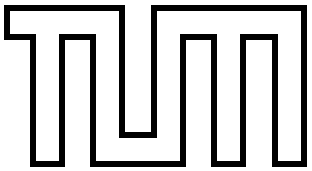


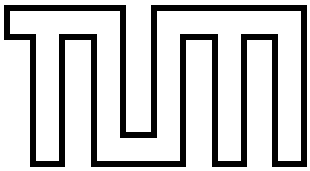
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Exploring the Discrete Element Method: Simulation of Granular Particles using AutoPas

Joon Kim



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Exploring the Discrete Element Method: Simulation of Granular Particles using AutoPas

Untersuchung von Discrete Element Method: Simulation von Granulat-Partikeln mit AutoPas

Author:	Joon Kim
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	Manish Kumar Mishra, M.Sc.
Date:	04.03.2025

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 04.03.2025 Joon Kim

Acknowledgements

I would like to thank Manish not only for the insightful discussions throughout the course of this thesis but also for his thoughtful and constructive feedback.

I am also grateful to the Chair of Scientific Computing and Professor Dr. Hans-Joachim Bungartz for providing me with the opportunity to work on this topic.

Moreover, I gratefully acknowledge the computational and data resources provided by the Leibniz Supercomputing Centre (www.lrz.de).

Abstract

Granular materials, such as powders and grains, are prevalent in both natural and industrial processes, including those in the pharmaceutical and food-processing sectors. To optimize the processing of such materials, simulation technology based on Discrete Element Method (DEM) is widely employed. DEM uses physics-based contact force and heat models to simulate interaction between individual particles. In this study, we examine the force and heat models for spherical particles and present an implementation of DEM using the node-level short-range particle simulation library, AutoPas. Furthermore, we show a straightforward implementation of non-spherical DEM by approximating arbitrary particle shapes using subspheres, utilizing md-flexible, a molecular dynamics simulator integrated within AutoPas.

Furthermore, to demonstrate the effectiveness of our approach, we provide simulation results from several scenarios, particularly a rotating square tumbler and a fluidized bed. In the rotating square tumbler, we observe particle heating resulting due to frictional interactions, as well as heat conduction along the tumbler walls. These thermal effects can either amplify or oppose each other, depending on the thermal settings of the walls. In the fluidized bed, solid particles are initially packed densely. As gas particles exert significant pressure on these solids, they disrupt the alignment of the solid particles, leading to a reduction in pressure and the formation of a fluidized state.

Additionally, we highlight the benefits of AutoPas's auto-tuning capabilities in improving the efficiency of these simulations. For the rotating tumbler, AutoPas selects the Verlet Cluster Lists algorithm for neighbor identification with the AoS (Array of Structures) particle data layout, which proves more efficient than other variants, such as the Linked Cells algorithm. In the fluidized bed scenario, which involves dynamic changes in particle density, AutoPas adapts by transitioning from Linked Cells with AoS to Linked Cells with SoA, and eventually to Verlet Cluster Lists with AoS, as the particle density increases.

Contents

1. Introduction

Advances of computing power and parallelization techniques in recent years opened doors to perform complex simulations of particle dynamics, particularly of granular materials. To model flow of such particles, simulation technology based on the Discrete Element Method (DEM) is widely employed. Here, the particulate system is modeled as an assembly of singular discrete and interacting particles. This approach allows researchers to deepen their understanding of granular motion, while engineers can optimize the design and operation of industrial systems that involve particulate materials. One example is the hopper, an inverted pyramidal container used in various industries, including the pharmaceutical and food-processing sectors [GDK21][BAC+14]. Hoppers store materials and discharge them from the bottom when required. However, the discharge rate, or even the ability of the particles to discharge, can be influenced by factors such as the angles of the walls and the shape of the particles [FLS21][Cle99]. By conducting DEM simulations of the hopper, engineers can identify the ideal wall angles and particle shapes, enabling them to optimize the industrial process (see Figure 1.1).

Given the broad applicability of DEM, several DEM libraries, such as LIGGGHTS [KGK+12], are already in use. However, there is still room for further performance optimization, as currently known DEM packages such as LIGGGHTS use fixed algorithmic configurations. In contrast, the node-level short-range particle simulation library, AutoPas, dynamically adapts its algorithmic configuration to the current simulation state, offering greater potential for performance improvements. This flexibility motivates us to implement a DEM simulator using AutoPas in this thesis and to validate the benefits of employing it.

This thesis begins by exploring the force and heat models and provides an overview of AutoPas, including its neighbor identification algorithms and other pre-implemented configurable options. Following this, we present the implementation of a DEM simulator for spherical particles and extend it to non-spherical particles using the multi-sphere method described in [FAKR99]. We then present DEM simulation results for scenarios such as the rotating square tumbler and fluidized bed, where the auto-tuning capabilities of AutoPas demonstrate significant benefits. Finally, the thesis concludes with suggestions for future research and a summary of findings.

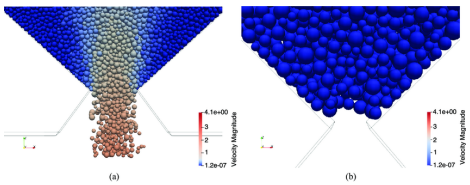


Figure 1.1.: DEM simulations of granular flow in a hopper at discharge. (a) active granular flow (b) no flow due to arching
Source: [FLS21]

2. Theory

2.1. Discrete spherical particle model

The body shape of granular materials that are subjects of DEM simulations could be various. For example, sand has its specific crystal shape, which might differ from other grains such as corn or salt. Moreover, some granular units can deform under stress, which could change their shape. However, in this study, we make following modeling assumptions to simplify the implementation:

- The granular units have a spherical shape.
- The shape of particles is unchangable.

These assumptions give rise to our discrete spherical particle model, where the interaction between a particle pair is determined by the extent of overlap between them (see Figure 2.1).

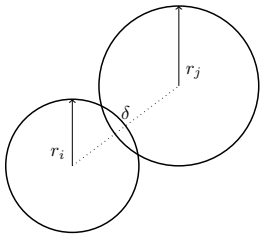


Figure 2.1.: Two spherical particles in contact with overlap δ
Source: [Lud08b]

2.2. Equations of Motion

The dynamics of granular particles can be expressed with Newton's equations, which can be reduced to a system of ordinary differential equations of translational and rotational motion:

$$m_i \frac{\mathrm{d}^2}{\mathrm{d}t^2} x_i = f_{contact} + f_{global}, \quad \text{and} \quad I_i \frac{\mathrm{d}^2}{\mathrm{d}t^2} \varphi_i = q_{contact} + q_{global} \tag{2.1}$$

with the mass m_i of particle i, its position x_i, and the moment of inertia $I_{i,sphere} = \frac{2}{5} m_i r_i^2$. The $f_{contact}$ and $q_{contact}$ can be obtained by the sum of all contact forces and torques, respectively, directed to particle i as we assume existing interaction force and torque only when contact between two particles is present:

$$f_{contact} = \sum_{j \in C_i, i \neq j} f_{ij}, \quad \text{and} \quad q_{contact} = \sum_{j \in C_i, i \neq j} q_{ij} \tag{2.2}$$

with C_i being the set of all particles in contact with i. The global force f_{global} and global torque q_{global} represent forces and torques, respectively, that act on all particles in a system, such as background friction.

For approximating the solutions of these ODEs, numerical integrators are applied, which will be presented next.

2.3. Time Discretization

Due to practical reasons, the aforementioned problem that is posed on a continuous time interval has to be transformed to a problem that is posed on discrete time steps. This brings the necessity of a numeric integrator that is applied to compute the new quantities for translational and rotational motion.

2.3.1. The Integration Method of Störmer-Verlet

In the context of translational motion, the task is to compute the new positions and velocities of the particles from the old positions, old velocities, and the relevant forces. The focus of our simulation lies on observing developments of macroscopic quantities e.g. Energy and Distribution of Particles. This application makes high-order integrators such as Runge-Kutta rather unsuitable due to its computational complexity and relatively poor memory scalability. Instead, a second order symplectic integrator such as Velocity-Störmer-Verlet seems more promising due to its lower computational complexiy and capability to still capture long-time patterns well. The formulas for Störmer-Verlet is presented in the following.

$$x_i(t_{n+1}) = x_i(t_n) + \Delta t \cdot v_i(t_n) + \frac{(\Delta t)^2 F_i(t_n)}{2m_i}, \tag{2.3}$$

$$v_i(t_{n+1}) = v_i(t_n) + \Delta t \cdot \frac{F_i(t_n) + F_i(t_{n+1})}{2m_i}, \tag{2.4}$$

with the position x_i, the velocity v_i, the mass m_i, and the force F_i of particle i and the old time step t_n and the new time step t_{n+1} with $\Delta t = t_{n+1} - t_n$.

2.3.2. The Integration Method of Explicit Euler

In the context of rotational motion, we can simplify the integration due to following reasons.

- Due of the assumption of the spherical particle model, the relevance of orientation, which corresponds to the position in the translational motion, disappears.
- Our main interest lies on observing the development of rotational motion and its macroscopic quantities such as the rotational energy, rather than calculating the angular velocity of a particle accurately in the given time step.

This requirement allows us to use a simple first-order integrator of Explicit Euler with its computational and memory advantages:

$$w_i(t_{n+1}) = w_i(t_n) + \Delta t \cdot \frac{q_i(t_n)}{I_i}, \tag{2.5}$$

with the angular velocity w_i and the moment of Inertia I_i of particle i.

With the same reasoning, the method of Explicit Euler is also applied for thermal development:

$$T_i(t_{n+1}) = T_i(t_n) + \Delta t \cdot \frac{\sum_{j \in C_i} \phi_{ji}}{m_i \cdot C_i^p}, \tag{2.6}$$

with the temperature T, the heat flux ϕ and the specific heat C_i^p. Details follow in Sec. 2.5.1.

2.4. Contact Force Laws

The interaction force models can be divided into two main classes: normal and tangential forces / torques. Here, we introduce the physics-based force and torque models presented by [Lud08b].

2.4.1. Normal Contact Force Law

Linear Normal Contact Model

The model used for expressing the normal force between particles is the Maxwell model [Roy01], which consists of a spring generating a linear repulsive force and a damper realizing a dissipative viscosity.

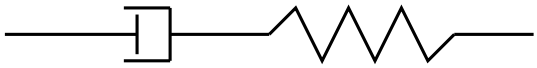


Figure 2.2.: Maxwell's spring dashpot model with a dashpot and a spring placed in series

This model becomes active if and only if the overlap δ between two particles is positive, which can be computed as the sum of the radii of the particles subtracted by their distance:

$$\delta_{ij} = (r_i + r_j) - \|\mathbf{x_i} - \mathbf{x_j}\|_2 = \delta_{ji} \tag{2.7}$$

with r being the radius of the corresponding particle. As this model expresses force in the normal direction, which is parallel to the branch vector $\mathbf{l_{ji}}$ pointing from center position of particle j to i, the unit vector $\mathbf{n_{ji}}$ in the normal direction should additionally be computed to provide a directional meaning to the overlap-dependent normal force:

$$\mathbf{l_{ji}} = \mathbf{x_i} - \mathbf{x_j}, \quad \mathbf{n_{ji}} = \frac{\mathbf{l_{ij}}}{\|\mathbf{l_{ij}}\|_2} \tag{2.8}$$

For every contact with positive overlap δ, the rising normal force f_{ji}^n on particle i from particle j can be computed by taking into account the repulsion of the spring and the dissipation of the damper (see Figure 2.2):

$$f_{ji}^n = f_{elastic}^n + f_{dissipative}^n, \tag{2.9}$$

$$f_{elastic}^n = k^n \delta, \quad f_{dissipative}^n = -\gamma_n v_{ji}^n, \tag{2.10}$$

$$v_{ji}^n = \mathbf{v_{ji}} \cdot \mathbf{n_{ji}} = (\mathbf{v_i} - \mathbf{v_j}) \cdot \mathbf{n_{ji}}. \tag{2.11}$$

with the spring stiffness k^n, the viscous damping coefficient γ_n, and the relative velocity in normal direction v_{ji}^n. The negative sign of the term $-\gamma_n v_{ji}^n$ explains the damping effect, reducing the amount of force according to the damping coefficient γ_n. Multiplying this scalar force, i.e. $f_{ji}^n \in R$, with the according normal vector $\mathbf{n_{ji}}$, produces a three-dimensional force vector f_{ji}^n with a directional meaning:

$$\mathbf{f_{ji}^n} = f_{ji}^n \cdot \mathbf{n_{ji}} \tag{2.12}$$

However, the viscous damping coefficient γ_n must be set carefully due to following reasons:

- a large value of γ_n might lead to unphysical behavour such as abrupt increase of the force.
- a large value of γ_n might increase the duration of particle contact. Here, it should apply: $\Delta t \ll t_{\text{contact}}$.

Refer to A.1.1 and A.1.1 in the appendix for more details and some experiments with different values of γ_n.

2.4.2. Tangential Contact Force and Torque Laws

For the tangential contact forces and torques, three laws should be considered:

1. sliding and friction forces and the resulting frictional torque
2. rolling-resisting torque
3. torsion-resisting torque

Sliding and Static Friction

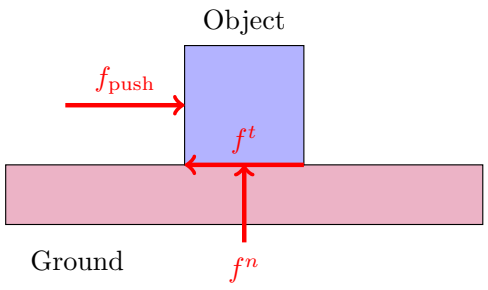


Figure 2.3.: Frictional, Normal, and external forces acting on a square object on a flat ground
Source: [Lud98]

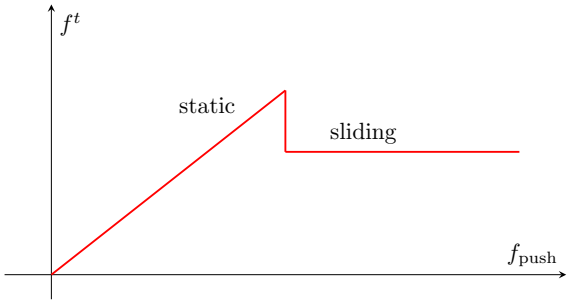


Figure 2.4.: Static and sliding friction in relation to the external force

Sliding and friction forces describe the force resisting the relative motion of surfaces. Figure 2.3 illustrates the forces acting on the contact surface of a simplified example of a square object and a flat ground. Besides the normal force f^n, which acts as a reaction to the gravity, the frictional force f^t opposes the direction of the relative motion, caused by induced by the external force f_{push}. A typical relationship between frictional and external forces is illustrated in Figure 2.4, which demonstrates the distinct behaviors of static and sliding friction. While the static friction is proportional to the external force f_{push}, the sliding friction is independent of f_{push}. Moreover, one usually observes greater static friction than sliding friction, which can be modeled with different coefficients of friction μ, i.e. $\mu_s > \mu_d$, with μ_s and μ_d denote static and sliding (dynamic) friction coefficients respectively. Using these coefficients, the frictional forces can be modeled as follows:

$$\begin{aligned} f^t_{\text{static}} &\leq \mu_s \cdot f^n = f^t_C \\ f^t_{\text{sliding}} &= \mu_d \cdot f^n \end{aligned} \tag{2.13}$$

with f^t_C frequently referred to as the Coulomb limit [Lud08b], as it sets an upper bound to the static friction. One way to model the static friction with a better approximation than in Eq. 2.13 is to use relative velocity, similar to the linear spring model in Sec. 2.4.1 [Lud08b]:

$$\mathbf{f^t} = \begin{cases} -\gamma_t \cdot \mathbf{v_t} & \text{if static} \\ \mu_d \cdot f^n \cdot \mathbf{t} & \text{if sliding} \end{cases} \tag{2.14}$$

In the static case, the model utilizes the tangential dissipation parameter γ_t, and the tangential relative velocity $\mathbf{v_t}$, following a similar approach to the linear spring model in Sec. 2.4.1. However, unlike the linear spring model, we omit the tangential spring that would correspond to the overlap δ. Although this tangential spring would contribute an additional force along $\mathbf{v_t}$, it is a relatively minor detail. Moreover, it introduces memory overhead, as each particle would need to store the state of its tangential spring for all surrounding particles in contact. Refer to [Lud08b] for further details. Furthermore, the negative sign is attached to account for the fact that the frictional force acts in the opposite direction of the relative motion $\mathbf{v_t}$. The description of the tangential relative velocity $\mathbf{v_t}$ follows:

$$\begin{aligned} \mathbf{v_t} &= \mathbf{v_{ij}} - \mathbf{n_{ji}} \cdot (\mathbf{n_{ji}} \cdot \mathbf{v_{ij}}) \\ \mathbf{v_{ij}} &= (\mathbf{v_i} + \omega_\mathbf{i} \times (-\alpha_\mathbf{i} \mathbf{n_{ij}})) - (\mathbf{v_j} + \omega_\mathbf{j} \times (\alpha_\mathbf{j} \mathbf{n_{ji}})) \\ &= \mathbf{v_i} - \mathbf{v_j} + \alpha_i \mathbf{n_{ji}} \times \omega_\mathbf{i} + \alpha_j \mathbf{n_{ji}} \times \omega_\mathbf{j} \end{aligned} \tag{2.15}$$

with $\alpha_{i/j} = r_{i/j} - \frac{\delta}{2}$ being the corrected radius, which measures the distance between the center of the particle and the contact point. For the calculation of the total relative velocity $\mathbf{v_{ij}}$, additional rotational terms are appended to incorporate the velocities occurred from rotational motion at the contact point. Moreover, to obtain the relative velocity along the tangential plane, the normal component is subtracted from the total relative velocity in Eq. 2.15.

