Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

MUSEN: An open-source framework for GPU-accelerated DEM simulations

Maksym Dosta [*], Vasyl Skorych

Hamburg University of Technology, Institute of Solids Process Engineering and Particle Technology, Denickestrasse 15, 21073 Hamburg, Germany

A R T I C L E   I N F O

Article history:
Received 13 August 2020
Received in revised form 29 September 2020
Accepted 30 October 2020

Keywords:
Discrete element method
CPU–GPU computing
Simulation framework
Granular materials

A B S T R A C T

The conceptual design, implementation aspects and main features of an open-source DEM simulation framework MUSEN have been described. MUSEN has been developed for efficient calculations that can be performed on personal computers equipped with general-purpose graphics processing units (GPUs). A very intuitive graphical user interface significantly simplifies usage of the system and reduces learning time. It makes this framework an ideal software tool for educational purposes and for solution of classical problems of solids process engineering. Users without significant experience can easily specify initial scenes, perform simulations and analyse results. In the same time, the modular-based structure of the system allows its extension with new components, calculation algorithms or contact models.

© 2020 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Code metadata

Current code version	*1.65.0*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00025
Code Ocean compute capsule	–
Legal Code License	*BSD-3-Clause*
Code versioning system used	*git*
Software code languages, tools, and services used	*C++, Qt, CUDA, GLSL, zlib, protobuf*
Compilation requirements, operating environments & dependencies	*Windows, Visual Studio 15 2017, Qt 5.12.3 msvc2017_64, Qt Visual Studio Tools for Visual Studio 2017, CUDA 10, Git*
If available Link to developer documentation/manual	https://msolids.net/musen/documentation/
Support email for questions	dosta@tuhh.de

Software metadata

Current software version	*1.65.0*
Permanent link to executables of this version	https://msolids.net/musen/download/
Legal Software License	*BSD-3-Clause*
Computing platforms/Operating Systems	*Microsoft Windows, Ubuntu*
Installation requirements & dependencies	*Visual C++ Redistributable for Visual Studio 2017, CUDA 10*
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://msolids.net/musen/documentation/
Support email for questions	dosta@tuhh.de

1. Motivation and significance

The discrete element method (DEM) is the standard approach for microscale simulation of granular materials. It is applied for a wide spectrum of tasks starting with classical problems related to modelling mechanical behaviour (Zhu et al. 2008 [1]), ending with the advanced applications like simulating sintering processes (Besler et al. 2016 [2]; Dosta et al. 2020 [3]). Discrete element method implies calculating the motion of a large number of small objects (most often spherical particles) by computing Newton's equations for each of them. This approach makes DEM relatively simple and highly flexible, but at the same

[*] Corresponding author.
E-mail address: dosta@tuhh.de (M. Dosta).

time, computationally intensive. More detailed information on the mathematical aspects of the method can be obtained from (Pöschel et al. 2005 [4]; Kruggel-Emden et al. 2007 [5]; Zhu et al. 2007 [6]; O'Sullivan, 2011 [7]). There exist different open-source and proprietary software tools where DEM calculations can be performed. Some of the widely used proprietary systems are: EDEM (EDEM online [8]), PFC3D (PFC online [9]), Rocky DEM (Rocky online [10]), etc. Examples of open source systems include LIGGGHTS (Kloss et al. 2011 [11], LIGGGHTS online [12]), YADE (Kozicki & Donze, 2009 [13], Yade online [14]), Woo (Woo online [15]), Mercury DPM (Weinhart et al. 2020 [16], MercuryDPM online [17]), GranOO (Andrè et al. 2014 [18], GranOO online [19]), MechSys (MechSys online [20]), etc. The general calculation algorithm in all these systems is similar. It usually consists of three basic steps: contact detection, calculation of forces and integration of motion. Moreover, depending on the type of problems to be solved, these steps can be extended by the generation of new particles, their removing, connection to the fluid phase (Norouzi et al. 2016 [21]; Kieckhefen et al. 2020 [22]), etc. However, there are significant differences in these systems, when it comes to implementation aspects and support of various features and extensions. Among them can be distinguished:

- graphical user interface;
- methods for contact detection;
- coupling with other simulation approaches (CFD, FEM, SPH, etc.) and external program packages;
- modelling of non-spherical, composite, deformable and destroyable objects;
- parallelization of calculations;
- storage and analysis of simulation results.

In this contribution, we present an open-source simulation framework MUSEN. Among its distinguishing features may be highlighted:

- intuitive easy-to-use graphical user interface;
- high computational efficiency due to support for calculations on general-purpose graphics processing units (GPGPU);
- efficient multigrid contact detection algorithms;
- cross-platform availability;
- simple installation procedure.

Thanks to these characteristics, MUSEN has high potential to be used for educational and research purposes.

