated to the local memory. In comparison to CPU parallelisation, GPU parallelisation is more sensitive to memory access patterns. To achieve a high memory read/write bandwidth, the access to the global memory on each thread should have coalesced. Therefore, parallelisation of the procedure "Update kinematic state of particles" is more efficient, as its threads write variables to the global memory in a consecutive sequence.

Algorithm 4 GPU parallelisation of 'Update kinematic state of particles'.

```
For i = 0 to N (divided by threads)
Equations (13)–(18)
End
```

3. Performance Evaluation

Three benchmark cases, the triaxial test of a sand specimen, cone penetration test and sand flow due to dam break, were used to evaluate the performance of the GPU parallel computing strategy. The parallel computing was conducted on a workstation with a GPU NVIDIA GeForce Titan V and a 6-core (12-thread) CPU Intel i7-6850K with a frequency of 3.6 GHz. On the GPU, a maximum of 3840 cores were available, and the global memory was 12 GB. For comparison purposes, the benchmark cases were simulated with a single-CPU core, multiple-CPU cores (OpenMP) and a single-GPU (CUDA), respectively, with a well-tuned code on each platform. All the simulations were using double-precision numbers as single-precision seems to be insufficiently accurate to describe the tiny particles with a radius of less

than 0.1 mm [60]. A "speedup" factor for the whole incremental step or each operation within the incremental step is used to quantify the effect of parallel computing, expressed as:

$$\text{Speedup} = T_{\text{Sequential}}/T_{\text{Parallel}} \tag{19}$$

where $T_{\text{Sequential}}$ and T_{Parallel} represent the average runtimes of CPU sequential and CPU or GPU parallel calculations per increment.

3.1. Triaxial Test

The micro characteristics of the granular material are often calibrated using the triaxial element test by comparing the macro response of the samples. Simulation of triaxial tests was performed to evaluate the performance of the GPU parallel strategy. A cylindrical element of a standard sand specimen, with a height H of 8 cm and a diameter D of 4 cm, was modelled and enclosed by smooth boundary walls, as shown in Figure 4a. The Fujian standard sand—a marine sand—with the particle size distribution shown in Figure 4b, was considered. In order to reduce the computing time, particle diameters in the simulations were scaled up to 7.5 times the real sand particles, which were represented as d_{10} = 2.4 mm, d_{50} = 3.2 mm and d_{90} = 5 mm. The uniformity coefficient was $C_u = d_{60}/d_{10}$ = 1.57 (well sorted). The initial void ratio was 0.592. Therefore, a total of 3920 spheres were generated and randomly distributed, corresponding to the characteristic ratio D/d_{50} as 12.5. Given the real particle sizes are used, more than 1,650,000 particles would be required. The density of the particles was 2643 kg/m^3. The self-gravity of the particles was not considered. The normal and tangential stiffness (K^n, K^s) were taken as 300 MPa and 200 MPa, respectively, and the friction coefficient was 0.5. A local damping ratio of 0.7 was used to accelerate the equilibrium process. A confining pressure of 300 kPa was imposed prior to the loading procedure through a servo system on the boundary walls. The under-compaction method was adopted to prepare a homogenous specimen [61], then the specimen was loaded with the top and bottom plates moving inward at a velocity of 1 cm/s (i.e., a strain rate of 12.5%), while the confining pressure in the lateral direction was maintained.

The stress–strain curves from the CPU and GPU computations are shown in Figure 5a. Considering the uncertainty due to the relatively small particle numbers, the predictions with different tools agree well with each other. The derivative stress converges to a critical value of 320 kPa. Figure 6 presents the transient velocity, the accumulative displacement, and the force chains on the particles at an axial strain of 6.2%. Another sequential CPU analysis using the widely used software package Particle Flow Code in 3 Dimensions (PFC3D) was also performed, and the results agree well with the CPU and GPU curves at the plastic stage of deformation, which verifies the accuracy of the tools developed in this study. An average runtime of 618 s was consumed in the CPU sequential calculations for 10,000 incremental steps, in which around 78% of the efforts were taken over the procedure "Calculate contact forces", and 22% over the procedure "Update kinematic state of particles". As 10 threads were invoked in the CPU parallel computation, a speedup of 4.3 was obtained over the CPU sequential calculation (Table 1). In comparison, the GPU parallelisation accelerates the computations 33 times over the CPU calculations using a single thread. In particular, the speedups of the GPU parallelisation for the procedures "Calculate contact forces" and "Update kinematic state of particles" are 26 and 272 (Table 2), respectively.

Then, the ratio of the specimen diameter to the mean particle diameter, D/d_{50}, was increased from 25 to 31.5, 40, 50, 72, 100, 126, 158, 200 and 316 by enlarging the model sizes to increase the computational scale, while the aspect ratio H/D of the specimen remained as 2; the corresponding particle numbers increased from 3920 to 7840, 15,680, 31,360, 100,880, 250,880, 501,760, 2,007,040 and 4,997,600. Theoretically, the space of the global memory (12 GB) of the GPU used in this study is capable of accommodating particles up to 27,000,000, for which the total runtime to finish an integrated case would be unacceptably long. Therefore, the maximum number of particles was kept below 5,000,000 as the concern of many geotechnical problems. The loading velocity remained at 1 cm/s. The stress–strain curves obtained with 31,360 and 250,880 particles present better convergence and stability

(Figure 5b). The average runtime of the CPU sequential simulation in 10,000 incremental steps was investigated for a comparison with the CPU and GPU parallel simulations. For the CPU sequential calculations, the runtime is nearly linearly proportional to the number of particles (Table 1). The CPU parallel simulations were also performed using the OpenMP with up to 10 cores. The speedups of the CPU parallel simulations increase with the number of CPU cores (see Table 1 and Figure 7a). The maximum speedup with 10 threads is 5.7. With more CPU threads mobilised, the acceleration efficiency wanes, which is due to the overhead on the scheduling manipulations.

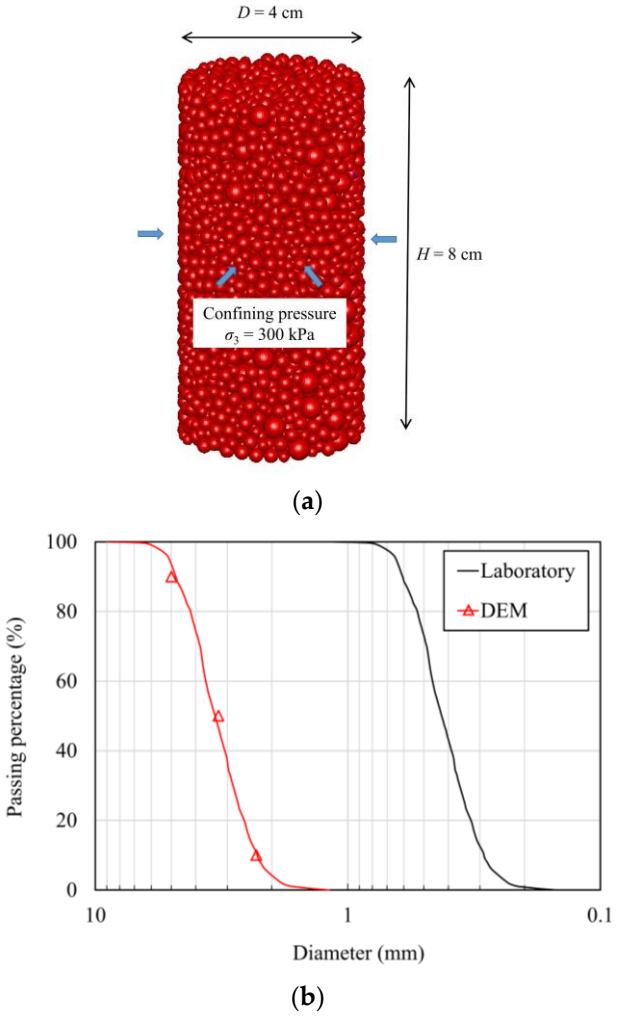


Figure 4. Schematic for discrete element model of the triaxial test. (**a**) specimen setup (**b**) particle size distribution.