In the sliding case of Eq. 2.14, the tangential unit vector is approximated as $\mathbf{t} = -\frac{\mathbf{v_t}}{\|\mathbf{v_t}\|_2}$ providing the direction of the frictional force.

Furthermore, the case distinction between static and sliding friction is derived from Eq. 2.13. if the magnitude of the computed tangential force $\mathbf{f^t}$, assuming a static case, is

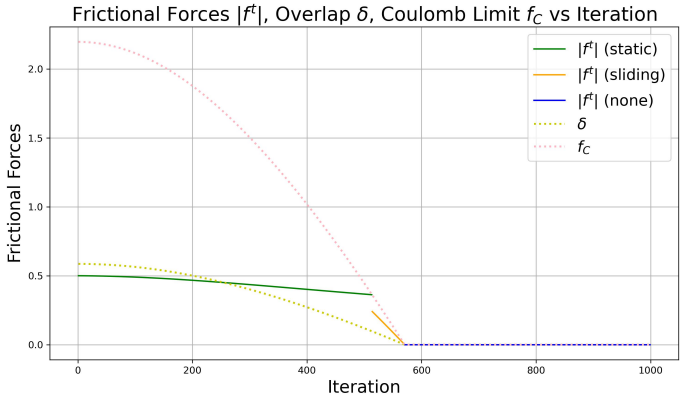


Figure 2.5.: Transition from static to sliding friction with their relations to overlap and Coulomb friction. Used simulation parameters: $k^n = 5, \gamma_n = 5 \cdot 10^{-5}, \gamma_t = 0.1, \mu_s = 0.75, \mu_d = 0.5$

less than or equal to the Coulomb limit f_C^t, then $\mathbf{f^t}$ can be used as the static frictional force. Otherwise, the sliding case applies.

Figure 2.5 illustrates the dynamics of an example contact between two particles as they transition from static to sliding friction, which was simulated using the aforementioned models. Initially, static friction prevails due to the relatively high overlap and Coulomb limit ($f_C \propto \delta$). Over time, the magnitude of static friction decreases as the tangential relative velocity diminishes, which is a result of the decelerating effect of the frictional force. Meanwhile, the Coulomb limit also decreases as the overlap between the particles reduces. When the static friction becomes weaker than the Coulomb limit, the transition to sliding friction occurs. The frictional force of the sliding case has a lower magnitude than the static case due to the typical relation $\mu_d < \mu_s$. For further experiments with tangential forces, refer to A.1.1 in the appendix.

Moreover, this tangential force also causes a torque, as the contact point has a positive distance to the center of mass i.e. $\alpha_{i/j}$:

$$\begin{aligned} \mathbf{q^t}_i &= (-\alpha_i \cdot \mathbf{n}) \times \mathbf{f}_i^t \\ \mathbf{q^t}_j &= (\alpha_j \cdot \mathbf{n}) \times \mathbf{f}_j^t \end{aligned} \tag{2.16}$$

with the sign in the formula for $\mathbf{q_i^t}$ accounting for the fact that the normal unit vector $\mathbf{n} = \mathbf{n_{ji}}$ points from particle j to i. Moreover, while the tangential forces are equal in magnitude via the third law of Newton, i.e. $\mathbf{f_i^t} = -\mathbf{f_j^t}$, such equality does not necessarily hold for torques, as the corrected radii, i.e. α_i and α_j, can have different values. However, unlike the forces, the torques are parallel. Inserting the equality $\mathbf{f_i^t} = -\mathbf{f_j^t}$ into Eq. 2.16 yields:

$$\mathbf{q_j^t} = \frac{\alpha_j}{\alpha_i} \cdot \mathbf{q_i^t} \tag{2.17}$$

For better visualization, Figure 2.7 illustrates the directions of the position vector, which points from the particle center to the contact point, the tangential force, and the resulting torque, which is perpendicular to both the force and the position vector.

Furthermore, the development of torque, tangential force, and angular velocities during an example particle contact is illustrated in Figure 2.6, where only sliding friction for the tangential force is considered to highlight the direction of the induced torque. The following observations can be made:

- Magnitudes of frictional forces and torques are proportional to f^n and therefore to overlap δ
- The plots of angular velocities match with those of frictional torques with its relation $\mathbf{q^t} = I \cdot \frac{d\omega}{dt}$

For further details and checks, refer to A.2 in the appendix.

Rolling resistance

Similar to the calculation of the relative tangential velocity $\mathbf{v_t}$ for determining tangential frictional forces, the relative rolling velocity $\mathbf{v_r}$ must also be computed and taken into account when calculating the torque for rolling resistance. While the relative tangential velocity is computed by taking the difference between the velocities from translational and rotational motion of the two contacting particles, the rolling velocity is obtained by accumulating the velocities from their rotational motion:

$$\mathbf{v_r^0} = (\omega_\mathbf{i} \times (-\alpha_i \mathbf{n})) + (\omega_\mathbf{j} \times (\alpha_j \mathbf{n})) \tag{2.18}$$

Here, the corrected radii, i.e. $\alpha_{i/j}$ can be further replaced by the reduced radius α_{ij}, which is always less than or equal to each corrected radius, i.e. $\alpha_{ij} \leq \alpha_i$ and $\alpha_{ij} \leq \alpha_j$. The formula for the reduced radius follows:

$$\frac{1}{\alpha_{ij}} = \frac{1}{\alpha_i} + \frac{1}{\alpha_j} \Leftrightarrow \alpha_{ij} = \frac{\alpha_i \cdot \alpha_j}{\alpha_i + \alpha_j} \tag{2.19}$$

Inserting this reduced radius into Eq. 2.18 yields:

$$\mathbf{v_r} = \alpha_{ij}(\mathbf{n} \times \omega_\mathbf{i} - \mathbf{n} \times \omega_\mathbf{j}) \tag{2.20}$$

The usage of the reduced radius simplifies the formula for the rolling velocity and is objective in general, which is further discussed in [Lud08a].

This rolling velocity expresses distances per time unit at which the contact surfaces of the particles roll over each other without translational motion ("slipping") with directional meaning [Lud08b] [BK04]. Based on this definition, this velocity $\mathbf{v_r}$ is identical for both particles, as they cover the same distance, excluding any "slipped" distances. The rolling resistance "force" $\mathbf{f_r}$, which only serves the purpose of calculating the torque and will not be applied elsewhere, opposes the rolling velocity and can be calculated using the same procedure as the tangential frictional forces described earlier [Lud08b]:

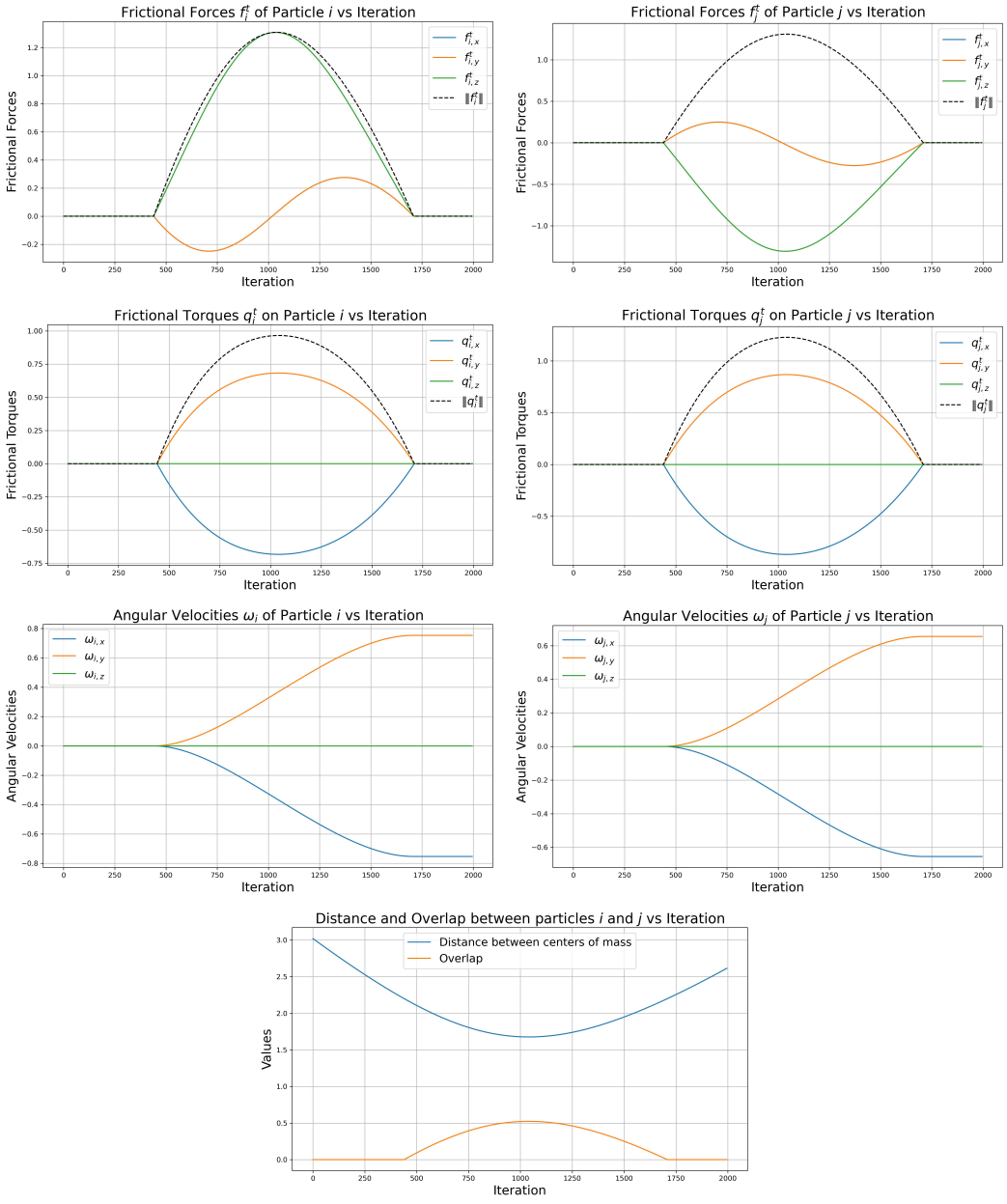


Figure 2.6.: Development of frictional forces $\mathbf{f^t}$, torques $\mathbf{q^t}$, angular velocities ω, overlap δ, and distance d_{ij} during a typical particle contact.

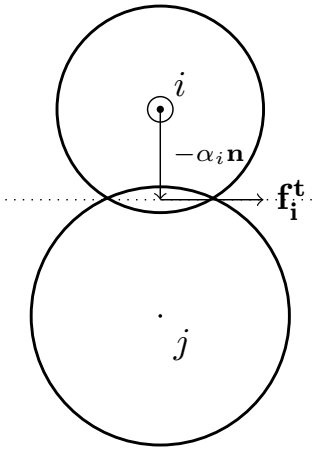


Figure 2.7.: Frictional Torque arised at contact between particles i and j. The torque $\mathbf{q_i^t}$ is perpendicular to both position vector $-\alpha_i \mathbf{n}$ and tangential force $\mathbf{f_i^t}$ and is directed outward from the figure, implying a counterclockwise torque on particle i.

$$\mathbf{f^r} = \begin{cases} -\gamma_r \cdot \mathbf{r} & \text{if static} \\ \mu_r \cdot f^n \cdot \mathbf{r} & \text{if sliding} \end{cases} \tag{2.21}$$

with the rolling viscosity γ_r, the rolling unit vector $\mathbf{r} = -\frac{\mathbf{v_r}}{\|\mathbf{v_r}\|_2}$, and the rolling "friction" coefficient μ_r with the typical relation $\mu_r < \mu_s$.

Due to identical rolling velocities, the forces $\mathbf{f_r}$ remain equal for both particles. With these forces $\mathbf{f_r}$, the rolling torques are computed similarly to the frictional torques from Eq. 2.16:

$$\begin{aligned} \mathbf{q_i^r} &= (-\alpha_{ij} \cdot \mathbf{n}) \times \mathbf{f^r} \\ \mathbf{q_j^r} &= (\alpha_{ij} \cdot \mathbf{n}) \times \mathbf{f^r} \end{aligned} \tag{2.22}$$

resulting in torques that are equal in magnitude but oppose in direction.

A typical scenario, in which such rolling-resisting torques arise, is illustrated in Figure 2.9. Two particles i and j have a positive overlap and their total rolling motion would slow down due to the "friction" the particles experience in the overlapped region. This reduction of rolling velocity is further visually verified in Figure 2.8, which illustrates an example particle contact with only rolling torque (with sliding case only) activated. Refer to A.3 in the appendix for more details.

Torsion resistance

Another torque to consider is torsion resistance, which acts against rotational deformation of particles. A typical scenario in which the torsion torque becomes active is as follows. Two particles are spinning along a common axis, e.g. z-axis, which is aligned with their normal

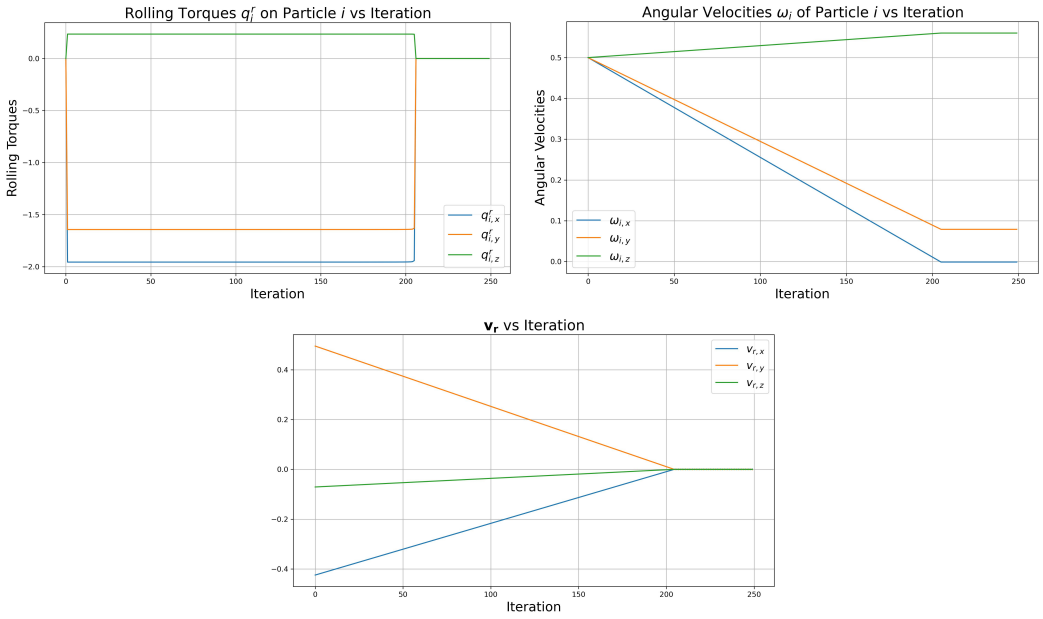


Figure 2.8.: Development of (dynamic) rolling torque $\mathbf{q^r}$, angular velocities ω, and $\mathbf{v_r}$ during an example particle contact. Torques and angular velocities of particle j are same in magnitude with those of particle i with an inverted sign. Distance and overlap between particles i and j remain constant as their positions are fixed.

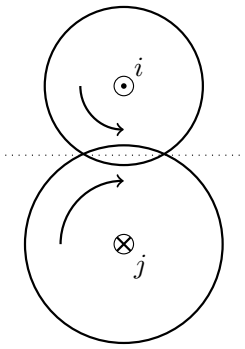
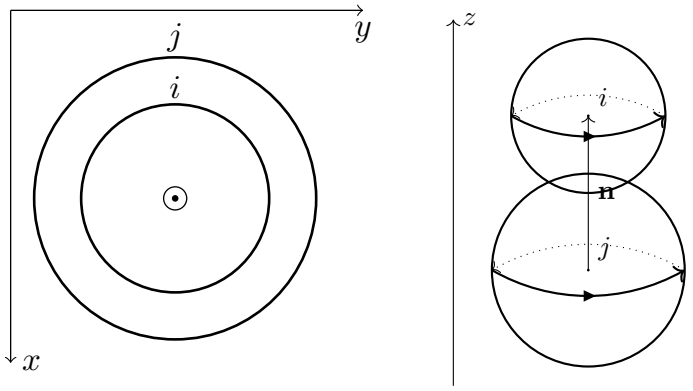


Figure 2.9.: Typical scenario of arising rolling torque. Two particles i and j are in contact and are spinning in anti-parallel directions in the tangential plane. The rolling torque is activated against their rotating direction, i.e. clockwise for particle i and counterclockwise for particle j.