2. Software description

2.1. Methods

The MUSEN framework is based on the soft-sphere discrete element method (DEM). The leap-frog algorithm is used as a time integration scheme. For contact detection, the Verlet list method (Verlet, 1967 [23]) combined with the linked-cell algorithm (Quenterec et al. 1973 [24]) is implemented. The entire simulation volume is discretized with a constant size mesh into a set of smaller cells, and contact detection is performed only between particles situated in adjacent cells. On top of that, a multigrid approach (Mio et al. 2007 [25]) is used for faster processing of particle sets with wide size distributions. Depending on the scatter of particles in size, the simulation domain is simultaneously discretized by several volumetric grids of different sizes. In this case, the contact detection between pairs of large particles as well as between large and small particles is performed on a coarse grid and between pairs of small particles — on a more suitable fine grid. This significantly reduces the number of comparison operations needed to build the Verlet list.

Finally, the Verlet list is populated with contacts that could potentially occur within a specific cut-off distance. This means that it can be used without recalculation for several consecutive calculation steps. Therefore, the selection of the optimal cut-off distance can significantly influence the performance of calculations. On the one hand, large values lead to a higher memory consumption and to larger lists of potential contacts, which should be analysed in each time step. Short distances, on the other hand, require very frequent recalculations of lists. Therefore, a strategy for automatic adjustment of the cut-off distance has been developed and implemented in MUSEN. The average execution time of one DEM simulation step is stored and analysed as the main performance characteristic. After a specific number of steps, the cut-off distance is recalculated based on the performance statistics collected for the previous cut-off distances. This allows to consider the actual load of computational resources (CPU and GPU) and choose an optimal value for the current conditions.

MUSEN offers different contact models, which are grouped into the following four categories:

- Particle–particle: interaction between particles;
- Particle–wall: interaction between particles and walls;
- Solid bonds: bonded-particle model (BPM) to simulate aggregates or heterogeneous materials;
- External field: to treat any type of external forces acting on particles.

MUSEN provides various built-in contact models, such as the widely used Hertz–Mindlin (Mindlin & Deresiewicz, 1953 [26]; Tsuji et al. 1992 [27]) or the JKR model (Johnson et al. 1971 [28]), as well as more specialized models to describe specific processes, such as sintering (Parhami & McMeeking, 1953 [29]). The contact radius of particles can deviate from its physical radius that allows to consider non-contact interactions, such as van der Waals or electrostatic forces.

To describe geometries and to calculate particle–wall contacts, a triangulated surface mesh is used. Here, the point of contact between the particle and the triangle can be located on the facet of the triangle, on its edge or on its vertex. Because contact detection is performed separately for each individual triangle, multiple redundant contacts with the geometry can be found for a single particle. To eliminate this, a uniqueness check is applied (Su et al. 2011 [30]). In MUSEN, there is no contact detection between walls, however, it is possible to simulate uniaxial motion of geometries caused by particle–wall stresses. For each geometry, one can specify the direction of free motion. In this case, the translational motion of the geometry will be calculated based on the overall force acting on it and its mass specified by the user.

It is possible to apply cubic periodic boundary conditions (PBC) that are active for particles, solid bonds, and walls. PBC can be activated not only for simulation itself, but also for the generation of initial particle packings or the formation of solid bonds. This allows to create scenes with consistent initial conditions (Dosta et al. 2020 [3]).

Non-spherical or irregular-shaped objects are represented in MUSEN using BPM as a set of smaller spheres connected with solid bonds (Dosta et al. 2019 [31], Kozhar et al. 2016 [32]. This allows to perform simulation of non-spherical, deformable or destroyable objects.

2.2. Software architecture

A simplified representation of the general architecture of the MUSEN system is shown in Fig. 1. The central role is played by *System Structure*, which contains all the fixed and time-dependent information about the process being simulated. Each scene is described by a set of *Physical Objects*, such as spheres, solid bonds, walls etc. The whole time-dependent information from *System Structure* can be saved to or loaded from the files in binary or text format. To cope with large amounts of data, the automatic caching on the hard drive is done at runtime. To perform various modification operations with files (for example, to remove specific time points from a file or to combine multiple files into one), *File Tools* are available.

The system has a cross-platform graphical and a command-line user interface. The command-line interface allows to run calculations in batch mode and to perform operations of data extraction and results analysis. The GUI also allows to visualize data and finely modify *System Structure*.

The DEM simulations can be performed either on CPU or on GPU. In both cases, light-weighted *Simplified Scene* is generated from *System Structure* and DEM simulations are executed with corresponding *Simulator*. During simulation, various types of contact models for particle–particle, particle–wall, external force field, etc. can be used.

There exist *Auxiliary Components* that simplify generation of scenes:

- *Package Generator*: forms packings of particles into a specified volume;
- *Bonds Generator*: generates bonds between primary particles;
- *Objects Generator*: generates new objects during simulation;
- *Results Analyser*: analyses simulation results during post-processing.