The significant reduction in runtimes by the GPU parallel strategy is clear. The speedup increases to less than 31,360 particles; if the computational scale is enlarged further, the GPU runs with a full load, and the improvement presents a good scaling behaviour and gradually stabilises to an average speedup of 84 (Figure 7b). The total runtime for the whole simulation process within an axial strain of 20% is proportional to the square of D/d_{50}, e.g., 8.3 h for D/d_{50} = 50 (31,360 particles) and 18.75 h for D/d_{50} = 75 (100,880 particles), and that for D/d_{50} = 158 (1,003,520 particles) is estimated as 75 h. However, for the real Fujian sand with d_{10} = 0.32 mm, d_{50} = 0.43 mm and d_{90} = 0.67 mm, the runtime is further increased by the leap of an incremental step. Therefore, the total runtimes for a standard sample with H = 8 cm and D = 4 cm accommodating 1,650,000 real sand grains is extrapolated to be more than 92 days, which remains unacceptable for quantitative analysis of the real sand.

Therefore, compromising techniques, such as decreasing the particles' stiffness, expanding the particle sizes and mass scaling, may remain necessary.

The performance of the GPU parallel computing varies between the two computational procedures, which are listed in Table 2. The efficiency of the procedure "Update kinematic state of particles" is enhanced tremendously with speedups of more than 150. In contrast, the acceleration effect of the "Calculate contact forces" procedure is much less, mainly due to the low arithmetic intensity in the operations. The overhead on updating the contact pair list accounts for about 15% of that on the whole "Calculate contact forces" procedure. The acceleration effect of the GPU parallelisation can be influenced by various factors, e.g., the size of the coarse cells and the number of empty cells. To accommodate all the potential connecting particles, the coarse mesh needs to be larger than $2R^{max}$. Given that a larger coarse mesh size is used, the cell number reduces and the particle number in each cell accordingly increases (Figure 8). The speedup increases from 82 to 123 with the coarse mesh size varying from $2R^{max}$ (0.005 m) to $4R^{max}$ (0.01 m), and then its value decreases with a coarse mesh larger than $4R^{max}$. For consistency purposes, the coarse mesh size remained as $2R^{max}$ in the following simulations. The speedups can be undermined by the existence of empty cells, which means some GPU threads are idle in the calculations. The influence of the empty cells can be roughly estimated by the ratio of the number of empty cells to engaged ones.

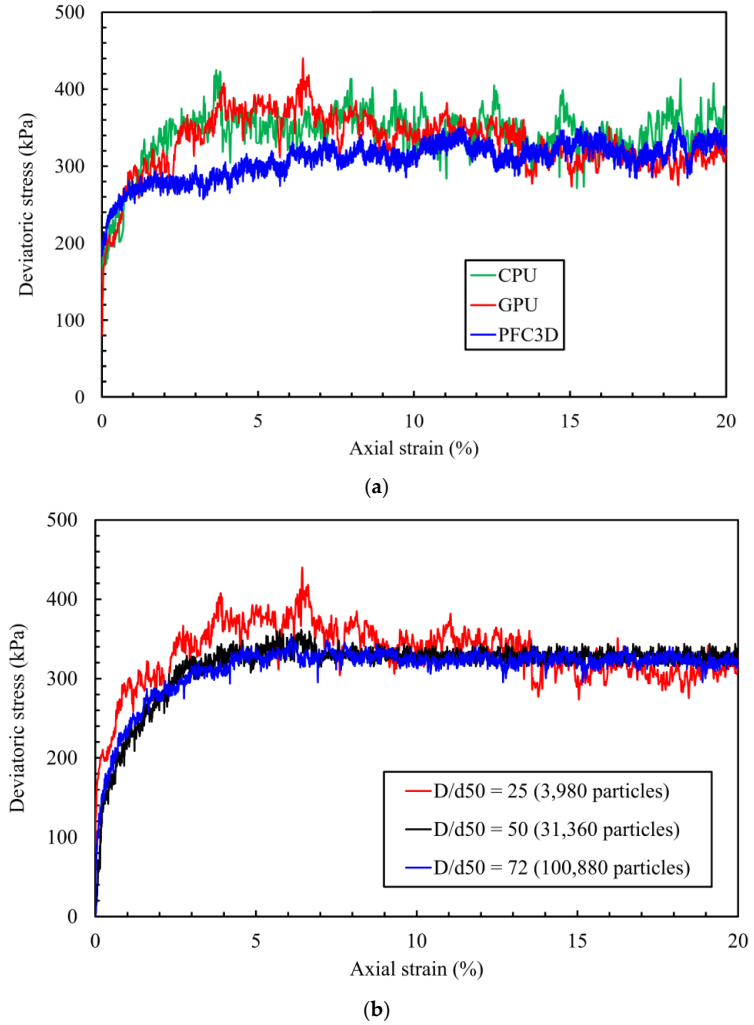


Figure 5. Deviator stress changes versus axial strain during triaxial tests with confining stress of 300 kPa. (**a**) with different tools (**b**) influence of model size.

