Computer Physics Communications 296 (2024) 109066

Contents lists available at ScienceDirect

Computer Physics Communications

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

Computational Physics

Comparing open-source DEM frameworks for simulations of common bulk processes ☆

M. Dosta [a,b], D. Andre [c], V. Angelidakis [d,e], R.A. Caulk [f], M.A. Celigueta [g], B. Chareyre [f], J.-F. Dietiker [h,i], J. Girardot [j], N. Govender [k], C. Hubert [l,m], R. Kobyłka [n], A.F. Moura [o], V. Skorych [a], D.K. Weatherley [p], T. Weinhart [q,*]

[a] Hamburg University of Technology, 21073 Hamburg, Germany
[b] Boehringer Ingelheim Pharma GmbH & Co. KG, Biberach 88400, Germany
[c] University of Limoges, IRCER, UMR CNRS 7315, 87000 Limoges, France
[d] Newcastle University, NE1 7RU, Newcastle Upon Tyne, UK
[e] Friedrich-Alexander-Universität Erlangen-Nürnberg, Institute for Multiscale Simulation, 91058 Erlangen, Germany
[f] Univ. Grenoble Alpes, CNRS, 3SR lab., 38000 Grenoble, France
[g] CIMNE, Campus Nord UPC, Edificio C1, 08034 Barcelona, Spain
[h] National Energy Technology Laboratory, Pittsburgh, PA 15236, USA
[i] Leidos Research Support Team, Pittsburgh, PA 15236-0940, USA
[j] Arts et Metiers Institute of Technology, I2M Bordeaux, UMR CNRS 5295, Talence F-33400, France
[k] Research Center Pharmaceutical Engineering, Graz, Austria
[l] Univ. Polytechnique Hauts-de-France, LAMIH, CNRS, UMR 8201, F-59313 Valenciennes, France
[m] INSA Hauts-de-France, F-59313 Valenciennes, France
[n] Institute of Agrophysics, Polish Academy of Sciences, 20-290 Lublin, Poland
[o] DCS Computing GmbH, Linz, Austria
[p] The University of Queensland, Sustainable Minerals Institute, St. Lucia, Queensland, 4072, Australia
[q] University of Twente, Drienerlolaan 5, 7522 NB Enschede, Netherlands

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.8252892

Keywords:
Open source
Discrete Element Method
Benchmark

ABSTRACT

Multiple software frameworks based on the Discrete Element Method (DEM) are available for simulating granular materials. All of them employ the same principles of explicit time integration, with each time step consisting of three main steps: contact detection, calculation of interactions, and integration of the equations of motion. However, there exist significant algorithmic differences, such as the choice of contact models, particle and wall shapes, and data analysis methods. Further differences can be observed in the practical implementation, including data structures, architecture, parallelization and domain decomposition techniques, user interaction, and the documentation of resources.

This study compares, verifies, and benchmarks nine widely-used software frameworks. Only open-source packages were considered, as these are freely available and their underlying algorithms can be reviewed, edited, and tested. The benchmark consists of three common bulk processes: silo emptying, drum mixing, and particle impact. To keep it simple and comparable, only standard features were used, such as spherical particles and the Hertz-Mindlin model for dry contacts. Scripts for running the benchmarks in each software are provided as a dataset.

1. Introduction

The Discrete Element Method (DEM) is a widely used approach for modelling granular processes in both engineering and nature. The traditional DEM algorithm focuses on rigid, unbreakable, spherical particles and employs force models for both pairwise interactions with other particles and walls, and external body forces. However, various extensions have been proposed since its introduction [1]. This includes models

☆ The review of this paper was arranged by Prof. Andrew Hazel.

* Corresponding author.
E-mail address: t.weinhart@utwente.nl (T. Weinhart).

https://doi.org/10.1016/j.cpc.2023.109066
Received 16 February 2023; Received in revised form 3 December 2023; Accepted 13 December 2023
Available online 19 December 2023

Table 1
Analysed DEM software packages in alphabetical order.

Software	Ref.	Link	Revision
Blaze-DEM	[26]	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00085	V_02_2015
GranOO	[27]	https://www.granoo.org	3.0
Kratos Multiphysics	[28]	https://github.com/KratosMultiphysics/Kratos	9.0
LIGGGHTS-public	[16]	https://github.com/CFDEMproject/LIGGGHTS-public	3.8.0
MercuryDPM	[29]	https://www.mercurydpm.org	1.0.Alpha
MFiX	[30]	https://mfix.netl.doe.gov	22.2.2
MUSEN	[31]	https://github.com/msolids/musen	v1.73.0
Yade	[32]	https://gitlab.com/yade-dev/trunk	2021.01a

for non-spherical particles [2], deformable particle shapes [3,4], particle breakage [5], particles of complex internal structure [6,7], bonding models for simulating fracture [5,8–10], multi-physics effects including thermo-mechanical or chemical processes [11–13], as well as different strategies for coupling DEM with other computational approaches such as fluid and solid dynamic models, using finite volume and finite element models [14–17], Lattice Boltzmann [18], or smoothed particle hydrodynamics [19]. The scope of DEM applications extends from the nanometre scale for modelling diffusion processes [20] to macroscale for modelling of Saturn rings [21].

There are many different software packages that perform DEM simulations. However, despite the wide variety of software and intensive research in the area, there are few studies available that compare calculation results and computational performance between codes. Chung and Ooi [22] proposed benchmark tests of individual particle collisions to verify the contact law implementation in two commercial codes. Holst et al. [23] proposed a benchmark test of silo flow simulations to compare the results of 16 different groups, showing significant differences between the implementations.

This paper was born from an initiative started at the DEM8 conference, which aimed to foster a closer collaboration between the various open-source DEM packages. The primary objectives were twofold: (A) enhance the interoperability and comparability of the software packages and (B) unify the previously distinct developer communities. This paper represents an important step towards those goals. It provides a comparative analysis of eight different DEM codes, by setting up transparent and reproducible benchmarks that can be run in any software. The analysed software packages are listed in Table 1. We focus exclusively on open-source software packages, as they are available to the entire academic community, and can be accessed freely.

A ninth code, ESyS-Particle, was part of the DEM8 initiative, but had to be excluded from the comparison as it lacks an implementation of frictional particle walls. Producing comparable results in ESyS would require a calibration of the ESyS contact model, but this would change the nature of this paper, and thus has not been attempted. Nevertheless, the ESyS developers participated in the development of the benchmarks and implemented them in their software, and are thus included in the author's list.

In section 2, we compare the software packages (including ESyS). In section 3, we verify the implementation of the contact model used in the benchmark simulation. In section 4, we present three case studies: silo emptying, particle mixing and dynamic impact on a particle bed. These benchmarks are common bulk processes that aim to test the basic functionality of different DEM modelling frameworks. For comparability, all case studies assume the use of spherical particles and triangulated walls, whose interaction is described by the viscoelastic Hertz-Mindlin model [24,25]. Depending on the case being solved, the number of modelled particles varies between 25 000 and 100 000.

The full description of benchmarks and their implementation in the respective codes are available in the supporting information, allowing users to repeat the benchmark tests on their own hardware.

2. Software packages

2.1. Methodological aspects

The basic DEM formulation is relatively trivial: the method is based on an iterative procedure consisting of three main steps: contact detection, computation of interaction forces and torques, and integration of the equations of motion. All analysed frameworks follow this approach. However, various modifications and extensions are used to improve computational performance. Consequently, numerous differences exist between the analysed packages:

Time integration: All software packages used explicit time integration: GranOO, LIGGGHTS and MercuryDPM use the Velocity-Verlet algorithm, MUSEN uses leapfrog integration, while Blaze, ESyS, MFiX, Kratos and Yade use symplectic Euler.

Contact detection: In all software packages, contact detection is based on spatial domain decomposition, with further analysis of contacts only for objects located in adjacent volumes. However, different decomposition algorithms are used: GranOO uses a single-level grid, while MercuryDPM and MUSEN apply a hierarchical multi-level grid [33,34]. MUSEN additionally uses Verlet lists with an automatically adjustable cut-off distance [31]. Yade implements a sweep-and-prune algorithm, which filters possible interactions based on axis-aligned bounding boxes [32]. Kratos Multiphysics uses a binary tree. Blaze-DEM implements a single-level grid [26] as well as a bounding-volume hierarchy [35]. ESyS-Particle employs a hybrid of the linked-cell and Verlet-list methods with a fixed single-level grid and contact list updates [36].

Coupling: Many of the software packages provide interfaces for coupling to a continuum phase solver. We distinguish three types of coupling: the interaction of discrete particles with fluids (CFD-DEM) or deformable solids (DEM-FEM), and multiscale coupling, where DEM and continuum models are used in different parts of the domain to describe the same bulk material on different scales:

- CFD-DEM coupling is implemented in LIGGGHTS [16], ESyS-Particle [37] and Yade [12,38] through a coupling to OpenFOAM [39], and in MFiX by coupling to its internal fluid solver. ESyS also couples to the Lattice-Boltzmann solver TCLB [40].
- DEM-FEM coupling is implemented in MercuryDPM [17] using the open-source FEM software oomph-lib [41]. Kratos Multiphysics [42] and Yade [38] couple with internal solvers.
- Multiscale coupling is implemented in MercuryDPM using oomph-lib [17]. In Yade, multiscale and hierarchical multidomain approaches employ a coupling to OOFEM [43].

Particle shape: Each software uses different approaches to model non-spherical particles:

- Bonded particles (all participating codes): objects are represented as a set of spheres connected by stiff bonds [6,9,44–47];
- Multispheres (Kratos Multiphysics, LIGGGHTS, MercuryDPM, Yade): non-spherical particles are modelled as a set of overlapping spheres;

Table 2
Parallelisation strategies available in the different software packages.

