Lethe-DEM : An open-source parallel discrete element solver with load balancing

Shahab Golshan[a], Peter Munch[b,c], Rene Gassmöller[d], Martin Kronbichler[c,e], Bruno Blais[a,*]

[a] *Research Unit for Industrial Flows Processes (URPEI), Department of Chemical Engineering, École Polytechique de Montréal, PO Box 6079, Stn Centre-Ville, Montréal, QC, Canada, H3C 3A7*
[b] *Continuum Simulations, Institute of Material Systems Modeling, Helmholtz-Zentrum Hereon GmbH, Germany, Max-Planck-Str. 1, 21502 Geesthacht, Germany*
[c] *Institute for Computational Mechanics, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany*
[d] *Department of Geological Sciences, University of Florida, Gainesville, FL, USA*
[e] *Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden*

Abstract

Approximately 75% of the raw material and 50% of the products in the chemical industry are granular materials. The Discrete Element Method (DEM) provides detailed insights of phenomena at particle scale and it is therefore often used for modeling granular materials. However, because DEM tracks the motion and contact of individual particles separately, its computational cost increases non-linearly $O(n_p \log(n_p)) - O(n_p^2)$ (depending on the algorithm) with the number of particles (n_p). In this article, we introduce a new open-source parallel DEM software with load balancing: Lethe-DEM. Lethe-DEM, a module of Lethe, consists of solvers for two-dimensional and three-dimensional DEM simulations. Load-balancing allows Lethe-DEM to significantly increase the parallel efficiency by $\approx 25 - 70\%$ depending on the granular simulation. We explain the fundamental modules of Lethe-DEM, its software architecture, and the governing equations. Furthermore, we verify Lethe-DEM with several tests including analytical solutions and comparison with other software. Comparisons with experiments in a flat-bottomed silo, wedge-shaped silo, and rotating drum validate Lethe-DEM. We investigate the strong and weak scaling of Lethe-DEM with $1 \leq n_c \leq 192$ and $32 \leq n_c \leq 320$ processes, respectively, with and without load-balancing. The strong-scaling analysis is performed on the wedge-shaped silo and rotating drum simulations, while for the weak-scaling analysis, we use a dam break simulation. The best scalability of Lethe-DEM is obtained in the range of $5000 \leq n_p/n_c \leq 15\,000$. Finally, we demonstrate that large scale simulations can be carried out with Lethe-DEM using the simulation of a three-dimensional

*Corresponding author
Email address: bruno.blais@polymtl.com (Bruno Blais)

cylindrical silo with $n_p = 4.3 \times 10^6$ on 320 cores.

Keywords: Discrete Element Methods (DEM), High-Performance Computing, Load-balancing, Silo, Rotating Drum

1. Introduction

Granular material is ubiquitous in nature in the form of soil, sand, or gravel, and the second most used and manipulated material in global industry, right after water [1, 2]. Approximately half of the products and three quarters of the raw material in the chemical industry is in the form of granular materials [3, 2]. Because of their prominence, an accurate modeling method for granular flows is critical to simulate the efficiency and safety of industrial processes involving such materials. There are two common approaches in the numerical modelling of granular flows, continuum models (Eulerian) and discrete models (Lagrangian). The continuum models, in which a constitutive model describes the relation between stresses and strains, ignore the discrete essence of granular systems and therefore suffer from multiple limitations, for instance in calculating solids holdup [4].

The Discrete Element Method (DEM) is a Lagrangian model that simulates the motion and collision of discrete particles with each other and with surfaces [5]. Due to the small modelling scale of individual particles and its discrete nature, DEM is accurate and provides detailed insights of phenomena at particle and system scale [6, 7, 5]. DEM has been used to simulate the granular systems in geotechnical [8, 9], material processing [10, 11], mining [12], chemical [6, 5], polymers [13], metallurgical [14], pharmaceuticals [15], agricultural [16, 4], and food [17] industries. Furthermore, DEM models coupled with Computational Fluid Dynamics (CFD) approaches, a.k.a. CFD-DEM, provide insight into solid-fluid systems, such as pneumatic conveying, fluidized, and spouted beds [6, 7, 18, 19]. In DEM, the flow properties of granular materials are simulated using the interactions between the individual discrete particles. Two main approaches exist to model collisions, namely hard-sphere (event-based) and soft-sphere (time-based) methods [7, 6]. In hard-sphere models, the simulation proceeds collision by collision. As a result, the model is unable to simulate multiple simultaneous collisions or quasi-static systems, making hard-sphere models only suitable for the simulation of dilute systems. On the other hand, in concentrated systems with a high number of particles, the collision frequency is high and multiple simultaneous collisions may occur. Soft-sphere models handle such dense systems by using a small time step ($dt \approx 10^{-5} - 10^{-6}$ s).

Figure 1 illustrates the major steps of a conventional DEM simulation [6]. All these steps have a computational cost that is proportional to the number of particles ($\mathcal{O}(n_p)$), except contact search and collision force calculations. After initialization of the parameters, DEM performs a contact search to obtain a list of all particle-particle and particle-wall contacts. In the worst-case scenario, the DEM has to investigate the possibility of contact of each particle with all other

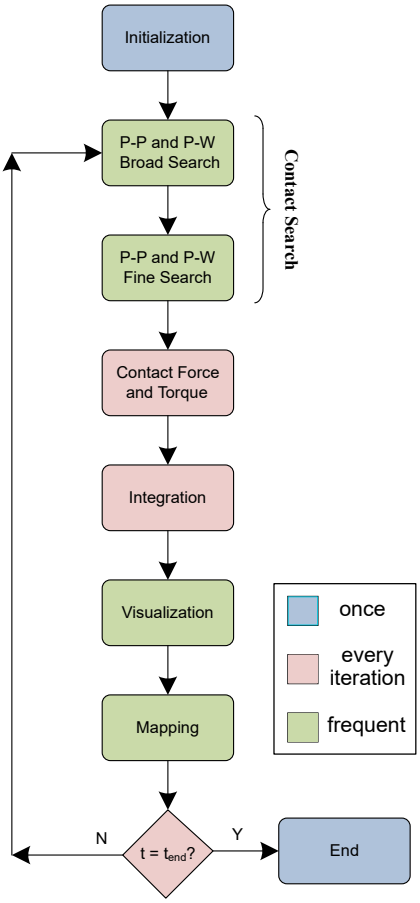


Figure 1: Major steps in a conventional DEM simulation. Blue, green and red colors show steps that are performed once, at every n-th time step and at each step throughout the simulation, respectively. P and W stand for particle and wall, respectively.

particles and the system walls at each step. This would result in a computational cost of $\mathcal{O}(n_p^2)$ per step. To avoid this, usually a two-step contact search method is employed, often called broad and fine searches, respectively. The broad search detects nearby particles as contact candidates, while the fine search finds the candidate contact pairs in a smaller domain. This two-step contact detection method decreases the computational cost to $\mathcal{O}(n_p \log n_p)$. In the next step, the contact list is used to compute the contact forces. Subsequently, the integration step of the DEM updates the accelerations, velocities, and positions of the particles, based on the accumulated forces. According to their new positions, the particles are mapped to the simulation grids for the next time step and visualization purposes.

Several software packages have been developed for DEM simulations, including LIGGGHTS [20], MercuryDPM [21], EDEM [22] and XPS [23]. In this

work, we introduce a new open-source, parallel and load-balanced software for the simulation of granular systems: Lethe-DEM. Lethe-DEM is a DEM solver in the framework of Lethe [24], a high-order CFD solver to simulate incompressible flows and targeting practice-oriented chemical engineering applications. Lethe is built upon deal.II [25], a well-established open-source finite element modelling library. Lethe-DEM uses the background mesh infrastructure of the deal.II library, which allows it to reuse many functionalities that have been implemented with FEM in mind and provides several advantages. First of all, it enables the use of high-order finite element mappings, enabling the simulation of complex geometries (such as cylinders and spheres) with high accuracy. Additionally, Lethe-DEM inherits deal.II's high-performance infrastructure, such as SIMD vectorization [26] and, more importantly, parallelization capabilities with the message passing interface (MPI) including load-balancing capabilities through the p4est library [27, 28]. Using load-balancing, Lethe-DEM redistributes the computational load among the CPUs throughout a simulation by ensuring that all cores have a similar amount of particles. Load-balancing diminishes the possibility of having idle cores during a simulation and decreases the simulation time significantly, especially in large-scale systems or when complex and sparse geometries are considered. Due to this redistribution of the computational load via load-balancing, Lethe-DEM shows good strong scaling.

The remainder of this work is organized as follows. First, we present the governing equations of the DEM models in Lethe-DEM in Section 2. Then, Section 3 describes the architecture of Lethe-DEM, including contact detection, parallelization and load-balancing strategies. In Section 4 we verify and validate the software with available correlations, results of other DEM codes and experimental data. Additionally, we investigate both the strong and weak scaling capabilities of the software. To the best of our knowledge, these are amongst the first DEM parallel scaling result of this kind that were demonstrated. Finally, Section 5 concludes the article and presents the future works of the software.

2. Governing equations and solution strategies

In DEM, Newton's second law of motion describes the evolution of the position and the velocity of all the particles through time:

$$m_i \boldsymbol{a} = m_i \frac{d\boldsymbol{v_i}}{dt} = \sum_{j \in \mathcal{C}_i} (\boldsymbol{F}_{ij}^n + \boldsymbol{F}_{ij}^t) + m_i \boldsymbol{g} + \boldsymbol{F}_i^{\text{ext}} \tag{1}$$

$$I_i \frac{d\boldsymbol{\omega_i}}{dt} = \sum_{j \in \mathcal{C}_i} (\boldsymbol{M}_{ij}^t + \boldsymbol{M}_{ij}^r) + \boldsymbol{M}_i^{\text{ext}} \tag{2}$$

where i is the particle index, m_i is the mass of particle i, $\boldsymbol{v}_i$ the velocity of particle i, $\boldsymbol{a}_i$ the acceleration of particle i, t denotes time, j runs through all particles $\mathcal{C}_i$ in the contact list of particle i, $\boldsymbol{F}_{ij}^n$ and $\boldsymbol{F}_{ij}^t$ are normal and tangential contact forces due to the contact between particles i and j, $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{F}^{\text{ext}}$ encompasses all other external forces, I_i is the moment of

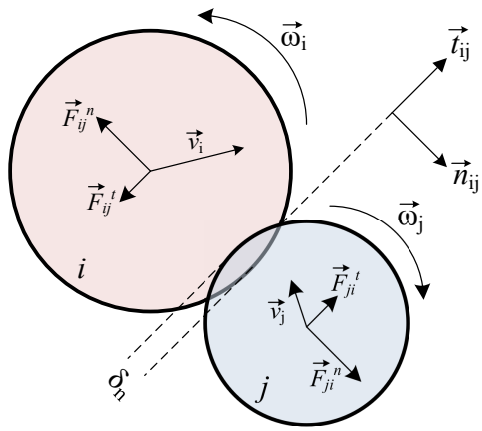


Figure 2: Typical particle-particle contact in the framework of soft-sphere DEM. For a description of the shown quantities see equations (1) and (2).

inertia of particle i, $\boldsymbol{\omega_i}$ denotes the angular velocity of particle i, $\boldsymbol{M}_{ij}^t$ and $\boldsymbol{M}_{ij}^r$ are tangential and rolling friction torques due to the contact between particle i and j, and $\boldsymbol{M}^{\text{ext}}$ denotes all other external torques. Figure 2 shows a typical contact between two particles in the framework of soft-sphere DEM. This figure illustrates some of the variables introduced in equations (1) and (2).