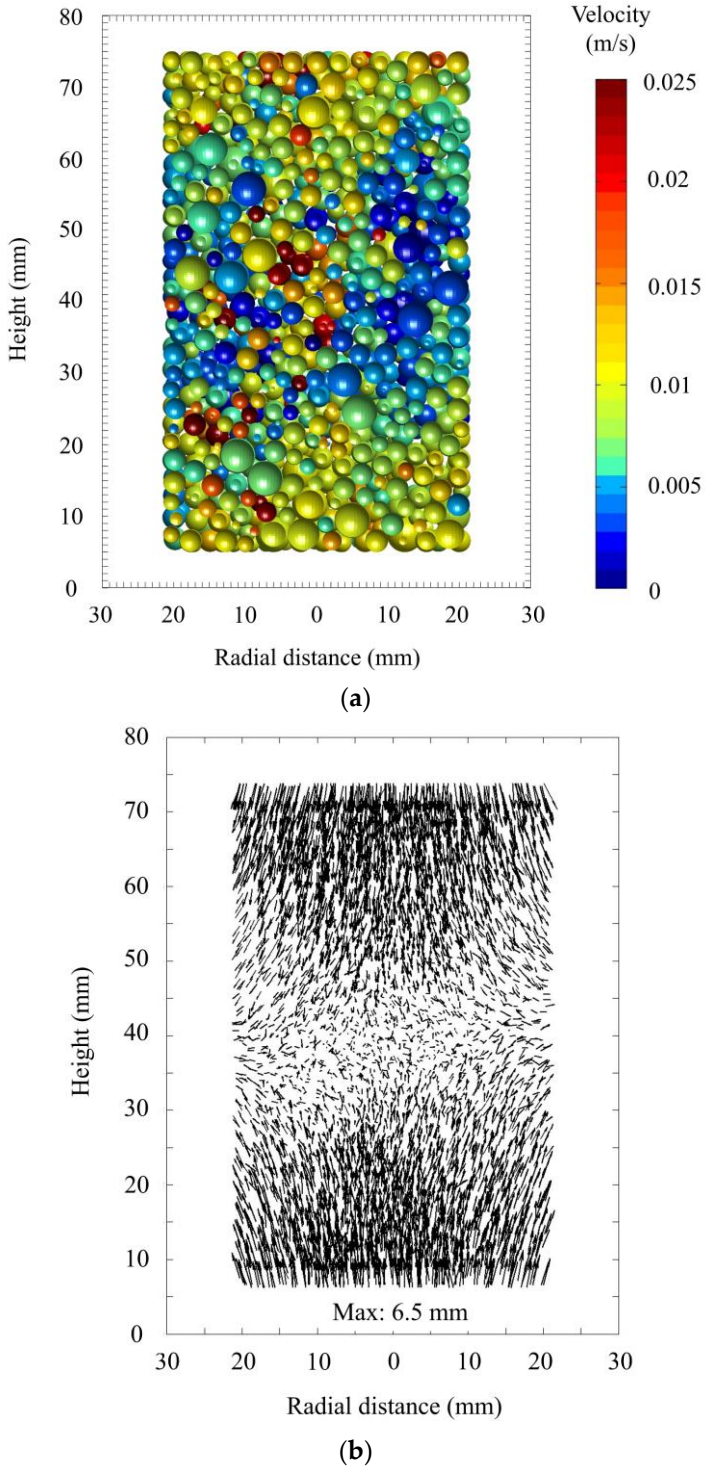


Figure 6. Deformation and mechanical profiles of sample at axial strain of 6.2% (3920 particles). (**a**) transient velocity (**b**) accumulative displacement.

Table 2. Speedups of GPU parallel calculations for the procedures in triaxial test cases.

Number of Particles	Calculate Contact Forces between Particles			Update Kinematic State of Each Particle		
	CPU Sequential	GPU Parallel	Speedup	CPU Sequential	GPU Parallel	Speedup
3920	482	18.5	26	136	0.5	272
7840	1900	47	40	348	1.8	193
15,680	5478	73	75	594	3.3	180
31,360	11,764	144	82	1283	5.9	217
100,880	36,404	455	80	3971	20	198
250,880	107,312	1369	78	8281	40	210
501,760	316,020	3784	83	17,580	95	185
1,003,520	627,622	8070	78	38,676	258	150
2,007,040	1,326,887	16,291	81	72,339	425	170
4,997,600	3,273,441	39,280	83	191,309	1007	190

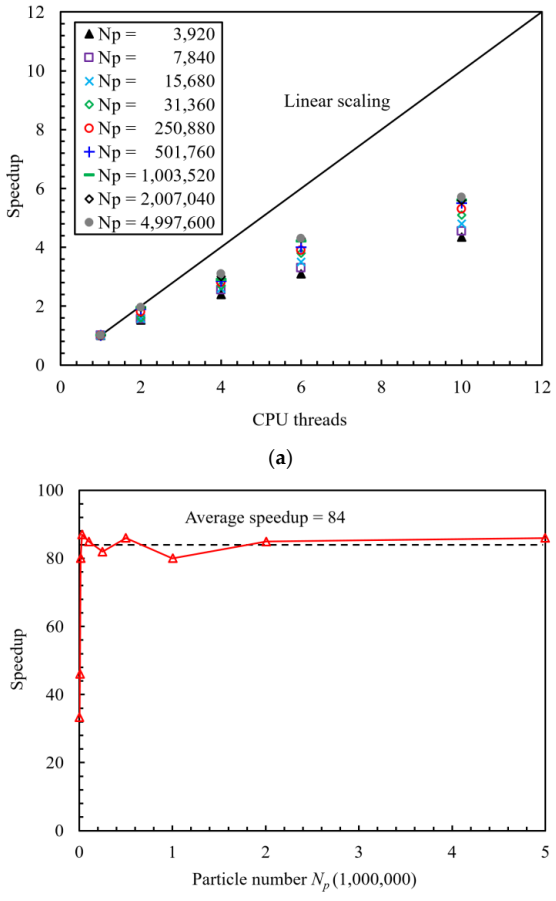


Figure 7. Speedups of the CPU and GPU parallelisation for the triaxial test. (**a**) CPU parallelization (**b**) GPU parallelisation.

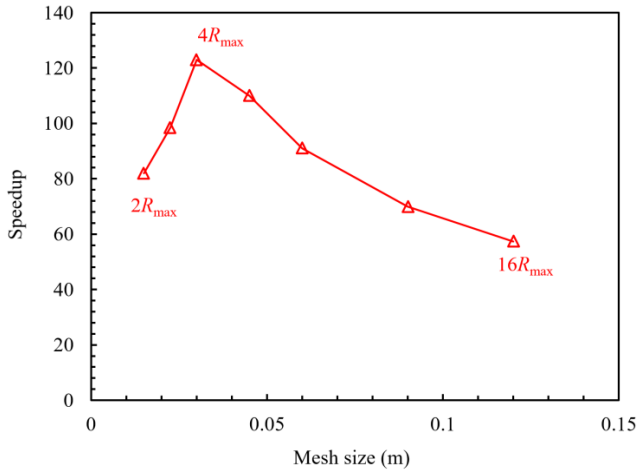


Figure 8. Influence factors on acceleration effect.

3.2. Cone Penetration Test

A series of cone penetration tests of Ticino sand in a virtual calibration chamber were performed using the DEM code PFC3D [62,63], which were simulated in this study for comparison. The chamber was a cylindrical container with a height H of 0.7 m and a diameter D of 1.2 m. The cone had a diameter d_c of 71.2 mm, and its tip had an angle of 60°, as shown in Figure 9a. The cone tip and frontal section of the shaft with a length of 0.1 m had a friction coefficient of 0.35.

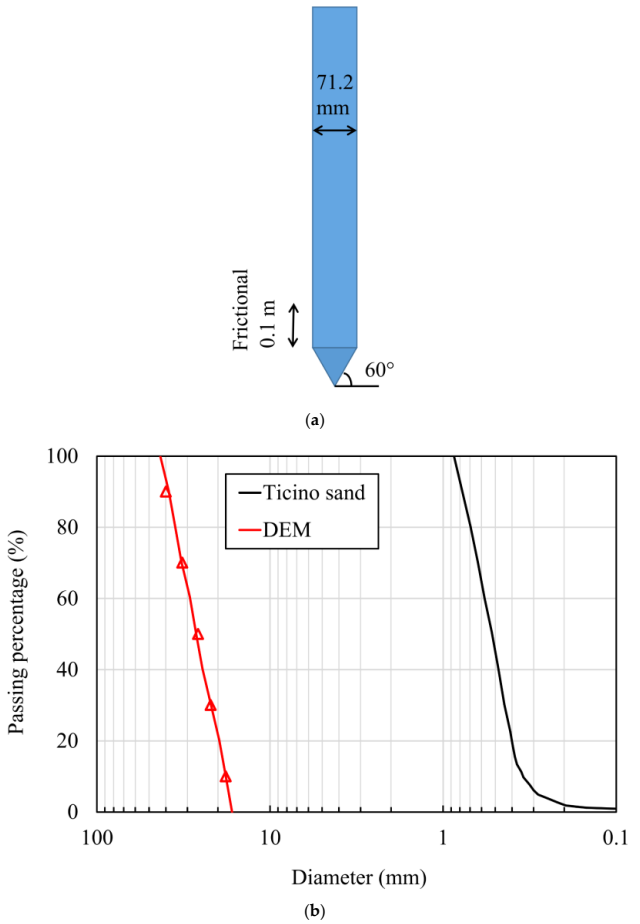


Figure 9. Configurations in the numerical model of cone penetration test (**a**) size of cone (**b**) particle size distribution of sand.