Package	Multithreaded CPU (shared memory)	Multiprocessor CPU (distributed memory)	GPU
Blaze-DEM			✓
ESyS-Particle		✓	
GranOO	✓		
Kratos Multiphysics	✓		
LIGGGHTS-public	✓	✓	
MercuryDPM	✓	✓	
MFiX	✓	✓	
MUSEN	✓		✓
Yade	✓	✓	

- Superquadrics (MercuryDPM, LIGGGHTS and MFiX): particle shape is described by the superquadric equation, containing five parameters to determine the shape [48,49];
- Polyhedra (Blaze-DEM, GranOO, Yade): the particle shape is represented as a composition of one or more convex polyhedra [50–53];
- Spheropolyhedra (Yade): the particle shape is the Minkowski sum [54] of a sphere and a polyhedron [3];
- Primitive geometries (GranOO): each object is the Minkowski sum [54] of a sphere and a basic shape, such as a cone, cylinder or box;
- Potential particles (Yade): the particle shape is represented by rounded, irregular, convex non-spherical particles, described by a single potential function [55];
- Level set (Yade): the particle shape is represented by a discrete distance field and surface nodes [56].

There are many more methodological differences between the software packages, such as different available contact laws, wall implementations, and modelling additional particle properties such as temperature and moisture, etc. However, discussing all of those would go beyond the scope of this paper.

2.2. Implementation aspects

Language: Most analysed DEM frameworks are implemented mainly in the C++ programming language; only the MFiX flow solver is written in Fortran 90. ESyS-Particle, GranOO, Kratos Multiphysics and Yade have further implemented Python [57] bindings that can be used for extending the framework, dynamically controlling model physics, harvesting data, and post-processing results.

Parallelisation: DEM calculations are computationally expensive, but can easily be parallelised to reduce simulation time. The software packages under consideration employ three different types of parallelization strategies, as presented in Table 2. To perform multithreading on architectures with shared memory, most software packages use the OpenMP interface; only MUSEN has developed its own multithreading approach, based on the thread pool pattern using the C++ standard library threads. For parallelization on multiprocessor systems with distributed memory, ESyS-Particle, MercuryDPM, MFiX, LIGGGHTS and Yade use the Message Passing Interface (MPI). Furthermore, Blaze-DEM and MUSEN support parallelisation by using Graphics Processing Units (GPU), with double precision. Both frameworks use the CUDA toolkit, which means that GPU-based calculations can only be executed on NVIDIA graphics cards.

User interfaces and visualisation: All software packages provide tools for operations in headless mode, which allow users to manage simulations without a GUI via python or programme-specific scripting languages. Most packages also offer a graphical user interface (GUI): Blaze-DEM, MUSEN, GranOO, MFiX and Yade have a built-in graphical interface for rendering scenes, specifying input parameters, analyzing results, etc, based on the Qt library. MUSEN, GranOO and Yade further use the OpenGL interface to efficiently render large number of objects. The main GUI of Kratos Multiphysics is based on the commercial pre- and post-processor GiD [58]. Furthermore, Yade couples to Matplotlib [59] for live selection and plotting of model quantities.

Data formats: Since DEM simulations are often performed for large particle collectives, efficient data storage plays an important role. In all frameworks, it is possible to save or export data in text format, which significantly simplifies visualisation and data analysis using external tools. Blaze-DEM, ESyS-Particle, GranOO, Kratos Multiphysics, LIGGGHTS, MercuryDPM, MFiX and Yade also support the widely used VTK file format to perform post-processing and visualization in Paraview [60]. To save data as binary files, some serialization libraries are employed, such as protobuf [61] in MUSEN or boost::serialization [62] in Yade and GranOO. The Yade serialization method allows users to save the entire DEM scene, to be reloaded from disk at a later date. ESyS-Particle and MercuryDPM provide similar checkpoint/restart facilities via text-based, human-readable output files.

2.3. Open-source aspects

All participating software packages are open-source. Thus their code base is visible and modifiable by anyone, and users can extend the software as it suits their needs. Newly developed features can then be shared with the community by adding them to the code base, either by committing them via pull request [63] or by contacting the developers.

Code development in an open-source environment offers several benefits, including promoting knowledge sharing, reproducibility and transparency, and avoiding the “black box” problem of commercial software, which does not allow code review. These advantages are particularly beneficial for research activities.

In fact, the present study demonstrates why open-source software is advantageous to scientific advancement. The collaborative and open nature of the present work has yielded better reproducibility of results, and increased access to these codes for the general public. It has also spawned an ongoing discussion on how to better connect the open-source community, use common data formats, etc. These issues are currently being addressed within the ON-DEM COST network.

3. Verification of the contact model

Next, we verify the implementation of the no-slip Hertz-Mindlin contact model [24], which is used in all benchmark cases. This model is implemented in all participating frameworks, though with small differences (Kratos employs a correction of the damping forces proposed by [36]; GranOO uses a non-standard way for computing tangential displacements; Yade employs a different damping coefficient, see equation (4.14) in [64], and modifies the end-of-contact behaviour as in [36] to avoid unphysical attractive normal forces).

Chung and Ooi [22] proposed eight benchmark tests to verify the implementation of the no-slip Hertz-Mindlin contact model; here we reproduce the first four of those test cases:

- Test 1: Elastic normal impact of two identical particles.
- Test 2: Elastic normal impact of a particle with a rigid wall.
- Test 3: Normal particle-wall contact with different restitution coefficients.
- Test 4: Oblique particle-wall contact at different incident angles.

The four other test cases were omitted as they did not add significant extra information. The test setups can be found in [22]; the material properties are given in Table 3. The results of the verification tests are plotted in Fig. 1. For each software, we plot the force-displacement graphs for Test 1-2 and the restitution coefficient for Test 3, together with the analytical solutions from [22]. For test 4, Chung and Ooi [22] provided no analytical solution, but analytical lower limits for the rebound angle and post-collision angular velocity, see equations (30) and (31) in [22]. Both limits are derived from the yield limit, $|F_t| \leq \mu F_n$, i.e.

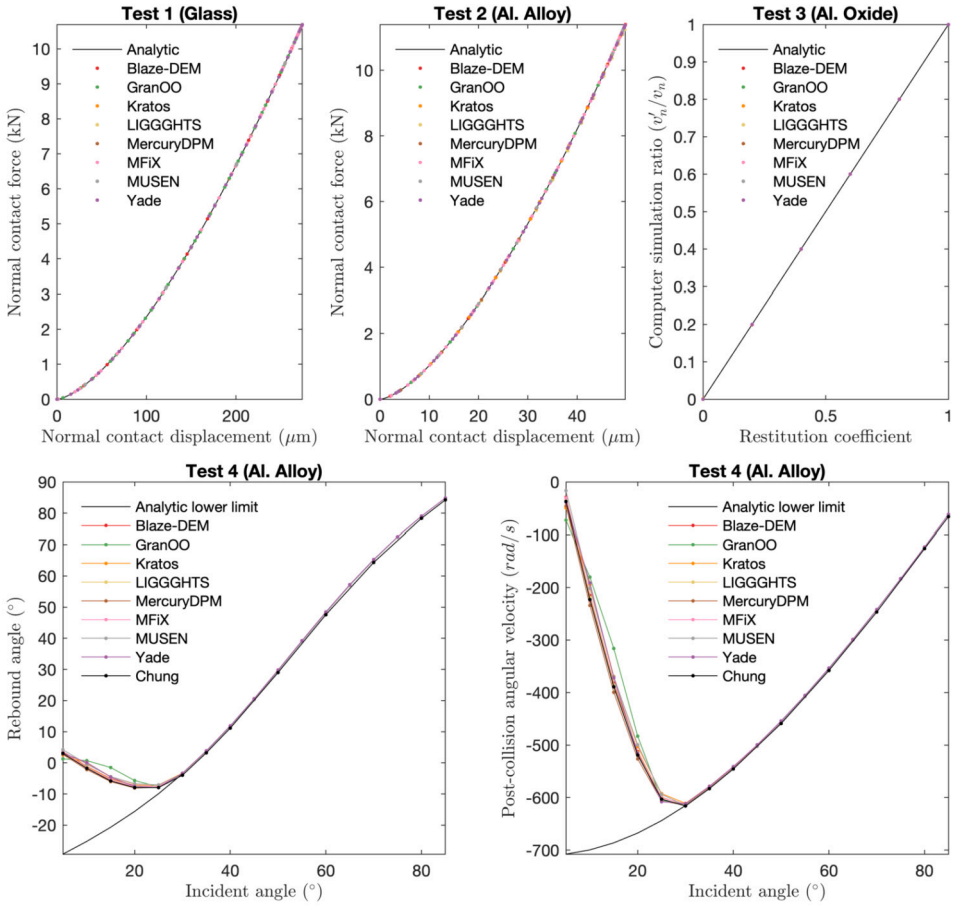

Fig. 1. Verification of the contact model: solutions to test cases of Chung and Ooi [22].

Table 3
Particle and contact properties of the Chung and Ooi verification cases.

Property	Test 1	Test 2	Test 3	Test 4
Young's modulus particle (GPa)	48	70	380	70
Poisson ratio	0.2	0.3	0.23	0.33
Friction coefficient	0.35	0	0	0.092
Restitution coefficient	1	1	varies	0.98
Density (kg/m^3)	2800	2699	4000	2700
Radius (mm)	10	100	2.5	2.5
Velocity (m/s)	±10	0.2	3.9	3.9

the lower limit would be achieved if the tangential force would always satisfy $|F_t| = \mu F_n$.

All codes show perfect agreement with the analytical results for test 1-3, and obey the analytical lower limits for test 4. The solutions to test 4 vary slightly between the different frameworks. However, they all stay close to the DEM solution obtained by Chung and Ooi; only the GranOO results show a slightly different behaviour at very low incident angles, which is likely due to the non-standard way in which it computes tangential displacements.

4. Case studies

Next we present three case studies of standard DEM applications. We used spherical particles and the no-slip Hertz-Mindlin contact model [24], since this contact model was available in all analysed frameworks. No rolling resistance was added since the different frameworks use different rolling resistance models [65].

All walls were represented as triangulated surfaces, and steel was chosen as the material for all geometries. The particles were modelled as arbitrary materials M1 and M2. The DEM-relevant material properties, as well as their interaction properties are listed in Tables 4 and 5. In all case studies, the DEM simulation time step was below 10% of the Rayleigh time step [66]. We checked that lowering the time step did not significantly change the results. For simplicity, frictional forces were set to zero at the initial state. For each case study, a common input file was defined to store the particle and wall positions, and a common output format and frequency was agreed upon (see dataset listed in Data availability section).