There are three different types of databases, which significantly simplify the reuse of previously defined entities. They are stored as separate files and contain information about different geometries, agglomerates and materials.

Most of components shown in Fig. 1 are compiled into static libraries and linked directly to the program. However, it is possible to create contact models of any type independently of the modelling system, implementing the proposed interfaces and using only a few source files. Fig. 2 schematically shows the interface for the particle–particle contact model. The user can provide an implementation for CPU (CalculatePPForce), GPU (CalculatePPForceGPU) or for both architectures at once. To enable such external models to be used, they are compiled into dynamic libraries and linked to the system at runtime.

2.3. Implementational aspects and software functionalities

MUSEN is written in C++ language using the object-oriented programming paradigm with intensive use of the standard template library and the features of the C++14 standard. The system consists of a set of classes grouped into several subcategories, like interaction models, Qt dialogues, databases, etc.

The simulation results are stored as a single binary file. Data serialization is performed using the Protocol Buffers [33] library. The use of this library, on the one hand, makes it easy to change and extend data structures, on the other hand, provides compatibility for various hardware configurations and operating systems. Additionally, Protobuf reduces the size of the stored data using the DEFLATE compression algorithm (Deutsch, 1996 [34]) from zlib library [35]. A more detailed description of the saving module and extended methods for data saving can be found in Dranyshnikov et al. (2019) [36].

The cross-platform graphical user interface (GUI) is implemented using the Qt library [37]. Using the GUI, the user can create, modify and visualize simulation scenes, specify simulation parameters and define all program settings. The main program window is shown in Fig. 3.

For fast and efficient 3D visualization of a large number of objects that may be present on the scene, GPU-accelerated rendering is used. The implementation utilizes the OpenGL API [38] via the Qt OpenGL module [39], which provides a platform-independent wrapper around the OS-specific graphical interfaces. To improve performance, the visualization of each object is described by means of vertex and fragment shaders written in the GLSL language. Moreover, the user can modify the representation of particles by uploading new or using built-in images.

Calculations in MUSEN are parallelized for multicore CPUs and for general-purpose GPUs. Parallelization for CPU is done according to the shared memory model using the software design pattern *Thread Pool* based on *std::thread* class from the C++ standard library. GPU calculations are performed using the CUDA computing platform, which allows scenes of millions of discrete objects to be simulated in a reasonable time (Dosta et al. 2020 [3]). All operations related to the integration of motion are performed in MUSEN with double precision.

In the case of GPU computing, a hybrid approach is used, where the contact detection is performed on the CPU, and the calculation of forces and integration of motion is performed on the GPU. The Verlet list calculated on the CPU contains not only currently active contacts, but a set of all potential contacts which can occur within a specific period of time determined by the cut-off distance. Thus, there is no need to transfer data between the CPU and GPU after each step, which significantly increases the calculations efficiency, but increases the memory requirements of the GPU. In each time step, the actual coordinates of the particles from the list of potential contacts are evaluated on the GPU to create a shorter list of currently active contacts. Contact models are calculated only for this list.

The data structure in the GPU memory, which is implemented as set of separate vectors, is schematically shown in Fig. 4. To utilize the features of coalesced data access and, as a result, to reduce the number of data access operations, the particles are sorted according to their coordinates. For this purpose, Morton space filling curve in 3 dimensions is used (Morton, 1966 [40]). The forces and moments calculated in the contact models are directly added to the particles using atomic CUDA operations, which avoids the use of a temporary buffer and, as a consequence, two redundant write and read operations. It should be noted, that due to the nondeterministic order of atomic operations and the limited precision of floating-point operations, slight deviations in the simulation results may occur.

Microsoft Visual C++ 2017 is used to compile and build MUSEN for Windows platform and cmake with g++ to compile it for Linux. The MUSEN installation package for Windows is assembled using the Inno Setup [41] utility.

3. Illustrative examples

To illustrate the applicability of the MUSEN framework, two simulation studies are shown below (Figs. 5 and 6). In the first example, the dynamics of two types of particles in a mixer equipped with vertical mixing blades is analysed. The apparatus is filled with approximately 3.87 million particles with a diameter of 1 mm. All properties of the particles are the same except for their colour. In Fig. 5, the initial ideally segregated state and the rearrangement of particles after 3 and 15 rotations are shown.

Table 1
Calculation time using CPU and hybrid CPU–GPU execution mode.