The grain size distribution of the Ticino sand is shown in Figure 9b, which was scaled by a factor of 50 and represented as d_{10} = 18 mm, d_{30} = 22 mm, d_{50} = 26 mm, d_{70} = 32 mm, and d_{90} = 40 mm. The uniformity coefficient was C_u = 1.61 (well sorted). Therefore, a total of 65,000 particles were initialised, corresponding to the characteristic ratios D/d_{50} = 46 and d_c/d_{50} = 2.74. The boundary effect from the container could be eliminated with the sufficiently large value of D/d_{50}, while the penetration resistance will fluctuate due to the relatively low value of d_c/d_{50}. Using the real particle sizes in reality, there would be more than 8,125,000,000 particles involved, which would overwhelm the computational capacity. The initial void ratio was 0.66. The density of the particles was 2643 kg/m^3. The self-gravity of the particles was not considered. The normal and tangential stiffness of the particles were taken as 300 MPa and 75 MPa, respectively, and the friction coefficient was 0.35. A local damping ratio of 0.7 was used to accelerate the equilibrium process. The specimens were created to the target void ratio using the radius expansion method under a confining pressure of 100 kPa. In the penetration operations, the cone penetration was driven at a rate of 0.1 m/s.

The key measurable quantity for the cone penetration test is the cone factor f_c, which is the force measured at the cone surface divided by the cone area. The DEM predictions with different particle numbers are shown in Figure 10, in which the raw vibrating curves are fitted using an equation [63]:

$$f_c = a[1 - \exp(-bw)] \tag{20}$$

where a and b are the fitting coefficients; w is the penetration depth. The value of a characterises the ultimate bearing capacity of the cone below a particular depth. The prediction of a in this study is 1.25, which is higher than that obtained by Butlanska et al. [63] of 1.17 for 7%, which is mainly attributed to the inaccuracy of the DEM algorithm itself.

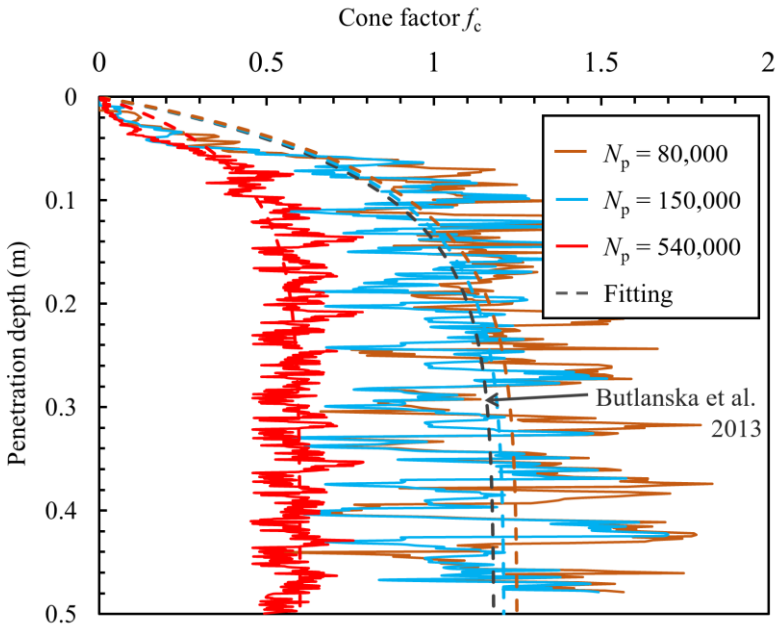
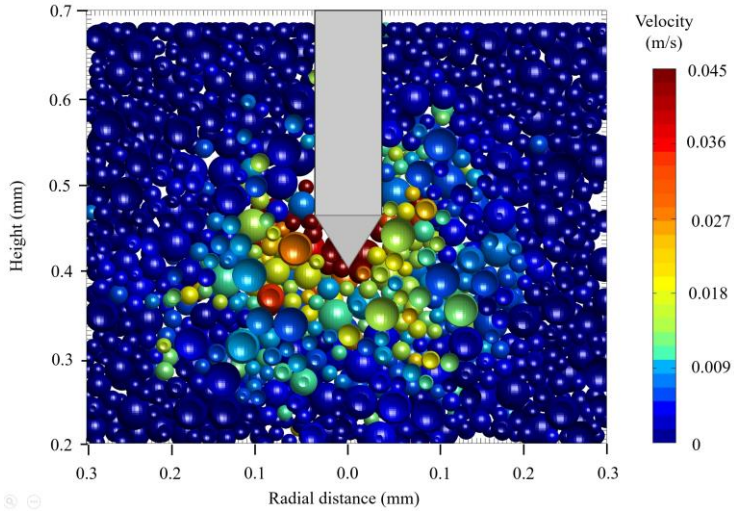


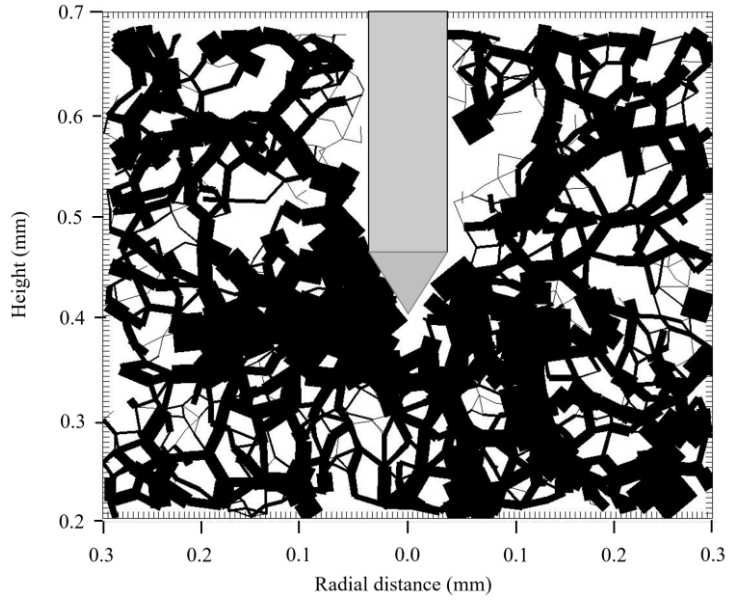
Figure 10. Cone factors predicted by DEM simulations.