Table 4
Material properties used in DEM case studies.

Property	M1	M2	Steel
Density [kg/m^3]	2500	2000	7200
Young's modulus [GPa]	1.0	0.5	210
Poisson ratio [-]	0.2	0.2	0.2

Note that the selection of case studies was primarily guided by the objective of comparing the results of a diverse range of computational codes, hence the study does not include validation experiments. Nevertheless, we will show in the following that the test cases represent physically meaningful behaviour. Furthermore, while the benchmark study of Holst et al. [23] focused on comparing steady-state quantities like the angle of repose and the wall pressure, we focus here on the dynamic behaviour in the early stage of simulation, both because this is rarely tested, and because it allows us to keep the simulations short.

4.1. Case study 1: silo emptying

The first benchmark case simulated the discharge of particles from a cylindrical silo. Two different (steel) silos were used, with small (0.04

Table 5
Interaction properties between different materials.

Interacting materials	Restitution coefficient	Sliding friction
M1 – M1	0.5	0.3
M1 – M2	0.45	0.2
M1 – Steel	0.4	0.2
M2 – M2	0.4	0.4
M2 – Steel	0.4	0.2
Steel – Steel	0.6	0.5

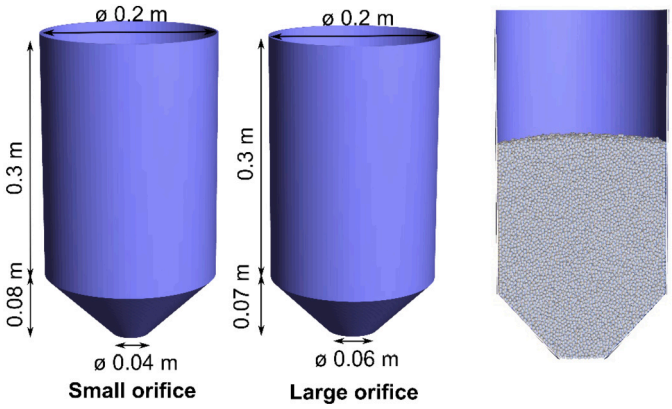

Fig. 2. Case study 1: dimensions of silos with small (left) and large (centre) orifice, and a cross-cut of the silo at the initial state of the case study (right).

m) and large (0.06 m) orifice diameters and external dimensions as shown in Fig. 2. The discretized geometry of each silo was represented by a mesh composed of 8636 and 7632 triangular elements, respectively; only in the case of GranOO native geometry shapes of cylinder and cone were used.

At the start of the simulation, each silo was filled with 100 000 particles of type M1 or M2 with a diameter of 4 mm. The system was relaxed under gravity and the resulting state, shown in Fig. 2, was used as the initial state of the case study. The first 5 seconds of emptying were simulated in each framework. Mass flow was observed in all cases. During emptying, the simulation time steps were 1.5 μs and 2 μs for Case M1 and Case M2, respectively. All DEM software packages recorded the number of remaining particles in the silo with respect to time. These results were sampled every 0.1 seconds.

4.2. Case study 2: drum mixer

The motion of particles in a rotating drum mixer was simulated in the second case study. As shown in Fig. 3, a cylindrical drum made out of steel with an inner diameter of 20 cm and an inner depth of 6 cm was filled with a bimodal particle size distribution. In the first stage, 8000 particles of material M2 of diameter 4 mm were placed on the bottom of the apparatus. Afterwards, 30 000 particles of material M1 with diameter 2 mm were generated on top of the M2 particles, and the entire system was relaxed under gravity. The resulting state, shown in Fig. 3a, was used as the initial state for the case study. Then the drum was rotated around its centre in counterclockwise direction with a rotational velocity of 2 rad/s (only MFiX rotated the gravity direction instead of the drum).

During rotation, the number of M1 and M2 particles located in Zone 1 and Zone 2 (Fig. 3b) with respect to time were collected by all software packages. These quantities enable rough estimates of the dynamic angle of repose and mixing/segregation effects occurring in the drum. The case study was performed with a time step of 0.8 μs for a duration of 5 s. All software saved zone data quantities with a frequency of 0.1 s.

4.3. Case study 3: particle impact

The impact and penetration of a steel particle with a diameter of 20 mm on a static particle bed was modelled in the third case study. This case study included three different particle beds consisting of 25 000, 50 000 or 100 000 M1 particles with a diameter of 2 mm (Fig. 4). All particle beds were settled under the influence of gravity. The steel particle impactor had an initial downwards velocity of 5 m/s and its initial position was 15 cm above the bottom plate. The total duration of the particle impact simulations was 0.1 s and the time-dependent vertical position of the impactor was measured as the result of this case study. The simulation time step was 0.5 μs, and the impactor position was recorded every 1 ms.

5. Results

The simulation results described below were computed using either:

1. A single threaded CPU (GranOO);
2. Multithreaded CPUs (Kratos, LIGGGHTS, MercuryDPM, MFiX, Yade);
3. GPUs (Blaze-DEM, MUSEN).

We have checked that both the GPU and CPU version of MUSEN produce the same results.

5.1. Simulation results

5.1.1. Case study 1: silo emptying

Fig. 5 shows snapshots from all simulations of the large-orifice silo filled with M1 particles at $t = 2$, when about half the particles have exited the silo. While the positions of individual particles differ in each simulation, the flow behaviour seems similar, except for the case of GranOO, which shows a flat bulk surface instead of a conical shape. Note that some simulations deleted particles as soon as they exited the silo orifice, whereas others deleted outflowing particles a bit below the orifice. To account for these differences, only the particles above the orifice were considered for the analysis.

To obtain a quantitative measure of comparison on the particle scale, we plot a histogram of all three components of the particle velocity for each code. Fig. 6 shows these histograms for M1 particles for the large-orifice silo at $t = 2$. The histograms show that all simulations constitute a mass flow, as all particles are in motion ($v_z < 0$). As with the snapshots we observe good agreement between the software packages; only the GranOO data shows significant differences. A current investigation (not shown) indicates that the differences in velocity distribution and surface profile are due to the use of perfect primitive geometries instead of faceted ones. Thus, even if the shapes are geometrically close, slight variations in the dimensions introduce changes in the repulsion forces with the wall.

For a macroscale comparison, we plot the number particles remaining in each silo in Fig. 7. For all software, the number of remaining particles decreases almost linearly over time. This means that the flow rate is constant, which is in good agreement with experimental observations for large grains [8]. For both the small- and large-orifice silo, the flow rates vary by less than 5% between the software packages. Note that there is no consistent trend which programme produces the lowest or highest flow rates; hence we cannot attribute the variations to a specific difference in the algorithms.

The flow rate within the large-orifice silo surpasses that of the small orifice silo by a factor of 3.10 ± 0.06, averaged over all software packages. This can be compared with the Beverloo law [67], which states that for mass-flow silos the flow rate $\dot{m}$ depends on the orifice diameter D and the particle diameter d as follows, $\dot{m} \propto (D - kd)^5/2$, where k is a fitting parameter in a range of $1 < k < 2$ [68], depending on the particle and hopper properties. The observed increase in flow rate corresponds

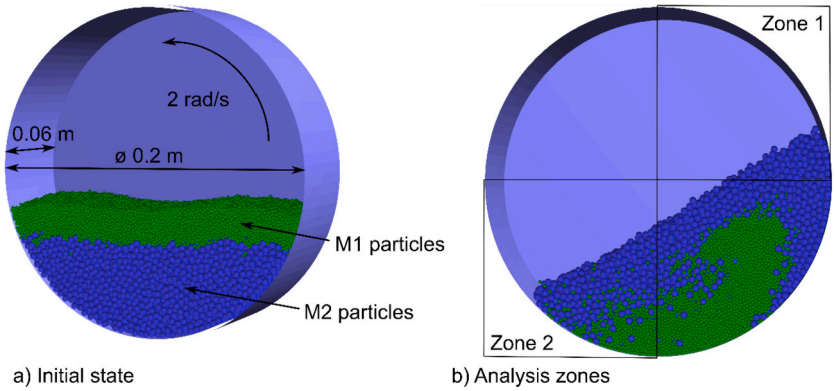

Fig. 3. Case study 2: particle dynamics in a rotating drum. a) initial state; b) after 3 seconds of rotation.

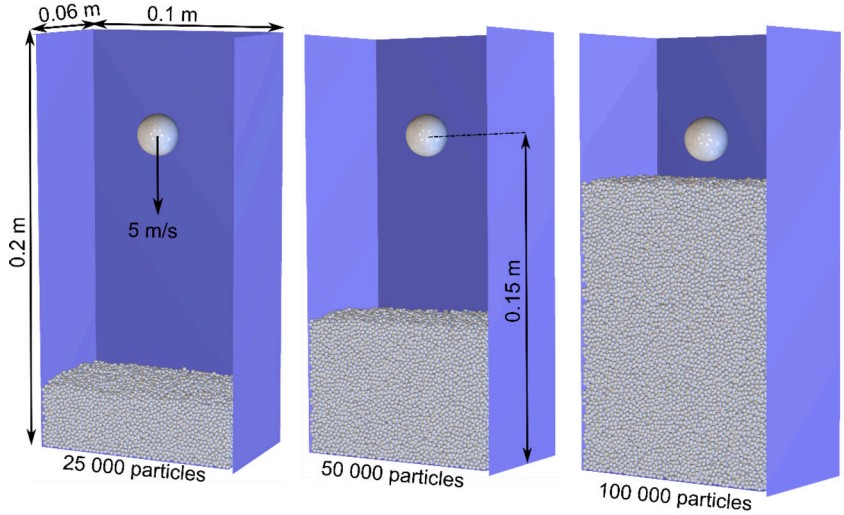

Fig. 4. Case study 3: initial state of the impact test with a varied number of bed particles.

to $k = 1.26$ which is close to the data measured. This alignment with the Beverloo law reinforces the connection between the observed phenomenon and its theoretical underpinnings.