2.1. Contact force and torque models

The particles can overlap in the soft-sphere model (see Figure 2), and the contact forces in the normal and tangential directions are defined using the normal and tangential overlaps. The normal overlap (δ_n) between the particles i and j is computed as:

$$\delta_n = R_i + R_j - \|\boldsymbol{x}_j - \boldsymbol{x}_i\| \tag{3}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the position vectors of the particle centers, and R_i, R_j are radii of particles i and j, respectively. Consequently, the normal overlap is positive when the distance between two particles i and j is smaller than the sum of their radii, that is, when particles are undergoing collision.

The contact normal vector ($\boldsymbol{n}_{ij}$), as illustrated in Figure 2, is a vector pointing from particle i to particle j in a collision, computed as:

$$\boldsymbol{n}_{ij} = \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|} \tag{4}$$

The tangential overlap ($\boldsymbol{\delta}_t$) in the contact models of Lethe-DEM depends on the contact history. At the beginning of each contact, the tangential overlap is equal to zero. The following equation is then used to update the tangential overlap during a contact [7]:

$$\boldsymbol{\delta}_{ij}^{t,\text{new}} = \boldsymbol{\delta}_{ij}^{t,\text{old}} + \boldsymbol{v}_{rt} dt \tag{5}$$

Here, dt denotes time-step, v_{rt} is the contact relative velocity in tangential direction and is computed as described below.

This approach is different from the one taken in MercuryDPM [21], which uses a two-step procedure to update the tangential overlap during a contact. This two-step procedure consists of removing the normal component of the tangential overlap using the updated normal vector in the first step, and updating the tangential overlap using the updated relative velocity in the tangential direction in the second step. Even though this could theoretically lead to some minor differences when a particle rolls on top of another particle, we did not observe any significant benefit in several benchmarks. As a result, we followed the procedure which is also utilized in other software packages including LIGGGHTS [20]. The contact relative velocity in the tangential direction ($\boldsymbol{v}_{rt}$) is given by:

$$\boldsymbol{v}_{rt} = \boldsymbol{v}_{ij} - \boldsymbol{v}_{rn} \tag{6}$$

where the contact relative velocity ($\boldsymbol{v}_{ij}$) and relative velocity in the normal direction ($\boldsymbol{v}_{rn}$) are given respectively given by:

$$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j + (R_i\boldsymbol{\omega}_i + R_j\boldsymbol{\omega}_j) \times \boldsymbol{n}_{ij} \tag{7}$$

$$\boldsymbol{v}_{rn} = (\boldsymbol{v}_{ij}.\boldsymbol{n}_{ij})\,\boldsymbol{n}_{ij} \tag{8}$$

The computed normal and tangential overlaps are then used to calculate normal and tangential contact forces. For the calculation of contact forces and torques, previous studies have proposed several models. Linear viscoelastic models (a.k.a. Hookean models) [29], in which the contact force is a linear function of the contact overlap, and non-linear viscoelastic models (a.k.a. Hertzian models), in which the relation between contact force and overlap is non-linear, are the most well-known models.

In linear and non-linear viscoelastic models, normal and tangential contact forces are calculated using the following equations [29, 30]:

$$\boldsymbol{F}_{ij}^n = -(k_n\delta_n)\boldsymbol{n}_{ij} - (\eta_n\boldsymbol{v}_{rn}) \tag{9}$$

$$\boldsymbol{F}_{ij}^t = -(k_t\boldsymbol{\delta}_t) - (\eta_t\boldsymbol{v}_{rt}) \tag{10}$$

where k is the spring constant and η is the damping constant. In linear models, spring and damping constants do not depend on the normal or tangential overlaps, whereas in non-linear models, these constants are a function of the overlaps.

Several models have been proposed to calculate the spring and damping constant in the context of linear models. In Lethe-DEM, we use the model expressed with the following equation [20]:

$$k_n = \frac{16}{15}\sqrt{R_e}Y_e\left(\frac{15m_eV^2}{16\sqrt{R_e}Y_e}\right)^{0.2} \tag{11}$$

$$\eta_n = \sqrt{\frac{4 m_e k_n}{1 + \left(\frac{\pi}{\ln e}\right)^2}} \tag{12}$$

where R_e, Y_e, m_e, V and e are the effective radius, effective Young's modulus, effective mass, characteristic impact velocity, and coefficient of restitution, respectively. Spring and damping constants in the tangential direction are also calculated using Equations (11) and (12).

On the other hand in non-linear viscoelastic models, the following equations calculate the normal and tangential spring and damping constants [20]:

$$k_n = \frac{4}{3} Y_e \sqrt{R_e \delta_n} \tag{13a}$$

$$\eta_n = -2\sqrt{\frac{5}{6}} \beta \sqrt{S_n m_e} \tag{13b}$$

$$k_t = 8 G_e \sqrt{R_e \delta_n} \tag{13c}$$

$$\eta_t = -2\sqrt{\frac{5}{6}} \beta \sqrt{S_t m_e} \tag{13d}$$

Table 1 reports the parameters of Equations(13a)–(13d).

Table 1: Parameters in Equations (13a)-(13d)

Parameter	**Equation**
Effective mass	$\frac{1}{m_e} = \frac{1}{m_i} + \frac{1}{m_j}$
Effective radius	$\frac{1}{R_e} = \frac{1}{R_i} + \frac{1}{R_j}$
Effective shear modulus	$\frac{1}{G_e} = \frac{2(2-\nu_i)(1+\nu_i)}{Y_i} + \frac{2(2-\nu_j)(1+\nu_j)}{Y_j}$
Effective Young's modulus	$\frac{1}{Y_e} = \frac{\left(1-\nu_i^2\right)}{Y_i} + \frac{\left(1-\nu_j^2\right)}{Y_j}$
β	$\beta = \frac{\ln e}{\sqrt{\ln^2 e + \pi^2}}$
S_n	$S_n = 2 Y_e \sqrt{R_e \delta_n}$
S_t	$S_t = 8 G_e \sqrt{R_e \delta_n}$

The following equation calculates the tangential torque in Lethe-DEM:

$$\boldsymbol{M}_{ij}^t = R_i \boldsymbol{n}_{ij} \times \boldsymbol{F}_{ij}^c \tag{14}$$

Two models, namely constant [31, 7] and viscous [32, 7], are available for the calculation of the rolling friction torque. Table 2 reports these two models and their corresponding parameters.

Table 2: Rolling friction torque models in Lethe-DEM and the corresponding parameters. $\boldsymbol{F}_{ij}^{n}$ and μ_r are contact force in normal direction and rolling friction coefficient, respectively.

Model	**Equation**
Constant torque model [31]	$\boldsymbol{M}_{ij}^{r} = -\mu_r R_e \left\lvert \boldsymbol{F}_{ij}^{n} \right\rvert \hat{\boldsymbol{\omega}}_{ij}$
Viscous torque model [32]	$\boldsymbol{M}_{ij}^{r} = -\mu_r R_e \left\lvert \boldsymbol{F}_{ij}^{n} \right\rvert \left\lvert \boldsymbol{V}_{\omega} \right\rvert \hat{\boldsymbol{\omega}}_{ij}$
Parameters	$\hat{\boldsymbol{\omega}}_{ij} = \frac{\boldsymbol{\omega}_i - \boldsymbol{\omega}_j}{\lvert \boldsymbol{\omega}_i - \boldsymbol{\omega}_j \rvert}$
	$\boldsymbol{V}_{\omega} = (\boldsymbol{\omega}_i \times R_i \boldsymbol{n}_{ij} - \boldsymbol{\omega}_j \times R_j \boldsymbol{n}_{ji})$

If $\left\lvert \boldsymbol{F}_{ij}^{t} \right\rvert \geq \mu \left\lvert \boldsymbol{F}_{ij}^{n} \right\rvert$ occurs during a collision, Coulomb's criterion is violated (a.k.a. gross sliding). In this case, the magnitude of the exerted kinetic friction is independent of the magnitude of particles' velocities, and the tangential overlap is limited to [29]:

$$\boldsymbol{\delta}_t = \frac{\tilde{\boldsymbol{F}}_{ij}^{t}}{-k_t} \tag{15}$$

where $\tilde{\boldsymbol{F}}_{ij}^{t}$ is the limited elastic tangential force (tangential force without the damping force),

$$\tilde{\boldsymbol{F}}_{ij}^{t} = \hat{\boldsymbol{F}}_{ij}^{t} + \eta_t \boldsymbol{v}_{rt} \tag{16}$$

in which $\hat{\boldsymbol{F}}_{ij}^{t}$ is the limited tangential force:

$$\hat{\boldsymbol{F}}_{ij}^{t} = \mu \left\lvert \boldsymbol{F}_{ij}^{n} \right\rvert \frac{\boldsymbol{F}_{ij}^{t}}{\left\lvert \boldsymbol{F}_{ij}^{t} \right\rvert} \tag{17}$$

To limit the tangential force to the Coulomb limit, two different approaches are common in the literature [21, 30, 20, 7]. Some researchers only limit the tangential force to the Coulomb's limit, while others limit the tangential overlap first and then recalculate the tangential force using this limited overlap. In Lethe-DEM, we use the second approach, similar to the method used in LIGGGHTS [20], to ensure that the tangential overlap does not accumulate during gross sliding. Furthermore, we noticed that using the equation $\boldsymbol{\delta}_t = \boldsymbol{F}_{ij}^{t}/k_t$ is not consistent with tangential force models including damping force. Tangential force models with damping lose their continuity when they exceed the Coulomb's limit (with $\boldsymbol{\delta}_t = \boldsymbol{F}_{ij}^{t}/k_t$). As a result, we use equations (15)–(17) to maintain the continuity of the tangential force when it is limited to the Coulomb's limit.

For particle-wall contacts, we apply the same model and equations of particle-particle contacts. We use the background triangulation and mapping, which are available using deal.II, to obtain the normal vectors at the location of the contact and the normal overlap with each boundary wall.

2.2. Integration methods

Equation (1) converts the net force exerted on a particle to its acceleration at each DEM time step. Several integration methods can be used in DEM [7]. In Lethe-DEM, we have implemented the Explicit Euler and the Velocity Verlet methods.

2.2.1. Explicit Euler

In the explicit Euler method, the following equations calculate the velocity and position of a particle at each time step:

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \boldsymbol{a}_i^n dt \tag{18}$$

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \boldsymbol{v}_i^n dt \tag{19}$$

in which n and $n+1$ denote the current and the next step, respectively. This scheme is first order accurate for both position and velocity. It is very simple to implement, but its accuracy is limited [7]. The same method is applied to the calculation of angular velocity using the momentum exerted on a particle.