Smaller particle sizes were adopted with the ratio d_c/d_{50} modified to 3.45, 5.5, 11, 22 and 55, making the particle numbers increase to 150,000, 520,000, 1,001,000, 2,002,000 and 4,881,200. The fitted value of a by the DEM using 150,000 particles (1.21) is similar to the fitted a with 65,000 particles (1.25), both of which are much higher than the value with 540,000 particles (0.6). Fewer fluctuations occur in the resistance profile using more particles. The comparison hints that an obvious scale effect exists in the cone penetration test relating to the value of d_c/d_{50}. The scenario with smaller particles is not simulated in

this study, as the focus here is on the parallel effect. The scale effect in the cone penetration test has been studied by References [64,65]. The transient velocity and the force chain of the particles around the cone with d_c/d_{50} = 3.45 (65,000 particles) are shown in Figure 11. The average runtimes in 10,000 incremental steps were compared between the CPU sequential, CPU parallel, and GPU parallel simulations (Table 3). The speedups for the GPU parallel simulations are relatively stable, implying that the GPU cores are running under full loads. The maximum speedup of the CPU parallel simulations with 10 threads is 5.0, which is much less than the average speedup of 73 with the GPU parallelisation (Figure 12). The average speed up for the cone penetration tests is close to that for the triaxial tests with a difference of around 15%. Therefore, it is expected that the GPU parallelisation accelerates the three-dimensional simulations by about 80 times. However, the current single-GPU parallelisation is not sufficient for the real-sized simulation of the cone penetration element test, with more than 8,125,000,000 particles involved, which will be even more particles for the in situ cone penetration tests for geotechnical investigations. Calibration of scaled numerical models of the cone penetration test with the real-sized physical ones is then necessary.



(**a**) transient velocity



(**b**) force chain

Figure 11. Displacement of particles around the cone (d_c/d_{50} = 3.45; maximum: 0.037 m).

Table 3. Speedups of CPU and GPU parallel simulations in cone penetration test cases (10,000 incremental steps).

Number of Particles	d_c/d_{50}	D/d_{50}	CPU Sequential	CPU Parallel (10 Threads)		GPU Parallel (Double-Precision)	
			Runtime (s)	Runtime (s)	Speedup	Runtime (s)	Speedup
65,000	2.74	46	16,918	3525	4.8	236	72
150,000	3.45	58	35,336	7362	4.8	476	74
520,000	5.5	92	179,411	36,615	4.9	2347	76
1,001,000	11	184	373,927	76,312	4.9	5125	73
2,002,000	22	368	748,954	149,790	5.0	9981	75
4,881,200	55	920	1,852,845	370,569	5.0	25,798	72

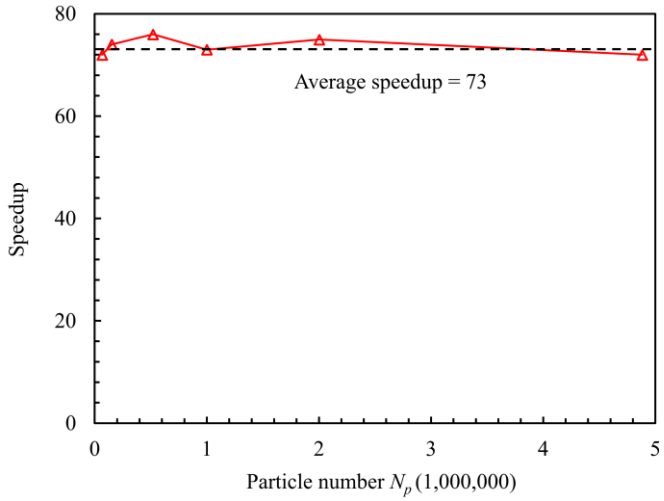


Figure 12. Speedups of GPU parallelisation for cone penetration test.

3.3. Two-Dimensional Dam Break

A planar granular runout induced by a dam break was carried out by instantaneously releasing a sand pile along a smooth bed, which initially had a height H of 0.2 m and a length L of 0.2 m (Figure 13). The particle size distribution was d_{10} = 1.4 mm, d_{30} = 2 mm, d_{50} = 3 mm, d_{70} = 3.4 mm and d_{90} = 4 mm. A total of 4000 particles were initially generated. Before the release of the dam, the uniform packing of sand particles was initially consolidated under their own self-weight, obtaining a height of 0.172 m and a void ratio of 0.28. The density of the particles was 2643 kg/m^3. The normal and tangential stiffness were taken as 50 MPa and 20 MPa, respectively, and the friction coefficient was 0.5. A local damping ratio of 0.7 was adopted. Verification of the code in this study was performed by comparing the runout morphologies with those using PFC3D (Figure 13). The velocity field and force chains of the particles at 50 s are shown in Figure 14. The average runtimes in 10,000 steps are 20 s and 6 s for the CPU- and GPU-parallel computations, respectively, which are much smaller than those for the three-dimensional cases with 144 s and 19 s for counterparts in the triaxial test case with 3920 particles. This is the reason for the wide application of two-dimensional simulations of the DEM.

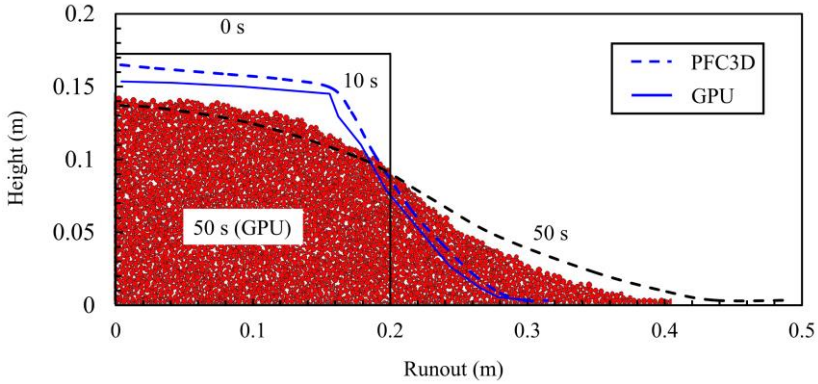
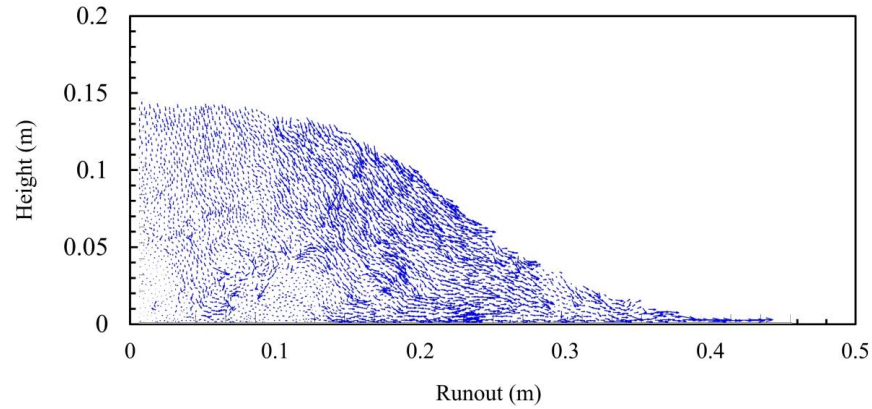
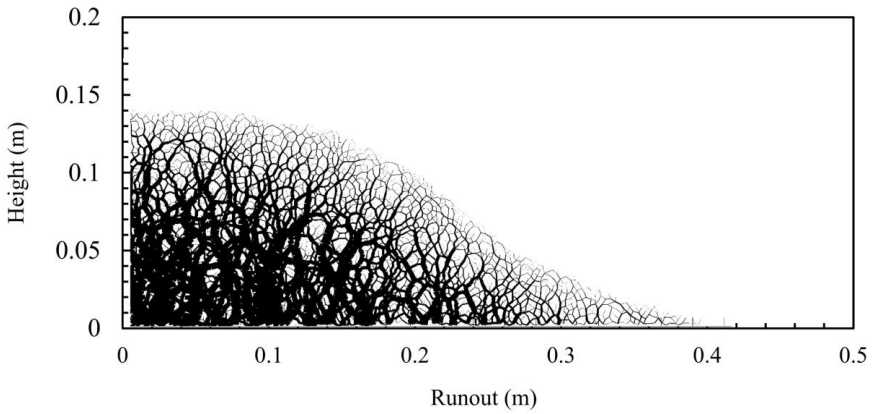


Figure 13. Runout profiles of sand grains due to dam break.