5.2. Case study 2: drum mixer

Fig. 8 shows the time-dependent change in the number of M1 particles located in Zone 2 (see Fig. 3). The results for M2 particles and particles in Zone 1 can be found in Appendix A. The oscillating behaviour is due to the initial segregation of the material. At some intervals, Zone 2 is occupied only by M2 particles, and the number of M1 particles drops to zero. The amplitude of the oscillation in the particle numbers is decreasing with time as the particles get better mixed. All codes show similar results; only the results of MFiX slightly differ, which might be due to them rotating gravity instead of the drum.

To show how the particles segregate by size, we compute the centre-of-mass of all M1 particles as $(x^{\mathrm{M1}}, y^{\mathrm{M1}}, z^{\mathrm{M1}}) = \sum_{i \in \mathcal{M}_1} (x_i, y_i, z_i)/|\mathcal{M}_1|$ where $\mathcal{M}_1$ is the set of all M1 particles and $|\mathcal{M}_1|$ denotes the number of particles in $\mathcal{M}_1$. We further compute the centre-of-mass of all M2 particles, defined similarly. The results are shown in Fig. 9. As expected, the smaller M1 particles aggregate in the centre of the particle bulk, thus their centre-of-mass of the M1 particles varies less over time than the centre-of-mass of the M2 particles. Again, all codes (except MFiX, see above) show good agreement.

For interested readers we also provide snapshots of the final state for each software in Appendix A.

5.3. Case study 3: particle impact

The simulation results of the third case study, the impact of a steel ball on the bed material, are shown in Fig. 10. The diagrams show the relative vertical position of the steel particle for the first 0.1 seconds for different number of bed particles.

For 25 000 bed particles (Fig. 10 top), the bed is so shallow that the steel particle contacts the bottom of the container, after approximately 30 ms. All software packages produce similar results during the penetration phase. However, the results differ slightly during the rebound phase ($> 30\,ms$). This behaviour is likely due to the sensitivity of the rebound behaviour on the positions of individual bed particles: i.e. when the steel particle hits the bottom wall, it rebounds very differently depending on the number and location of bed particles trapped between the steel particle and the wall. Note that this cannot be avoided: slight differences in the algorithm cause differences in the position of individual particles, even though the exact same initial conditions are used for each run. The differences can be minuscule, such as the order of thread execution. Thus, even the same code run twice can produce slightly different results. Therefore, we have simulated the 25 000 particle case for ten different initial conditions, which only differ in the way the particles are placed. For each software, we then average over the ten different trajectories and plot the results in Fig. 11. As expected, deviations are observed, but are within one standard deviation from the mean result, thus statistically indifferent. Only in Yade the rebound is systematically lower; the cause of this difference is yet unknown, despite some investigations.

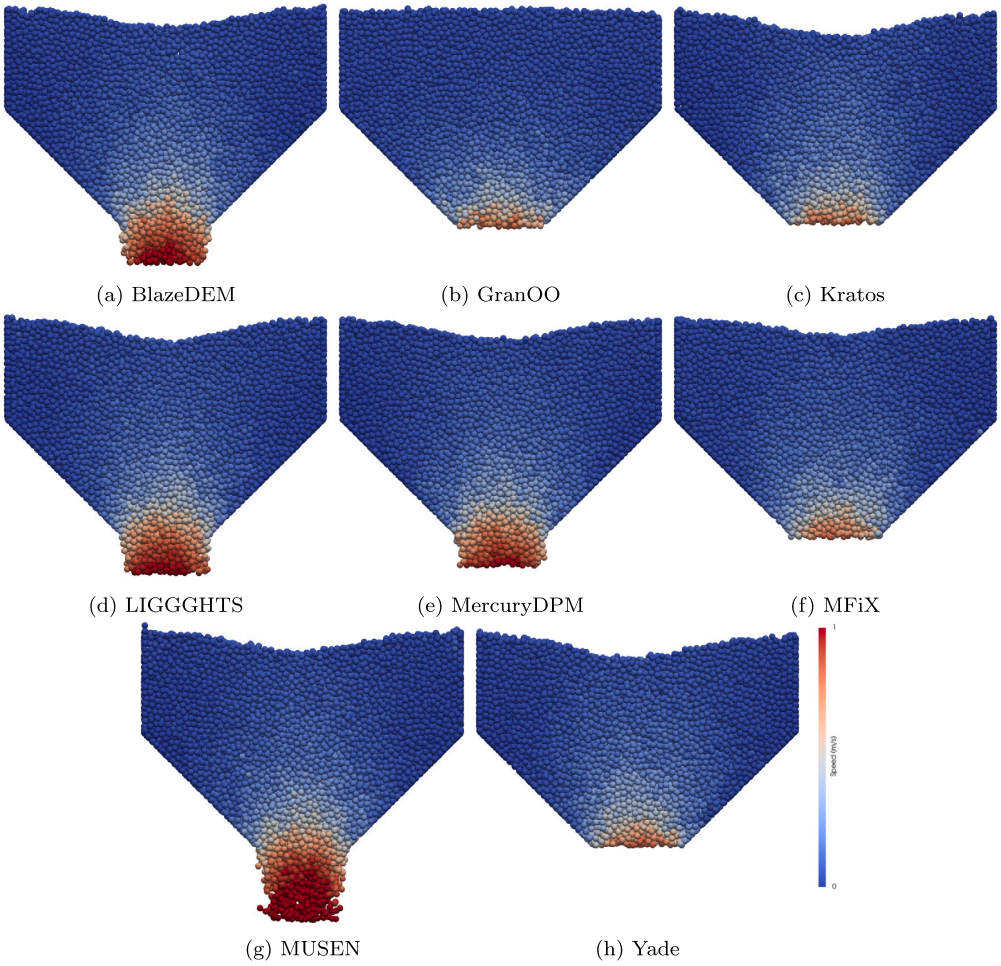

Fig. 5. Case study 1: simulations snapshots of the large-orifice silo with M1 particles at $t = 2$ s, coloured by particle speed. Only particles located within 1 cm of the silo axis in depth direction are shown, thus the selection represents a cross-section through the centre of the silo. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

For 50 000 bed particles (Fig. 10 middle), the rebound is much weaker as more energy is dissipated by the particle contacts. There is also a bigger layer of particles between the steel particle and the bottom of the container, thus the difference between the software packages during the rebound phase is smaller. For 100 000 bed particles (Fig. 10 bottom), there is no rebound. Again, all software packages produce similar results, whereas only Kratos shows a slightly deeper penetration depth.

5.4. Performance analysis

A performance analysis was carried out by running simulations in all frameworks on the same personal computer with the following configuration:

- CPU: AMD Ryzen 9 3900X with reference settings
- GPU: TITAN RTX with reference settings connected over PCIe 3.0 x16
- RAM: 64 GB DDR4 @ 2800 MT/s
- OS: Ubuntu 20.04.2 LTS

To allow the reader to judge the relative merits, we will list the approximate cost, power usage, and specifications of the CPU and GPU: As of August 2023, the list price for the CPU is 499 EUR, for the GPU it is 2499 EUR. The Thermal Design Power of the CPU is 105 watts, for the GPU it is around 280 watts. The Ryzen 9 3900X is a 12-core, 24-thread processor with a base clock of 3.8 GHz. The TITAN RTX is a high-end graphics card featuring 4,608 CUDA cores and a base clock of 1,350 MHz.

For each case study, the time needed to perform simulations in single-threaded and multi-threaded mode on the CPU as well as on the GPU was measured. Since some of the software packages do not support multithreading mode or do not have GPU parallelization, each specific execution mode was tested only for suitable software packages. It should further be noted that in the scope of this work no performance analysis on multi-processor systems (with distributed memory) was performed, which could lead to significant speed-up, especially for large setups consisting of more than 10^5 particles.

The computation times for each case study are listed in Table 6. For Case 1, only the statistics for the silo with the large-orifice filled with M1 particles is shown. During the tests, it was observed that the computational performances of software packages depend strongly on the type of the problem being solved and on the hardware configuration. The configuration can influence not only the average calculation time but also changes the order of which software needed more or less computational time. Also, the optimal number of threads was observed to depend on the case study, the number of modelled objects, the software and the hardware. Considering all these factors, only average times as well as the range are provided in Table 6. The full table of results is available in Appendix B.

Three main conclusions can be drawn regarding the performance of the analysed software:

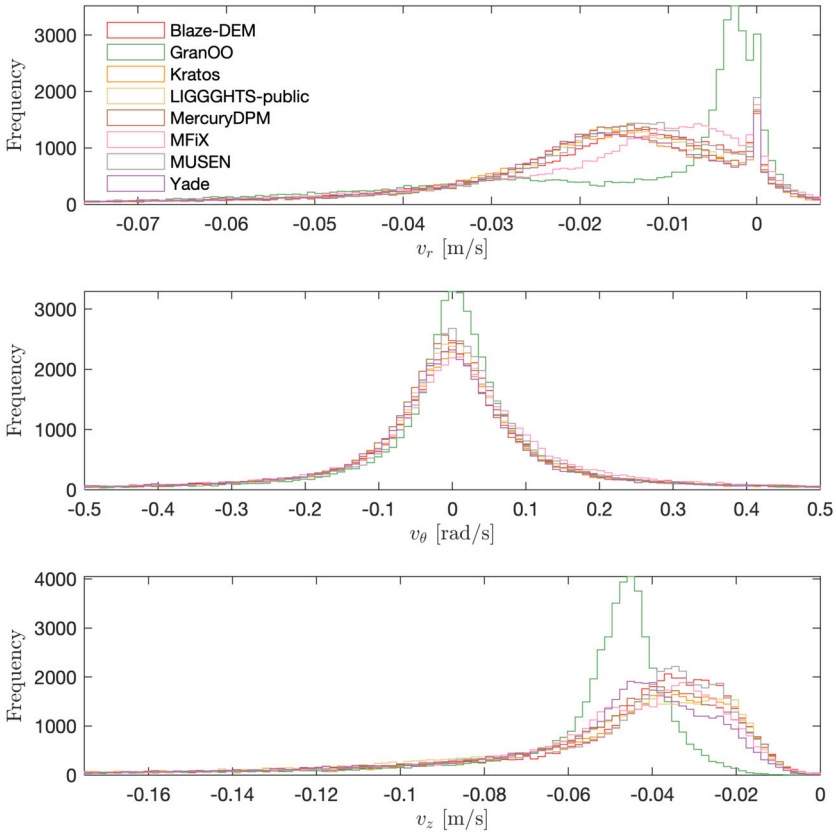

Fig. 6. Case study 1: histogram of radial, angular and vertical velocity components in large-orifice silo with M1 particles at $t = 2$ s.