2.2.2. Velocity Verlet

The velocity Verlet integration scheme uses a half-step velocity $\boldsymbol{v}_i^{n+\frac{1}{2}}$ to update the position of the particles:

$$\boldsymbol{v}_i^{n+\frac{1}{2}} = \boldsymbol{v}_i^n + \boldsymbol{a}_i^n \frac{dt}{2} \tag{20}$$

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \boldsymbol{v}_i^{n+\frac{1}{2}} dt \tag{21}$$

The new velocity of the particle is:

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^{n+\frac{1}{2}} + \boldsymbol{a}_i^{n+1} \frac{dt}{2} \tag{22}$$

When the force only depends on the position, the Velocity Verlet scheme is second-order accurate for position and velocity. In this context, the Velocity Verlet scheme is symplectic (see [33] for a full demonstration). When the force depends on the velocity, as is the case when the coefficient of restitution is not one, the scheme is first-order accurate for both position and velocity [33]. In practice, however, it is still significantly more accurate than an explicit Euler scheme. Since the only velocity-dependent force in DEM are dissipative forces, the loss of symplectic characteristic of the integration scheme does not pose a problem since there is no more energy to conserve.

3. Software description and architecture

In this section, the architecture of Lethe-DEM as well as the main features and algorithms of this software are explained.

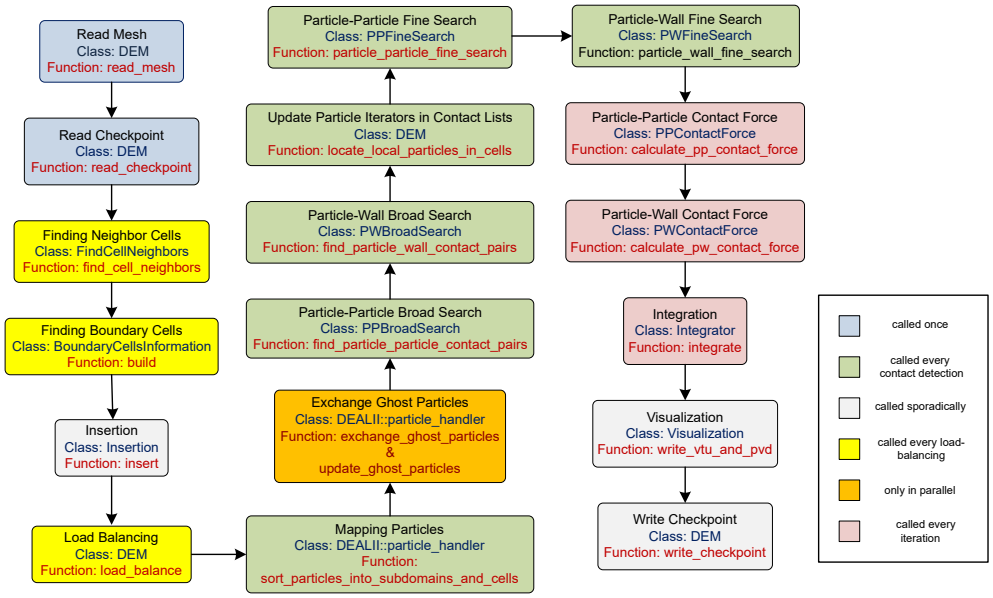


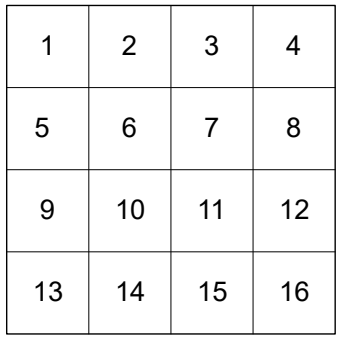
Figure 3: Lethe-DEM fundamental steps, classes and function.

3.1. Overview of Lethe-DEM

The fundamental steps, features, and architecture of Lethe-DEM are illustrated in Figure 3. The main steps are divided into six main groups, depending on whether a particular step is called once, at every contact detection, sporadically, at every load balancing operation, only in parallel or at every iteration.

At the beginning, the simulation domain is defined using a triangulation (grid), which defines the system outer and inner boundaries. Similar to Lethe [24], unstructured meshes (using the GMSH [34] file format) and native meshes produced by the deal.II library (ranging from a simple ball and cube to more complex ones like airfoils) are supported. Lethe-DEM supports writing and reading checkpoint files, which enables the user to quickly restart an unfinished simulation.

Once the triangulation has been initialized, the neighbor lists of all cells are produced (`Finding Neighbor Cells` block in Figure 3). Cell neighbor lists is a data container in which the neighbors of all the cells sharing a vertex/node are stored. To make broad and fine searches more efficient, repetitions are avoided in the neighbor list. For instance, if cell j is in the neighbor list of cell i, we do not reconsider cell i in the neighbors list of cell j. Figure 4 gives as an example a two-dimensional triangulation and its cell neighbor lists. The same procedure defines the neighbor lists also in three-dimensional triangulations. In load-balancing iterations, cell neighbor lists have to be reconstructed since the owners of the cells change. The details of load balancing are explained in more detail in Section 3.5. Similar to the cell neighbor lists, boundary cells are also identified within the triangulation (`Finding Boundary Cells` block in



Main cell	Neighbor list
1	2, 5, 6
2	3, 5, 6, 7
3	4, 6, 7, 8
4	7, 8
5	6, 9, 10
6	7, 9, 10, 11
7	8, 10, 11, 12
8	11, 12
9	10, 13, 14
10	11, 13, 14, 15
11	12, 14, 15, 16
12	15, 16
13	14
14	15
15	16
16	-

Figure 4: A sample triangulation and the corresponding cell neighbors lists.

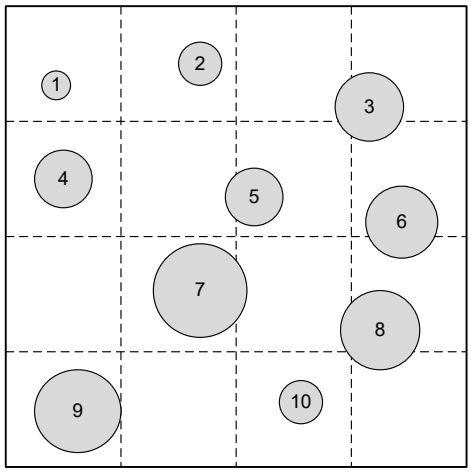
Figure 3). For the sample triangulation in Figure 4, cells 1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15 and 16 are the boundary cells.

In the insertion step (`Insertion` block in Figure 3), particles with specified properties (e.g., size) are inserted in the simulation domain. In Lethe-DEM, two particle size types, namely uniform and normal size distributions, can be used. A uniform size distribution results in a constant diameter for all particles. A normal size distribution for a parameter ξ is defined as:

$$f(\xi) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{\xi-\mu}{\sigma}\right)^2} \tag{23}$$

where μ and σ are average and standard deviation of the normal distribution. Multiple particle types, with different size distributions (uniform or normal) and mechanical properties, can be inserted by defining a new particle type in the parameter handler file. Given a batch of particles, the software searches the surrounding cell of each particle and assigns the responsibility of the particle to that cell. Using this information, broad and fine contact searches obtain particle-particle (`Particle-Particle Broad Search` and `Particle-Particle Fine Search` blocks in Figure 3) and particle-wall (`Particle-Wall Broad Search` and `Particle-Wall Fine Search` blocks in Figure 3) contact lists. Lethe-DEM does not call broad and fine contact searches every iteration. It supports two approaches to find contact search iterations: `constant` and `dynamic`. In the `constant` approach, the simulation calls broad and fine searches at a frequency of every n time steps, defined by a user defined parameter. On the other hand, in the `dynamic` method, Lethe-DEM automatically estimates the iterations for calling broad and fine searches by tracking the maximal displacement of particles and comparing it with a user-defined value.

In the next step, Equations (3)–(9) calculate the contact forces of the pairs present in the particle-particle and particle-wall contact lists (`Particle-Particle Contact Force` and `Particle-Wall Contact Force` blocks in Figure 3). After calculating the contact forces, the integration class (`Integration` block in



Particle i	Contact candidates
1	2, 4
2	4, 5
3	5, 6
4	7
5	6, 7, 8
6	8
7	9, 10
8	10
9	-
10	-

Figure 5: Contact pair candidates (which is the output of broad search and input of fine search) in a sample system.

Figure 3) updates the locations and velocities of all the particles by using Equations (18)–(22). Finally, when required, the software outputs the necessary post-processing files (`Visualization` block in Figure 3) and the checkpoints (`Write Checkpoint` block in Figure 3).

3.2. Contact detection algorithm

In the broad search, we use the cell neighbor lists to find contact pair candidates. To this end, Lethe-DEM loops through all the cells in the triangulation: for each cell it loops through all the particles, and for each particle, it adds other particles in the main cell or in neighbor cells to the contact pair candidates. Figure 5 shows the contact pair candidates in a sample system.

The fine search uses these contact pair candidates as an input, and calculates the distance between these pair candidates. If the distance between a pair is smaller than a used-defined threshold (generally $1.3d_p$, where d_p is the maximum particle diameter), the fine search adds the pair to the contact list. For instance, in the system illustrated in Figure 6, the red circle shows the domain for the fine search. In this system, particle 9 is in the contact list of particle 7, while particle 10 is not in this list. This contact list is the output of the fine search. The contact forces are calculated using this contact list. Contact list as well as contact pair candidates are updated at the contact search iterations.

3.3. User input and parameter files

To enable full control over the simulations, Lethe-DEM is parameterized using an input file in the native deal.II *prm* format. There are separated sections, such as `Simulation control`, `Model parameters`, `Physical properties`, `Insertion information`, `Mesh reading/generation`, and `Boundary motion` in

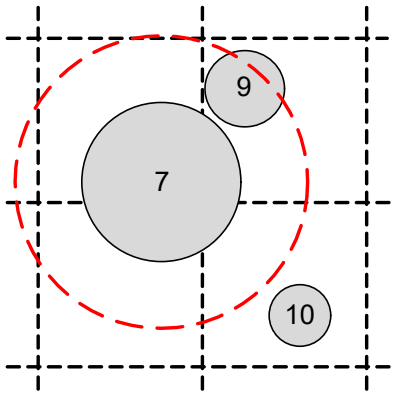


Figure 6: A sample system for showing contact list (which is the output of fine search and input of contact force class) members.

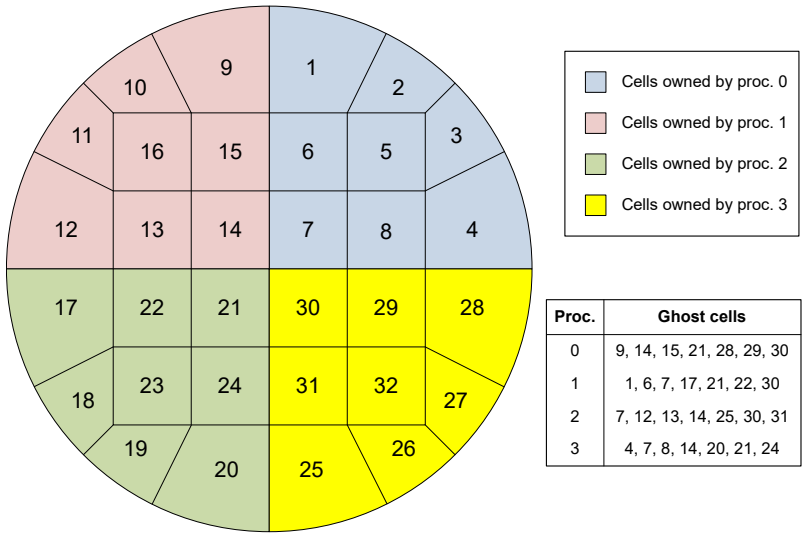
an input file. Listing 1 in Appendix A shows examples of an input parameter file for a DEM simulation.