(**a**) velocity



(**b**) force chain

Figure 14. Velocity and force chain profiles at 50 s.

The model heights were then increased to 0.28, 0.448, 0.632, 1.264, 2.528, 3.576, 5.056, and 7.152, while the aspect ratio H/L of the specimen remained as 1. The corresponding particle numbers increased to 8000, 20,000, 40,000, 160,000, 640,000, 1,280,000, 2,560,000, and 5,000,000. The average runtimes in 10,000 incremental steps were compared between the CPU sequential, CPU parallel and GPU parallel simulations (see Table 4). The speedups by the CPU parallelisation are relatively stable, varying between 4.4 and 4.8. With the GPU parallelisation, the speedup increases for less than 40,000 particles as the GPU cores are not fully loaded, which then stabilises around 60. The average speedup of the GPU parallelisation for the two-dimensional (60) (Figure 15) scenario appears to be lower than that for the three-dimensional ones (84 and 73), which hints that the three-dimensional simulations have higher computational intensity and better parallelisability. Particularly for the largest model with 1,280,000 particles in this study, the average runtime in 10,000 steps

is around 378 s, which means that the GPU parallelised two-dimensional DEM can be used to simulate large-scale problems with better consideration of the real shape and size of the particles. The granular runout with 1,280,000 particles was terminated at 25 s, with the runout morphology shown in Figure 16. The simulation took 5,270,000 incremental steps and a physical runtime of 55 h.

Table 4. Speedups of CPU and GPU parallel simulations in dam break cases (10,000 incremental steps).

Number of Particles	H/d_{50}	CPU Sequential	CPU Parallel (10 Threads)		GPU Parallel (Double-Precision)	
		Runtime (s)	Runtime (s)	Speedup	Runtime (s)	Speedup
4000	67	89	20	4.4	6	15
8000	93	200	45	4.4	8	25
20,000	150	398	88	4.5	7	57
40,000	210	756	164	4.6	12	63
160,000	424	2814	611	4.6	47	60
640,000	848	11,217	2386	4.7	193	58
1,280,000	1,200	22,311	4648	4.8	378	59
2,560,000	1700	46,019	9587	4.8	742	62
5,000,000	2400	90,159	18,783	4.8	1502	60

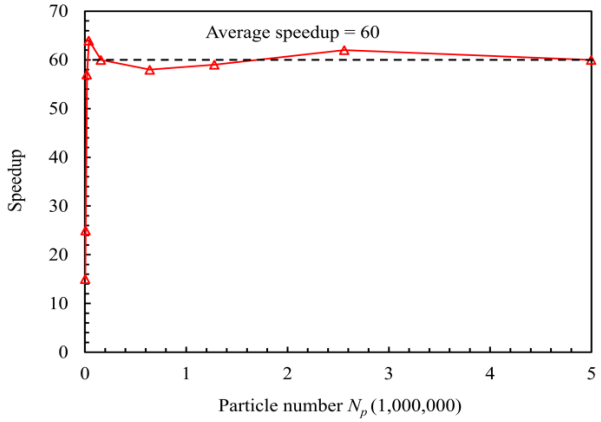


Figure 15. Speedups with different model sizes for dam break runouts.

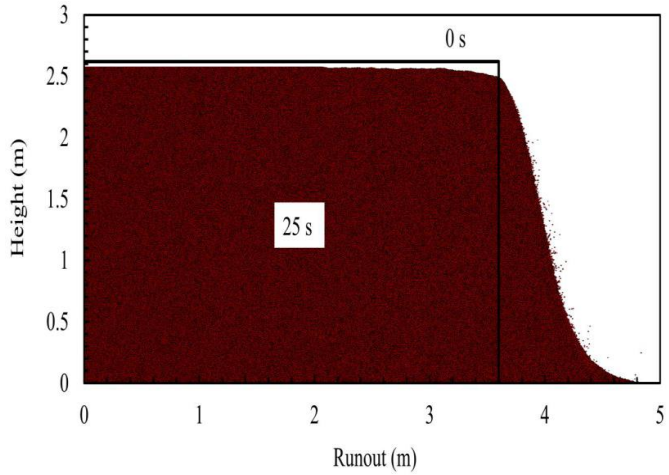


Figure 16. Runout profiles of sand pile with 1,280,000 particles.

4. Conclusions

The discrete element method (DEM), a discontinuum-based method to simulate the interaction between neighbouring particles of granular materials, has a very high computational intensity raised by the meso-scale representation of the macro information. A GPU-based parallel technique was developed to boost the computational efficiency of the DEM and allow a higher realism of the particle shapes. The computations were performed fully on the GPU cores with a platform of the computer unified device architecture (CUDA) rather than on the CPU cores as in the conventional calculations. The potential data race between the computing cores in the parallelisation was tackled by establishing a contact pair list with a hybrid technique, which was further optimised with better parallelisability.

Compared with the sequential CPU simulations, significant speedups of 84, 73 and 60, were obtained in the benchmark cases, triaxial test of a sand specimen, cone penetration test and granular flow due to a dam break, respectively. Higher speedups were gained for the parallelisation of the procedure "Update kinematic state of particles" (~150) than that of the "Calculate contact forces" procedure (~80). The influence of the coarse mesh size and empty cells on the parallelisation effect was also investigated. For the three-dimensional analysis, the real-sized sand grains remain unacceptable for quantitative analysis due to the very long runtimes of the whole computation process. Therefore, compromising techniques, such as increasing the particles' stiffness, expanding the particle sizes and mass scaling, may remain necessary. In comparison, simulations with a particle size of an order of millimetre were allowable, with a case of 1,280,000 particles completed in 55 h.