Table 6
Average, minimum and maximum computational time in minutes for three different execution modes.

Case	Single-threaded CPU		Multi-threaded CPU		GPU	
	Avg	[Min,Max]	Avg	[Min,Max]	Avg	[Min,Max]
Case 1	4267	[1456, 6864]	1408	[454, 3277]	22.5	[17, 28.1]
Case 2	4440	[1344, 8856]	1837	[438, 6120]	35.1	[30.7, 39.6]
Case 3 25 K	96	[30, 233]	41	[9, 122]	1.3	[1.2, 1.4]
Case 3 50 K	313	[71, 992]	96	[18, 333]	2.1	[2, 2.3]
Case 3 100 K	923	[152, 3540]	242	[51, 903]	3.7	[2.9, 4.5]

- There are significant deviations in calculation times between the software packages. E.g., using multithreaded execution on CPU, the computation times varied by a factor from 5 to 18, depending on the case. There are two main reasons: Firstly, inefficient parallelization or implementation of certain parts of the code. Secondly, some of the software packages were originally developed or focused, and therefore more optimized, for specific types of applications.
- Multithreaded parallelization can significantly improve execution time. For the used hardware configuration (12 cores, 24 threads), some of the software packages yielded a speedup factor of more than seven when parallelised.
- GPU-based computing is faster than the more common CPU-based calculations. GPU compute times are an order of magnitude less than the fastest calculation times on the CPU in multithreaded mode (for the considered hardware configuration and case studies).

6. Conclusion

In scope of this work, several case studies were proposed to test the basic functionality of several open-source DEM software packages. Four test cases were run to validate the implemented contact models. Three mid-size benchmark case studies were set up, consisting of up to 10^5 spherical particles, which can be efficiently simulated on a personal computer.

The analysis performed for the available open-source DEM simulation frameworks has shown that each software can be used for solving the proposed case studies. It has been shown that simulating the same initial setup leads to similar results in almost all cases. The deviations in the simulation results obtained for some packages have been attributed to differences in the implementation of contact models, especially in the treatment of tangential forces in particle-wall interactions, and to the sensitivity of results in the penetration test cases. Furthermore, the surface triangulation can have significant influence on the simulation results. Therefore, results obtained from the packages where the analytical primitives such as cylinder or cone are used, can deviate from systems where the triangulated surfaces are employed.

The performance of the simulation frameworks showed significant deviations, with calculation times varying by more than a factor of 15 on multithreaded CPUs. This is likely caused by differences in parallelisation strategies as well as in the efficiency of the implemented algorithms. Furthermore, the study has shown that running DEM codes on the GPU can reduce computational time by an order of magnitude compared to the fastest multithreaded simulations on the CPU.

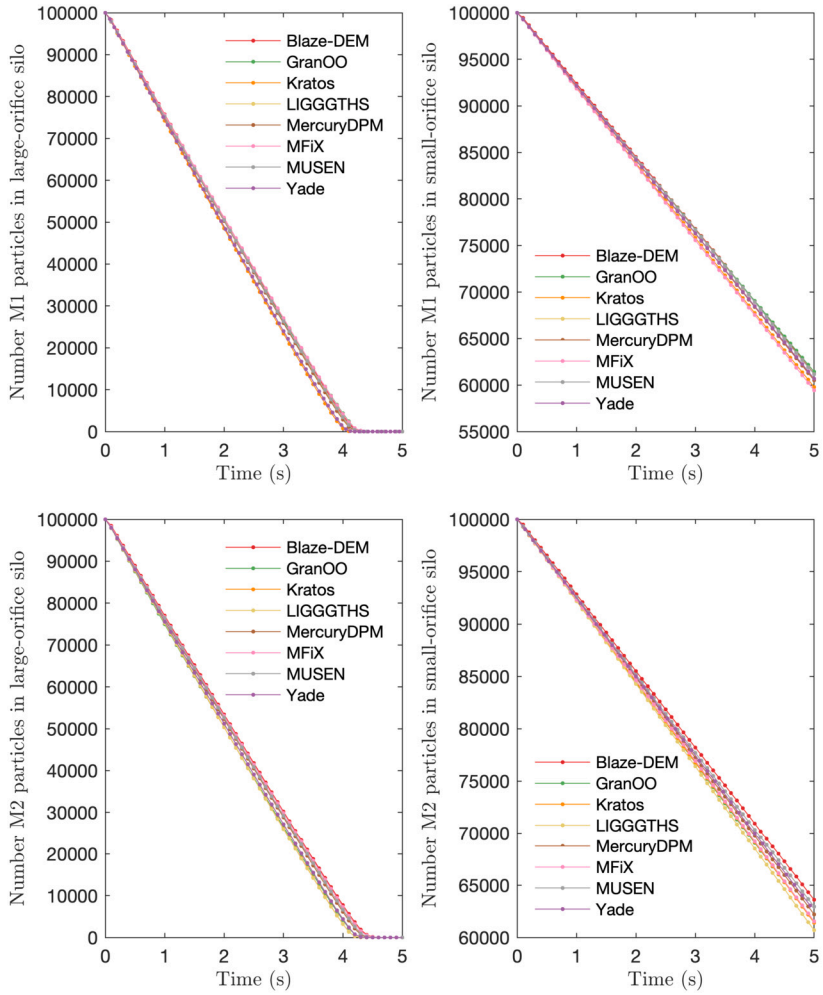

Fig. 7. Case study 1: the number of particles remaining in the silo over time for the large (*top left*) and small (*top right*) orifice filled with M1 particles and for the large (*bottom left*) and small (*bottom right*) orifice filled with M2 particles.

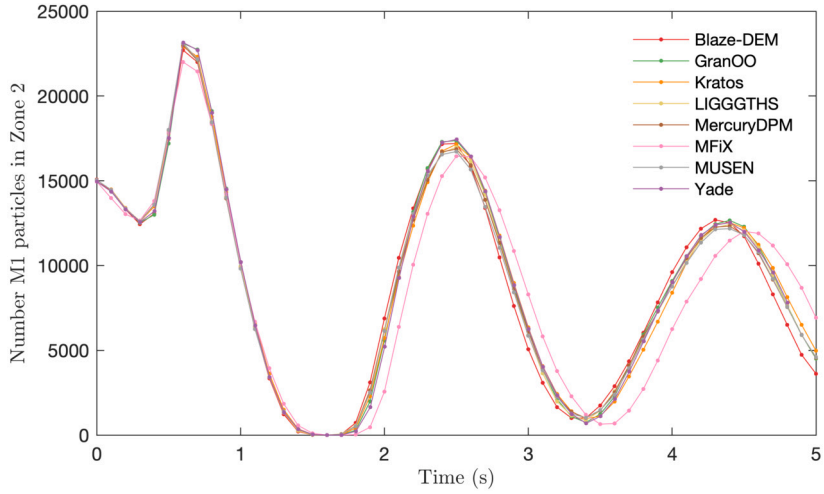

Fig. 8. Case study 2: the number of small M1 particles in Zone 2 over time.

The benchmarks are fully documented and individual scripts are provided as a dataset. This allows for checking agreement between the software frameworks, estimate the efficiency of new codes, and reproducing historic results. Furthermore, they can be used to check currently available commercial products.

CRediT authorship contribution statement

M. Dosta: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Supervision, Validation, Visualization, Writing – original draft, Resources. **D. Andre:** Conceptualization, Investigation,

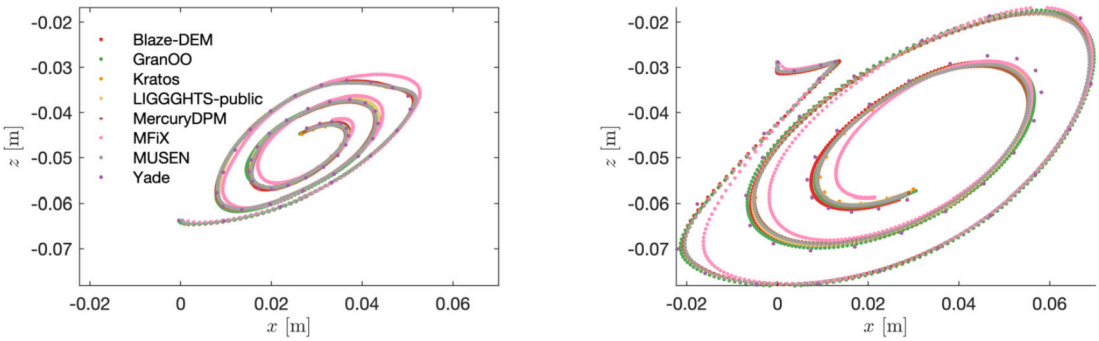

Fig. 9. Case study 2: the centre of mass of the small M1 particles (left) and the centre of mass of the large M2 particles (right) over time.