(a) From Above (b) From Side

Figure 2.10.: Typical scenario of arising torsion torque. Two particles i and j are in contact and are spinning with different angular velocities parallel to their normal direction, i.e., in this case, along the z-axis.

vector. However, the particles have different angular velocities along their normal vector, i.e., $\omega_{i,n} \neq \omega_{j,n}$. Due to the existing overlap, $\delta > 0$, this relative spin induces a twisting or torsional deformation between the particles. As a reaction, the particles exert restoring torques that resist such deformation. These restoring torques are referred to as torsion resistance and are modeled by the torsion torque $\mathbf{q^o}$ in this context [Lud08b]. This scenario is further illustrated in Figure 2.10.

Similar to the other torques discussed earlier, torsion torque $\mathbf{q^o}$ is computed with the relative torsion velocity $\mathbf{v_0}$ and the resulting torsion "force" $\mathbf{f^o}$:

$$\begin{aligned}
\mathbf{v_o} &= \alpha_{ij} \cdot (\mathbf{n}\cdot\omega_\mathbf{i} - \mathbf{n}\cdot\omega_\mathbf{j})\cdot\mathbf{n} \\
&= \alpha_{ij} \cdot \mathbf{n}\cdot(\omega_\mathbf{i} - \omega_\mathbf{j})\cdot\mathbf{n} \\
\mathbf{f^o} &= \begin{cases} -\gamma_o \cdot \mathbf{v_o} & \text{if static} \\ \mu_o \cdot f^n\cdot\mathbf{o} & \text{if sliding} \end{cases} \\
\mathbf{q_i^o} &= \alpha_{ij}\cdot\mathbf{f^o} = -\mathbf{q_j^o}
\end{aligned} \tag{2.23}$$

The relative torsion velocity $\mathbf{v_o}$ is a vector parallel to the normal direction of the two particles, quantifying the difference of their angular velocities along the normal direction, i.e. the projection of $\Delta\omega$ onto the normal vector $\mathbf{n}$. The torsion "force", which, like the rolling "force", only helps computing the torsion torque $\mathbf{q^o}$, is constituted by its factors: the torsion viscosity γ_o, the torsion friction coefficient μ_o, the normal force f^n and the torsion unit vector $\mathbf{o} = -\frac{\mathbf{v_o}}{\|\mathbf{v_o}\|_2}$. The resulting torsion torque $\mathbf{q^0}$ is parallel to its torsion unit vector $\mathbf{o}$ as well as the normal vector $\mathbf{n}$. These torques $\mathbf{q^o_{i/j}}$ are identical in magnitude and oppose in directions, changing their affected angular velocities $\omega_{\mathbf{i/j}}$ such that their rotations around the normal direction become identical, i.e. $\mathbf{v_o} \Rightarrow \mathbf{0}$, $\mathbf{n}\cdot\omega_\mathbf{i}^\mathbf{after} = \mathbf{n}\cdot\omega_\mathbf{j}^\mathbf{after} = \mathbf{n}\cdot\frac{\omega_\mathbf{i}^\mathbf{before}+\omega_\mathbf{j}^\mathbf{before}}{2}$ [Lud08b].

The formula of this torque exhibits structural similarities with the one of rolling torque,

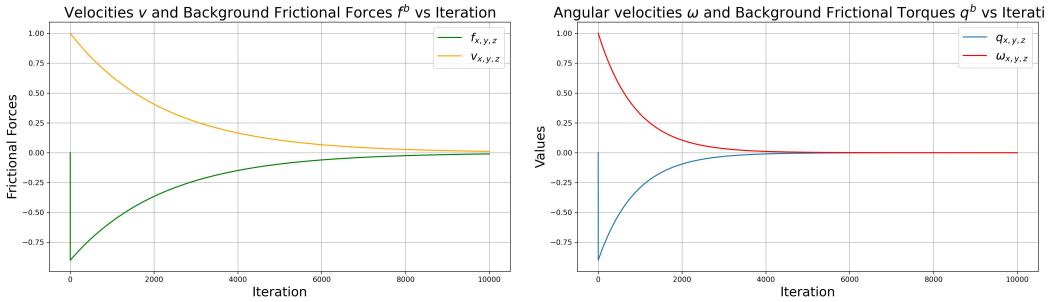


Figure 2.11.: Results of a simple simulation of background friction $\mathbf{f^b}$ and $\mathbf{q^b}$ with other forces and torques deactivated. A particle with following initial velocities and angular velocities is affected by background damping: $(v_x^0, v_y^0, v_z^0) = (\omega_x^0, \omega_y^0, \omega_z^0) = (1, 1, 1)$. Lines for each direction, i.e. x, y, and z, overlap due to symmetry. Simulation parameters: $r = m = 1, \gamma_b = \gamma_{br} = 0.9, \Delta t = 5 \cdot 10^{-4}$.

which results in similar behavior during a typical particle contact. For more details, refer to A.4 in the appendix.

2.4.3. Background Friction

Besides the aforementioned contact laws, a general background friction can be additionally considered, which models inter-particle medium. Comparing the collision of two particles in air and water, the forces and torques exerted on each other are significantly weaker in water due to the damping effect of the medium. Such "frictional" effect can be modeled as follows [Lud08b]:

$$\mathbf{f_i^b} = \gamma_b \mathbf{v_i}, \quad \mathbf{q_i^b} = \gamma_{br} r_i^2 \omega_\mathbf{i} \tag{2.24}$$

The translational and rotational background friction coefficients γ_b and γ_{br} must be chosen for each set of parameters, particularly for each mass m_i, due to the relations between force and velocity, as well as torque and angula velocty in Eq. 2.1. Since damping acts against the current (angular) velocity of the particle, its translational and rotational motion are reduced in magnitude. Incorporating this background frictional damping results in following total forces and torques:

$$\mathbf{f_i} = \sum_{C_i} \left(f^n \mathbf{n} + f^t \mathbf{t} \right) - \mathbf{f_i^b}, \quad \mathbf{q_i} = \sum_{C_i} \left(\mathbf{q^t} + \mathbf{q^r} + \mathbf{q^o} \right) - \mathbf{q_i^b} \tag{2.25}$$

Here, C_i denotes the set of all particles in contact with particle i, and f^n and f^t represent the sums of normal and frictional forces, respectively.

Results of a simple simulation for background friction are shown in Figure 2.11, where both the velocity and angular velocity are effectively damped over time. Since all other forces and torques are deactivated, the force and torque at any iteration correspond to the background frictional force and torque, which are equal to the damping coefficient,

$\gamma_{b/br}$, multiplied by the velocity and angular velocity at that iteration. Subsequently, the background frictional force and torque modify the velocity and angular velocity according to the integration methods outlined in Sec. 2.3. Therefore, the lines exhibit exponential growth/decay, as the change in (angular) velocities is directly proportional to the current (angular) velocity, i.e.,

$$\Delta\mathbf{v} \propto \gamma_b \cdot \mathbf{v} \quad \text{and} \quad \Delta\omega \propto \gamma_{br} \cdot \omega.$$

Moreover, the angular velocity ω decays faster than the velocity $\mathbf{v}$ for two reasons:

1. For the calculation of velocity and angular velocity changes ($\Delta\mathbf{v}$ and $\Delta\omega$), the Störmer-Verlet method in Eq. 2.4 uses the average of the current and previous forces, i.e., $\frac{\mathbf{f(t_n)}+\mathbf{f(t_{n+1})}}{2}$, whereas the explicit Euler method only considers the current torque, i.e., $\mathbf{q(t_n)}$. As a result, the velocity changes computed using Störmer-Verlet are smaller than those computed using explicit Euler when the force and torque are monotonically decreasing. However, this leads to only minor differences in the evolution of the two methods, as the chosen time step, $\Delta t = 5 \cdot 10^{-4}$, is sufficiently small.
2. In the explicit Euler method, the current torque is divided by the moment of inertia, i.e., $\frac{\mathbf{q(t_n)}}{I}$, which becomes smaller than one when the simulation parameters are inserted, i.e., $I_{\text{sphere}} = \frac{2}{5} \cdot m \cdot r^2 = 0.4 < 1$. Dividing by a number smaller than one increases the change in angular velocity ($\Delta\omega$). In contrast, in the Störmer-Verlet method, the averaged forces are divided by the mass of the particle, which is one in this scenario, i.e., $m = 1$, and therefore does not increase the change in velocity ($\Delta\mathbf{v}$) subsequently.

2.5. Heat Models

2.5.1. Heat Transfer Model

The models presented above focus solely on motion. To incorporate thermal effects, a heat transfer model presented by [CMT06] is briefly introduced below.

$$\begin{aligned} \phi_{ji} &= H_c \cdot (T_j - T_i) \\ H_{ij}^c &= 2k_S\left(\frac{3}{4} \cdot \frac{f^{n*} \cdot \sqrt{r_i r_j}}{E_{ij}}\right)^{1/3} \\ f^{n*} &= k^n \cdot \delta \end{aligned} \tag{2.26}$$

Here, ϕ_{ji} denotes the flux of heat transferred from particle i to j, and $T_{i/j}$ is the temperature of the corresponding particle. Furthermore, k_S represents the thermal conductivity of the material, E_{ij} is the effective Young's modulus for the particles i and j, and C_i^p is the specific heat of the material. Moreover, f^{n*} denotes the simplified normal force, where the dashpot term is omitted to ensure that the force remains positive throughout the simulation, i.e. $f^{n*} > 0$. This is necessary because, as shown in Eq. 2.26, H_{ij}^c becomes a complex number for a negative f^n.

In sum, this is a linear model where the heat flux is directly proportional to the normal force, and thus to the overlap between surfaces. This assumption is based on the idea that

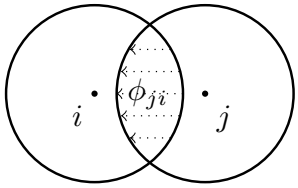


Figure 2.12.: Heat Flux ϕ_{ji} between two particles in contact

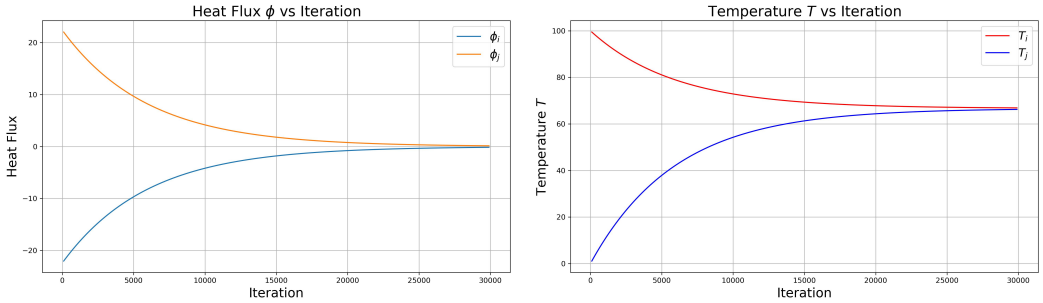


Figure 2.13.: Results of a simple simulation of heat transfer with other models deactivated. Simulation parameters: $\delta = 0.25, k^n = 50, \gamma_n = 10^{-3}, k_S = 0.05, E = C^p = 1, ri = r_j = 1, m_i = 2 \cdot m_j = 2, \Delta t = 5 \cdot 10^{-4}$. The Linear normal force model was used solely to compute f^n without applying the force.

a larger contact area or volume enables greater heat transfer (see Figure 2.12). Further assumptions are following [CMT06]:

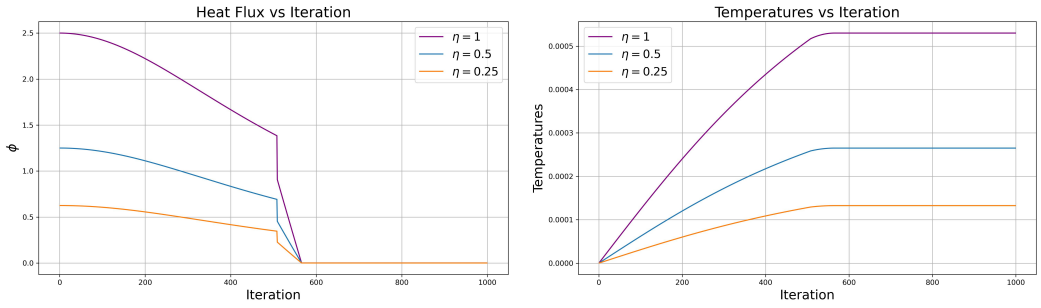
1. k_S, E_{ij}, and C_i^p (see Eq. 2.6) are constant.
2. Temperature T is uniformly distributed over the particle's volume (cf. Sec. 2.1).

Again, a simulation of an example scenario verifies this heat transfer effect (see Figure 2.13). The magnitude of heat flux in both directions is identical, as $\phi_{ji} = -\phi_{ij}$. The temperature evolution of particle i is slower due to its doubled mass. The converged temperature is as follows:

$$T_{i/j}^{after} = T_{i/j}^{before} + \frac{1}{m_{i/j} C_{i/j}^p} \cdot \int_C \phi_{i/j} dt$$
$$\Rightarrow T_i^{after} = T_i^{before} - \frac{1}{3} \Delta T_{ij}^{before} \approx 66.6 = T_j^{after}$$

as the particles' positions are fixed in our setup.

Moreover, to note is that this model solely accounts for heat transfer between particles through conduction and does not account for other forms of heat transfer, such as convection or radiation. That is, according to our model assumptions, heat transfer occurs only between overlapping particles. Additionally, since this model only balances initial thermal differences through heat transfer and does not generate heat, no thermal dynamics will be observed if



(a) (b)

Figure 2.14.: Results of a simple simulation of heat generation with other models deactivated. The same scenario as in Figure 2.5.

no initial temperature differences are provided for the particles. To further account for the conversion of kinetic energy into heat, an additional model for heat generation is introduced in the following.

2.5.2. Heat Generation Model

We assume the generation of heat by frictional forces. To simplify the model, we only consider tangential frictional forces and do not incorporate rolling and torsion friction:

$$\phi_{ji} = \eta \cdot \|\mathbf{f^t}\|_2 \cdot \|\mathbf{v^t}\|_2 = \phi_{ij} \tag{2.27}$$

This model resembles the one presented in [HRK+24] with the difference that [HRK+24] only considers dynamic friction in $\|\mathbf{f^t}\|$. The parameter $\eta < 1$ indicates the proportion of the work done by the frictional forces that is converted into heat. According to the definition of this model, the resulting heat flux is always non-negative and is equally distributed between the two particles, i.e., $\phi_{ji} = \phi_{ij} \geq 0$.

Figure 2.14 shows the development of heat flux and temperature in the scenario depicted in Figure 2.5. The lines in Figure 2.14 (a) resemble those in Figure 2.5, as heat flux is directly influenced by the frictional force. Specifically, the higher the value of the parameter η, the greater the heat flux. Figure 2.14 (b) illustrates the relationship between heat flux and temperature, as shown in Figure 2.13, where a higher value of η leads to a higher final temperature.

2.6. Non-spherical particle model

While the spherical particle model provides a reasonable approximation for granular particles, there is still room for refinement when modeling non-spherical particles. In reality, particles can take on a variety of shapes depending on their crystalline forms. To better capture this diversity, the multi-sphere model can be employed. This model approximates the particle's shape effectively by using multiple smaller sub-spheres (see Figure 2.15). However, this approach introduces several complexities:

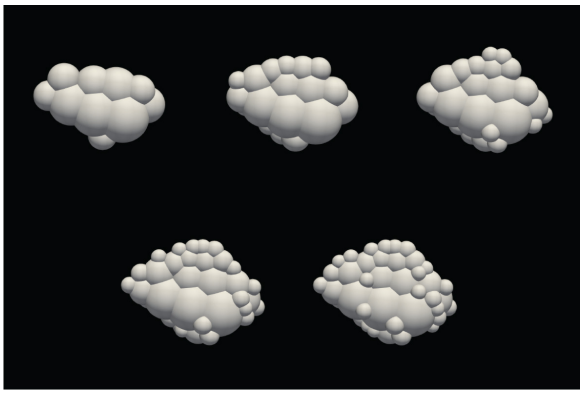


Figure 2.15.: Approximation of a non-spherical particle using the multi-sphere model with an increasing number of sub-spheres
Source: [WCNT23]

- Unlike the spherical model, this multi-sphere approach requires accounting for spatial orientation, making the integration of rotational motion more complex. A rotational velocity Verlet algorithm, as presented in [RK10], can be used for this purpose.
- To maintain the structure of the glued sub-spheres, the relative positions between them must remain constant. This can be achieved by modeling these non-spherical particles as rigid bodies with fixed relative positions between sub-spheres.

Refer to [NGNB23] for further details.