	CPU [h]	CPU–GPU [h]
Mixing: for 1 s of process time	212	9
Bending test: until breakage	40	2.05

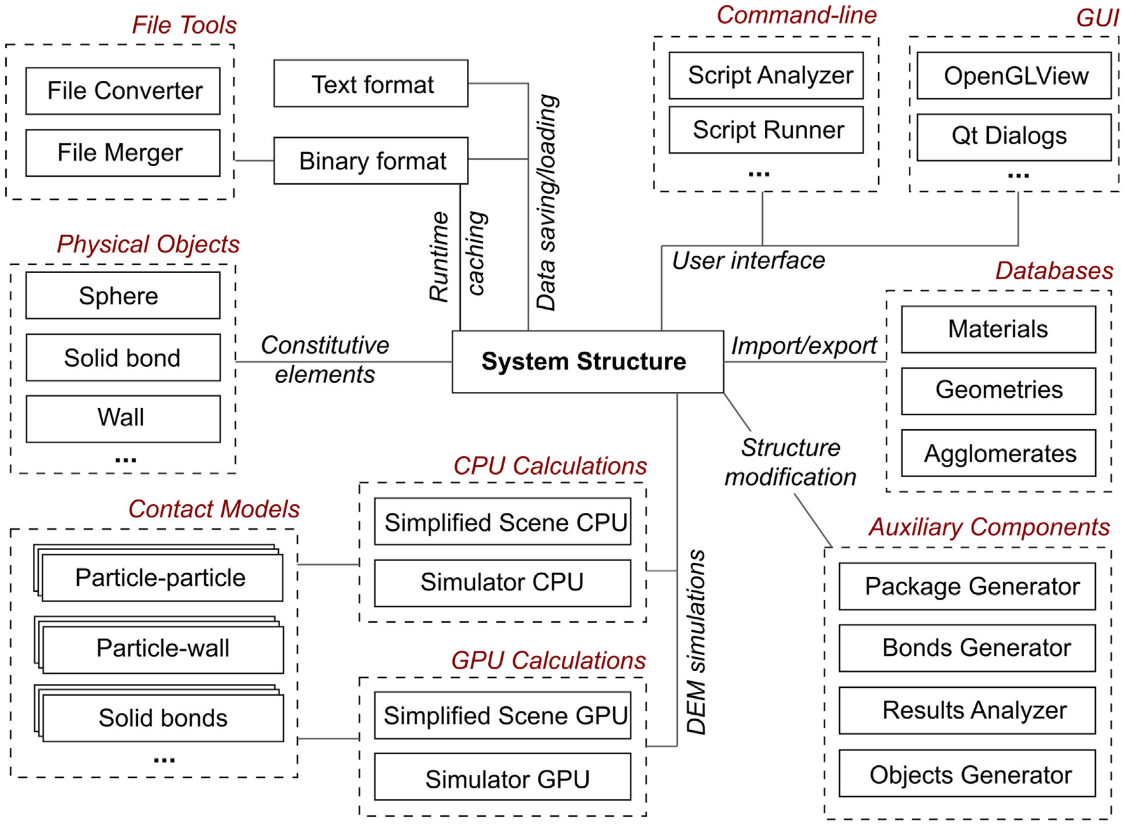


Fig. 1. Schematic representation of the main components of the MUSEN framework.

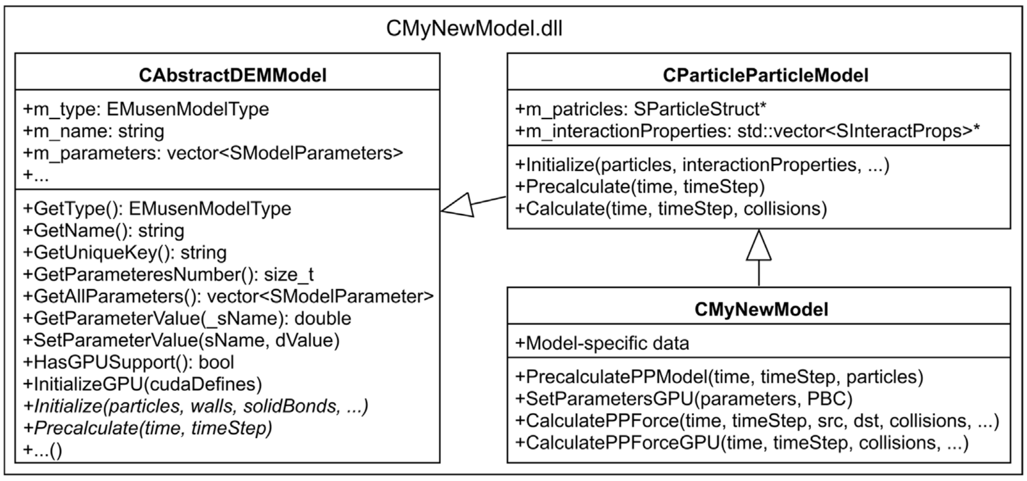
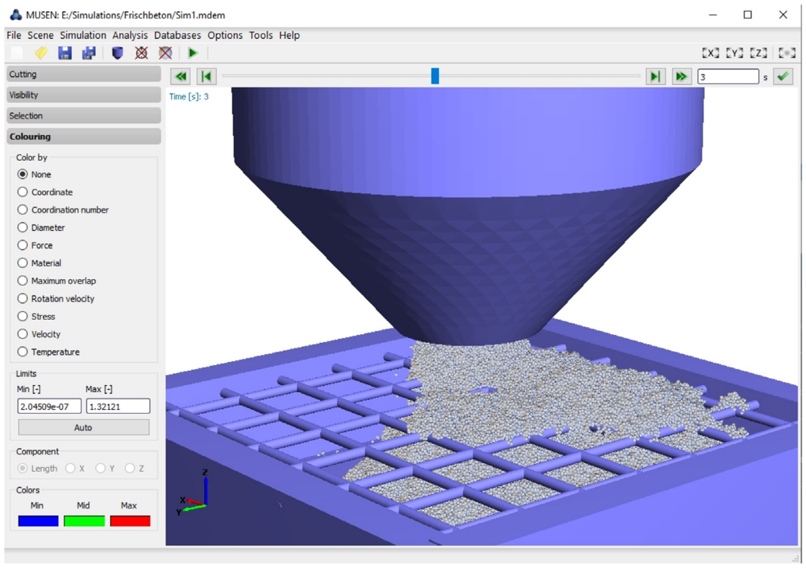