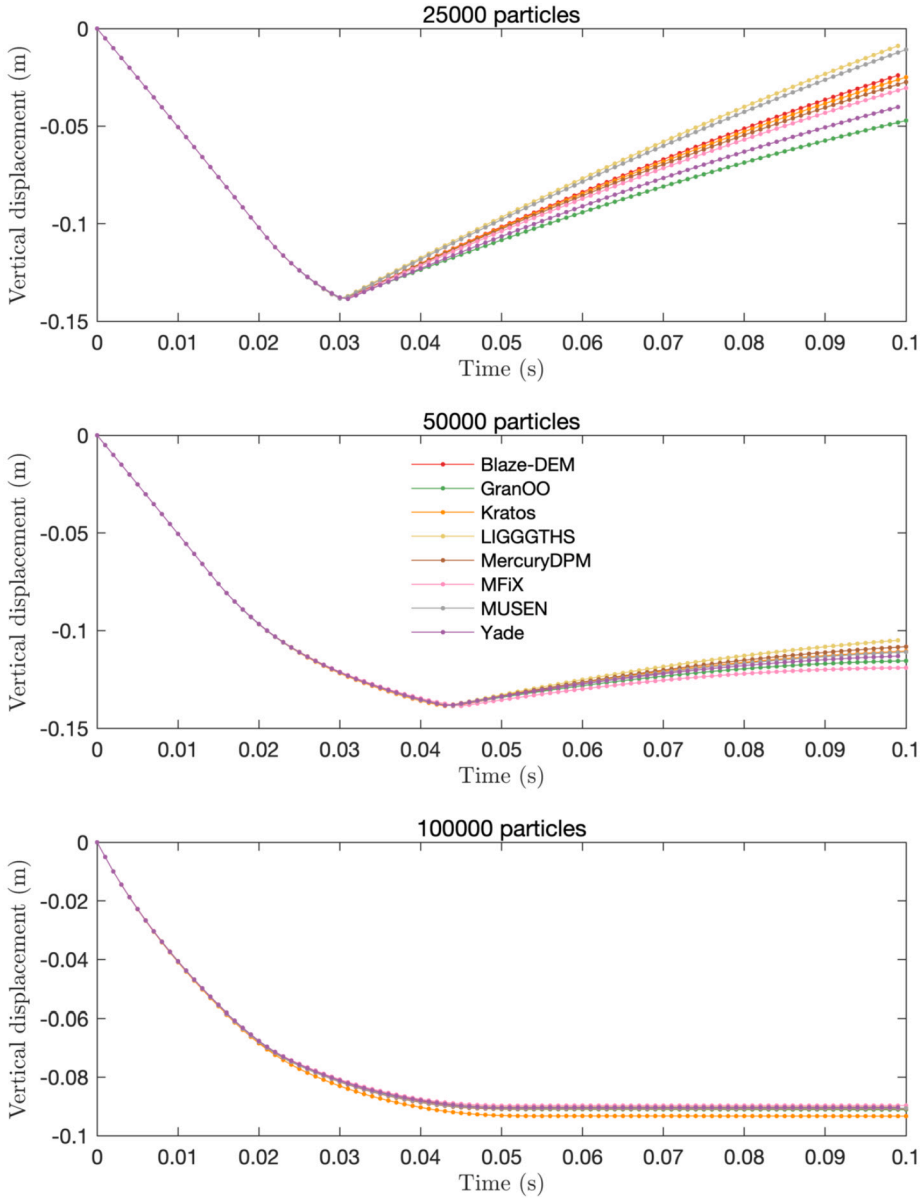

Fig. 10. Case study 3: vertical displacement of the steel ball with 25 000 (top), 50 000 (middle) and 100 000 (bottom) particles of bed material.

Writing – review & editing. **V. Angelidakis:** Conceptualization, Investigation, Writing – review & editing. **R.A. Caulk:** Conceptualization, Investigation, Writing – review & editing. **M.A. Celigueta:** Conceptualization, Investigation, Writing – review & editing. **B. Chareyre:** Conceptualization, Investigation, Writing – review & editing. **J.-F. Dietiker:** Conceptualization, Investigation, Writing – review & editing. **J. Girardot:** Conceptualization, Investigation, Writing – review & editing. **N. Govender:** Conceptualization, Investigation, Writing – review & editing. **C. Hubert:** Conceptualization, Investigation, Writing – review & editing. **R. Kobyłka:** Conceptualization, Investigation, Writing – re-

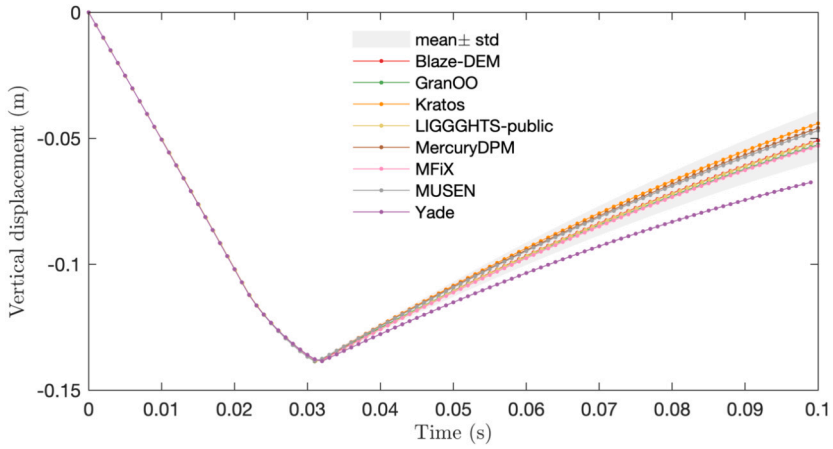

Fig. 11. Case study 3: vertical displacement of the steel ball with 25 000 particles of bed materials, averaged over ten different initial conditions for each software. Shaded area in the background shows the region of one standard deviation around the mean value of displacement, averaged over all 70 cases (excluding the Yade data).

view & editing. **A.F. Moura:** Conceptualization, Investigation, Writing – review & editing. **V. Skorych:** Conceptualization, Formal analysis, Investigation, Validation, Visualization, Writing – review & editing, Data curation, Resources. **D.K. Weatherley:** Conceptualization, Investigation, Writing – review & editing. **T. Weinhart:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Supervision, Validation, Visualization, Writing – review & editing.

Declaration of competing interest

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

Data availability

The scripts for running the benchmarks in each software are available as a dataset at https://doi.org/10.5281/zenodo.8252892, so the reader can perform the test on their own hardware, and compare the results and performance with other codes that are not part of this study.

Acknowledgements

The developers of MUSEN acknowledge the financial support of the German Research Foundation (DFG) via project 418788750 (DO 2026/6-1). The developers of Blaze-DEM acknowledge NVIDIA (Applied Research Accelerator Program) for the donation of GPUS. The developers of MercuryDPM acknowledge the support of the Dutch Research Council (NWO), projects 16604 and 15050.

Appendix A. Supplementary plots

For completeness, we present the Case 2 results omitted in the main paper in Fig. A.12, as well as snapshots of the final state of the drum cases for each software in Fig. A.13.

Appendix B. Individual results of the benchmark cases

In Table B.7, we present simulation times for each case and software package.

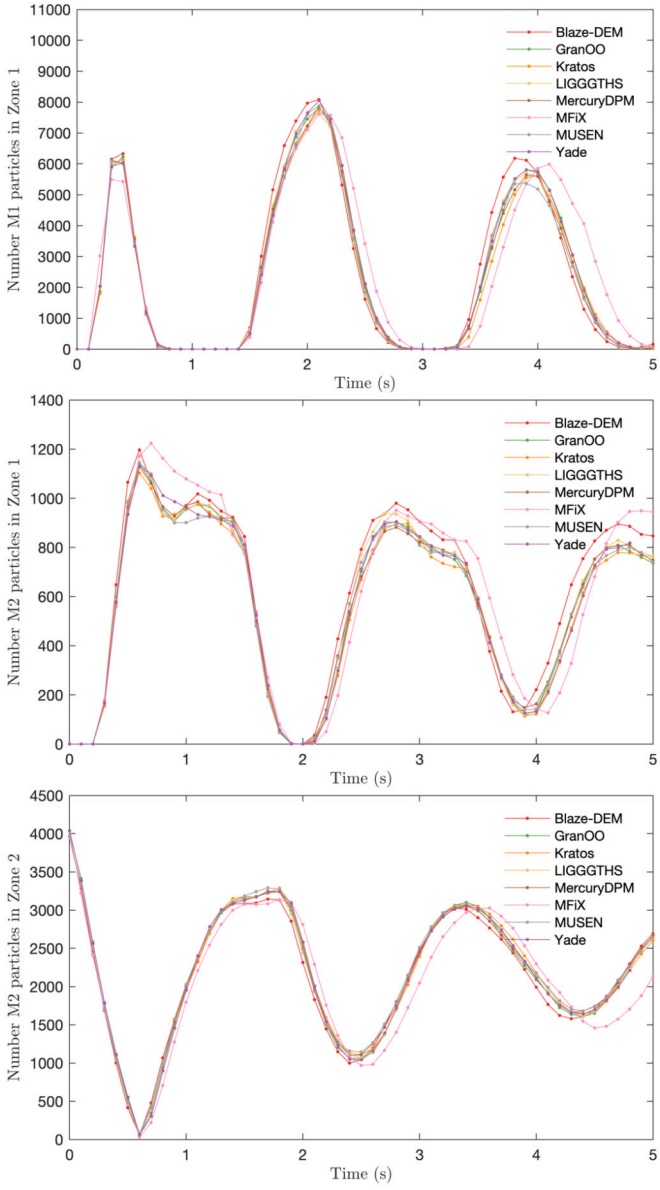

Fig. A.12. Case study 2: the number of M1 particles in Zone 1 (*top*), M2 particles in Zone 1 (*middle*) and Zone 2 (*bottom*) over time.