3.4. Version control, continuous integration, and testing

Lethe-DEM is directly integrated within Lethe [24]. It is built as separate executable and is configured using CMake [35]. A public GitHub repository `https://github.com/lethe-cfd/lethe` keeps Lethe under version control. The Wiki documentation of the project is also on the GitHub repository. Lethe-DEM is distributed with several unit tests for individual functions and classes, and application tests for all the solvers. GitHub Actions instances are used for continuous integration and to verify all the tests (including unit and application tests). This ensures the stability of the master branch and ensure quality control when merging pull requests. Lethe is distributed under an LGPL 3.0 license.

3.5. Parallelization

Lethe-DEM is parallelized using MPI [36]. This allows the simulation of large problems on distributed computer architectures. Lethe-DEM leverages the p4est library via deal.II to handle mesh partitioning [27, 37]. First, a coarse triangulation is created, either from a GMSH file or a deal.II native triangulation. This coarse triangulation, which in most cases consists of not more than 100–10,000 cells, is replicated on all processes. This triangulation is then refined locally or globally to a desired element size in a forest-of-tree manner, partitioning the final triangulation with a well-established domain-decomposition approach. Each locally-owned subdomain is surrounded by a halo layer of the width of a single cell. Particles which reside in the ghost cells of a process are defined as ghost particles. In Figure 7, a sample domain is distributed among four processes. For each process, we specify the locally owned and the ghost cells.



Proc.	Ghost cells
0	9, 14, 15, 21, 28, 29, 30
1	1, 6, 7, 17, 21, 22, 30
2	7, 12, 13, 14, 25, 30, 31
3	4, 7, 8, 14, 20, 21, 24

Figure 7: A sample distributed triangulation on four processes, local and ghost cells are specified for each process.

All the operations in Lethe-DEM except particle-particle and particle-wall contacts occur in the local domain of each process. For each process, we categorize the particle-particle collisions into two types: local-local and local-ghost contacts. In local-local contacts between particles i and j, the process owns both particles i and j, while in local-ghost contacts, the process owns particle i, while another process owns particle j located in a ghost cell. Therefore, the processes fully handle local-local collisions, while for local-ghost collisions, only the exerted contact force on particle i and the associated integration is performed using the information (position, velocity, etc.) of the ghost particle of index j. The `exchange_ghost_particles` and `update_ghost_particles` functions update the information of location and properties of the ghost particles.

The `exchange_ghost_particles` function creates and updates the ghost particles by removing particles that have left the ghost cells and as a consequence have become unnecessary, as well as by creating the required ghost particles incoming from other processes. Furthermore, it sets up the communication patterns while the `update_ghost_particles` function only updates the properties and location of the ghost particles. Therefore, we call the former only at contact detection iterations, where we are locating particles in the cells by calling the `sort_particles_into_subdomains_and_cells` function. In other iterations, we call the significantly cheaper `update_ghost_particles` function.

3.6. Load balancing

Lethe-DEM supports load-balancing for parallel computations by using p4est's and deal.II's functionalities. Each cell owned by the local process is assigned a weight according to its computational load. Lethe-DEM then distributes the

sum of these weights evenly among the available processes. Users can specify the load-balancing frequency in the parameter-handler file. At the load-balancing iterations, Lethe-DEM re-equalizes the computational loads on the process according to the number of particles in the distributed simulation domain. Here, a load L_c is attributed to each process, which is a linear combination of the number of cells n_e and particles n_p owned by the process:

$$L_c = \alpha n_p + \beta n_e \tag{24}$$

where α and β are weights of particles and cells. During the redistribution, this load is balanced between the processes. Currently, the defined ratio of the weights is $\alpha/\beta = 10$. Considering the high number of particles compared to the number of cells in a DEM simulation, this weighting strategy focuses mainly on the number of particles owned by each process. We should mention that future investigations are required to optimize the ratio of the weights and investigate the performance of this linear weighting strategy, in particular once we tackle the coupling of DEM and CFD which possibly have conflicting load balancing needs. The same load balancing algorithm applied to a different application has been shown to only modestly influence the performance as long as the ratio remains reasonably close to an optimal ratio that is model dependent [38].

3.7. Other features

Some of the other useful features of Lethe-DEM are covered in this section. First of all, Lethe-DEM makes use of C++ templates to create two separate solvers for two-dimensional (`dem_2d`), and three-dimensional (`dem_3d`) simulations. Hence, both 2D and 3D simulations are supported without loss of performance using the same code base. Another important feature of Lethe-DEM is particle-line and particle-point contacts. Lethe-DEM automatically checks the input triangulation for possible boundary lines (edges) or points (vertices). Particles may collide with boundary faces, as well as boundary lines and points in specific geometries. If boundary lines or points exist in the input triangulation, the software considers the contact between particles and these boundary elements. Boundary lines may only exist in three-dimensional triangulations, while boundary points may exist in both two-dimensional and three-dimensional triangulations.

Another feature of Lethe-DEM is the motion of boundaries. Boundaries can move in Lethe-DEM. Currently, users can define two types of motion, namely rotational (Figure 8a) and translational (Figure 8b), in the parameter handler file. By specifying the boundary id, type of motion, and motion properties, Lethe-DEM implements the motion in the simulation. Another feature of Lethe-DEM is the definition of floating walls, as illustrated in Figure 8c. By specifying the normal vector of the floating wall (arrow in Figure 8c), a point on the wall, start and end times of the wall, Lethe-DEM defines it automatically in the simulation. This feature is especially convenient when simulating filling and discharge of particles in containers.

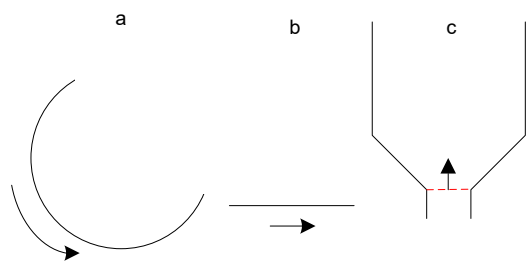


Figure 8: Schematics of a: rotational boundary motion, b: translational boundary motion, and c: a floating wall (red dashed line) with its normal vector (arrow).

4. Verification and validation

All the major functions and modules of Lethe are tested and verified using unit tests. New features added to the software are tested with at least one test to guarantee the accuracy and reproducibility of its outputs. Besides these tests, we have compared the outputs of Lethe-DEM with experimental data and simulations made using other software. We present the result of selected comparisons in the following.

4.1. Verification

In this section, we compare the outputs of Lethe-DEM and analytical solutions by three test cases.

4.1.1. Particle trajectory before and after a collision

In this test, we compare the height of a particle before and after a collision with a wall. The accuracy of particle-wall collisions extends to particle-particle collisions, since the same model and calculation procedure are employed for both. In Figure 9, we show the results of this comparison. The analytical solution for the position of the particle consists of three parts: free fall of the particle before the collision, contact with the wall, and motion after the collision. Equations (25a)–(25c) express the location of the particle during these steps [39, 40].

$$y(t) = h_0 - \frac{gt^2}{2} \tag{25a}$$

$$y(t-t_1) = (\alpha_1 \cos(w(t-t_1)) + \alpha_2 \sin(w(t-t_1))) \exp(-\beta_1 \omega_0 (t-t_1)) + r_p - \alpha_1 \tag{25b}$$

$$y(t-t_2) = r_p + v_2(t-t_2) - 0.5g(t-t_2)^2 \tag{25c}$$

where

$$\alpha_1 = \frac{g}{\omega_0^2}, \quad \alpha_2 = \frac{-\sqrt{2g(h_0 - r_p)} + \frac{\beta g}{\omega_0}}{w}, \quad \beta_1 = \frac{\eta_n}{2\sqrt{k_n m_i}}, \tag{26}$$

$$\omega_0 = \sqrt{\frac{k_n}{m_i}}, \quad w = \sqrt{1 - \beta^2 \omega_0}. \tag{27}$$

and h_0, y, t_1, t_2 and v_2 are initial particle height, vertical position, end times of steps one and two, and velocity of the particle at the end of step two, respectively. Relative percent error (RPE) values (0.82%, 0.68% and 0.32% for coefficients of restitution $e = 0.5$, 0.7 and 0.9, respectively for $\Delta t = 7 \times 10^{-6}$ s) demonstrate the accuracy of Lethe-DEM in modelling the contacts. The RPE is defined as:

$$\text{RPE} = \frac{\sum_{i=1}^{l} \left| \frac{f^{\text{exp}} - f^{\text{sim}}}{f^{\text{exp}}} \right|}{l} \cdot 100\% \tag{28}$$

where l, f^{exp} and f^{sim} denote the number of data points, the expected value from the analytical solution, and the simulated value from Lethe-DEM. The errors are attributed to the numerical integration, and they decrease with decreasing the time-step (following a second-order convergence for the Verlet scheme).

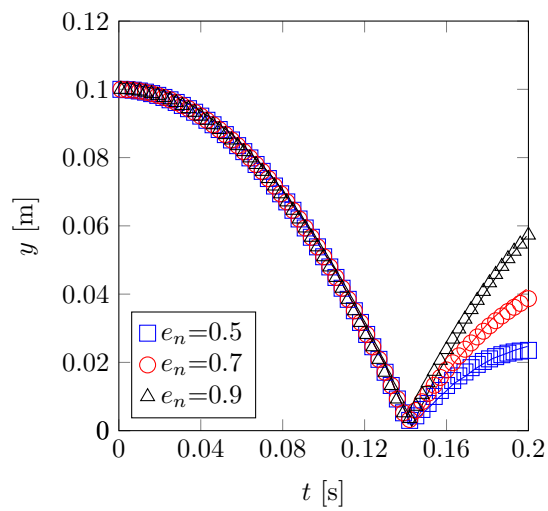


Figure 9: Comparison of particle position between Lethe-DEM (lines) and analytical solution (markers).

4.1.2. Orders of convergence of the integration schemes

We also measured the order of convergence for the explicit Euler and Velocity Verlet integration schemes in a unit test (`integration_schemes_accuracy`). This is an artificial problem with a known analytical solution which is very

similar to the collisions in DEM, and we use this test to investigate the orders of convergence of the integration schemes [5]. This test calculates and compares the errors with respect to the analytical solution of the explicit Euler and Velocity Verlet schemes in a pendulum oscillation problem. Table 3 reports the errors and orders of convergence. The calculated orders of convergence for the explicit Euler and the Velocity Verlet schemes are 1 and 2, respectively.

Table 3: Errors and orders of convergence of explicit Euler and Velocity Verlet integration schemes for a pendulum oscillation model (`integration_schemes_accuracy` unit test)

	`Explicit Euler`	`Velocity Verlet`
Error for $\Delta t = 0.1\,\mathrm{s}$	0.01275	0.00010
Error for $\Delta t = 0.05\,\mathrm{s}$	0.00634	2.63×10^{-5}
Error for $\Delta t = 0.025\,\mathrm{s}$	0.00316	6.57×10^{-6}
Order of convergence	≈ 1	≈ 2

4.1.3. Normal and tangential forces during a collision

In this section, we compare the normal and tangential forces of a particle-wall collision at a specified condition (reported in Table 4) with simulation results of another code (cemfDEM) [7]. Figure 10 shows the results of this comparison in a plot of the normal force over the normal overlap. The results show high accuracy (RPE = 2.98%, 3.54% and 3.52% for $e_n = 0.5$, 0.7 and 0.9, respectively) of Lethe-DEM compared to cemfDEM [40, 7]. Small discrepancies between the normal forces in Figure 10 arise from different non-linear contact models used in the codes. For instance, the coefficient and power of the normal overlap in the damping force are different.