CMyNewModel.dll
CAbstractDEMModel
+m_type: EMusenModelType
+m_name: string
+m_parameters: vector<SModelParameters>
+...
+GetType(): EMusenModelType
+GetName(): string
+GetUniqueKey(): string
+GetParameteresNumber(): size_t
+GetAllParameters(): vector<SModelParameter>
+GetParameterValue(_sName): double
+SetParameterValue(sName, dValue)
+HasGPUSupport(): bool
+InitializeGPU(cudaDefines)
+Initialize(particles, walls, solidBonds, ...)
+Precalculate(time, timeStep)
+...()
CParticleParticleModel
+m_patricles: SParticleStruct*
+m_interactionProperties: std::vector<SInteractProps>*
+Initialize(particles, interactionProperties, ...)
+Precalculate(time, timeStep)
+Calculate(time, timeStep, collisions)
CMyNewModel
+Model-specific data
+PrecalculatePPModel(time, timeStep, particles)
+SetParametersGPU(parameters, PBC)
+CalculatePPForce(time, timeStep, src, dst, collisions, ...)
+CalculatePPForceGPU(time, timeStep, collisions, ...)

Fig. 2. Interfaces for creating a new model of particle–particle interaction.

To quantify the mixing efficiency the Lacey mixing index (Lacey, 1943 [42]) is calculated. For this, 20 virtual samples were taken using *Particle Analyser* at different time points and the number of particles of each type is calculated there. As a result of this study, it was noted that the vertical blades used do not reveal high mixing efficiency, and even after 8 rotations the mixing index is still below 0.9.

In the second case study, the mechanical behaviour of composites during a four-point flexural test is investigated using the bonded-particle model (Potyondy, 2015 [43]). The sample is described with 0.58 million primary particles connected by approximately 4.3 million solid bonds. During the simulation, two upper punches are moving downwards with a constant velocity until fracture occurs. Fig. 6 shows the sample at the initial, intermediate and final stages. At the intermediate stage, the bonds are coloured according to their strain, which allows to identify different stress regions. When the stress in the bonds exceeds their strength, the bonds start to break that leads to the fracture of the sample. To estimate the stiffness and strength of the material, the forces acting on the upper punches can be extracted from the MUSEN simulations using the *Results Analyser* component.

The calculation times for the both scenes using the hardware configuration Intel Core i7-6800K with NVIDIA GeForce GTX 1080 Ti are provided in Table 1.


MUSEN: E:/Simulations/Frischbeton/Sim1.mdem
File Scene Simulation Analysis Databases Options Tools Help
Cutting
Visibility
Selection
Colouring
Time [s]: 3
Min
Mid
Max

Color by: (●) None ○ Coordinate ○ Coordination number ○ Diameter ○ Force ○ Material ○ Maximum overlap ○ Rotation velocity ○ Stress ○ Velocity ○ Temperature

Limits: Min [-] 2.04509e-07 Max [-] 1.32121 Auto

Component: (●) Length ○ X ○ Y ○ Z

Fig. 3. Main window of the graphical user interface.

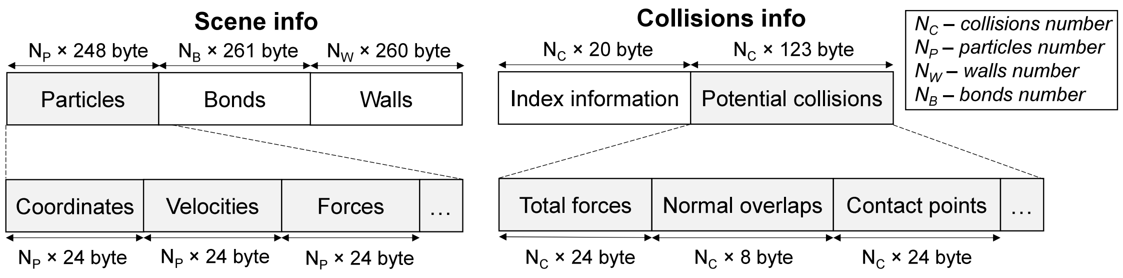


Fig. 4. General representation of the data structure in GPU memory.