3. AutoPas

The aforementioned contact models are implemented with the usage of a particle simulation library, AutoPas. In this chapter, AutoPas and its unique abilities to dynamically tune a simulation configuration will be presented.

AutoPas is a particle simulation library that arised from the following nature of problem. Many particle simulations are based on short-range pairwise interactions such as the contact force laws in Sec. 2.4 or the Lennard-Jones (LJ) Potential [GSBN21], which is characterized by strong short-range repulsive forces and weaker mid-range attractive forces. A key computational bottleneck in evaluating these short-range forces is the process of identifying neighboring particles, between which the forces must be applied. Fortunately, various algorithms exist to accelerate this neighbor identification process. However, each algorithm comes with its own trade-offs between computational power and memory usage. Interestingly, their performance can vary significantly depending on a range of factors as the simulation scenario, the current system state, and the nature of the interaction forces. In addition to neighbor identification algorithms, other configuration components, such as the choice of data structures and traversal methods, can also affect performance, with their impact varying depending on these factors. AutoPas tackles this challenge by dynamically selecting the optimal algorithmic configuration through auto-tuning, based on the current state of the simulation. The developers of AutoPas describe their library as "an open-source C++ library delivering optimal node-level performance by providing the ideal algorithmic configuration for an arbitrary scenario in a given short-range particle simulation" [GSBN21]. '

3.1. Neighbor Identification Algorithms

AutoPas internally implements four distinct algorithms for identifying neighboring particles within the so-called cutoff radius. The cutoff radius, d_{cutoff}, defines the maximum distance beyond which the short-range interaction force is considered negligible and is thus ignored, i.e.:

$$f_{ji} \approx 0 \quad \forall d_{ij} > d_{cutoff}.$$

In the case of DEM, which is based on particle contacts, the cutoff distance is given by $d_{cutoff} = 2 \cdot r_{max}$ with the maximum particle radius r_{max}, particularly in the case of polydisperse particles. By introducing the cutoff, small interaction forces—such as the weak attractive forces from the LJ potential at longer distances—are neglected, leading to significant computational time savings. The following section provides a brief introduction to these four algorithms.

- **Direct Sum**:
  The Direct Sum algorithm naively computes the distances to all other particles in

the domain and checks whether each calculated distance is within the cutoff radius. In Figure 3.1 (a), all particles, except the red one, are colored grey, indicating that distances have been calculated to all other particles in the domain. Therefore, this algorithm needs to compute distances between all particle pairs in the domain and scales very badly, i.e. $O(N^2)$.

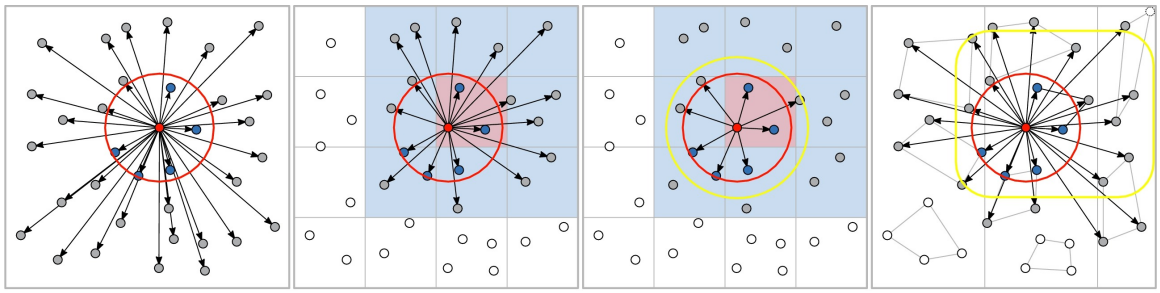
- **Linked Cells**:
  LinkedCells subdivides the domain into a grid of cells. Each cell has a width greater than or equal to the cutoff radius, which is, in turn, at least as large as the diameter of any particle, i.e.

$$d_{\text{cell}} \geq d_{\text{cutoff}} \geq 2 \cdot r_{max},$$

  and tracks the particles within it by maintaining a list of particles inside each cell. This grid structure significantly reduces the number of distance calculations, as computations are only necessary inside the region surrounding the corresponding cell, also referred to as base cell (see Figure 3.1(b)). For particles in cells outside this region, it is safe to assume they are beyond the cutoff radius, thanks to the relationship between cell width and the cutoff radius. Compared to DirectSum (Figure 3.1a), Linked Cells reduces the number of distance calculations considerably (Figure 3.1b), mitigating the time complexity to $O(N)$ [GSBN21]. The main disadvantage is the computational and memory overhead associated with updating the list of particles within each cell in every iteration.

- **Verlet Lists**:
  Figure 3.1(b) shows that the Linked Cells algorithm still computes distances to many grey particles that are far beyond the cutoff radius, which is due to the overly large search (blue) region. In contrast, the Verlet Lists algorithm maintains a list of particles within the so-called interaction length [GSBN21], indicated by the yellow circle in Figure 3.1(c), which is typically larger than the cutoff radius. By selecting an appropriate interaction length, the Verlet Lists can be maintained for several iterations. For (re)generation of Verlet Lists, which should occur periodically when their lifetime "expires", a background grid structure like Linked Cells can be used to identify particles within the interaction length. Since the enlarged interaction length circle better approximates the cutoff radius than the 3×3 grid of Linked Cells, the Verlet Lists algorithm often enjoys a small constant-factor advantage in hitrate and time complexity over the LinkedCells algorithm. However, it still has memory overhead, as the Verlet Lists must be stored across multiple iterations [GST+19a][GST+19b].

- **Verlet Cluster Lists**:
  The Verlet Cluster Lists algorithm takes advantage of the fact that nearby particles often share similar Verlet Lists by clustering a predefined number of neighboring particles together. Each cluster maintains a Verlet list for the entire cluster rather than for individual particles, reducing both the number of Verlet lists and their lengths. The rest of the algorithm operates similarly to the Verlet Lists method. While Verlet Cluster Lists offer a memory advantage by clustering neighboring particles, they may lower the hitrate and increase the number of distance calculations, as the interaction



(a) Direct Sum (b) Linked Cells (c) Verlet Lists (d) Verlet Cluster Lists

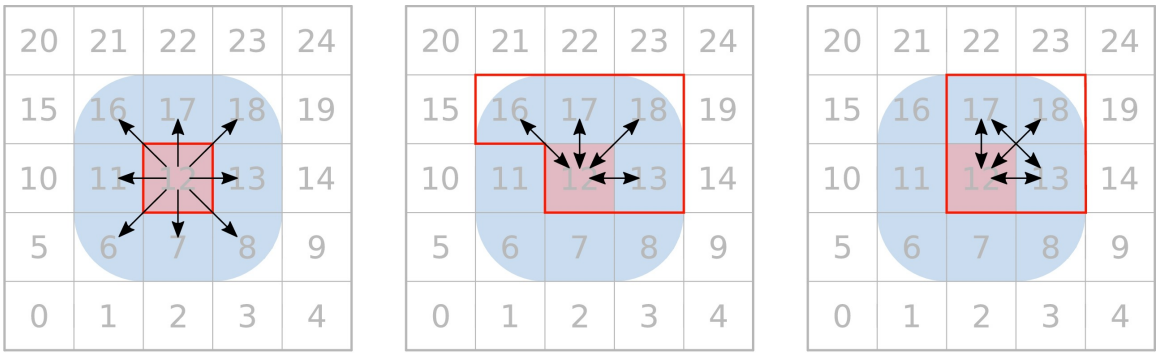
Figure 3.1.: Neighbor identification algorithms used in AutoPas:
The red circle represents the cutoff radius of the red particle. The color distinctions (white, grey, blue) indicate the need for a distance calculation and the actual distance to each corresponding particle. White particles are considered outside the cutoff radius according to the algorithm, so distance calculations to them are skipped. Grey particles require a distance calculation, and among them, the blue particles are within the cutoff radius, meaning that interaction forces will be evaluated for these particles. In the case of Verlet Lists (Figure 3.1(c)), many grey particles are not marked with an arrow because distance calculations are not required in every iteration; they are only necessary when the Verlet lists need to be (re)generated. For cell-based algorithms, the base cell, in which the red particle resides, is colored red. Source: [GSBN21]

length circle must now be centered around the entire cluster of particles (see the larger yellow circle in Figure 3.1(d)) [GSBN21].

3.2. Traversal Methods

Another component of the algorithmic configuration is the traversal method, which determines the order in which particles in the domain are processed. Specifically, for cell-based neighbor identification algorithms like Linked Cells, the 3 × 3 grid surrounding the base cell can be adjusted or reduced by applying Newton's third law, thereby improving computational efficiency. This reduced grid space, where distance calculations are necessary, is referred to as base steps in [GSBN21], representing the area covered by each forward step of the base cell as it progressively spans the entire domain. Three base steps implemented in AutoPas are presented briefly in the following.

- **c01 base step**:
  This standard base step considers the full 3 × 3 grid surrounding the base cell, as described earlier, and does not utilize Newton's third law further (see Figure 3.2(a)).
- **c18 base step**:
  This base step applies Newton's third law and only considers the "upper" half of its 3 × 3‘ surrounding grid space, including the base cell itself, for distance calculations. Due to Newton's third law, interactions with particles in the omitted "lower" half of the grid region are handled by the cells in the lower half (see Figure 3.2(b)).
- **c08 base step**: This base step applies Newton's third law and further reduces the full 3 × 3 grid to a 2 × 2 square, with the base cell positioned in the lower left corner



(a) c01 base step (b) c18 base step (c) c08 base step

Figure 3.2.: Base steps in AutoPas: The background displays the cell grid, where the cell width is equal to the cutoff radius. The base cell, colored red, is surrounded by the full 3×3 grid region, shown in blue. The reduced grid area covered by the base step is outlined with red lines. Black arrows indicate the necessary distance calculations between particles in the cells at the arrow's ends. Source: [GSBN21]

(Figure 3.2(c)). In addition, it accounts for the interactions between the particles in the upper left and lower right corner cells, ensuring that all interactions between the base cell and the surrounding cells of the original 3×3 grid are fully covered (see Figure 3.2(c)).

These base steps can be further optimized by combining them with coloring and slicing techniques, enabling more time-efficient processing through parallel computation. A comprehensive description of all available traversal methods can be found in [GSBN21].

3.3. Data Layouts

A further aspect of the configuration is the layout of particle data, which determines how particle attributes are stored in memory. AutoPas supports two primary approaches for this:

- **Array of Structures (AoS)**:
  In this layout, each particle is represented by a C++ structure (or object) that encapsulates its attributes, such as x/y/z-positions. These structures are then stored in a contiguous array, typically implemented as a `std::vector<Particle>`, where each element corresponds to a single particle.
- **Structures of Arrays (SoA)**:
  In this layout, the attributes of particles are stored in separate, contiguous arrays, which are grouped together within a structure. For instance, the x, y, and z coordinates of all particles could be stored in distinct arrays, such as `std::vector<double> posX`, `std::vector<double> posY`, and `std::vector<double> posZ`, all encapsulated within a SoA object.

Using the AoS layout for particles may be more efficient for random access to individual particles, as each particle's data is stored together. On the other hand, the SoA layout offers

improved memory access efficiency by storing attributes of all particles in contiguous arrays, enabling operations to be performed on specific attributes across all particles. This layout is particularly advantageous in performance-critical scenarios, as it enables easy parallelization or vectorization of operations [GSBN21].

3.4. Further Configurable Options

Furthermore, AutoPas offers two further configuration options, i.e. Cell Size and Newton3.

- **Cell Size**:
  The cell size defines the width of the cell grid used for Linked Cells or Verlet Lists. Depending on this, the number of necessary distance calculations and the overall time efficiency of the simulation might vary [GSBN21].
- **Newton3**:
  By applying Newton's third law, the number of force calculations is reduced by half, as the forces between pairs of particles are computed only once. However, this optimization can introduce challenges for parallelization, as force updates may involve interactions between particles in two different cells and therefore bring about synchronization overhead [GSBN21]. This trade-off makes the use of Newton3 a configurable option.

3.5. User Perspective

From the user's perspective, AutoPas functions as a black-box container that provides various methods for constructing a simulation loop. The user can add particles to the simulation using `addParticle(Particle)`, iterate over all particles with `begin()`, trigger the evaluation of interaction forces among all particle pairs via `iteratePairwise(Functor)`, or update the particle container (e.g., regenerating Verlet lists or re-tracking particles in each cell). These operations are invoked through an interface, while AutoPas handles the underlying logic, including auto-tuning, automatically. In addition to the aforementioned methods, which are visualized in Figure 3.3 AutoPas also offers options for configuring simulation-wide parameters, such as adjusting the domain size [GSBN21].

3.5.1. Custom Particle and Functor

To perform a particle simulation with custom interaction forces in AutoPas, the user must implement custom particle and functor classes, both inheriting from the base classes `Particle` and `Functor` [GSBN21]. Through inheritance, objects of these custom classes can call the interface methods outlined earlier (see Figure 3.3).

The custom particle class should encapsulate all the necessary attributes required to calculate the interaction force between particles. These attributes typically include particle position, velocity, mass, and other relevant properties. The interaction force itself should be implemented in the custom functor class, which should overwrite multiple methods defined in the base class. Each method should be designed to support different neighbor identification algorithms, tailored to various data layout types. The implementation of these methods is

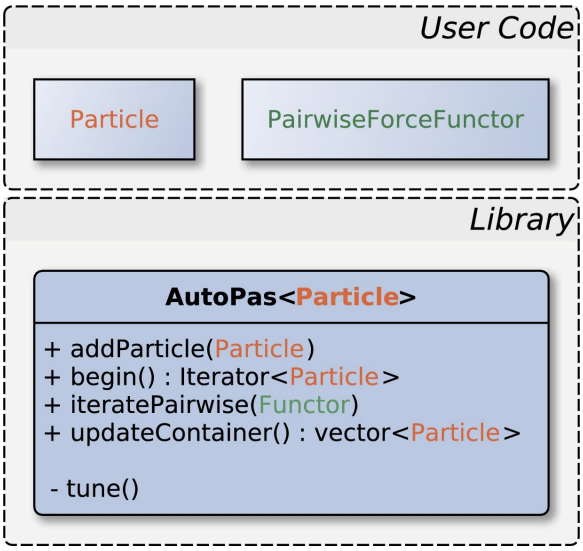


Figure 3.3.: The AutoPas interface, along with its key methods, which can be called with arguments from user code. Source: [GSBN21]

(a) Base Class `Particle` (b) Base Class `Functor`

Figure 3.4.: Simplified diagrams of class `Particle` and `Functor`. The symbol `<<get/set>>` indicates the presence of getter and setter methods for the corresponding attribute within the class.

necessary to fully leverage AutoPas's auto-tuning capabilities to use different algorithms and data layout types as configurable options.

The base classes `Particle` and `Functor` are illustrated in their simplified forms in Figure 3.4.

3.5.2. Simulation Loop

By utilizing the methods provided by the AutoPas interface, a simulation loop can be constructed. This loop evaluates interaction forces, computes the affected (angular) velocities based on the integration methods (see Sec. 2.3), and updates the particle positions. Additionally, various boundary conditions, such as reflective or periodic boundaries, can be applied to control the behavior of particles at the domain boundaries. An example of a simplified simulation code is shown in Listing 3.1.

```
1  // Create and initialize an AutoPas object:
2  AutoPas<CustomParticle> autopasContainer;
```

```
3   autopasContainer.init();
4
5   CustomParticle p;
6   autopasContainer.addParticle(p);
7
8   CustomForceFunctor f;
9
10  // Simulation Loop
11  for (needsMoreIterations()) {
12    calculatePositions();
13    autopasContainer.updateContainer();
14    applyBoundaryConditions(autopasContainer);
15    autopasContainer.iteratePairwise(&f);
16    calculateVelocities();
17    calculateAngularVelocities();
18  }
```

Listing 3.1: Simplified example of a simulation

3.6. `md-flexible`

To enhace the user friendlibility, AutoPas additionally provides code of an example simulation called `md-flexible`. md-flexible supports both single-site and multi-site molecular dynamics (MD) simulation using the Lennard-Jones 12-6 potential with Störmer-Verlet time integration [GSBN21][NGNB23].

Notably, the `md-flexible` framework for multi-site MD simulations includes already implemented algorithms for updating spatial orientation, along with angular velocities and torques, while ensuring the structural integrity of molecules by maintaining fixed positions of their sites. Advantageously, this framework is also applicable to the multi-sphere approach of DEM described in Sec. 2.6 where non-spherical particles are treated as molecules, with their sub-spheres acting as molecular sites. Here, users only need to specify the relative positions of these sites with respect to the center of mass and provide the inertia tensor of the molecule.

Moreover, `md-flexible` also offeres code for reflective and periodic boundary conditions, printing VTK files for visualization etc. The structure of the functor methods implemented for simulating the Discrete Element Method in Sec. 4.2 also closely follows the preimplemented methods in `md-flexible`.

4. Implementation

In this chapter, the implementation of the models from Sec. 2 will be described from the perspective of an user of AutoPas.