Author Contributions: Data curation, Y.D. and D.Y.; Formal analysis, Y.D. and L.C.; Investigation, Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the National Natural Science Foundations of China (Grants No. 51909248 and No. 41877260) and the Open Research Fund of State Key Laboratory of Geomechanics and Geotechnical Engineering, Institute of Rock and Soil Mechanics, Chinese Academy of Sciences (Grant No. Z018011). This work was also supported by the NVIDIA Corporation with the donation of the GPUs Geforce Titan Xp and GeForce Titan V.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: This paper was supported by the National Natural Science Foundations of China (Grants No. 51909248 and No. 41877260) and the Open Research Fund of State Key Laboratory of Geomechanics and Geotechnical Engineering, Institute of Rock and Soil Mechanics, Chinese Academy of Sciences (Grant No. Z018011). This work was also supported by the NVIDIA Corporation with the donation of the GPUs Geforce Titan Xp and GeForce Titan V.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cundall, P.A.; Strack, O.D.L. A discrete numerical model for granular assemblies. *Géotechnique* **1979**, *29*, 47–65. [CrossRef]
2. Hu, Y.; Randolph, M.F. A practical numerical approach for large deformation problems in soil. *Int. J. Numer. Anal. Methods Geomech.* **1998**, *22*, 327–350. [CrossRef]
3. Wang, D.; Randolph, M.F.; White, D.J. A dynamic large deformation finite element method based on mesh regeneration. *Comput. Geotech.* **2013**, *54*, 192–201. [CrossRef]
4. Dong, Y. Reseeding of particles in the material point method for soil-structure interactions. *Comput. Geotech.* **2020**, *127*, 103716. [CrossRef]
5. Soga, K.; Alonso, E.; Yerro, A.; Kumar, K.; Bandara, S. Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. *Géotechnique* **2016**, *66*, 248–273. [CrossRef]
6. Jiang, M.J.; Yu, H.S.; Leroueil, S. A simple and efficient approach to capturing bonding effect in naturally microstructured sands by discrete element method. *Int. J. Numer. Methods Eng.* **2007**, *69*, 1158–1193. [CrossRef]

7. Sufian, A.; Knight, C.; O'Sullivan, C.; Van Wachem, B.; Dini, D. Ability of a pore network model to predict fluid flow and drag in saturated granular materials. *Comput. Geotech.* **2019**, *110*, 344–366. [CrossRef]
8. Yuan, W.H.; Liu, K.; Zhang, W.; Dai, B.; Wang, Y. Dynamic modeling of large deformation slope failure using smoothed particle finite element method. *Landslides* **2020**, *17*, 1591–1603. [CrossRef]
9. Zhu, H.P.; Zhou, Z.Y.; Yang, R.Y.; Yu, A.B. Discrete particle simulation of particulate systems: A review of major applications and findings. *Chem. Eng. Sci.* **2008**, *63*, 5728–5770. [CrossRef]
10. Yazdchi, K.; Srivastava, S.; Luding, S. Micro–macro relations for flow through random arrays of cylinders. *Compos. Part A Appl. Sci. Manuf.* **2012**, *43*, 2007–2020. [CrossRef]
11. Kumar, N.; Luding, S.; Magnanimo, V. Macroscopic model with anisotropy based on micro–macro information. *Acta Mech.* **2014**, *225*, 2319–2343. [CrossRef]
12. Guo, P.J.; Stolle, D.F.E. Lateral pipe-soil interaction in sand with reference to scale effect. *J. Geotech. Geoenviron. Eng.* **2005**, *131*, 338–349. [CrossRef]
13. Jiang, M.J.; Yin, Z.Y. Analysis of stress redistribution in soil and earth pressure on tunnel lining using the discrete element method. *Tunn. Undergr. Space Technol.* **2012**, *32*, 251–259. [CrossRef]
14. Falagush, O.; McDowell, G.R.; Yu, H. Discrete element modeling of cone penetration tests incorporating particle shape and crushing. *Int. J. Geomech.* **2015**, *15*, 04015003. [CrossRef]
15. Shi, D.; Yang, Y.; Deng, Y.; Xue, J. DEM modelling of screw pile penetration in loose granular assemblies considering the effect of drilling velocity ratio. *Granul. Matter* **2019**, *21*, 74. [CrossRef]
16. Peters, J.F.; Jelinek, B.; Goodman, C.; Vahedifard, F.; Mason, G. Large-scale discrete-element modeling for engineering analysis: Case study for the mobility cone penetrometer. *J. Geotech. Geoenviron. Eng.* **2019**, *145*, 04019111. [CrossRef]
17. Gao, M.; Wang, X.; Wu, K.; Pradhana, A.; Sifakis, E.; Yuksel, C.; Jiang, C. GPU optimization of material point methods. *ACM Trans. Graph. (TOG)* **2018**, *37*, 1–12. [CrossRef]
18. Coop, M.R.; Sorensen, K.K.; Bodas Freitas, T.; Georgoutsos, G. Particle breakage during shearing of a carbonate sand. *Géotechnique* **2004**, *54*, 157–163. [CrossRef]
19. Zhang, W.C.; Wang, J.F.; Jiang, M.J. DEM-aided discovery of the relationship between energy dissipation and shear band formation considering the effects of particle rolling resistance. *J. Geotech. Geoenviron. Eng.* **2013**, *139*, 1512–1527. [CrossRef]
20. Singh, A.; Magnanimo, V.; Saitoh, K.; Luding, S. Effect of cohesion on shear banding in quasistatic granular materials. *Phys. Rev. E* **2014**, *90*, 022202. [CrossRef]
21. Cui, L.; Sheng, Q.; Dong, Y.; Xie, M. Unified elasto-plastic analysis of rock mass supported with fully grouted bolts for deep tunnels. *Int. J. Numer. Anal. Methods Geomech.* **2022**, *46*, 247–271. [CrossRef]
22. Johnson, D.H.; Vahedifard, F.; Jelinek, B.; Peters, J.F. Micromechanical modeling of discontinuous shear thickening in granular media-fluid suspension. *J. Rheol.* **2017**, *61*, 265–277. [CrossRef]
23. Kesseler, M.; Heller, V.; Turnbull, B. A laboratory-numerical approach for modelling scale effects in dry granular slides. *Landslides* **2018**, *15*, 2145–2159. [CrossRef]
24. Fan, N.; Jiang, J.; Dong, Y.; Guo, L.; Song, L. Approach for evaluating instantaneous impact forces during submarine slide-pipeline interaction considering the inertial action. *Ocean Eng.* **2022**, *245*, 110466. [CrossRef]
25. Potyondy, D.O.; Cundall, P.A. A bonded-particle model for rock. *Int. J. Rock Mech. Min. Sci.* **2004**, *41*, 1329–1364. [CrossRef]
26. O'Sullivan, C. Particle-based discrete element modeling: Geomechanics perspective. *Int. J. Geomech.* **2011**, *11*, 449–464. [CrossRef]
27. Coetzee, C.J. Calibration of the discrete element method. *Powder Technol.* **2017**, *310*, 104–142. [CrossRef]
28. Cleary, P.W. Industrial particle flow modelling using discrete element method. *Eng. Comput.* **2009**, *26*, 698–743. [CrossRef]
29. Furuichi, M.; Nishiura, D.; Kuwano, O.; Bauville, A.; Hori, T.; Sakaguchi, H. Arcuate stress state in accretionary prisms from real-scale numerical sandbox experiments. *Sci. Rep.* **2018**, *8*, 8685. [CrossRef]
30. Kuhn, M.R.; Bagi, K. Specimen size effect in discrete element simulations of granular assemblies. *J. Eng. Mech.* **2009**, *135*, 485–492. [CrossRef]
31. Thakur, S.C.; Ooi, J.Y.; Ahmadian, H. Scaling of discrete element model parameters for cohesionless and cohesive solid. *Powder Technol.* **2016**, *293*, 130–137. [CrossRef]
32. Lommen, S.; Mohajeri, M.; Lodewijks, G.; Schott, D. DEM particle upscaling for large-scale bulk handling equipment and material interaction. *Powder Technol.* **2019**, *352*, 273–282. [CrossRef]
33. Berger, K.J.; Hrenya, C.M. Challenges of DEM. II: Wide particle size distributions. *Powder Technol.* **2014**, *264*, 627–633. [CrossRef]
34. Cheng, K.; Wang, Y.; Yang, Q.; Mo, Y.; Guo, Y. Determination of microscopic parameters of quartz sand through tri-axial test using the discrete element method. *Comput. Geotech.* **2017**, *92*, 22–40. [CrossRef]
35. Gao, P.; Liu, Z.; Shi, D.; Wang, F. Investigation on perforation drilling to mitigate punch-through potential in sand overlying soft clay. *Appl. Ocean Res.* **2022**, *119*, 103026. [CrossRef]
36. Ferellec, J.F.; McDowell, G.R. A method to model realistic particle shape and inertia in DEM. *Granul. Matter* **2010**, *12*, 459–467. [CrossRef]
37. Zheng, J.; Hryciw, R.D. An image based clump library for DEM simulations. *Granul. Matter* **2017**, *19*, 1–15. [CrossRef]
38. Hopkins, M. Polyhedra faster than spheres? *Eng. Comput.* **2014**, *31*, 567–583. [CrossRef]
39. Kuang, D.; Long, Z.; Guo, R.; Yu, P. Experimental and numerical investigation on size effect on crushing behaviors of single calcareous sand particles. *Mar. Georesources Geotechnol.* **2020**, *39*, 1–12. [CrossRef]