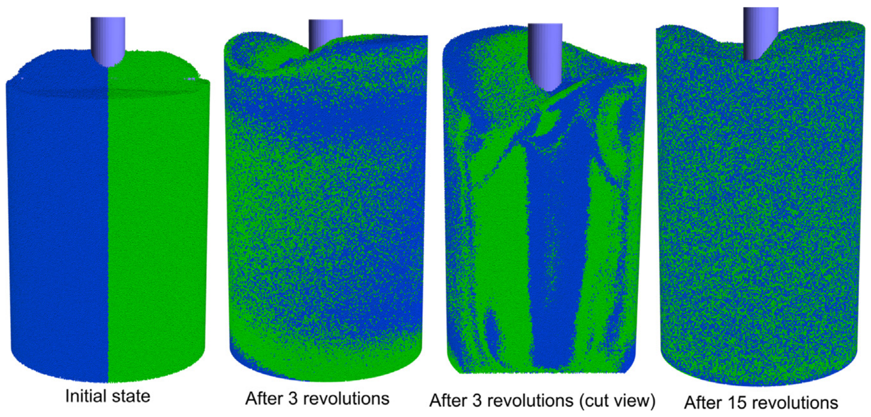


Fig. 5. Particle behaviour in mixer with vertical stirrer blades.

4. Impact

The MUSEN framework has been successfully applied to solve various tasks from different disciplines. Lee et al. 2017 [44] have analysed a wet granulation process through coupling of DEM results to a population balance model. Dosta et al. 2018 [45], using trial and error principle, adjusted unknown DEM model parameters by comparing simulation results with experimental data obtained from micro-computed tomography. More advanced linearization-based adjustment techniques were proposed for the modelling of mechanical properties with bonded-particle model (Jarolin et al. 2020 [46]). The sintering-induced deformation of metal–ceramic composites or macro-porous thermal barrier coatings were studied in Besler et al. 2016 [2] and Dosta et al. 2020 [3] accordingly. There exist numerous applications of MUSEN focused on modelling the deformational and breakage behaviour of different materials, such as biopolymer alginate aerogels (Dosta et al. 2019 [31]), titania or glass agglomerates (Kozhar et al. 2015 [32]; Dosta et al. 2016 [47]), etc.

Moreover, MUSEN is successfully incorporated into the educational process. Thanks to its user-friendly graphical interface, high computational efficiency and built-in tools for results analysis, students can configure initial scene, perform meaningful simulations, extract data and analyse results. All this can be done in

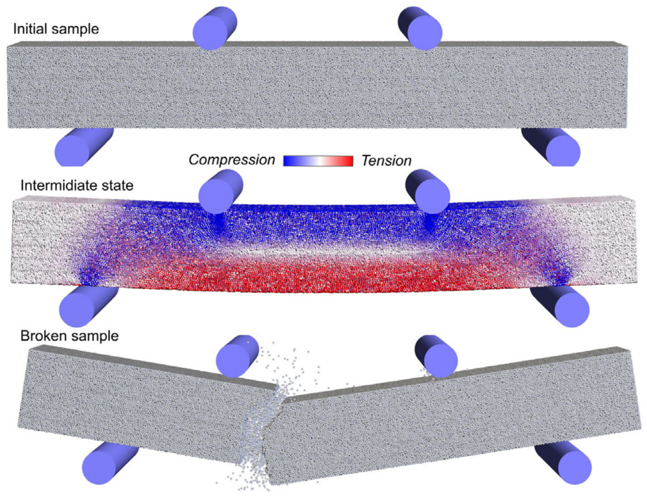


Fig. 6. Modelling of mechanical behaviour of composite material during four-point bending test.

one program on a conventional personal computer in the scope of standard 90 min classes.

It should be noted that nowadays there are alternative DEM software packages, and some of them provide functionality not available in MUSEN, for example, built-in coupling to CFD calculations or consideration of non-spherical particles. However, due to a number of the following benefits, such as:

- user-friendly intuitive graphical interface,
- high computational efficiency,
- open-source licensing,
- simple extendibility of the system

we expect MUSEN to be widely used in research, education and industry. Moreover, the object-oriented design and modular-based architecture strongly simplify the extension and modification of the framework in order to adapt it to specific processes or application patterns.