To simulate the Discrete Element Method (DEM), custom classes `GranularDEM` for particles and `DEMFunctor` for contact forces have been implemented, that are presented briefly in the following.

4.1. Custom Classes `GranularDEM` and `DEMFunctor`

On the one hand, the `GranularDEM` class extends the base `Particle` class by incorporating additional attributes that are needed to implement the contact force and heat models from Sec. 2.4 and 2.5:

- The attributes `angularVel` (angular velocity) and `torque` are added to model rotational dynamics. Due to the spherical particle model (see Sec.2.1), spatial orientation is neglected.
- The `oldForce` attribute is stored, as the Störmer-Verlet integration method is used for translational motion (see Sec.2.3).
- The `typeId` attribute is utilized to differentiate particles with varying radii, provided that the number of distinct radii remains manageable.
- The attributes `temperature` and `heatFlux` are included for heat modeling.

On the other hand, the `DEMFunctor` class implements interaction forces based on the contact force and heat models. It encapsulates the coefficients used in these models and provides various functor methods. The structures of these base classes are further illustrated in Figure 4.1, while the details of these functor methods are presented in the following.

4.2. Functor Methods in `DEMFunctor`

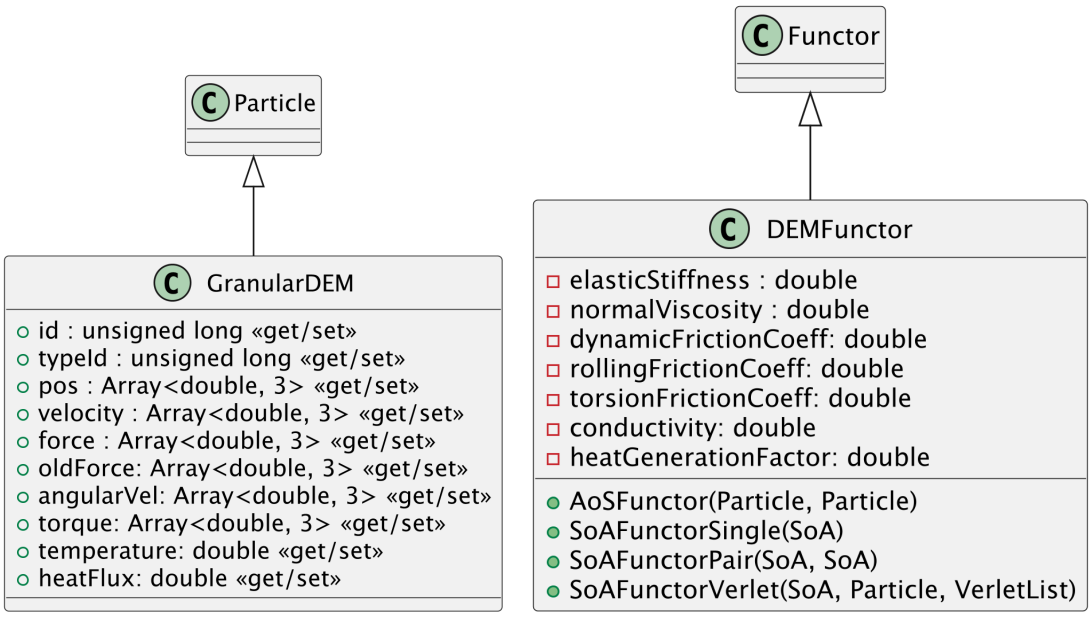
As illustrated in Figure 4.1, the `DEMFunctor` class encompasses multiple methods designed to utilize configurable options such as data layouts and neighbor identification algorithms. The following provides a concise overview of the structure of these methods.

4.2.1. AoSFunctor

The AoSFunctor takes `C++` Particle objects as arguments, along with a boolean flag indicating whether Newton's third law should be applied. Due to the AoS layout, particle properties, such as position, can be directly accessed and modified from the particle objects. An example structure of this method is provided in Listing 4.1.

```
1  void AoSFunctor(Particle &i, Particle &j, bool newton3) {
2    // Ignore particles outside the cutoff radius
3    const double distance = calculateDistance(i.getPos(), j.getPos());
4    if (distance > cutoff) return;
5
6    // Compute overlap, normal unit vector, translational relative velocity, ...
7
8    // Compute linear normal force
9    const double normalContactForceScalar = elasticStiffness * overlap -
         normalViscosity * dot(normalUnitVec, relVel);
10   const std::array<double, 3> normalForceVector = mulScalar(normalUnitVec,
         normalContactFScalar);
11
12   // Compute other forces
13
14   i.addF(totalF);
15   if (newton3) {
16     j.subF(totalF);
17   }
18
19   // Compute and apply torques
20   // ...
21
22 }
```

Listing 4.1: Simplified version of AoSFunctor()



(a) Cutustom Particle Class `GranularDEM` (b) Custom Functor Class `DEMFunctor`

Figure 4.1.: Simplified diagrams of custom class `GranularDEM` and `DEMFunctor`.

4.2.2. SoAFunctor

The SoAFunctor accepts a `SoAView` as argument, which mirrors the concept of `std::string_view` from C++17, offering a view into a segment of the actual SoA by referencing its start and endpoint [GSBN21]. In contrast to AoS, SoA requires an access via pointer-indexing to retrieve a particle's attributes. Initializing such pointers can be simplified by defining an inner enum class `AttributeNames` inside the custom particle class, which contains the names of all particle attributes. This enum can then be used by `SoAView` to provide pointers to the corresponding attribute arrays. Moreover, in contrast to AoS, the SoAFunctor evaluates the interaction forces between multliple particles "simultaneously". This enables vectorization with e.g. OpenMP, using SIMD instructions with a reduction to the force and torque accumulators. An example is provided in Listing 4.2.

```
1  void SoAFunctor(SoAView soa, bool newton3) {
2    // Initialize pointers to SoA data
3    const auto *const  xptr = soa.template begin<Particle::AttributeNames::
         posX>();
4    const auto *const  yptr = soa.template begin<Particle::AttributeNames::
         posY>();
5    //...
6    double *const fxptr = soa.template begin<Particle::AttributeNames::forceX
         >();
7    double *const fyptr = soa.template begin<Particle::AttributeNames::forceY
         >();
8     // ...
9
10   // Nested for-loops
11   for (unsigned int i = 0; i < soa.size(); i++) {
12     // Initialize accumulators for force and torque
13     double fxacc = 0.;
14     double fyacc = 0.;
15     // ...
16
17     #pragma omp simd reduction(+ : fxacc, fyacc, ...) // vectorization
18     for (unsigned int j = i + 1; j < soa.size(); ++j) {
19       // Force and torque calculations
20       // ...
21
22       // Add computed forces and torques to accumulators
23       fxacc += totalFX;
24       fyacc += totalFY;
25       // ...
26
27       if (newton3) { // Apply forces and torques to Particle j (newton3)
28         fxptr[j] -= totalFX;
29         fyptr[j] -= totalFY;
30         // ...
31       }
32     }
33     // Apply forces and torques
34     fxptr[i] += fxacc;
35     fyptr[i] += fyacc;
36     // ...
37   }
38 }
```

Listing 4.2: Simplified version of SoAFunctor() using `omp simd`, though other vectorization methods are also possible.

4.2.3. SoAFunctorVerlet

The method `SoAFunctorVerlet` takes as arguments a Verlet-List (`&neighborList`), the index of the base particle (`indexParticleI`), and the corresponding `SoAView`. It defines the hyper-parameter `vecsize`, which specifies the number of particles processed "simultaneously" in a vectorized manner. The method handles $\mathtt{n} \cdot \mathtt{vecsize}$ particles from the Verlet-List in parallel, where

$$\mathtt{n} = \lfloor \frac{\mathtt{neighborList.size()}}{\mathtt{vecsize}} \rfloor.$$

The remaining $\mathtt{neighborList.size()} - \mathtt{n} \cdot \mathtt{vecsize}$ are handled in a non-vectorized manner. An example is shown in Listing 4.3.

```
1  void SoAFunctorVerlet(SoAView soa, const size_t indexParticleI, const
       NeighborList &neighborList, bool newton3) {
2    // Initialize pointers to SoA data
3    const auto *const xptr = soa.template begin<Particle::AttributeNames::posX
       >();
4    // ...
5
6    // Initialize accumulators
7    double fxacc = 0.;
8    // ...
9
10   constexpr size_t vecsize = 12; // hyper-parameter
11
12   if (neighborList.size() >= vecsize) {
13     // Initialize registers to load computation-relevant values and store
         the results
14     alignas(64) std::array<double, vecsize> xtmp, ..., xArr, ..., fxArr, ...
15
16     // Load values of Particle i to registers
17     for (size_t tmpj = 0; tmpj < vecsize; tmpj++) {
18       xtmp[tmpj] = xptr[indexParticleI];
19       // ...
20     }
21
22     // Loop over the Verlet-List from index 0 to the highest multiple of
         vecsize
23     for (size_t joff = 0; joff < neighborList.size() - vecsize + 1; joff +=
         vecsize) {
24
25       // Load values of Particle j
26       #pragma omp simd safelen(vecsize)
27       for (size_t tmpj = 0; tmpj < vecsize; tmpj++) {
28         xArr[tmpj] = xptr[mapToSoAIndex(joff + tmpj)];
29         // ...
30       }
31
32       #pragma omp simd reduction(+ : fxacc, ...) safelen(vecsize)
```

```
33          for (size_t j = 0; j < vecsize; j++) {
34            // Actual force and torque calculations
35            // ...
36
37            // Add computed forces and torques to accumulators
38            fxacc += totalFX;
39            // ...
40            if (newton3) {
41              // Store the values temporarily to registers
42              fxArr[j] = totalFX;
43              // ...
44            }
45          }
46
47          if (newton3) { // Apply the values that were stored in registers
48            #pragma omp simd safelen(vecsize)
49            for (size_t tmpj = 0; tmpj < vecsize; tmpj++) {
50              const size_t j = mapToSoAIndex(joff + tmpj);
51              fxptr[j] -= fxArr[tmpj];
52              // ...
53            }
54          }
55        }
56      }
57
58      // Handle the remaining particles (fewer than vecsize) without
            vectorization (similar to approach in SoAFunctor())
59      // ...
60    }
```

Listing 4.3: Simplified version of SoAFunctorVerlet() using `omp simd`

4.3. Simulation Code

Using the presented custom particle and functor classes, the simulation code can be constructed similar to the structure in Listing 3.1. However, a few further elements have been added (see Figure 4.2), which are presented briefly in the following.

4.3.1. Background Friction

Inside the functor methods described above, all contact models can be implemented, except for the background friction model. The background friction model subtracts the background friction force from the sum of all applied forces by neighboring particles, making its implementation more complex within these functor methods. However, at the simulation loop level, after evaluating all interaction forces, this can be easily done in an "embarrassingly" parallel manner.

```
1   void Simulation::calculateBackgroundFriction(const double gamma_b, const
        double gamma_br) {
2     #pragma omp parallel shared(autopasContainer)
3     for (auto p = autopasContainer->begin(); particle.isValid(); ++particle) {
4     const auto dampedForce = p->getV() * gamma_b;
5     p->subF(dampedForce);
```

```
6
7       const double radius = retrieveRadius(p->getTypeId());
8       const auto dampedTorque = p->getAngularVel() * (radius * radius * gamma_br
            );
9       p->subTorque(dampedTorque);
10      }
11  }
```

Listing 4.4: A simplified version of the `Simulation::calculateBackgroundFriction()` method, designed for use within the simulation loop of the `Simulation` class.

4.3.2. `StatisticsCalculator`

The `StatisticsCalculator` class computes various statistical values, including stress and density, and outputs these results into a CSV file.

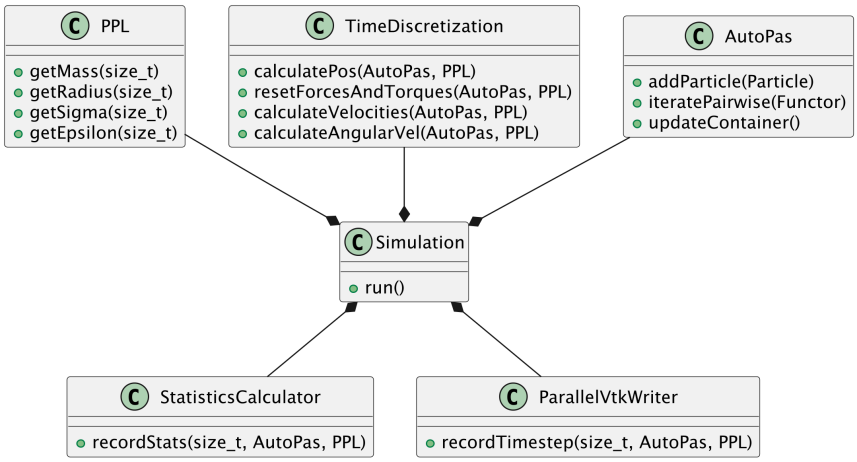


Figure 4.2.: Simplified diagram of classes used to perform a simulation. `PPL` abbreviates the original class name `ParticlePropertiesLibrary`, which stores different particle attributes in a map with keys as typeIds (`size_t`).

4.4. Implementation of Non-spherical Model

For the non-spherical model from Sec. 2.6, one can utilize the multi-site MD framework in `md-flexible` without changes to contact force or heat models.

As mentioned in Sec. 3.6, the user only needs to provide the relative positions of the sub-spheres with respect to the center of mass and the inertia tensor. Specifying these relative positions is straightforward. However, obtaining an analytical solution for the inertia tensor of an arbitrary particle shape is often challenging. Therefore, it is typically approximated using the Monte Carlo method.

In this approach, a large number of infinitesimally small mass points are sampled uniformly from the volume of the desired non-spherical particle shape. The inertia tensor for each point mass is computed using the analytical solution for a single point mass, and the results are summed to approximate the inertia tensor of the entire non-spherical particle. The

Algorithm 1: Approximation of Inertia Tensor using Monte Carlo method

Input: num_samples, mass, domain
Output: inertia_tensor

```
1  Function compute_pointMass_inertia(position, mass):
2      Ixx ← mass · (position.y^2 + position.z^2);
3      Iyy ← mass · (position.x^2 + position.z^2);
4      Izz ← mass · (position.x^2 + position.y^2);
5      ...
6      return Tensor(Ixx, Ixy, Ixz, ...)

   // Actual Approximation
7  Function approximate_inertia(num_samples, mass, domain):
       // Initialize parameters
8      dm ← total_mass / num_samples; drawn_num_samples ← 0;
        current_inertia_tensor ← 0
9      while drawn_num_samples < num_samples do
10         sample ← sample_uniformly_from_domain (domain);
11         current_inertia_tensor += compute_pointMass_inertia(sample, dm);
12         ++drawn_num_samples;
13     return current_inertia_tensor
```

Figure 4.3.: Algorithm to approximate the inertia tensor using Monte Carlo method. One samples point masses from the volumetric domain of the multispherical particle and adds up the inertia of the point masses.

algorithm for this approach is shown in Figure 4.3, and the generated point masses for approximation of an example non-spherical particle shape are illustrated in Figure 4.4.

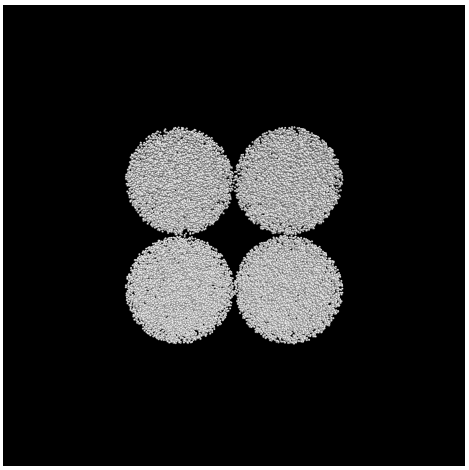


Figure 4.4.: Approximation of the inertia tensor using the Monte Carlo method, with generated point masses shown. Together, these point masses approximate the particle's shape, represented by four sub-spheres in a quadratic alignment.

5. Simulation Results

This chapter presents results of six simulation scenarios with the relevant macroscopic statistical values and the auto-tuning results.

5.1. Settling

Settling scenario simulates falling and settling down of particles. For this simulation, all contact force models have been used.