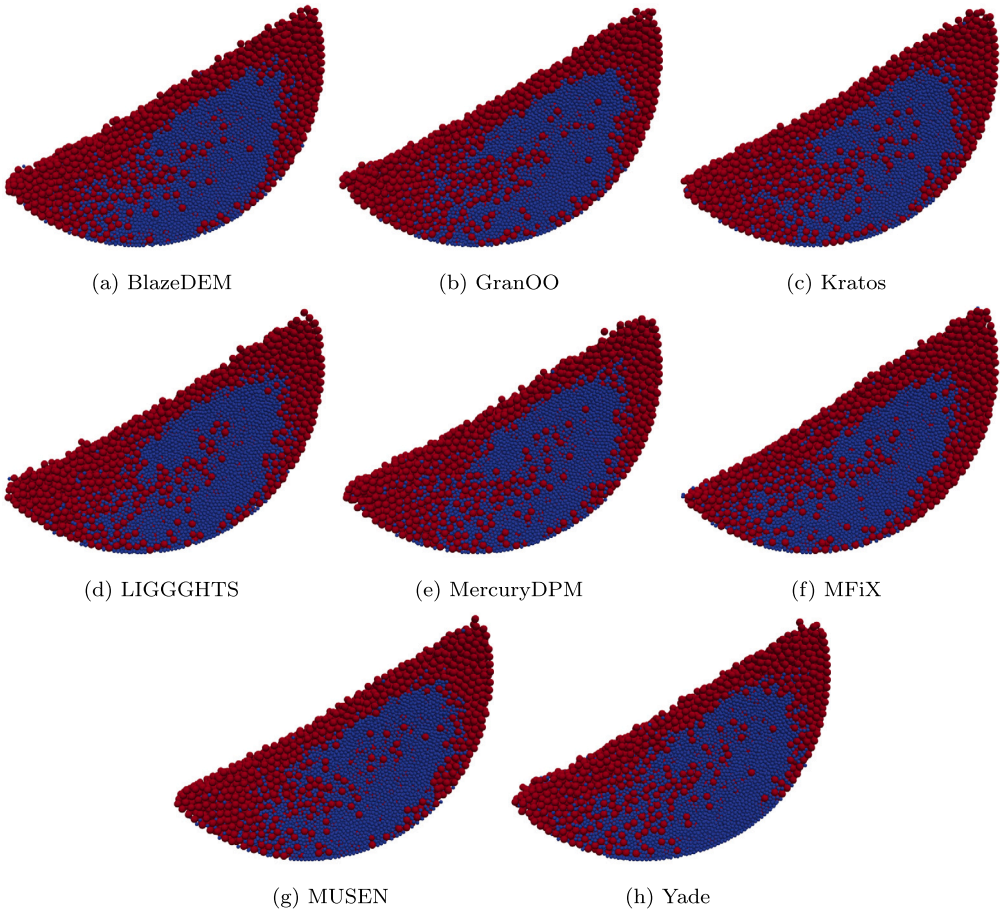

Fig. A.13. Case study 2: simulation snapshots of the final state of the drum simulation.

Table B.7
Simulation time in minutes.

	Single-threaded						
	GranOO	Kratos	LIGGGHTS	MercuryDPM	MFiX	MUSEN	Yade
Case 1	4400	27830	1701	6864	1456	5241	5944
Case 2	4014	64866	1345	8856	1527	3242	7660
Case 3 25 K	87	1362	30	233	58	38	130
Case 3 50 K	208	4306	72	993	171	148	289
Case 3 100 K	472	10856	153	3540	422	378	576

	Multi-threaded					
	Kratos	LIGGGHTS	MercuryDPM	MFiX	MUSEN	Yade
Case 1	3278	1725	1454	454	572	1152
Case 2	6121	448	2235	445	438	1477
Case 3 25 K	122	25	48	13	9	37
Case 3 50 K	333	33	111	32	18	74
Case 3 100 K	903	51	351	75	52	136

	GPU	
	Blaze	MUSEN
Case 1	17	28.2
Case 2	30.7	39.6
Case 3 25 K	1.2	1.5
Case 3 50 K	2	2.3
Case 3 100 K	2.9	4.5

References

[1] P.A. Cundall, O.D. Strack, A discrete numerical model for granular assemblies, Géotechnique 29 (1979) 47–65, https://doi.org/10.1680/GEOT.1979.29.1.47.

[2] G. Lu, J.R. Third, C.R. Müller, Discrete element models for non-spherical particle systems: from theoretical developments to applications, Chem. Eng. Sci. 127 (2015) 425–465, https://doi.org/10.1016/J.CES.2014.11.050.

[3] A. Effeindzourou, B. Chareyre, K. Thoeni, A. Giacomini, F. Kneib, Modelling of deformable structures in the general framework of the discrete element method, Geotext. Geomembr. 44 (2016) 143–156, https://doi.org/10.1016/J.GEOTEXMEM.2015.07.015.

[4] J. Rojek, A. Zubelewicz, N. Madan, S. Nosewicz, The discrete element method with deformable particles, Int. J. Numer. Methods Eng. 114 (2018) 828–860, https://doi.org/10.1002/NME.5767.

[5] G.Y. Liu, W.J. Xu, N. Govender, D.N. Wilke, Simulation of rock fracture process based on GPU-accelerated discrete element method, Powder Technol. 377 (2021) 640–656, https://doi.org/10.1016/J.POWTEC.2020.09.009.

[6] S. Rybczynski, M. Dosta, G. Schaan, M. Ritter, F. Schmidt-Döhl, Numerical study on the mechanical behavior of ultrahigh performance concrete using a three-phase discrete element model, Struct. Concr. 23 (2022) 548–563, https://doi.org/10.1002/SUCO.202000435.

[7] M. Dosta, K. Bistreck, V. Skorych, G.A. Schneider, Mesh-free micromechanical modeling of inverse opal structures, Int. J. Mech. Sci. 204 (2021) 106577, https://doi.org/10.1016/J.IJMECSCI.2021.106577.

[8] A. Janda, I. Zuriguel, D. Maza, Flow rate of particles through apertures obtained from self-similar density and velocity profiles, Phys. Rev. Lett. 108 (2012) 248001, https://doi.org/10.1103/physrevlett.108.248001.

[9] D. André, J. Girardot, C. Hubert, A novel DEM approach for modeling brittle elastic media based on distinct lattice spring model, Comput. Methods Appl. Mech. Eng. 350 (2019) 100–122, https://doi.org/10.1016/J.CMA.2019.03.013.

[10] L. Orefice, J.G. Khinast, Deformable and breakable DEM particle clusters for modelling compression of plastic and brittle porous materials — model and structure properties, Powder Technol. 368 (2020) 90–104, https://doi.org/10.1016/J.POWTEC.2020.04.035.

[11] D. André, B. Levraut, N. Tessier-Doyen, M. Huger, A discrete element thermo-mechanical modelling of diffuse damage induced by thermal expansion mismatch of two-phase materials, Comput. Methods Appl. Mech. Eng. 318 (2017) 898–916, https://doi.org/10.1016/J.CMA.2017.01.029.

[12] R. Caulk, L. Sholtès, M. Krzaczek, B. Chareyre, A pore-scale thermo–hydro-mechanical model for particulate systems, Comput. Methods Appl. Mech. Eng. 372 (2020) 113292, https://doi.org/10.1016/J.CMA.2020.113292.

[13] N. Govender, P.W. Cleary, M. Kiani-Oshtorjani, D.N. Wilke, C.Y. Wu, H. Kureck, The effect of particle shape on the packed bed effective thermal conductivity based on DEM with polyhedral particles on the GPU, Chem. Eng. Sci. 219 (2020) 115584, https://doi.org/10.1016/J.CES.2020.115584.

[14] P. Kieckhefen, S. Pietsch, M. Dosta, S. Heinrich, Possibilities and limits of computational fluid dynamics–discrete element method simulations in process engineering: a review of recent advancements and future trends, Annu. Rev. Chem. Biomol. Eng. 11 (2020) 397–422, https://doi.org/10.1146/annurev-chembioeng-110519-075414.

[15] S. Golshan, R. Sotudeh-Gharebagh, R. Zarghami, N. Mostoufi, B. Blais, J.A. Kuipers, Review and implementation of CFD-DEM applied to chemical process systems, Chem. Eng. Sci. 221 (2020) 115646, https://doi.org/10.1016/J.CES.2020.115646.

[16] C. Kloss, C. Goniva, A. Hager, S. Amberger, S. Pirker, Models, algorithms and validation for opensource DEM and CFD-DEM, Prog. Comput. Fluid Dyn. 12 (2012) 140–152, https://doi.org/10.1504/pcfd.2012.047457.

[17] H. Cheng, A.R. Thornton, S. Luding, A.L. Hazel, T. Weinhart, Concurrent multi-scale modeling of granular materials: role of coarse-graining in FEM-DEM coupling, Comput. Methods Appl. Mech. Eng. 403 (2023) 115651, https://doi.org/10.1016/j.cma.2022.115651.

[18] E.P. Montellà, C. Yuan, B. Chareyre, A. Gens, A hybrid multiphase model based on lattice Boltzmann method direct simulations, arXiv.org, 10.48550/arXiv.1906.04722, 2019.

[19] J.C. Joubert, D.N. Wilke, N. Govender, P. Pizette, U. Tuzun, N.E. Abriak, 3D gradient corrected SPH for fully resolved particle–fluid interactions, Appl. Math. Model. 78 (2020) 816–840, https://doi.org/10.1016/J.APM.2019.09.030.

[20] P.N. Depta, U. Jandt, M. Dosta, A.P. Zeng, S. Heinrich, Toward multiscale modeling of proteins and bioagglomerates: an orientation-sensitive diffusion model for the integration of molecular dynamics and the discrete element method, J. Chem. Inf. Model. 59 (2019) 386–398, https://doi.org/10.1021/ACS.JCIM.8B00613.

[21] E. Ortega Roano, MercuryDPM: from a chip to Saturn, Master's thesis, University of Twente, 2019, https://www2.msm.ctw.utwente.nl/sluding/THESIS/MSc_OrtegaRoano.pdf.

[22] Y.C. Chung, J.Y. Ooi, Benchmark tests for verifying discrete element modelling codes at particle impact level, Granul. Matter 13 (5) (2011) 643–656, https://doi.org/10.1007/S10035-011-0277-0.

[23] J.M.F.G. Holst, J.M. Rotter, J.Y. Ooi, G.H. Rong, Numerical modeling of silo filling. II: discrete element analyses, J. Eng. Mech. 125 (1999) 104–110, https://doi.org/10.1061/(ASCE)0733-9399(1999)125:1(104).

[24] Y. Tsuji, T. Tanaka, T. Ishida, Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe, Powder Technol. 71 (1992) 239–250, https://doi.org/10.1016/0032-5910(92)88030-L.

[25] R.D. Mindlin, Compliance of elastic bodies in contact, J. Appl. Mech. 16 (1949) 259–268, https://doi.org/10.1115/1.4009973.

[26] N. Govender, D.N. Wilke, S. Kok, Blaze-DEMGPU: modular high performance DEM framework for the GPU architecture, SoftwareX 5 (2016) 62–66, https://doi.org/10.1016/J.SOFTX.2016.04.004.