5. Conclusions

The microscale modelling of granular materials with the discrete element method is widely applied in different engineering areas, from solids process engineering to material science. Thus, the novel, highly efficient and user friendly open-source DEM simulation systems can significantly improve developments in various fields. In this contribution, the architecture and conceptual design of the MUSEN framework have been introduced. A combination of the visualization methods, algorithms for contact detection, parallelization strategies and other relevant features distinguishes MUSEN from other existing DEM software packages.

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.

Acknowledgements

A part of this work has been financially supported by the German Research Foundation (DFG) via projects DFG DO 2026/1-1 and DFG DO 2026/6-1.

References

[1] Zhu HP, Zhou ZY, Yang RY, Yu AB. Discrete particle simulation of particulate systems: A review of major applications and findings. Chem Eng Sci 2008;63:5728–70. http://dx.doi.org/10.1016/j.ces.2008.08.006.
[2] Besler R, da Silva MR, Dosta M, Heinrich S, Janssen R. Discrete element simulation of metal ceramic composite materials with varying metal content. J Eur Ceram Soc 2016;36(9):2245–53. http://dx.doi.org/10.1016/j.jeurceramsoc.2015.12.051.
[3] Dosta M, Furlan K, Skorych V, Heinrich S, Janssen R. Influence of pores arrangement on stability of photonic structures during sintering. J Eur Ceram Soc 2020;40(13):4562–71. http://dx.doi.org/10.1016/j.jeurceramsoc.2020.04.019.
[4] Pöschel T, Schwager T. Computational granular dynamics. Springer-Verlag Berlin Heidelberg; 2005, http://dx.doi.org/10.1007/3-540-27720-X.
[5] Kruggel-Emden H, Simsek R, Rickelt S, Wirtz S, Scherer V. Review and extension of normal force models for the discrete element method. Powder Technol 2007;171(3):157–73. http://dx.doi.org/10.1016/j.powtec.2006.10.004.
[6] Zhu HP, Zhou ZY, Yang RY, Yu AB. Discrete particle simulation of particulate systems: Theoretical developments. Chem Eng Sci 2007;62:3378–96. http://dx.doi.org/10.1016/j.ces.2006.12.089.
[7] O'Sullivan C. Particulate discrete element modelling. A geomechanics perspective. CRC Press; 2011.
[8] EDEM – The leading Discrete Element Method (DEM) software. www.edemsimulation.com. [Accessed September 2020].
[9] PFC – General Purpose Distinct-Element Modeling Framework. www.itasca.de/software/PFC. [Accessed September 2020].
[10] Rocky DEM – The Most Powerful Particle Simulation Software. rocky.esss.co. [Accessed September 2020].
[11] Kloss C, Goniva C. LIGGGHTS – Open source discrete element simulations of granular materials based on lammps. In: Supplemental proceedings: Materials fabrication, properties, characterization, and modeling, vol. 2. 2011, p. 781–8. http://dx.doi.org/10.1002/9781118062142.ch94.
[12] LIGGGHTS – Open source discrete element method particle simulation code – CFDEM®project. www.cfdem.com/liggghts-open-source-discrete-element-method-particle-simulation-code. [Accessed September 2020].
[13] Kozicki J, Donze FV. YADE-OPEN DEM: an open-source software using a discrete element method to simulate granular material. Eng Comput 2009;26(7):786–805. http://dx.doi.org/10.1108/02644400910985170.
[14] Yade – Open Source Discrete Element Method. yade-dem.org. [Accessed September 2020].
[15] Woo - extensible DEM simulations. woodem.org. [Accessed September 2020].
[16] Weinhart T, Orefice L, Post M, et al. Fast, flexible particle simulations – An introduction to MercuryDPM. Comput Phys Comm 2020;249. http://dx.doi.org/10.1016/j.cpc.2019.107129.
[17] MercuryDPM: Fast, flexible particle simulations. www.mercurydpm.org. [Accessed September 2020].
[18] Andrè D, Charles J.-L. Iordanoff I, Nèauport J. The GranOO workbench, a new tool for developing discrete element simulations, and its application to tribological problems. Adv Eng Soft 2014;74:40–8. http://dx.doi.org/10.1016/j.advengsoft.2014.04.003.
[19] GranOO: A free C++/Python discrete Element workbench. www.granoo.org. [Accessed September 2020].
[20] MechSys – Multi-Physics simulation library. mechsys.nongnu.org. [Accessed September 2020].
[21] Norouzi HR, Zarhami R, Sotudeh-Gharebagh R, Mostoufi N. Coupled CFD-DEM modeling: Formulation, implementation and application to multiphase flows. John Wiley & Sons; 2016, http://dx.doi.org/10.1002/9781119005315.
[22] Kieckhefen P, Pietsch S, Dosta M, Heinrich S. 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 2020;11:397–422. http://dx.doi.org/10.1146/annurev-chembioeng-110519-075414.
[23] Verlet L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys Rev 1967;159:98–103. http://dx.doi.org/10.1103/physrev.159.98.
[24] Quenterec B, Brot C. New method for searching for neighbors in molecular dynamics computations. J Comput Phys 1973;13:430–2. http://dx.doi.org/10.1016/0021-9991(73)90046-6.
[25] Mio H, Shimosaka A, Shirakawa Y, Hidaka J. Cell optimization for fast contact detection in the discrete element method algorithm. Adv Powder Technol 2007;18(4):441–53. http://dx.doi.org/10.1163/156855207781389519.
[26] Mindlin RD, Deresiewicz H. Elastic sphere in contact under varying oblique force. Trans ASME J Appl Mech 1953;20:327–44.
[27] Tsuji Y, Tanaka T, Ishida T. Lagrangian numerical simulation of plug flow of cohesionless particles in horizontal pipe. Powder Technol 1992;71:239–50. http://dx.doi.org/10.1016/0032-5910(92)88030-L.