5.1.1. Scenario Description

Initially, $N = 1500$ particles were positioned in a cubic arrangement, uniformly distributed at a height of $z = 2$ above the ground within a relatively small domain D. This domain is defined by the corners $D_{\min} = (0, 0, 0)$ and $D_{\max} = (25.25, 25.25, 50.25)$, with surrounding reflective boundaries. The particles are subject to gravity, $g_z = -9.8$, causing them to settle downward. As the reflective boundaries do not dissipate the particles' energy e.g. as heat, the friction coefficients have been increased, and a background friction has been introduced to accelerate the settling procedure. The following parameters were used:

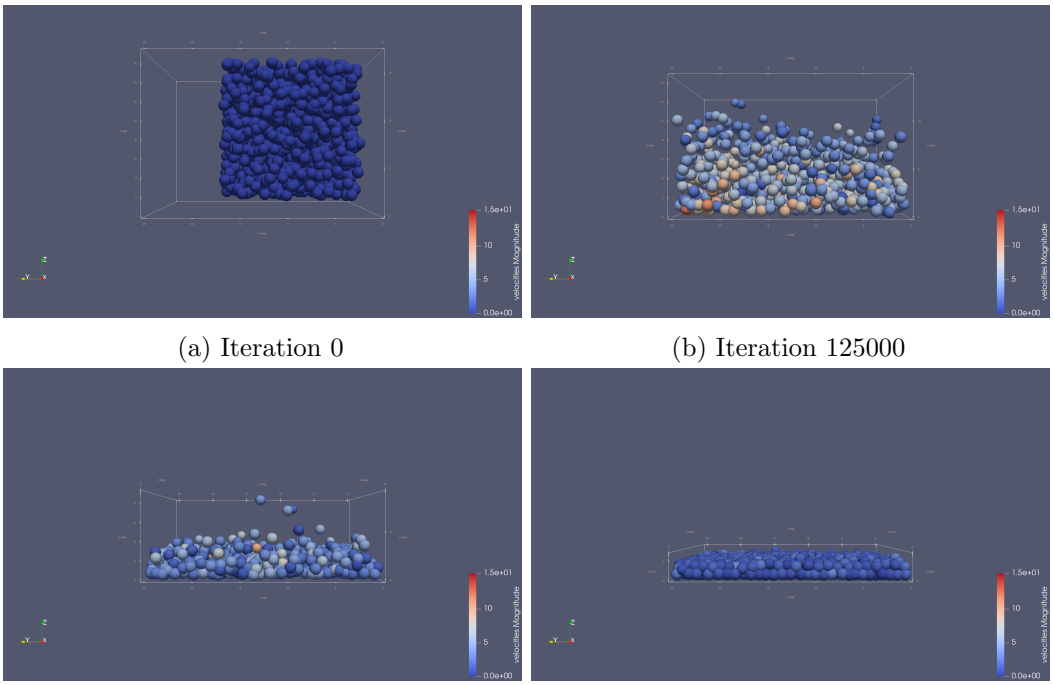
$$\begin{aligned} k^n &= 50, \\ \gamma_n &= 10^{-3}, \gamma_t = 10^{-1}, \gamma_r = \gamma_o = 10^{-2}, \gamma_{b/br} = 0.05 \\ \mu_s &= 7.5, \mu_d = \mu_r = \mu_o = 5, \\ d_{cutoff} &= 3, \Delta t = 5 \cdot 10^{-5} \end{aligned}$$

5.1.2. Simulation Results

The plots in Figure 5.2 provide evidence of particle settling, with a gradual decrease in overall movement over time.

In particular, the general decreasing development of kinetic energy shown in Figure 5.2 (a) effectively illustrates the settling process. The mean kinetic energy in all directions starts at zero, as no initial velocities are provided. The kinetic energy in the z-direction initially increases due to gravity but rapidly decreases as particles at the bottom of the initial cube bounce upward and collide with the still-falling particles above. This process repeats, exhibiting a quasi-"damped" oscillation, until the bouncing ceases and the kinetic energy approaches zero. As many collisions occur along the z-axis, tangential forces mostly act along the xy-plane. This is observable in the nearly identical kinetic energy increments along the x- and y-axes, which also settle over time.

A similar trend is observed for the rotational energy in Figure 5.2 (b). Rotational energy increases in all directions as particles collide with each other. However, as mentioned



(a) Iteration 0 (b) Iteration 125000

(c) Iteration 250000 (d) Iteration 375000

Figure 5.1.: Snapshots of Settling Scenario

previously, the rotational energy along the x- and y-axes increases more rapidly, as these axes are parallel to the tangential plane of many collisions, and reaches a higher peak than the rotational energy along the z-axis. Nevertheless, the rotational energy along the z-axis also increases continuously, as the domain size is larger than the initial cube. This allows the particles to spread out more, which results in more frequent collisions along tangential planes parallel to the z-axis. Again, due to friction, the rotational energy in all directions gradually settles down over time.

The graph of the number of contacts in Figure 5.2 (c) particularly confirms the aforementioned development of kinetic energy in the z-direction. The oscillation of the number of contacts at the beginning corresponds with the kinetic energy in the z-direction in such a way that an increase in contacts results in a decrease in kinetic energy due to friction. The number of contacts then reaches a plateau as the particles settle down.

Finally, the development of potential energy, which is proportional to the mean elevation of all particles along the z-direction, as shown in Figure 5.2(d), further supports the observed bouncing phenomenon and the general settling process.

5.2. Thermal Equilibrium of Stacked Particles

The thermal equilibrium scenario simulates the settling process in a more confined and elongated domain, incorporating the heat transfer model from Sec. 2.5.1.

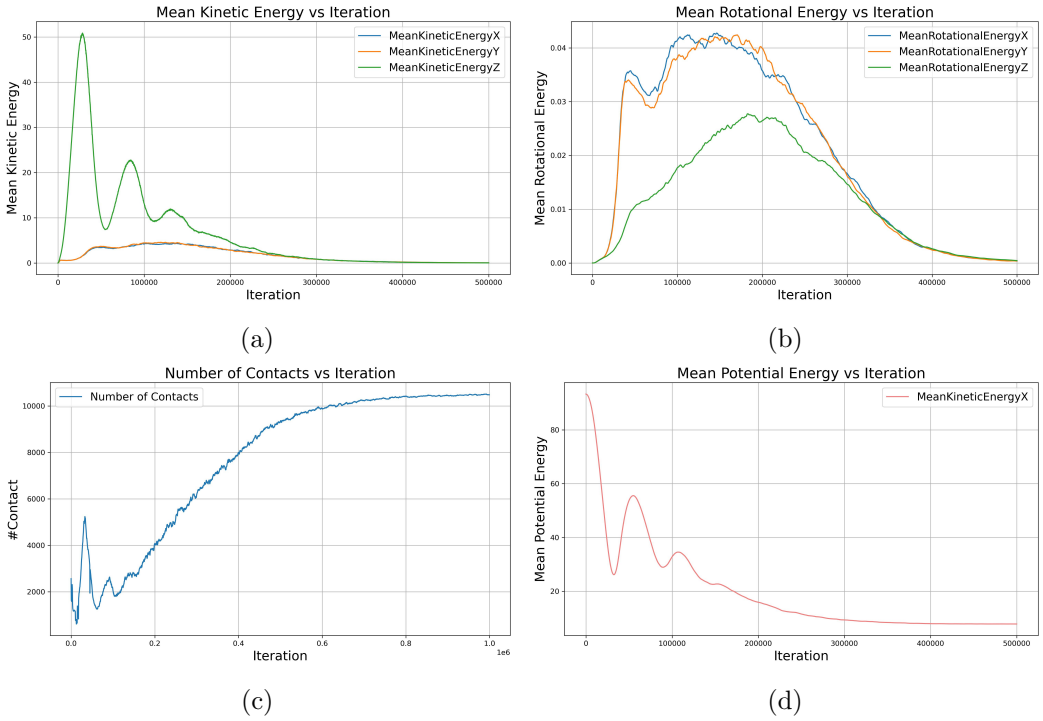


Figure 5.2.: Plots of Settling Scenario: Evolution of mean kinetic, rotational, and potential energy, and number of particle contacts throughout the simulation

5.2.1. Scenario Description

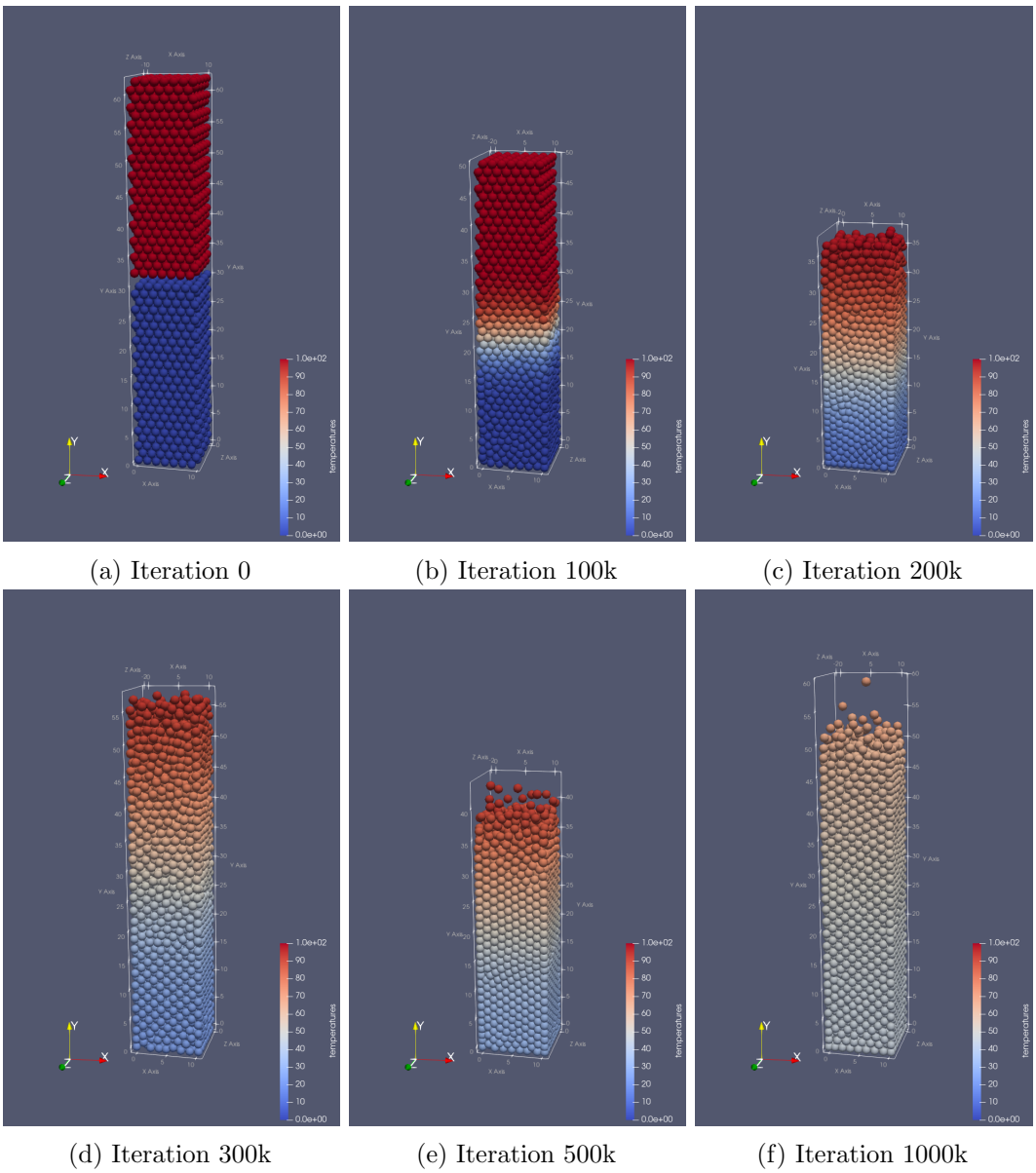
Two cubic blocks, each consisting of $N = 1484$ particles, are positioned above one another with a distance of 2 units. Initially, the two blocks have a temperature difference of 100 units, and they gradually undergo thermal equilibration as they settle down due to the downward gravity. Used parameters follow:

$$k^n = 100, k_S = E = C^p = 1, \|\mathbf{g}\| = 1$$

All other parameters are the same as those used in the settling scenario. The magnitude of the downward gravitational force has been reduced to account for the large number of particles stacked on top of each other.

5.2.2. Simulation Results

The snapshots and plots generally show strong similarities to the settling scenario, with a few key differences (see Figures 5.3 and 5.4). In Figure 5.4 (a), the first bump represents the initial settling phase. Following this, the entire particle pillar bounces upward in the second bump. During this second bump, the upward movement of the lower particles (`typeId`=0) is blocked by the upper particles (`typeId`=1), pushing the upper particles even further upward (similar to two balls bouncing on top of each other). This is further evidenced by the great difference in kinetic energy along y-axis between upper and lower particles. The kinetic



(a) Iteration 0 (b) Iteration 100k (c) Iteration 200k

(d) Iteration 300k (e) Iteration 500k (f) Iteration 1000k

Figure 5.3.: Snapshots of Thermal Equilibrium of Stacked Particles Scenario. Color is scaled by temperature.

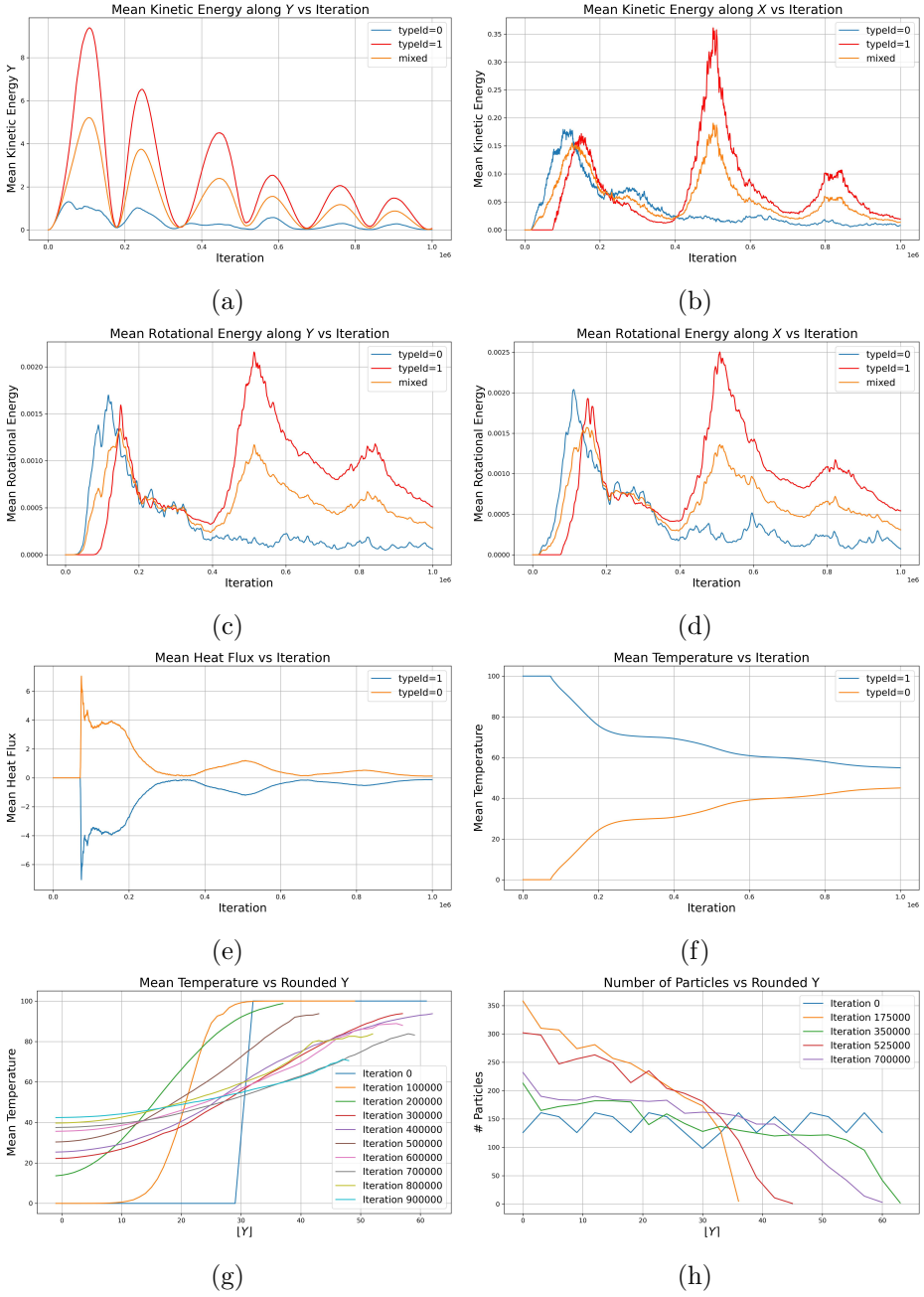


Figure 5.4.: Plots of Thermal Equilibrium of Stacked Particles Scenario: Evolution of mean kinetic and rotational energy, mean heat flux and temperature, profiles of temperature and number of particles along y-axis.

energy plots along the x- axis, as well as the rotational energy, show similar trends, while the energy evolution along the z-axis follows a pattern comparable to that along the z-axis. This occurs because the lower particles are spatially constrained as they are blocked by the upper particles, leaving little room for movement or rotation (for snapshots, refer to Figure A.11 in the appendix). Additionally, the relatively high density further limits their motion (see Figure 5.4 (h)). In contrast, the upper particles have more space to move and rotate due to the relatively low density in the upper half of the pillar (see Figure 5.4 (h)). This dynamic is further reflected in Figure 5.4 (b) to (d). The energy for the lower particles levels off after the first bump, while the upper particles move and rotate freely after they are bounced up.