Table 4: DEM parameters of the normal and tangential force calculations in Figures 10 and 11

Property	**Value**
d_p	$2\,\mathrm{mm}$
ρ_p	$7850\,\mathrm{kg\,m^{-3}}$
Y_p, Y_w	$200\,\mathrm{GPa}$
ν_p, ν_w	0.3
μ_p, μ_w	0.3
e_p, e_w	$0.5 - 0.9$
Impact velocity	$0.1 - 2\,\mathrm{m\,s^{-1}}$

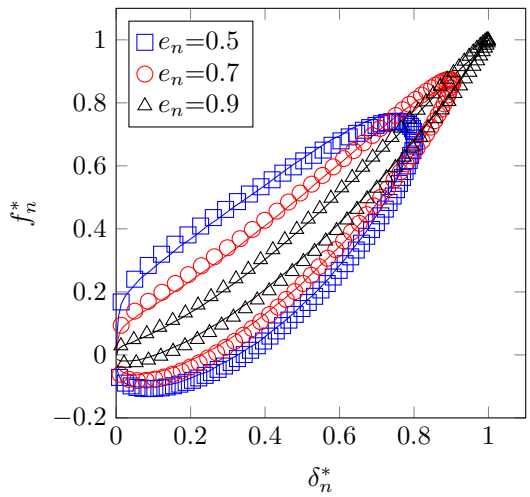


Figure 10: Comparison of normal overlap - force between Lethe-DEM (lines) and cemfDEM [7] (markers).

With the DEM parameters reported in Table 4, we calculate the contact tangential force at three different contact angles: 5°, 25° and 65° (Figure 11). At contact angles of 25° and 65°, the tangential force is limited by Coulomb's limit (Equation (15)). Other research [41, 7] report similar trends of tangential force at various contact angles.

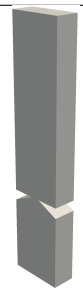
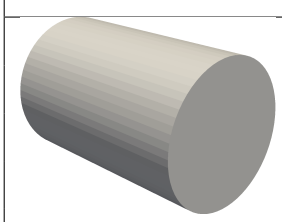
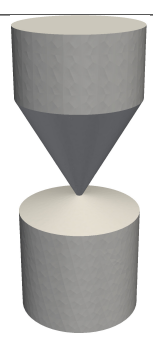
4.2. Validation

We validate Lethe-DEM with experimental results of a flat-bottomed silo (Table 5-a), a wedge-shaped silo (Table 5-b), and a rotating drum (Table 5-c) filled with granular material. We chose these systems since both the normal and tangential forces govern their dynamics. In other words, the normal and tangential force calculations and integration are tested in these validations.

4.2.1. Flat-bottomed silo

Balevivcius et al. [42] measured the particle velocity using particle image velocimetry and DEM in a flat-bottomed silo with $n_p = 18,000$ particles. Table 5-a shows the geometry of the silo and reports the physical properties of the simulation [42]. An animation (silo1.mp4) of this simulation is available in the supplementary materials. Figure 12 shows the lateral distributions of the vertical (Figure 12-a) and lateral (Figure 12-b) components of particle velocity in this flat-bottomed silo at three heights ($z = 0.05\,\mathrm{m}$, $0.1\,\mathrm{m}$, and $0.15\,\mathrm{m}$). RPE values of the axial velocity distributions are 15.4%, 25.6% and 19.0% at z= 0.05, 0.1 and 0.15, respectively. RPE values of lateral velocity distributions are also less than 17%. Lethe-DEM predicts the trends and velocity magnitudes with acceptable accuracy, and comparable DEM results in the same work [42]

Table 5: Geometries of the validation and scalability simulations and the corresponding DEM parameters. (a) flat-bottomed silo, (b) wedge-shaped silo, (c) rotating drum, and (d) cylindrical silo

Geometry	**Properties**	**Value**	**Case**
	d_p	$7.5\,\mathrm{mm}$	
	ρ_p	$1290\,\mathrm{kg\,m^{-3}}$	
	ν_p, ν_w	0.2	(a)
	μ_p, μ_w	$0.29, 0.2$	
	e_p, e_w	0.56	
	$\mathrm{d}t$, t_f	10^{-5}s, $6\,\mathrm{s}$	
	d_p	$5.83\,\mathrm{mm}$	
	ρ_p	$600\,\mathrm{kg\,m^{-3}}$	
	ν_p, ν_w	0.5	
	μ_p, μ_w	0.5	(b)
	e_p, e_w	0.7	
	f_{LB}	$0.5\,\mathrm{Hz}$	
	$\mathrm{d}t$, t_f	10^{-5}s, $40\,\mathrm{s}$	
	d_p	$3\,\mathrm{mm}$	
	ρ_p	$2500\,\mathrm{kg\,m^{-3}}$	
	ν_p, ν_w	0.2	
	μ_p, μ_w	0.85	(c)
	e_p, e_w	$0.97, 0.85$	
	t_{LB}	$1.5\,\mathrm{s}$	
	$\mathrm{d}t$, t_f	10^{-5}s, $10\,\mathrm{s}$	
	d_p	$4\,\mathrm{mm}$	
	ρ_p	$600\,\mathrm{kg\,m^{-3}}$	
	ν_p, ν_w	0.5	
	μ_p, μ_w	0.5	(d)
	e_p, e_w	0.7	
	f_{LB}	$2\,\mathrm{Hz}$	
	$\mathrm{d}t$, t_f	10^{-5}s, $6\,\mathrm{s}$	

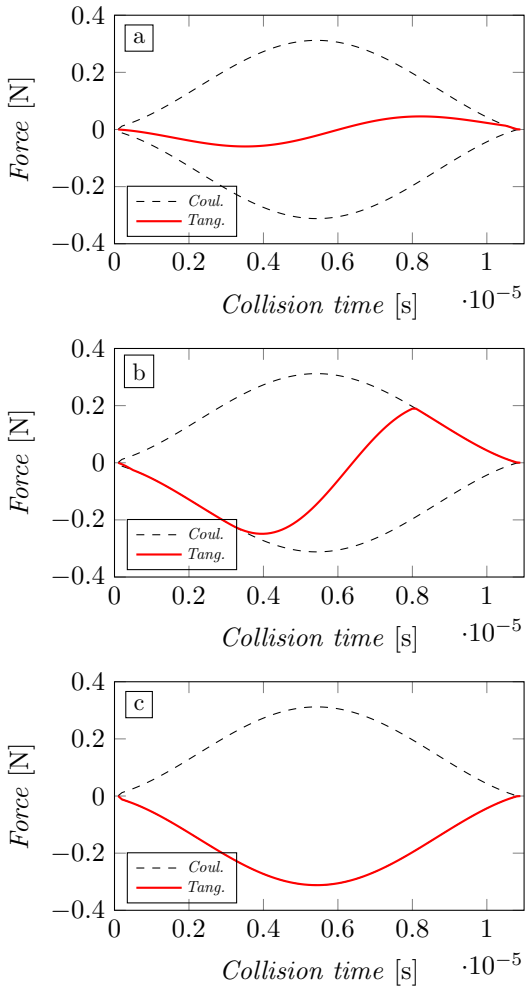


Figure 11: Tangential force at the physical conditions reported in Table 4 and contact angles of (a) 5°, (b) 25°, and (c) 65° and the corresponding Coulomb's limit.

have similar error values. Since both DEM results show deviations from the experiments, the remaining errors are attributed to systematic differences between the DEM model setup and the experimental setup – for example in the material properties or the used contact model parameters – and not an error in the numerical implementation of Lethe-DEM.

4.2.2. Wedge-shaped silo

Similar to the flat-bottomed silo, we compare the lateral distributions of axial and lateral components of particle velocity in a wedge-shaped silo with

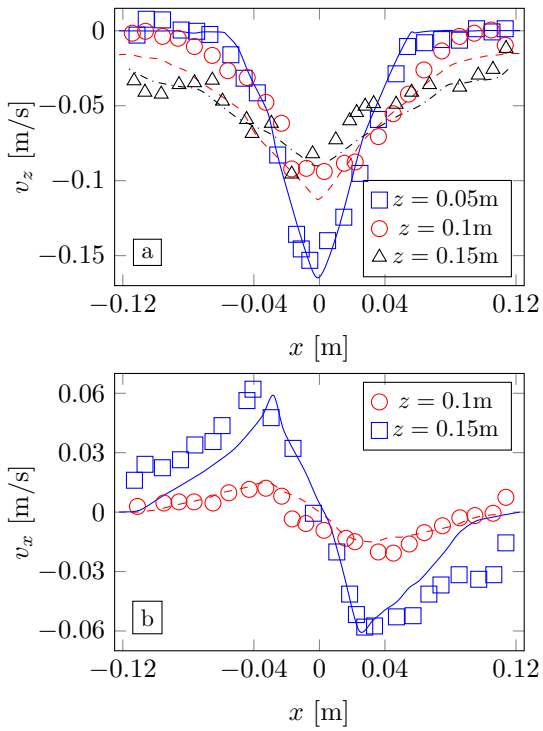


Figure 12: Lateral distributions of (a) vertical, and (b) lateral components of particles velocity in a flat-bottomed silo, obtained from Lethe-DEM (lines) and experiments (markers) using PIV [42].

$n_p = 132\,300$ particles. Golshan et al. [43] measured time-averaged particle velocity in a wedge-shaped silo using particle image velocimetry (PIV) and DEM. Table 5-b reports the DEM physical properties of this simulation [43]. In the experiments, non-spherical barley grains were used, while in the DEM simulations we replaced them with spheres with the same volume as the non-spherical particles. Interested readers may find the simulation animation (silo2.mp4) in the supplementary materials. Figure 13 shows the lateral distributions of vertical and lateral components of particle velocity in this silo. RPE values of the comparison between DEM and experiments are less than 20% for all the velocity distributions in Figure 13. Errors (overestimation of the velocity values) are attributed to replacing non-spherical barley grain with spherical particles.

Load-balancing redistributes the computational load on the processes every 2 s in the wedge-shaped simulation. Figure 14 shows the distribution of core domains during the simulation (on 32 processes). At $t = 8$ s (iteration 8×10^6), the majority of the cores (26 processes) are in the hopper section (upper part), and as the simulation proceeds the processes move toward the bottom of the geometry. At $t = 38$ s (iteration 38×10^6), the particles are in the bottom of

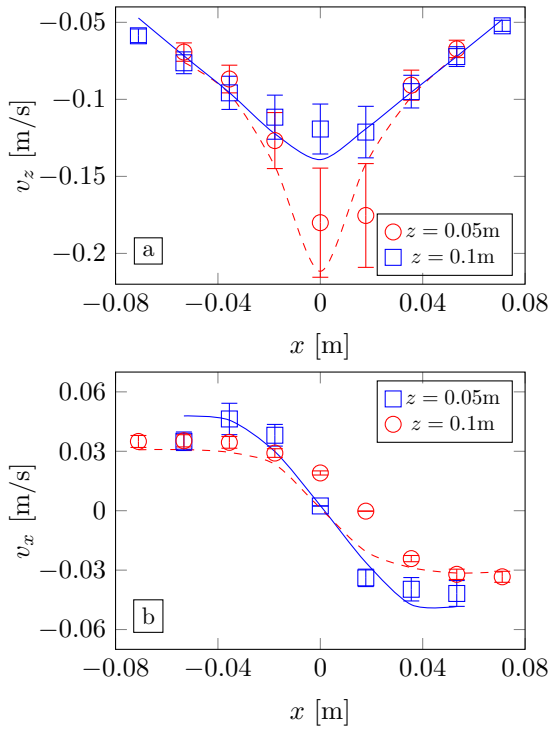


Figure 13: Lateral distributions of (a) vertical, and (b) lateral components of particles velocity in a wedge-shaped silo, obtained from Lethe-DEM (lines) and experiments (markers) using PIV [43].

the container and only one process handles all the cells located in the hopper section. We also compared the velocity distributions of particles calculated from simulations on 1-128 processes and observed identical velocity distributions. It means that the accuracy of a simulation does not diminish by performing it in parallel.