40. Wang, J.; Schweizer, D.; Liu, Q.; Su, A.; Hu, X.; Blum, P. Three-dimensional landslide evolution model at the Yangtze River. *Eng. Geol.* **2021**, *292*, 106275. [CrossRef]
41. Yan, Z.; Wilkinson, S.K.; Stitt, E.H.; Marigo, M. Discrete element modelling (DEM) input parameters: Understanding their impact on model predictions using statistical analysis. *Comput. Part. Mech.* **2015**, *2*, 283–299. [CrossRef]
42. Rapaport, D.C. Multi-million particle molecular dynamics I. Design considerations for vector processing. *Comput. Phys. Commun.* **1991**, *62*, 198–216. [CrossRef]
43. Nishiura, D.; Sakaguchi, H. Parallel-vector algorithms for particle simulations on shared-memory multiprocessors. *J. Comput. Phys.* **2011**, *230*, 1923–1938. [CrossRef]
44. Amritkar, A.; Deb, S.; Tafti, D. Efficient parallel CFD-DEM simulations using OpenMP. *J. Comput. Phys.* **2014**, *256*, 501–519. [CrossRef]
45. Yan, B.; Regueiro, R.A. Comparison between pure MPI and hybrid MPI-OpenMP parallelism for Discrete Element Method (DEM) of ellipsoidal and poly-ellipsoidal particles. *Comput. Part. Mech.* **2019**, *6*, 271–295. [CrossRef]
46. NVIDIA. *CUDA Programming Guide, Version 5.5*; NVIDIA: Santa Clara, CA, USA, 2013.
47. Dong, Y.; Wang, D.; Randolph, M.F. A GPU parallel computing strategy for the material point method. *Comput. Geotech.* **2015**, *66*, 31–38. [CrossRef]
48. Dong, Y.; Cui, L.; Zhang, X. Multiple-GPU for three dimensional MPM based on single-root complex. *Int. J. Numer. Methods Eng.* **2021**, *123*, 1481–1504. [CrossRef]
49. Dong, Y.; Grabe, J. Large scale parallelisation of the material point method with multiple GPUs. *Comput. Geotech.* **2018**, *101*, 149–158. [CrossRef]
50. Zhang, W.; Zhong, Z.H.; Peng, C.; Yuan, W.H.; Wu, W. GPU-accelerated smoothed particle finite element method for large deformation analysis in geomechanics. *Comput. Geotech.* **2021**, *129*, 103856. [CrossRef]
51. Govender, N.; Wilke, D.N.; Kok, S.; Els, R. Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs. *J. Comput. Appl. Math.* **2014**, *270*, 386–400. [CrossRef]
52. Nishiura, D.; Matsuo, M.Y.; Sakaguchi, H. ppohDEM: Computational performance for open source code of the discrete element method. *Comput. Phys. Commun.* **2014**, *185*, 1486–1495. [CrossRef]
53. Šmilauer, V.; Catalano, E.; Chareyre, B.; Dorofeenko, S.; Duriez, J.; Gladky, A.; Kozicki, J.; Modenese, C.; Scholtès, L.; Sibille, L.; et al. Yade reference documentation. *Yade Doc.* **2010**, *474*, 1–510.
54. Itasca Consulting Group Inc. PFC manual, version 5.0. Minneapolis. 2016. Available online: https://www.itascacg.com/software/pfc (accessed on 3 July 2021).
55. Lin, X.; Ng, T.T. Contact detection algorithms for three-dimensional ellipsoids in discrete element modelling. *Int. J. Numer. Anal. Methods Geomech.* **1995**, *19*, 653–659. [CrossRef]
56. Munjiza, A.; Andrews, K.R.F. NBS contact detection algorithm for bodies of similar size. *Int. J. Numer. Methods Eng.* **1998**, *43*, 131–149. [CrossRef]
57. Perkins, E.; Williams, J.R. A fast contact detection algorithm insensitive to object sizes. *Eng. Comput.* **2001**, *18*, 48–61. [CrossRef]
58. Nezami, E.G.; Hashash, Y.M.A.; Zhao, D.; Ghaboussi, J. Shortest link method for contact detection in discrete element method. *Int. J. Numer. Anal. Methods Geomech.* **2006**, *30*, 783–801. [CrossRef]
59. Williams, J.R.; O'Connor, R. Discrete Element Simulation and the Contact Problem. *Arch. Comput. Methods Eng.* **1999**, *6*, 279–304. [CrossRef]
60. Whitehead, N.; Fit-Florea, A. Precision & Performance: Floating Point and IEEE754 Compliance for NVIDIA GPUs. 2011. Available online: https://developer.nvidia.com/sites/default/files/akamai/cuda/files/NVIDIA-CUDA-Floating-Point.pdf (accessed on 5 October 2020).
61. Jiang, M.J.; Konrad, J.M.; Leroueil, S. An efficient technique for generating homogeneous specimens for DEM studies. *Comput. Geotech.* **2003**, *30*, 579–597. [CrossRef]
62. Arroyo, M.; Butlanska, J.; Gens, A.; Calvetti, F.; Jamiolkowski, M. Cone penetration tests in a virtual calibration chamber. *Géotechnique* **2011**, *61*, 525–531. [CrossRef]
63. Butlanska, J.; Arroyo, M.; Gens, A.; O'Sullivan, C. Multi-scale analysis of cone penetration test (CPT) in a virtual calibration chamber. *Can. Geotech. J.* **2014**, *51*, 51–66. [CrossRef]
64. Eid, W.K. Scaling Effect in Cone Penetration Testing in Sand. Doctoral Dissertation, Virginia Polytechnic Institute and State University, 1987. Available online: https://vtechworks.lib.vt.edu/handle/10919/49849?show=full (accessed on 3 July 2021).
65. Eslami, A.; Moshfeghi, S.; Molaabasi, H.; Eslami, M.M. Piezocone and Cone Penetration Test (CPTu and CPT) Applications in Foundation Engineering. *Butterworth-Heinemann*. 2019. Available online: https://www.elsevier.com/books/piezocone-and-cone-penetration-test-cptu-and-cpt-applications-in-foundation-engineering/eslami/978-0-08-102766-0 (accessed on 3 July 2021).