Moreover, since only the heat transfer model from Section 2.5.1 is used here (and not the heat generation model), the heat flux is proportional to the overlap between particles. This overlap is maximized when the particles have settled or bounced down. As a result, the heat flux exhibits peaks after the first, third, and fifth bumps of the kinetic energy along the y-axis (see Figure 5.4 (e)) as the bumps correspond to alternating phases of settling and upward bouncing, beginning with settling. The heat flux also shows a bump during the first phase due to the great initial temperature difference and the present overlaps between particles during the initial settling phase. Furthermore, the temperature plot in Figure 5.4 (f) corresponds to the heat flux plot in such a way that the slope and change in temperature since the beginning of the simulation align with the value and integral of the heat flux, respectively. Lastly, Figure 5.4 (g) shows the mean temperatures along the y-axis, with the initial thermal difference gradually balancing out over time.

In summary, this scenario shares many similarities with the settling scenario. However, the effect of stacked balls is amplified due to the smaller bottom ground area and the greater number of particles stacked along the y-axis. Additionally, the effect of thermal equilibrium is observable.

5.3. Square Tumbler

This scenario simulates a tumbler, which is a machine commonly used in industry to granularize, dry, or to serve other purposes [GSL+16]. However, instead of the typical circular form, we use a square-shaped tumbler. The circular boundary complicates the boundary conditions, so a square domain is chosen to simplify the problem.

5.3.1. Scenario Description

This simulation models dynamics of $N = 4500$ particles confined within a domain surrounded by reflective boundaries. Each particle has a radius $r = 0.5$ and mass $m = 1$. After initialization, gravity is applied in the negative y-direction, simulating a settling scenario similar to the one described in Sec. 5.1. After the particles have settled, the direction of gravity is rotated to simulate the motion of a rotating tumbler. The rotation of the gravitation is formulated as follows:

$$\begin{aligned} \theta(t) &= \omega \cdot (t - t_0) \\ g_x(t) &= g \cdot \sin(\theta(t)) \\ g_y(t) &= g \cdot \cos(\theta(t)) \end{aligned} \tag{5.1}$$

The angular frequency ω represents the rotational speed, t_0 denotes the time at which the rotation begins after the settling phase, and g defines the magnitude of the gravitational force. Used parameters are presented in the following:

$$D_{min} = (-0.5, -0.5, -0.5), D_{max} = (30.5, 30.5, 7.5)$$
$$\gamma_{b/br} = 0.01, g = 17.5, \omega = \frac{\pi}{16}, \Delta t = 5 \cdot 10^{-4}$$

All other parameters are the same as those used in the settling scenario.

5.3.2. Simulation Results

Figure 5.5 presents snapshots of the square tumbler scenario. Snapshots (a) and (b) depict the initial settling process, followed by a quarter-turn rotation of the gravity (see Figure 5.5 (c)–(f)). Unlike a circular tumbler, which allows particles to cascade continuously, the square tumbler causes cascading movements to occur in bursts every quarter turn.

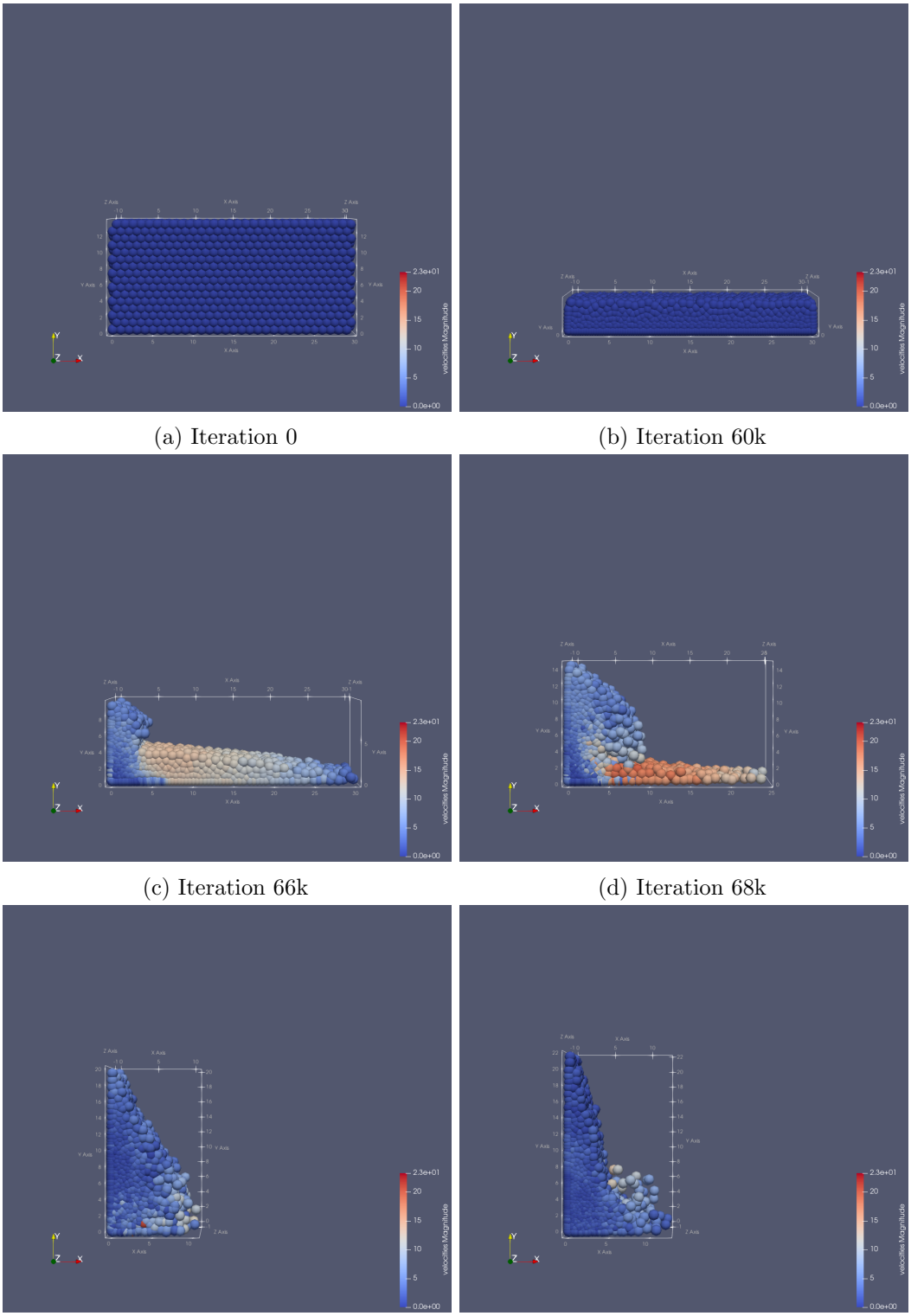
This effect is further evidenced in the plots in Figure 5.6. All three plots initially depict the settling process as expected. The mean kinetic energy decreases rapidly with some oscillations. This kinetic energy is subsequently converted into rotational energy, which also decays quickly. The number of contacts starts at zero and increases, reaching a constant value after some oscillation. Once the gravity begins to rotate, the mean kinetic energy increases periodically, alternating between the x- and y-directions, corresponding to the cascades occurring with each quarter turn of the xy-plane. This periodic increase in kinetic energy also subsequently leads to a rise in rotational energy, which predominantly occurs along the z-axis, perpendicular to the rotating xy-plane. The cascading motion first decreases the total number of contacts, but it later increases them again as the particles reach the opposite corner. The number of iterations that correspond to a quarter turn is computed as follows:

$$f = \frac{\omega}{2\pi} = 32(\text{s}) = N_{Iterations}^{full} \cdot \Delta t$$
$$N_{Iterations}^{quarter} = \frac{32}{4 \cdot \Delta t} = 1.6 \cdot 10^4,$$

i.e. the particles cascade every $N_{Iterations}^{quarter} = 1.6 \cdot 10^4$ iterations, which matches Figure 5.5 (a) approximately.

5.3.3. Auto-Tuning and Performance

Figure 5.7 presents the auto-tuning results of this scenario. Following aspects can be noticed:



(a) Iteration 0

(b) Iteration 60k

(c) Iteration 66k

(d) Iteration 68k

(e) Iteration 70k

(f) Iteration 72k

Figure 5.5.: Snapshots of Square Tumbler Scenario

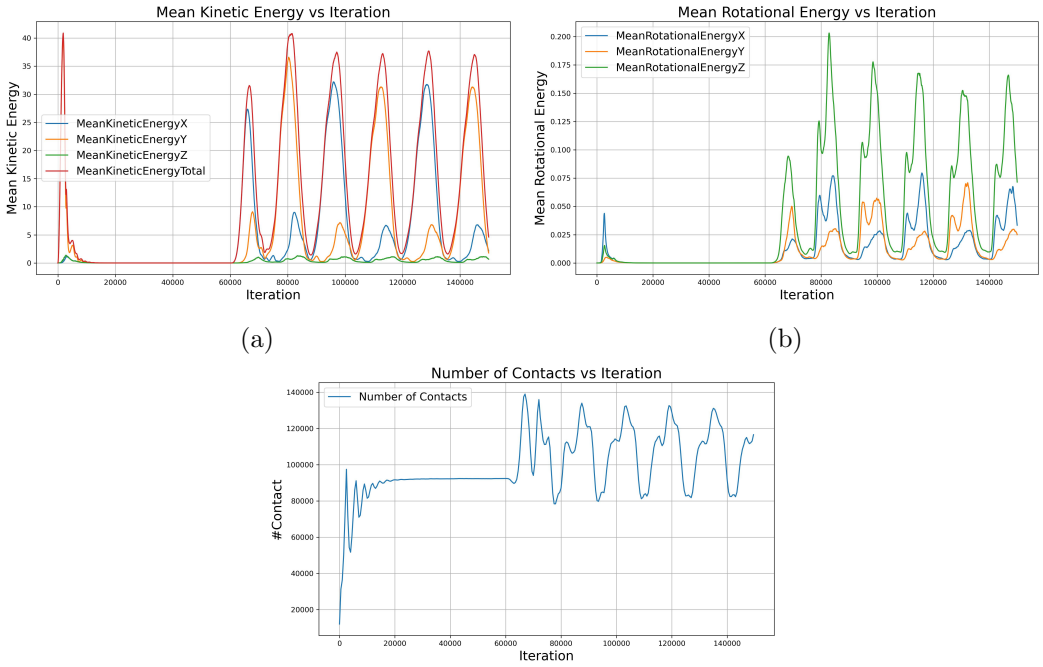


Figure 5.6.: Plots of Square Tumbler Scenario: Evolution of the mean kinetic and rotational energy, as well as the number of particle contacts, throughout the simulation.

- At the start of the simulation, the traversal method switches from `vcl_cluster_iteration` to `vcl_c01_balanced`, which governs the remainder of the simulation. The `vc_c01_balanced` traversal utilizes Verlet Cluster Lists (VCL) for neighbor identification and `c01` base step, as described in [GSBN21].
- In Figure 5.7 (a) and (b), the computation time for VCL shows the typical two-line behavior, as rebuild iterations require significantly more time compared to regular iterations.
- In Figure 5.7 (a) and (b), computation time of both Verlet Cluster Lists (VCL) and Linked Cells depend on the number of contacts (or particle density) in the iteration. However, computation time of Linked Cells exhibits higher fluctuations.
- AutoPas selects VCL over Linked Cells, which turns out to be reasonable. VCL shows faster computation time and lower fluctuations throughout the simulation.
- One explanation for the more efficient VCL compared with Linked Cells is the high density of the system of particles (due to high gravity). Higher hitrate of Verlet Cluster Lists profits from the high density showing its effect on lower computation time and lower fluctuation.
- A further reason for favoring VCL over Linked Cells and Verlet Lists (VL) is the continuity of neighborhoods and the lower cost of rebuilding Verlet Lists. In this scenario, particles that are neighbors directly after initialization remain neighbors throughout the settling and rotation phases of the tumbler, at least at short distances.

This leads to neighboring particles also being located close to each other in memory. As a result, VCL offers efficiency benefits, as particles that are neighboring in memory are grouped together in the same Verlet List, reducing data loading costs. Moreover, using VCL over VL benefits from shorter rebuild iterations, likely as a constant facter fewer VLs are needed, outweighing the drawback of the relatively lower hitrate.

- Figure 5.7 (c) compares the data layouts AoS and SoA with each other for VCL. The typical double lines are visible again due to rebuild iterations. AoS presents higher computational efficiency in the current implementation.

5.4. Heating Square Tumbler

This scenario incorporates the two heat models from Section 2.5 into the square tumbler scenario from above.

5.4.1. Scenario Description

The simulation setting is largely identical to the square tumbler scenario, with the key differences being the increased number of particles and the reduced magnitude of the gravitational force:

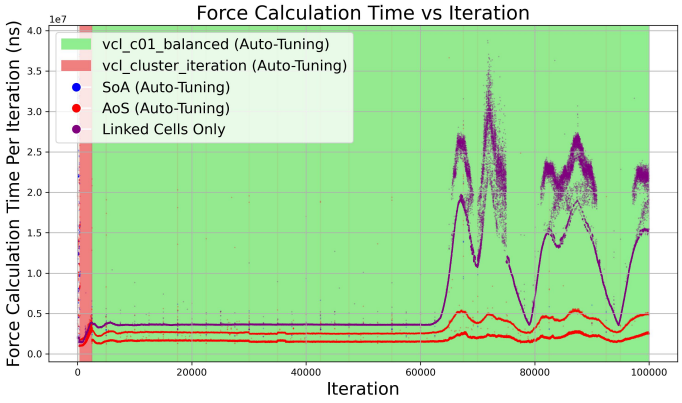
$$\begin{aligned} N_{particles} &= 1620, \\ k^n &= 100, k_S = E = C^p = 1, \eta = 0.1, \\ \gamma_{b/br} &= 0.1, g = 2.5, \omega = \frac{\pi}{16}, \Delta t = 10^{-4} \end{aligned}$$

5.4.2. Simulation Results

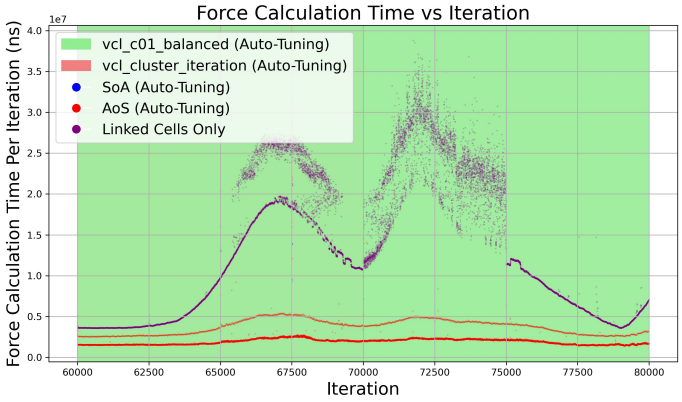
In general, the simulation results show strong similarities to the square tumbler scenario, especially the development of kinetic and rotational energy (see Figure 5.9 (a) and (b)).

Moreover, Figure 5.9 (c) and (d) show the developments of heat flux and temperature for different rotational speed variants, i.e., $\omega \in \{\omega_0 = \frac{\pi}{16}, 1.5 \cdot \omega_0, 2 \cdot \omega_0\}$. Here, the magnitude of gravity has been scaled by the square of the rotational speed multiplication factor, i.e., $g \in \{g_0 = 2.5, 1.5^2 \cdot g_0, 2^2 \cdot g_0\}$, according to the centrifugal force $F^{centrifugal} = m \cdot \omega^2 \cdot r_{domain}$. It can be observed that in Figure 5.9 (c) and (d), the multiplication factors of gravity correspond to the values of heat flux and temperature, respectively. Specifically, from about iteration 200k, the heat flux and temperature values for the doubled rotational speed, i.e., $\omega = 2 \cdot \omega_0$, are quadrupled compared to those for the original rotational speed. A similar pattern is observed for the rotational speed $\omega = 1.5 \cdot \omega_0$.

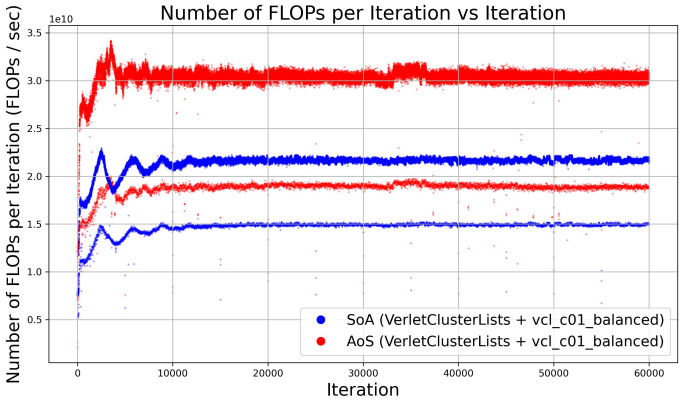
An additional observation from the snapshots is that particles near the boundaries or at the corners have higher temperatures compared to those located more centrally within the domain. Due to gravity, particles near the boundaries are compressed more as they must endure the weight of the particles above them. This results in a higher normal force needed. Since frictional forces are proportional to normal forces, the particles near the boundaries experience greater heat generation.