[27] D. André, J.L. Charles, I. Iordanoff, 3D Discrete Element Workbench for Highly Dynamic Thermo-Mechanical Analysis: GranOO, vol. 3, Wiley, 2015, https://doi.org/10.1002/9781119116356.

[28] P. Dadvand, R. Rossi, E. Oñate, An object-oriented environment for developing finite element codes for multi-disciplinary applications, Arch. Comput. Methods Eng. 17 (3) (2010) 253–297, https://doi.org/10.1007/S11831-010-9045-2.

[29] T. Weinhart, L. Orefice, M. Post, M.P. van Schrojenstein Lantman, I.F. Denissen, D.R. Tunuguntla, J.M. Tsang, H. Cheng, M.Y. Shaheen, H. Shi, P. Rapino, E. Grannonio, N. Losacco, J. Barbosa, L. Jing, J.E.A. Naranjo, S. Roy, W.K. den Otter, A.R. Thornton, Fast, flexible particle simulations — an introduction to MercuryDPM, Comput. Phys. Commun. 249 (2020) 107129, https://doi.org/10.1016/J.CPC.2019.107129.

[30] M. Syamlal, W. Rogers, T. O'Brien, MFIX documentation theory guide, Technology 1004 (12 1993), https://doi.org/10.2172/10145548.

[31] M. Dosta, V. Skorych, MUSEN: an open-source framework for GPU-accelerated DEM simulations, SoftwareX 12 (2020) 100618, https://doi.org/10.1016/J.SOFTX.2020.100618.

[32] V. Smilauer, V. Angelidakis, E. Catalano, R. Caulk, B. Chareyre, W. Chèvremont, S. Dorofeenko, J. Duriez, N. Dyck, J. Elias, B. Er, A. Eulitz, A. Gladky, N. Guo, C. Jakob, F. Kneib, J. Kozicki, D. Marzougui, R. Maurin, C. Modenese, G. Pekmezi, L. Scholtès, L. Sibille, J. Stransky, T. Sweijen, K. Thoeni, C. Yuan, Yade Documentation, The Yade Project, 2021, https://doi.org/10.5281/zenodo.5705394.

[33] V. Ogarko, S. Luding, A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects, Comput. Phys. Commun. 183 (2012) 931–936, https://doi.org/10.1016/J.CPC.2011.12.019.

[34] H. Mio, A. Shimosaka, Y. Shirakawa, J. Hidaka, Cell optimization for fast contact detection in the discrete element method algorithm, Adv. Powder Technol. 18 (2007) 441–453, https://doi.org/10.1163/156855207781389519.

[35] R. Lubbe, W.J. Xu, D.N. Wilke, P. Pizette, N. Govender, Analysis of parallel spatial partitioning algorithms for GPU based DEM, Comput. Geotech. 125 (2020) 103708, https://doi.org/10.1016/J.COMPGEO.2020.103708.

[36] T. Pöschel, T. Schwager, Computational granular dynamics: models and algorithms, in: Computational Granular Dynamics: Models and Algorithms, 2005, pp. 1–322, https://doi.org/10.1007/3-540-27720-X/COVER.

[37] T. Zhao, G.T. Houlsby, S. Utili, Investigation of granular batch sedimentation via DEM–CFD coupling, Granul. Matter 16 (2014) 921–932, https://doi.org/10.1007/S10035-014-0534-0/FIGURES/16.

[38] D. Kunhappan, B. Harthong, B. Chareyre, G. Balarac, P.J. Dumont, Numerical modeling of high aspect ratio flexible fibers in inertial flows, Phys. Fluids 29 (2017) 093302, https://doi.org/10.1063/1.5001514.

[39] H.G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Comput. Phys. 12 (6) (1998) 620–631, https://doi.org/10.1063/1.168744.

[40] J.W. McCullough, S.M. Aminossadati, C.R. Leonardi, Transport of particles suspended within a temperature-dependent viscosity fluid using coupled LBM–DEM, Int. J. Heat Mass Transf. 149 (2020) 119159, https://doi.org/10.1016/J.IJHEATMASSTRANSFER.2019.119159.

[41] M. Heil, A.L. Hazel, Oomph-lib - an object-oriented multi-physics finite-element library, Lect. Notes Comput. Sci. Eng. 53 (2006) 19–49, https://doi.org/10.1007/3-540-34596-5_2.

[42] G. Casas-Gonzalez, E. Oñate, R. Rossi, Numerical analysis of particle-laden flows with the finite element method, Ph.D. thesis, University of Cambridge, 2018, https://www.scipedia.com/public/Casas_et_al_2019a.

[43] B. Patzak, OOFEM — an object-oriented simulation tool for advanced modeling of materials and structures, Acta Polytech. 52 (2012) 59–66, https://doi.org/10.14311/1678.

[44] G.Y. Liu, W.J. Xu, Q.C. Sun, N. Govender, Study on the particle breakage of ballast based on a GPU accelerated discrete element method, Geosci. Front. 11 (2020) 461–471, https://doi.org/10.1016/J.GSF.2019.06.006.

[45] A.G. Pagano, V. Magnanimo, T. Weinhart, A. Tarantino, Exploring the micromechanics of non-active clays by way of virtual DEM experiments, Géotechnique 70 (2020) 303–316, https://doi.org/10.1680/JGEOT.18.P.060.

[46] L. Lu, X. Gao, M. Shahnam, W.A. Rogers, Simulations of biomass pyrolysis using glued-sphere CFD-DEM with 3-D intra-particle models, Chem. Eng. J. 419 (2021) 129564, https://doi.org/10.1016/J.CEJ.2021.129564.

[47] L. Scholtès, F.V. Donzé, Modelling progressive failure in fractured rock masses using a 3D discrete element method, Int. J. Rock Mech. Min. Sci. 52 (2012) 18–30, https://doi.org/10.1016/J.IJRMMS.2012.02.009.

[48] I.F.C. Denissen, On segregation in bidisperse granular flows, Ph.D. thesis, University of Twente, 11 2019, https://doi.org/10.3990/1.9789036548762.

[49] X. Gao, J. Yu, R.J. Portal, J.F. Dietiker, M. Shahnam, W.A. Rogers, Development and validation of SuperDEM for non-spherical particulate systems using a superquadric particle method, Particuology 61 (2022) 74–90, https://doi.org/10.1016/J.PARTIC.2020.11.007.

[50] 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.

[51] N. Govender, D.N. Wilke, C.Y. Wu, J. Khinast, P. Pizette, W. Xu, Hopper flow of irregularly shaped particles (non-convex polyhedra): GPU-based DEM simulation and experimental validation, Chem. Eng. Sci. 188 (2018) 34–51, https://doi.org/10.1016/J.CES.2018.05.011.

[52] J. Eliáš, Simulation of railway ballast using crushable polyhedral particles, Powder Technol. 264 (2014) 458–465, https://doi.org/10.1016/J.POWTEC.2014.05.052.

[53] C.W. Boon, G.T. Houlsby, S. Utili, A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method, Comput. Geotech. 44 (2012) 73–82, https://doi.org/10.1016/J.COMPGEO.2012.03.012.

[54] E.G. Gilbert, D.W. Johnson, S.S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space, IEEE J. Robot. Autom. 4 (1988) 193–203, https://doi.org/10.1109/56.2083.

[55] C.W. Boon, G.T. Houlsby, S. Utili, A new contact detection algorithm for three-dimensional non-spherical particles, Powder Technol. 248 (2013) 94–102, https://doi.org/10.1016/J.POWTEC.2012.12.040.
[56] J. Duriez, C. Galusinski, A level set-discrete element method in YADE for numerical, micro-scale, geomechanics with refined grain shapes, Comput. Geosci. 157 (2021) 104936, https://doi.org/10.1016/J.CAGEO.2021.104936.
[57] G. Rossum, Python Reference Manual, CWI, 1995, https://www.narcis.nl/publication/RecordID/oai:cwi.nl:5008.
[58] M.P. de Riera, E.E. Tercero, A.C. Sans, A.M. Ribera, A.M. Bellart, J.G. Vidiella, M.R.P. Rubio, GiD v. 16 user manual, https://www.gidsimulation.com/gid-for-science/support/manuals/, 2018.
[59] J.D. Hunter, Matplotlib: a 2D graphics environment, Comput. Sci. Eng. 9 (2007) 90–95, https://doi.org/10.1109/MCSE.2007.55.
[60] J. Ahrens, B. Geveci, C. Law, ParaView: an end-user tool for large-data visualization, in: C.D. Hansen, C.R. Johnson (Eds.), Visualization Handbook, Butterworth-Heinemann, Burlington, 2005, pp. 717–731, https://doi.org/10.1016/B978-012387582-2/50038-1.
[61] Google Developers, Protocol buffers, https://developers.google.com/protocol-buffers/. (Accessed 1 February 2023).
[62] D. Abrahams, A. Gurtovoy, C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond, Addison Wesley Professional, 2004.
[63] PagerDuty, What is a pull request?, https://www.pagerduty.com/resources/learn/what-is-a-pull-request/. (Accessed 16 August 2023).
[64] C. Thornton, Granular Dynamics, Contact Mechanics and Particle System Simulations: A DEM Study, vol. 24, Springer International Publishing, 2015, https://doi.org/10.1007/978-3-319-18711-2.
[65] J. Ai, J.F. Chen, J.M. Rotter, J.Y. Ooi, Assessment of rolling resistance models in discrete element simulations, Powder Technol. 206 (2011) 269–282, https://doi.org/10.1016/J.POWTEC.2010.09.030.
[66] Y.J. Huang, O.J. Nydal, B. Yao, Time step criterions for nonlinear dense packed granular materials in time-driven method simulations, Powder Technol. 253 (2014) 80–88, https://doi.org/10.1016/J.POWTEC.2013.10.010.
[67] W.A. Beverloo, H.A. Leniger, J. van de Velde, The flow of granular solids through orifices, Chem. Eng. Sci. 15 (3–4) (1961) 260–269, https://doi.org/10.1016/0009-2509(61)85030-6.
[68] R. Nedderman, C. Laohakul, The thickness of the shear zone of flowing granular materials, Powder Technol. 25 (1) (1980) 91–100, https://doi.org/10.1016/0032-5910(80)87014-8.