4.2.3. Rotating drum

Here we compare the simulation results of a rotating drum containing $n_p = 226\,080$ particles with the experiments of Alizadeh et al. [44]. Table 5-c reports the simulation parameters according to the physical properties of glass beads (particles) and plexiglass (wall). In the simulation of the wedge-shaped silo, we used a frequent load-balancing with the frequency of $f_{LB} = 0.5\,\text{Hz}$ since the particles move from the hopper to the bottom container during the simulation. In the simulation of the rotating drum, after reaching the steady-state condition ($t \approx 1.5\,\text{s}$), the particles occupy a specific part of the rotating cylinder, and the rest of the simulation domain is without any particles. From this point on, no load-balancing is performed anymore. As a result, we use a single-step load

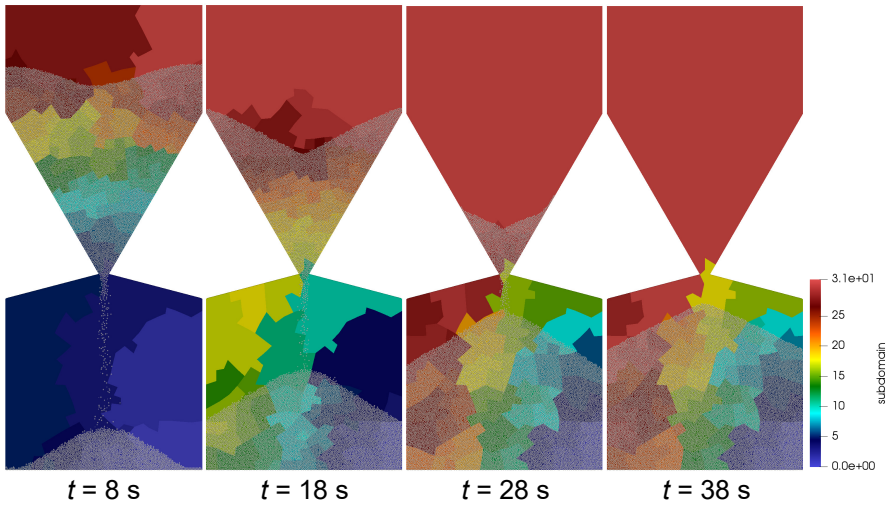


Figure 14: Distribution of core domains as a function time in the simulation of wedge-shaped silo on 32 cores.

balance at $t = 1.5\,\mathrm{s}$, when all the particles have been inserted and have reached the particle bed.

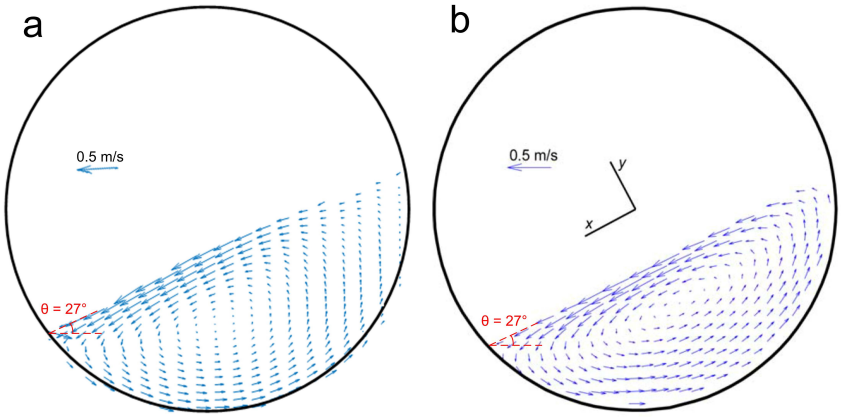


Figure 15: Time-averaged velocity field of the particles in a rotating drum from (a) Lethe-DEM simulation and (b) experiments [44]. The free surface angle is equal to 27° from the simulation and experiments.

Figure 15 shows the time-averaged velocity field obtained from the simulation and experiments [44]. We also compare the free surface angle with a horizontal line on this figure. Figure 16 compares the streamwise velocity profiles in x and y directions (these directions are defined in Figure 15) between Lethe-DEM and experiments carried out using radioactive particle tracking [44]. These profiles show the particles' velocity in (a) parallel and (b) perpendicular planes to the

free surface. RPE values of these profiles are less than 20%. Interested readers may find an animation of this simulation (drum.mp4) in the supplementary materials.

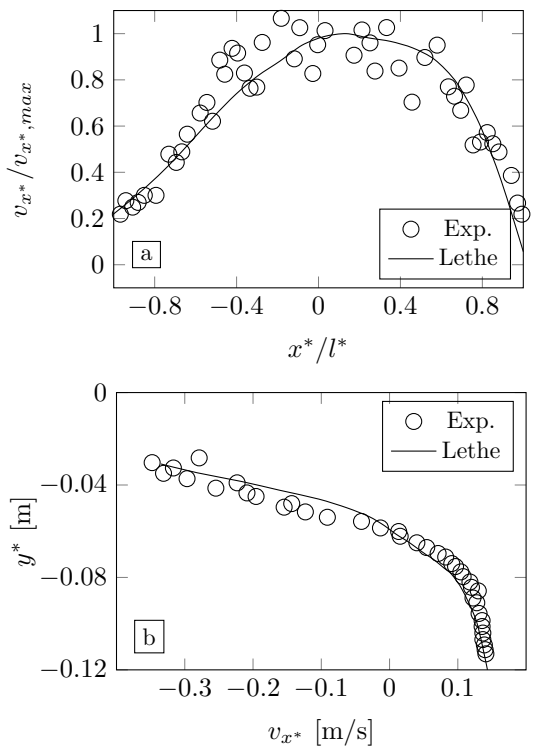


Figure 16: Streamwise velocity profiles in (a) x (parallel to the free surface) and (b) y (perpendicular to the free surface) directions.

4.3. Scalability

In this section, we present the results of strong and weak scaling analyses. For the strong scaling, we use the simulations of the wedge-shaped silo (Table 5-b) and the rotating drum (Table 5-c), while for the weak scaling analysis, we use a dam break case. We used the Graham cluster for the strong scaling analysis on the wedge-shaped silo as well as the Cedar cluster for strong scaling analysis on the rotating drum and weak scaling analysis. Each node on both clusters consists of 32 cores in the form of 2 x Intel E5-2683 v4 Broadwell @ 2.1 GHz processors with an available memory of 128 GiB per node.

4.3.1. Strong scaling

Strong scaling studies are not prevalent for DEM codes, since DEM codes generally fail to provide high speed-up values on large core numbers for constant

problem size. For Lethe-DEM however, due to the unique design of the software, we report strong scaling analyses for the two cases investigated with and without load-balancing. To the authors' best knowledge, these results are amongst the few strong scaling analysis of DEM softwares.

As a first test, we consider a wedge-shaped silo on 1, 2, 4, 8, 16, 32, and 64 cores (i.e., using up to two nodes), and measure the simulation time, keeping the number of particles and the size of the domain fixed. The simulation of wedge-shaped silo calls load balancing every 2 s (i.e. every 2×10^6 steps) to distribute the computational load of the simulations on all the cores as particles move in the domain. Figure 17 shows the result of this comparison. For lower core counts ($n_c < 16$), the strong scaling (with load-balancing) is nearly optimal. However, the speed-up deviates from the ideal line as the number of cores increases and the number of particles per core drops below 8,000 particles per core. This is because with increasing number of cores, the ratio of ghost particles to owned particles per process increases, which adds a non-negligible additional cost to the simulation. Furthermore, higher usage of shared resources like the memory bandwidth (up to 32 cores) or the network (64 cores) also leads to a deviation from the ideal behavior. Enabling load-balancing decreases the computational time by around 30% depending on the number of processors. The simulation time per iteration decreases from 2.07 s to 0.086 s for this simulation on 1 to 64 processes with load-balancing.

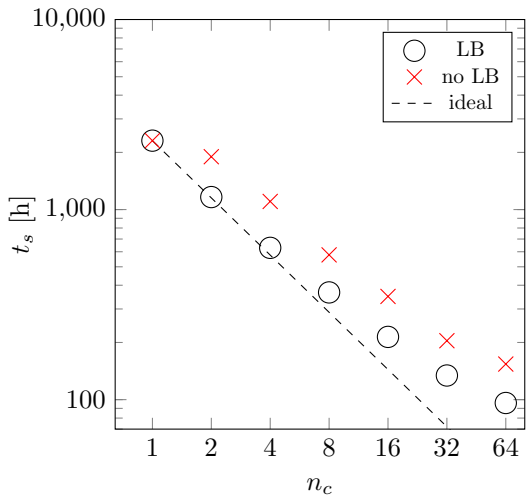


Figure 17: Simulation times of the wedge-shaped silo with (LB) and without load-balancing (no LB).

As a second test, we simulate the rotating drum on 1 to 6 compute nodes, each with 32 cores. Figure 18 shows the simulation times of these simulations. Similar to the simulation of the wedge-shaped silo (Figure 14), the deviation from the ideal speed-up increases with the number of cores. The best scalability

is obtained at $4000 < n_p/n_c < 20\,000$ (1–3 nodes or 32–96 cores for the 226 080 particles in this simulation). Using load-balancing decreases the computational time by around 35%. The simulation time per iteration decreases from 0.099 s to 0.025 s for this simulation on 1 to 6 nodes with load-balancing.

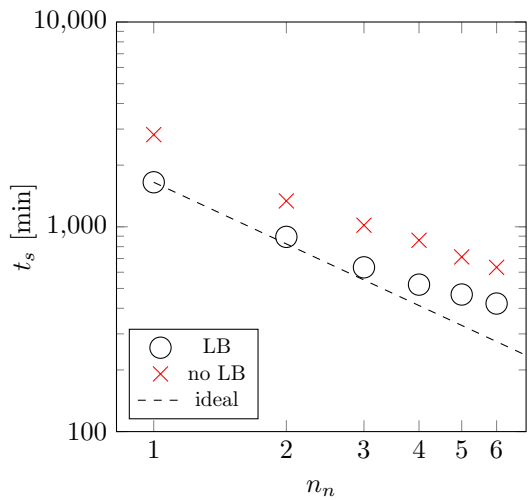


Figure 18: Simulation time of rotating drum case as a function of number of nodes with and without load-balancing. Each node has 32 cores.