[28] Johnson KL, Kendall K, Roberts AD. Surface energy and the contact of elastic solids. A Math Phys Sci 1971;324(1558):301–13. http://dx.doi.org/10.1098/rspa.1971.0141.

[29] Parhami F, McMeeking R. A network model for initial stage sintering. Mech Mater 1998;2:111–24. http://dx.doi.org/10.1016/S0167-6636(97)00034-3.

[30] Su J, Gu Z, Xu XY. Discrete element simulation of particle flow in arbitrarily complex geometries. Chem Eng Sci 2011;66:6069–88. http://dx.doi.org/10.1016/j.ces.2011.08.025.

[31] Dosta M, Jarolin K, Gurikov P. Modeling of mechanical behavior of biopolymer alginate aerogels using the bonded-particle model. Molecules 2019;24(14):2543. http://dx.doi.org/10.3390/molecules24142543.

[32] Kozhar S, Dosta M, Antonyuk S, Heinrich S, Bröckel U. DEM simulations of amorphous irregular shaped micrometer-sized titania agglomerates at compression. Adv Powd Technol 2015;26(3):767–77. http://dx.doi.org/10.1016/j.apt.2015.05.005.

[33] Protocol Buffers – Google Developers. developers.google.com/protocol-buffers. [Accessed June 2020].

[34] Deutsch P. DEFLATE compressed data format specification Version 1.3. RFC 1951. 1996, http://dx.doi.org/10.17487/rfc1951.

[35] zlib – A Massively Spiffy Yet Delicately Unobtrusive Compression Library. www.zlib.net. [Accessed September 2020].

[36] Dranyshnikov S, Dosta M. Advanced approach for simulation results saving from discrete element method. Adv Eng Softw 2019;136:102694. http://dx.doi.org/10.1016/j.advengsoft.2019.102694.

[37] Qt – Cross-platform software development for embedded & desktop. www.qt.io. [Accessed September 2020].

[38] OpenGL – The Industry Standard for High Performance Graphics. www.opengl.org. [Accessed September 2020].

[39] Qt OpenGL. doc.qt.io/qt-5/qtopengl-index.html. [Accessed September 2020].

[40] Morton GM. A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Technical report, Ottawa, Canada: IBM Ltd; 1966.

[41] Inno Setup. jrsoftware.org/isinfo.php. [Accessed September 2020].

[42] Lacey PMC. The mixing of solid particles. Trans Inst Chem Eng 1943;21:53–9. http://dx.doi.org/10.1016/S0263-8762(97)80004-4.

[43] Potyondy DO. The bonded-particle model as a tool for rock mechanics research and application: current trends and future directions. Geosyst Eng 2015;18:1–28. http://dx.doi.org/10.1080/12269328.2014.998346.

[44] Lee KF, Dosta M, McGuire AD, Mosbach S, Wagner W, Heinrich S, Kraft M. Development of a multi-compartment population balance model for high-shear wet granulation with discrete element method. Comput Chem Eng 2017;99(6):171–84. http://dx.doi.org/10.1016/j.compchemeng.2017.01.022.

[45] Dosta M, Bröckel U, Gilson L, Kozhar S, Auernhammer GK, Heinrich S. Application of micro computed tomography for adjustment of model parameters for discrete element method. Chem Eng Res Des 2018;135:121–8. http://dx.doi.org/10.1016/j.cherd.2018.05.030.

[46] Jarolin K, Dosta M. Linearization-based methods for the calibration of bonded-particle models. Comput Part Mech 2020. http://dx.doi.org/10.1007/s40571-020-00348-z.

[47] Dosta M, Dale S, Antonyuk S, Wassgren C, Heinrich S, Litster JD. Numerical and experimental analysis of influence of granule microstructure on its compression breakage. Powder Technol 2016;299:87–97. http://dx.doi.org/10.1016/j.powtec.2016.05.005.