4.3.2. Weak scaling

We use dam break simulations for weak-scaling analysis of Lethe-DEM. In this case (illustrated in Figure 19), particles are inserted and packed in the left half of a box (grey zone in Figure 19). A floating wall dam (red line in Figure 19) holds the particles in the left half. At $t = 0.5$ s, the dam is removed and particles fill the entire length of the box. We choose this scenario for weak-scaling analysis, since all the major functions of Lethe-DEM (including insertion, particle-particle and particle-wall contacts) are included and the particles can flow in the entire simulation domain. In weak-scaling analyses, the number of particles per process is kept constant. To this end, we increase the depth (z in Figure 19) of the box proportionally to the number of particles in the simulation. Consequently, particles owned by different processes are in contact. We use four values of n_p/n_c : 5k, 10k, 15k and 20k. Table 6 reports the values of z and n_p in weak-scaling simulations. Dynamic load-balancing redistributes the computational load between the processes during the weak-scaling simulations. In the dynamic load balancing approach, the code automatically detects the load-balancing steps from the distribution of particles and cells on the processes.

Figure 20 shows the results of the weak-scaling analysis by reporting the simulation times (t_s) against the number of nodes. The simulation times increase slightly with the number of processes at all the n_p/n_c values. The ratio

Table 6: Number of particles and depth of the box (illustrated in Figure 19) used for weak-scaling analysis. k and M stand for $\times 10^3$ and $\times 10^6$, respectively.

		n_p/n_c			
		5k	**10k**	**15k**	**20k**
n_c	**32**	$z = 0.01\,\mathrm{m}$ $n_p = 160\,\mathrm{k}$	$z = 0.02\,\mathrm{m}$ $n_p = 320\,\mathrm{k}$	$z = 0.03\,\mathrm{m}$ $n_p = 480\,\mathrm{k}$	$z = 0.04\,\mathrm{m}$ $n_p = 640\,\mathrm{k}$
	64	$z = 0.02\,\mathrm{m}$ $n_p = 320\,\mathrm{k}$	$z = 0.04\,\mathrm{m}$ $n_p = 640\,\mathrm{k}$	$z = 0.06\,\mathrm{m}$ $n_p = 960\,\mathrm{k}$	$z = 0.08\,\mathrm{m}$ $n_p = 1.28\,\mathrm{M}$
	96	$z = 0.03\,\mathrm{m}$ $n_p = 480\,\mathrm{k}$	$z = 0.06\,\mathrm{m}$ $n_p = 960\,\mathrm{k}$	$z = 0.09\,\mathrm{m}$ $n_p = 1.44\,\mathrm{M}$	$z = 0.12\,\mathrm{m}$ $n_p = 1.92\,\mathrm{M}$
	128	$z = 0.04\,\mathrm{m}$ $n_p = 640\,\mathrm{k}$	$z = 0.08\,\mathrm{m}$ $n_p = 1.28\,\mathrm{M}$	$z = 0.12\,\mathrm{m}$ $n_p = 1.92\,\mathrm{M}$	$z = 0.16\,\mathrm{m}$ $n_p = 2.56\,\mathrm{M}$
	160	$z = 0.05\,\mathrm{m}$ $n_p = 800\,\mathrm{k}$	$z = 0.1\,\mathrm{m}$ $n_p = 1.6\,\mathrm{M}$	$z = 0.15\,\mathrm{m}$ $n_p = 2.4\,\mathrm{M}$	$z = 0.2\,\mathrm{m}$ $n_p = 3.2\,\mathrm{M}$
	224	$z = 0.07\,\mathrm{m}$ $n_p = 1.12\,\mathrm{M}$	$z = 0.14\,\mathrm{m}$ $n_p = 2.24\,\mathrm{M}$	$z = 0.21\,\mathrm{m}$ $n_p = 3.36\,\mathrm{M}$	$z = 0.28\,\mathrm{m}$ $n_p = 4.48\,\mathrm{M}$
	256	$z = 0.08\,\mathrm{m}$ $n_p = 1.28\,\mathrm{M}$	$z = 0.16\,\mathrm{m}$ $n_p = 2.56\,\mathrm{M}$	$z = 0.24\,\mathrm{m}$ $n_p = 3.84\,\mathrm{M}$	$z = 0.32\,\mathrm{m}$ $n_p = 5.12\,\mathrm{M}$
	320	$z = 0.1\,\mathrm{m}$ $n_p = 1.6\,\mathrm{M}$	$z = 0.2\,\mathrm{m}$ $n_p = 3.2\,\mathrm{M}$	$z = 0.3\,\mathrm{m}$ $n_p = 4.8\,\mathrm{M}$	$z = 0.4\,\mathrm{m}$ $n_p = 6.4\,\mathrm{M}$

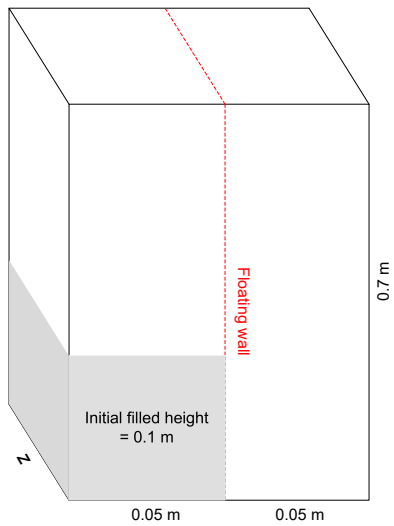


Figure 19: Geometry of breaking of dam simulations used for weak-scaling anslysis.

of simulation times on the maximum and minimum number of processes are 2.5 and 1.2 for $n_p/n_c = 5$k and 20k, respectively. Note that for $n_p/n_c = 20$k on more than 6 nodes the simulation time does not further increase, which represents near ideal weak scalability in this range and suggests Lethe-DEM is well suited for large-scale parallel simulations. We also performed the weak-scaling analysis for $n_p/n_c = 10$k without load-balancing. We observe that the ratio of simulation times without and with load-balancing ($t_s^{\text{no LB}}/t_s^{\text{LB}}$) is ≈ 2.4 at $n_p/n_c = 10$k more or less independent of model size and therefore particle number. This means not only is the load-balancing very effective for this application, it is also scaling to larger problem sizes. The simulation time per iteration on 1–10 nodes changes from 0.112 s to 0.285 s for 5k/core and from 0.578 s to 0.691 s for 20k/core. Interested readers may find a simulation animation (weak_scaling.mp4) in the supplementary materials.

4.4. Large scale feasibility

We investigate the performance of Lethe-DEM in a simulation of a large system. To this end, we simulate the packing of $n_p > 4.3 \times 10^6$ particles in a three-dimensional silo. The silo geometry is a three-dimensional (cylindrical) version of the silo used in Section 4.2.2. Table 5-d reports the properties of the particles and simulation parameters. Figure 21 shows the packing configuration at $t = 6$ s. This simulation is performed on 320 cores and takes ≈ 6 days ($\approx 46\,800$ core-hours) for 650 000 time steps.

5. Impact and conclusions

In this work, we introduced an open-source parallel DEM software with load-balancing: Lethe-DEM. Lethe-DEM consists of two solvers for two-dimensional

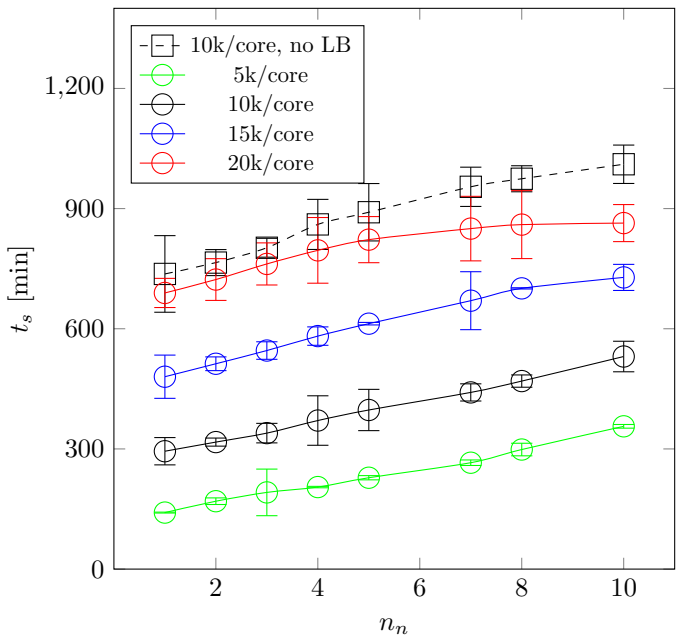


Figure 20: Simulation times of weak-scaling analysis on up to $n_n = 10$ nodes, at different values of n_p/n_c, illustrated in Table 6. Each node has 32 cores. Error bars indicate the standard deviation across three runs of the same simulations.

Figure 21: Packing of 4.3M particles in a three-dimensional silo computed on 320 cores.

(`dem_2d`) and three-dimensional (`dem_3d`) simulations. We designed this software in the framework of Lethe, a high-order CFD solver for incompressible flows.

Lethe-DEM uses a background grid that is compatible with other solvers in Lethe, which will allow coupling DEM and CFD simulations in the future. Several tests and comparisons with analytical solutions, models, and experiments verify and validate Lethe-DEM. For validation of Lethe-DEM, we compared its results with analogous experiments in a flat-bottomed silo, a wedge-shaped silo, and a rotating drum. Load-balancing allows Lethe-DEM to decrease the simulation times by roughly $25 - 70\%$ in parallel simulations compared to unbalanced parallel computations. To investigate the strong scalability of Lethe-DEM, we compared the simulation time of benchmark cases on 1–192 processes with and without load-balancing. Results show that Lethe-DEM scales optimally when there are $\approx 15\,000$ particles per process, while scalability decreases for fewer particles per process due to increased communication costs. We also performed a weak-scaling analysis of Lethe-DEM using a dam break simulation on 32–320 processes and showed optimal weak scalability for more than $\approx 20\,000$ particles per process. We finally demonstrated the computational performance and scalability of Lethe-DEM in the simulation of a cylindrical silo with 4.3×10^6 particles on 320 cores.

Lethe-DEM is under continuous development and we expect to add the following features in the near future:

- Unresolved coupling with CFD [19, 6];
- Heat transfer via conduction, convection and radiation mechanisms;
- Particle size change (particle size growth and shrinkage);
- Cohesive force models;
- Coarse graining;
- Electrostatic force.

The growing availability of large-scale massively parallel computing systems allows the modeling of physical processes at the pilot and industrial scales that can not be resolved by current simulations. However, in order to take advantage of this computing power we need DEM software that can leverage next generation HPC systems. Lethe-DEM was designed to use this opportunity and eventually enable coupled CFD-DEM models on the next generation of supercomputers.

6. Acknowledgements

The authors would like to acknowledge support received by the deal.II community. Without such rigorously developed open source project such as deal.II, the present work could have never been achieved. The authors would like to acknowledge the support received from Calcul Québec and Compute Canada. Computations shown in this work were made on the supercomputer Beluga, Cedar and Graham managed by Calcul Québec and Compute Canada. The

operation of these supercomputers is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la science et de l'innovation du Québec (MESI) and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT). This project was partially funded by the Natural Sciences and Engineering Research Council via NSERC Grant RGPIN-2020-04510. RG was supported by the Computational Infrastructure for Geodynamics (CIG), through the National Science Foundation under Award No. EAR-1550901, administered by The University of California-Davis.

7. Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix A. An example of the input files

Listing 1: Example of sections in a PRM file

```
# ----------------------------------------------------
# Simulation and IO Control
#----------------------------------------------------
subsection simulation control
  set time step                                     = 1e-5
  set time end                                      = 0.4
  set log frequency                                 = 1000000
  set output frequency                              = 1000000
end

#----------------------------------------------------
# Restart
#----------------------------------------------------
subsection restart
  set restart                                       = true
  set checkpoint                                    = true
  set filename                                      = sliding_restart
end

#----------------------------------------------------
# Timer
#----------------------------------------------------
subsection timer
  set type                                          = none
end

#----------------------------------------------------
# Test
#----------------------------------------------------
```

```
subsection test
  set enable                                        = true
end

# ----------------------------------------------------
# Model parameters
#----------------------------------------------------
subsection model parameters
  set contact detection method                      = constant
  set contact detection frequency                   = 20
  set neighborhood threshold                        = 1.3
  set particle particle contact force method        = pp_nonlinear
  set particle wall contact force method            = pw_nonlinear
  set rolling resistance torque method              = constant_resistance
  set integration method                            = velocity_verlet
end

#----------------------------------------------------
# Physical Properties
#----------------------------------------------------
subsection physical properties
  set gx                                            = 0.0
  set gy                                            = 0.0
  set gz                                            = -9.81
  set number of particle types                      = 1
  subsection particle type 0
    set size distribution type                      = uniform
    set diameter                                    = 0.005
    set number                                      = 20
    set density                                     = 2000
    set young modulus particle                      = 1000000
    set poisson ratio particle                      = 0.3
    set restitution coefficient particle            = 0.3
    set friction coefficient particle               = 0.1
    set rolling friction particle                   = 0.05
  end
  set young modulus wall                            = 1000000
  set poisson ratio wall                            = 0.3
  set restitution coefficient wall                  = 0.3
  set friction coefficient wall                     = 0.1
  set rolling friction wall                         = 0.05
end

#----------------------------------------------------
# Insertion Info
#----------------------------------------------------
subsection insertion info
  set insertion method                              = non_uniform
  set inserted number of particles at each time step= 20
  set insertion frequency                           = 2000000
```

```
  set insertion box minimum x                       = -0.05
  set insertion box minimum y                       = -0.05
  set insertion box minimum z                       = -0.06
  set insertion box maximum x                       = 0.05
  set insertion box maximum y                       = 0.05
  set insertion box maximum z                       = 0
  set insertion distance threshold                  = 2
  set insertion random number range                 = 0.75
  set insertion random number seed                  = 19
end

#-----------------------------------------------
# Mesh
#-----------------------------------------------
subsection mesh
  set type                                          = dealii
  set grid type                                     = hyper_cube
  set grid arguments                                = -0.07:0.07:true
  set initial refinement                            = 3
end

#-----------------------------------------------
# Boundary Motion
#-----------------------------------------------
subsection boundary motion
  set number of boundary motion                     = 1
  subsection moving boundary 0
    set boundary id                                 = 4
    set type                                        = translational
    set speed x                                     = 0.15
    set speed y                                     = 0
    set speed z                                     = 0
  end
end
```

References

[1] P. Richard, M. Nicodemi, R. Delannay, P. Ribiere, D. Bideau, Slow relaxation and compaction of granular systems, Nature materials 4 (2) (2005) 121–128.

[2] S. Larsson, Particle methods for modelling granular material flow, Ph.D. thesis, Luleå University of Technology (2019).

[3] R. M. Nedderman, Statics and kinematics of granular materials, Cambridge University Press, 2005.

[4] S. Golshan, R. Zarghami, K. Saleh, Modeling methods for gravity flow of granular solids in silos, Reviews in Chemical Engineering 1 (ahead-of-print) (2019).

[5] B. Blais, D. Vidal, F. Bertrand, G. S. Patience, J. Chaouki, Experimental methods in chemical engineering: Discrete element method—dem, The Canadian Journal of Chemical Engineering 97 (7) (2019) 1964–1973.

[6] S. Golshan, R. Sotudeh-Gharebagh, R. Zarghami, N. Mostoufi, B. Blais, J. Kuipers, Review and implementation of CFD-DEM applied to chemical process systems, Chemical Engineering Science (2020) 115646.

[7] H. R. Norouzi, R. Zarghami, R. Sotudeh-Gharebagh, N. Mostoufi, Coupled CFD-DEM modeling: formulation, implementation and application to multiphase flows, John Wiley & Sons, 2016.

[8] H.-C. Dan, Z. Zhang, J.-Q. Chen, H. Wang, Numerical simulation of an indirect tensile test for asphalt mixtures using discrete element method software, Journal of Materials in Civil Engineering 30 (5) (2018) 04018067.

[9] C. Coetzee, Calibration of the discrete element method, Powder Technology 310 (2017) 104–142.

[10] M. F. Wolff, V. Salikov, S. Antonyuk, S. Heinrich, G. A. Schneider, Three-dimensional discrete element modeling of micromechanical bending tests of ceramic–polymer composite materials, Powder technology 248 (2013) 77–83.

[11] F. A. Tavarez, M. E. Plesha, Discrete element method for modelling solid and particulate materials, International journal for numerical methods in engineering 70 (4) (2007) 379–404.

[12] B. K. Cook, R. P. Jensen, Discrete element methods, American Society of Civil Engineers, 2002.

[13] Y. Ismail, Y. Sheng, D. Yang, J. Ye, Discrete element modelling of unidirectional fibre-reinforced polymers under transverse tension, Composites Part B: Engineering 73 (2015) 118–125.

[14] J.-F. Jerier, B. Hathong, V. Richefeu, B. Chareyre, D. Imbault, F.-V. Donze, P. Doremus, Study of cold powder compaction by using the discrete element method, Powder Technology 208 (2) (2011) 537–541.

[15] W. R. Ketterhagen, M. T. am Ende, B. C. Hancock, Process modeling in the pharmaceutical industry using the discrete element method, Journal of pharmaceutical sciences 98 (2) (2009) 442–470.

[16] E. Tijskens, H. Ramon, J. De Baerdemaeker, Discrete element modelling for process simulation in agriculture, Journal of sound and vibration 266 (3) (2003) 493–514.

[17] J. M. Boac, R. K. Ambrose, M. E. Casada, R. G. Maghirang, D. E. Maier, Applications of discrete element method in modeling of grain postharvest operations, Food Engineering Reviews 6 (4) (2014) 128–149.

[18] B. Blais, M. Lassaigne, C. Goniva, L. Fradette, F. Bertrand, Development of an unresolved cfd–dem model for the flow of viscous suspensions and its application to solid–liquid mixing, Journal of Computational Physics 318 (2016) 201–221.

[19] A. Bérard, G. S. Patience, B. Blais, Experimental methods in chemical engineering: Unresolved cfd-dem, The Canadian Journal of Chemical Engineering 98 (2) (2020) 424–440.

[20] C. Kloss, C. Goniva, A. Hager, S. Amberger, S. Pirker, Models, algorithms and validation for opensource dem and cfd–dem, Progress in Computational Fluid Dynamics, an International Journal 12 (2-3) (2012) 140–152.

[21] T. Weinhart, L. Orefice, M. Post, M. P. van Schrojenstein Lantman, I. F. Denissen, D. R. Tunuguntla, J. Tsang, H. Cheng, M. Y. Shaheen, H. Shi, et al., Fast, flexible particle simulations—an introduction to mercurydpm, Computer physics communications 249 (2020) 107129.

[22] D. Solutions, Edem 2.3 user guide, Edinburgh, Scotland, UK (2010).

[23] T. Forgber, P. Toson, S. Madlmeir, H. Kureck, J. G. Khinast, D. Jajcevic, Extended validation and verification of xps/avl-fire™, a computational cfd-dem software platform, Powder Technology 361 (2020) 880–893.

[24] B. Blais, L. Barbeau, V. Bibeau, S. Gauvin, T. El Geitani, S. Golshan, R. Kamble, G. Mikahori, J. Chaouki, Lethe: An open-source parallel high-order adaptative cfd solver for incompressible flows, SoftwareX 12 (2020) 100579.

[25] D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, et al., The deal. ii library, version 9.2, Journal of Numerical Mathematics 28 (3) (2020) 131–146.

[26] M. Kronbichler, K. Kormann, A generic interface for parallel cell-based finite element operator application, Computers and Fluids 63 (2012) 135–147.

[27] C. Burstedde, L. C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133.

[28] C. Burstedde, Parallel tree algorithms for amr and non-standard data access, ACM Trans. Math. Softw. 46 (4) (Nov. 2020). doi:10.1145/3401990. URL https://doi.org/10.1145/3401990

[29] P. A. Cundall, O. D. Strack, A discrete numerical model for granular assemblies, geotechnique 29 (1) (1979) 47–65.

[30] Y. Tsuji, T. Tanaka, T. Ishida, Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe, Powder technology 71 (3) (1992) 239–250.

[31] Y. Zhou, B. Wright, R. Yang, B. H. Xu, A.-B. Yu, Rolling friction in the dynamic simulation of sandpile formation, Physica A: Statistical Mechanics and its Applications 269 (2-4) (1999) 536–553.

[32] N. V. Brilliantov, T. Pöschel, Rolling friction of a viscous sphere on a hard plane, EPL (Europhysics Letters) 42 (5) (1998) 511.

[33] B. Delacroix, A. Bouarab, L. Fradette, F. Bertrand, B. Blais, Simulation of granular flow in a rotating frame of reference using the discrete element method, Powder Technology 369 (2020) 146–161.

[34] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities, International journal for numerical methods in engineering 79 (11) (2009) 1309–1331.

[35] K. Martin, B. Hoffman, Mastering CMake: a cross-platform build system, Kitware, 2010.

[36] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the mpi message passing interface standard, Parallel computing 22 (6) (1996) 789–828.

[37] W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Trans. Math. Softw. 38 (2) (Jan. 2012). doi:10.1145/2049673.2049678.
URL https://doi.org/10.1145/2049673.2049678

[38] R. Gassmöller, H. Lokavarapu, E. Heien, E. G. Puckett, W. Bangerth, Flexible and scalable particle-in-cell methods with adaptive mesh refinement for geodynamic computations, Geochemistry, Geophysics, Geosystems 19 (9) (2018) 3596–3604.

[39] R. Garg, J. Galvin, T. Li, S. Pannala, Open-source mfix-dem software for gas–solids flows: Part i—verification studies, Powder Technology 220 (2012) 122–137.

[40] H. R. Norouzi, R. Zarghami, N. Mostoufi, New hybrid CPU-GPU solver for CFD-DEM simulation of fluidized beds, Powder Technology 316 (2017) 233–244.

[41] H. Kruggel-Emden, S. Wirtz, V. Scherer, A study on tangential force laws applicable to the discrete element method (dem) for materials with viscoelastic or plastic behavior, Chemical Engineering Science 63 (6) (2008) 1523–1541.

[42] R. Balevičius, A. Maknickas, I. Sielamowicz, Experimental, continuum-and dem-based velocities in a flat-bottomed bin, Powder Technology 377 (2021) 297–307.

[43] S. Golshan, B. Esgandari, R. Zarghami, B. Blais, K. Saleh, Experimental and dem studies of velocity profiles and residence time distribution of non-spherical particles in silos, Powder Technology 373 (2020) 510–521.

[44] E. Alizadeh, O. Dubé, F. Bertrand, J. Chaouki, Characterization of mixing and size segregation in a rotating drum by a particle tracking method, AIChE Journal 59 (6) (2013) 1894–1905.