(a) Auto-Tuning



(b) Auto-Tuning (Zoomed in)



(c) AoS vs SoA

Figure 5.7.: Auto-tuning results of Square Tumbler scenario. (a) Auto-tuning results compared with using Linked Cells only as a fixed configuration: The colored vertical lines in the background indicate the selected traversal method by auto-tuning, while the color of the scatter points represents the chosen data layout (AoS or SoA) determined by auto-tuning. (b) Zoomed-in version of Plot (a) focusing on the interval with rotations of tumbler. (c) Comparison of AoS and SoA with fixed configuration (VerletClusterLists with `vcl_c01_balanced`). Relevant tuning parameters: `cell-size` = 1, `tuning-strategy` = `predictive-tuning`, `verlet-rebuild-frequency` = 10, `tuning-interval` = 2500

Lastly, Figure 5.4 (e) illustrates the development of temperature variance among the particles. Initially, the variance is negligible, but it increases after the first cascade and eventually stabilizes due to the relatively high thermal conductivity.

5.4.3. Variant with surrounding Walls

A variant of the heating square tumbler scenario can be created by adding surrounding walls. Reflecting the assumption that the surface of the tumbler walls generates more friction than the granular particles inside in reality, these walls are assigned friction coefficients that are three times greater than those of the particles themselves:

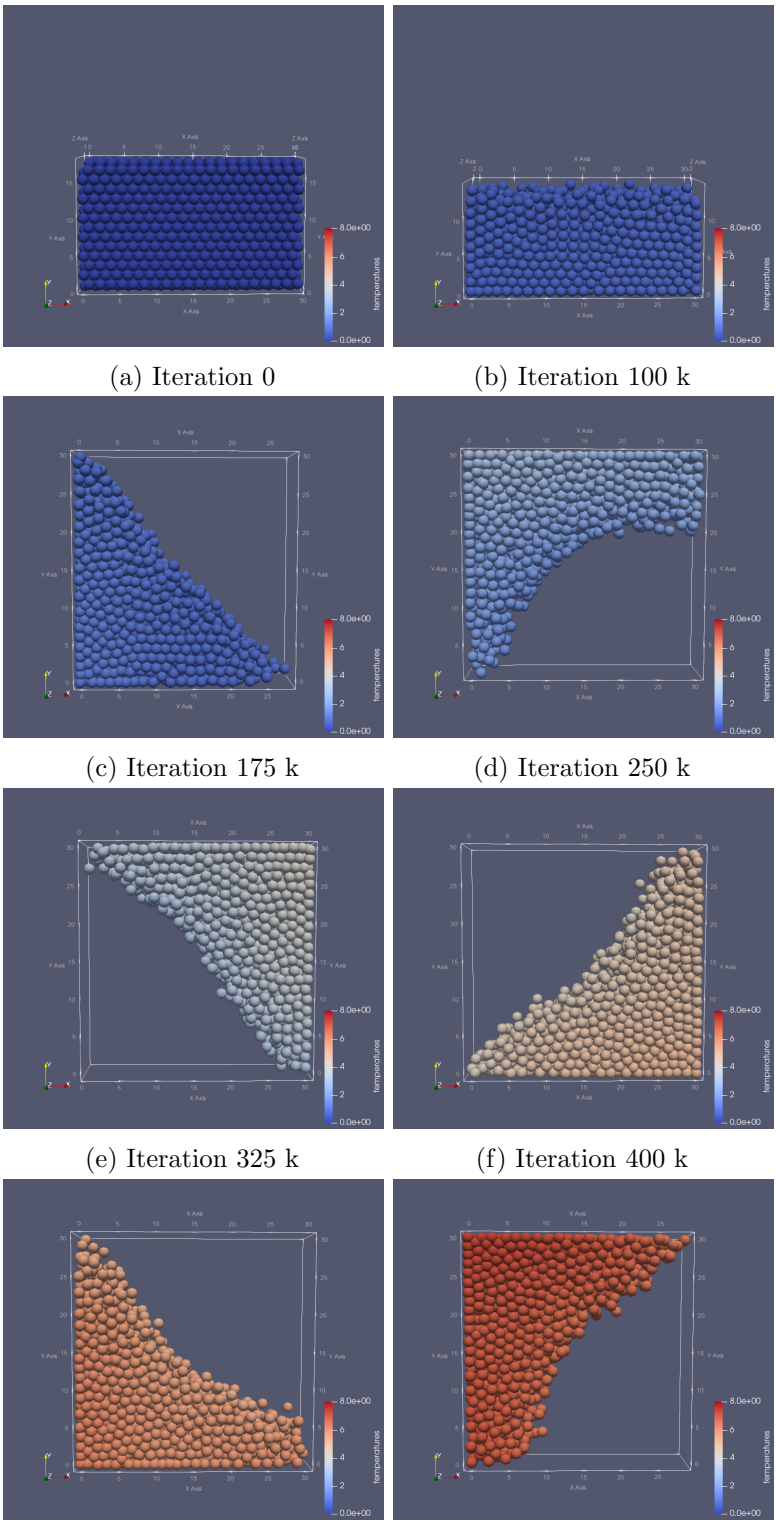
$$k^n_{wall} = 3 \cdot k^n_{solid}, \gamma_{n,wall} = 3 \cdot \gamma_{n,solid}, ...$$

Moreover, to enhance the effect of the surrounding walls, the heat conductivity of the granular particles has been reduced, i.e. $k_S = 0.05$. The results are presented in Figure 5.10 and 5.12. The following observations can be made from the snapshots in Figure 5.10:

- Particles near boundaries experience higher friction and increased heat generation (as in Sec. 5.4.2, see Figure 5.10 (b)).
- The higher temperature of the walls, relative to that of the particles, further amplifies the heat generation near the walls (see Figure 5.10 (d)).

These observations are also reflected in the plots of Figure 5.12:

- Figure 5.12 (a)
  - Adiabatic walls increase heat generation compared to the scenario without walls.
  - Cooled non-adiabatic walls produce two competing effects. On the one hand, the high friction between the particles and the walls attempts to increase the temperature of the particles. On the other hand, the low temperature of the walls, which remains nearly constant due to their extremely high specific heat, acts to cool the particles down. In this scenario, the cooling effect has a stronger influence, as evidenced by the lower temperature compared to the case without walls. However, the cooling effect does not completely dominate over the friction between particles and between the walls and particles, as the overall temperature of the particles still increases over time.
  - Heated non-adiabatic walls initially enhance heat generation. However, as the temperature of the particles surpasses that of the walls, the heating of the particles slows down, as seen in the decreasing slope from around Iteration 350k.
- Figure 5.12 (b)
  - The variants without walls and with adiabatic walls exhibit similar behavior. After the settling, the variance of temperature increases as particles near the boundaries or walls heat up but it eventually stabilizes over time. However, the variant with adiabatic walls reaches a higher maximum variance (relative to the variance in the settled state) due to the increased friction on the walls, and decreases more quickly as the relatively cooler particles heat up more easily when they come into contact with the walls.



(a) Iteration 0 (b) Iteration 100 k

(c) Iteration 175 k (d) Iteration 250 k

(e) Iteration 325 k (f) Iteration 400 k

(g) Iteration 475 k (h) Iteration 550 k

Figure 5.8.: Snapshots of Heating Square Tumbler Scenario. Color is scaled by temperature.

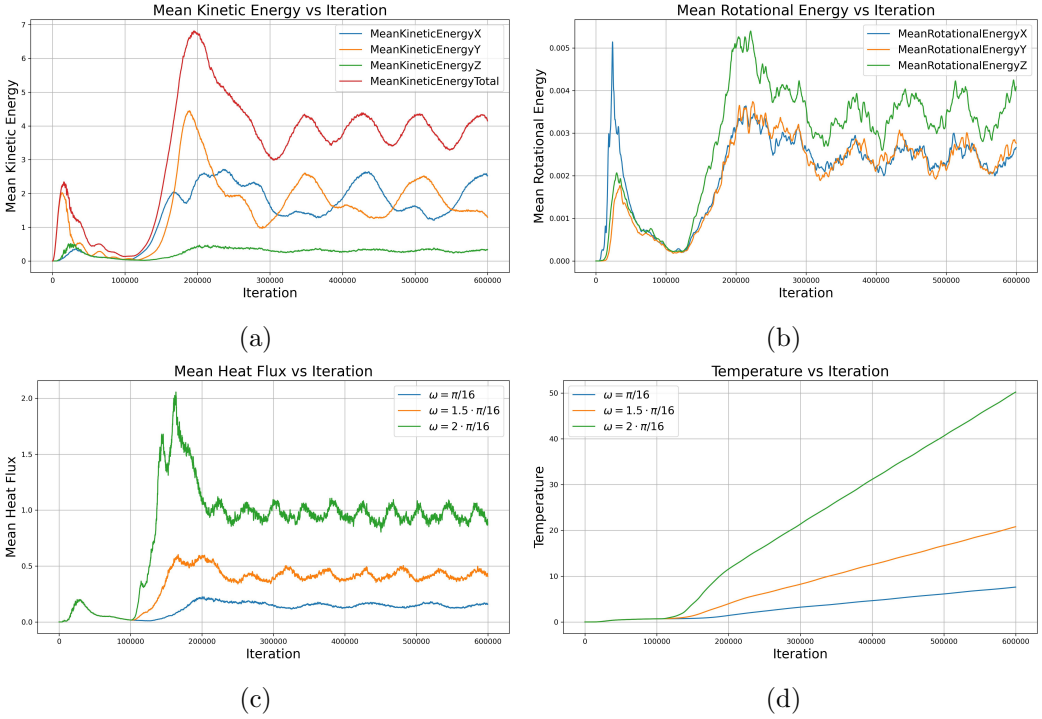


Figure 5.9.: Plots of Heating Square Tumbler Scenario: Evolution of mean kinetic and rotational energy, mean heat flux and temperature over time. Plots (a) and (b) show the scenario with $\omega = \frac{\pi}{16}$. Start of rotation is at Iteration 100 k.

  - The variant with cold non-adiabatic walls (i.e., $T^0_{wall} = 0$) shows a generally upward trend in variance. This is due to the continuous thermal discrepancies created by heating inside the particles (from friction) and cooling near the walls (see Figure 5.11). However, the variance plateaus after a sufficiently high number of iterations due to the positive conductivity of the particles (see Figure 5.12 (e)).
  - The variant with hot non-adiabatic walls (i.e., $T^0_{wall} = 5$) initially shows an increasing variance as the hot walls accelerate the heating process. However, after a certain point, the variance decreases due to the same reason as for adiabatic walls. Eventually, once the temperature of the particles increases sufficiently above that of the walls, the variance rises again. This occurs as the relatively cooler walls begin to cool the particles near them, while the inner particles continue to heat up due to friction, analogous to the behavior seen in the variant with cold non-adiabatic walls.

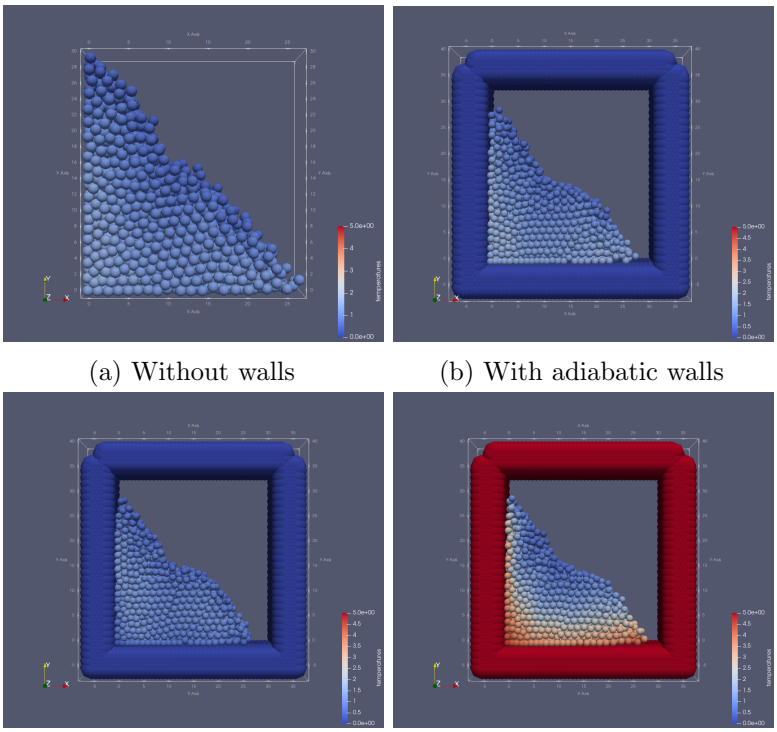
- Figure 5.12 (c) and (d)
  - The plots show the mean temperature along the axes for different variants after the first one-eighth turn (see Figure 5.10).
  - Temperature profiles in Figure 5.12 (c) exhibit U-shaped patterns. At lower x-coordinates, the mean temperature is higher due to the close distance to the left wall. At higher x-coordinates, the mean temperature increases as well, because the number of particles near the wall rises relative to those positioned further inside (see Figure 5.10. In contrast, at intermediate x-coordinates, the majority of particles are away from the walls, resulting in a lower mean temperature. The exception occurs in the variant with cold walls. At lower x-coordinates, the mean temperature is higher due to friction and insufficient time for cooling. However, at higher x-coordinates, the mean temperature decreases as a greater number of particles have already cooled down. A similar trend is also shown for y-coordinates in Figure 5.12 (d).

5.5. Fluidized Bed

A fluidized bed is a physical phenomenon in which granular material is subjected to both downward gravitational forces and upward gas or fluid flow. This upward flow, which opposes the gravity, converts the solid particles into a dynamic fluid-like state, resulting in a massive increase in the surface area exposed to the surrounding environment. This fluidized state enhances processes such as heat transfer or chemical reactions, e.g. under addition of a catalyst. Due to its effectiveness, fluidization is widely used in various industrial applications as a critical step in the processing [CKK14].

5.5.1. Scenario Description

The system contains $N_{total} = 26155$ particles, comprising $N_{solid} = 2142$ solid particles and $N_{gas} = 24013$ gas particles. For solid particles, reflective boundary conditions (BC) are applied to all six walls, while, for gas particles, periodic boundary conditions are used for the top and bottom walls. The implementation of two different BCs on a single wall is achieved



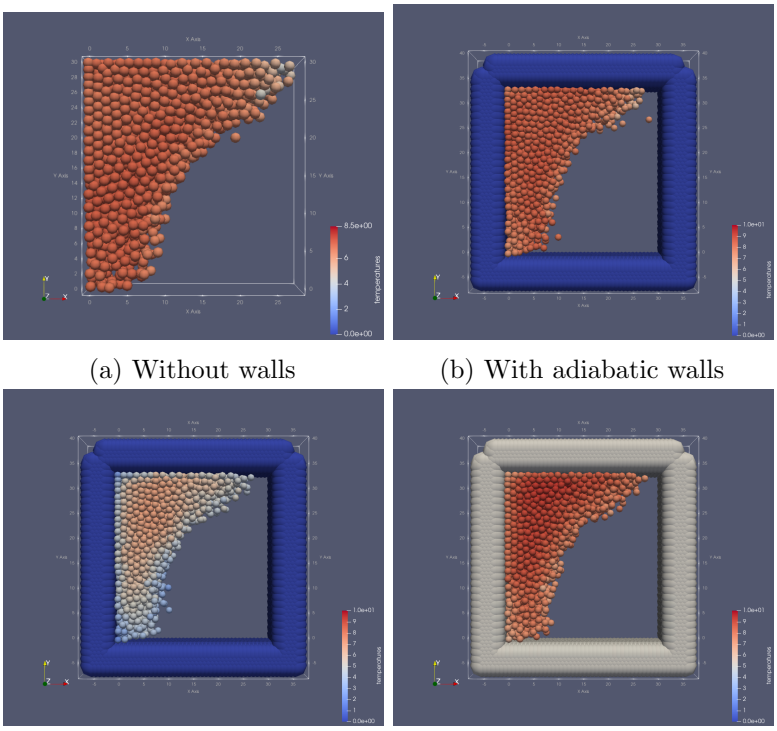
(a) Without walls

(b) With adiabatic walls

(c) With non-adiabatic walls, $T^0_{wall} = 0$

(d) With non-adiabatic walls, $T^0_{wall} = 5$

Figure 5.10.: Snapshots of heating square tumbler scenario with surrounding walls at Iteration 225k. Color is scaled by temperature. Start of rotation is at Iteration 150 k with $\omega = \frac{\pi}{16}$.



(a) Without walls

(b) With adiabatic walls

(c) With non-adiabatic walls, $T^0_{wall} = 0$

(d) With non-adiabatic walls, $T^0_{wall} = 5$

Figure 5.11.: Snapshots of heating square tumbler scenario with surrounding walls at Iteration 600 k. Color is scaled by temperature. Start of rotation is at Iteration 150 k with $\omega = \frac{\pi}{16}$.

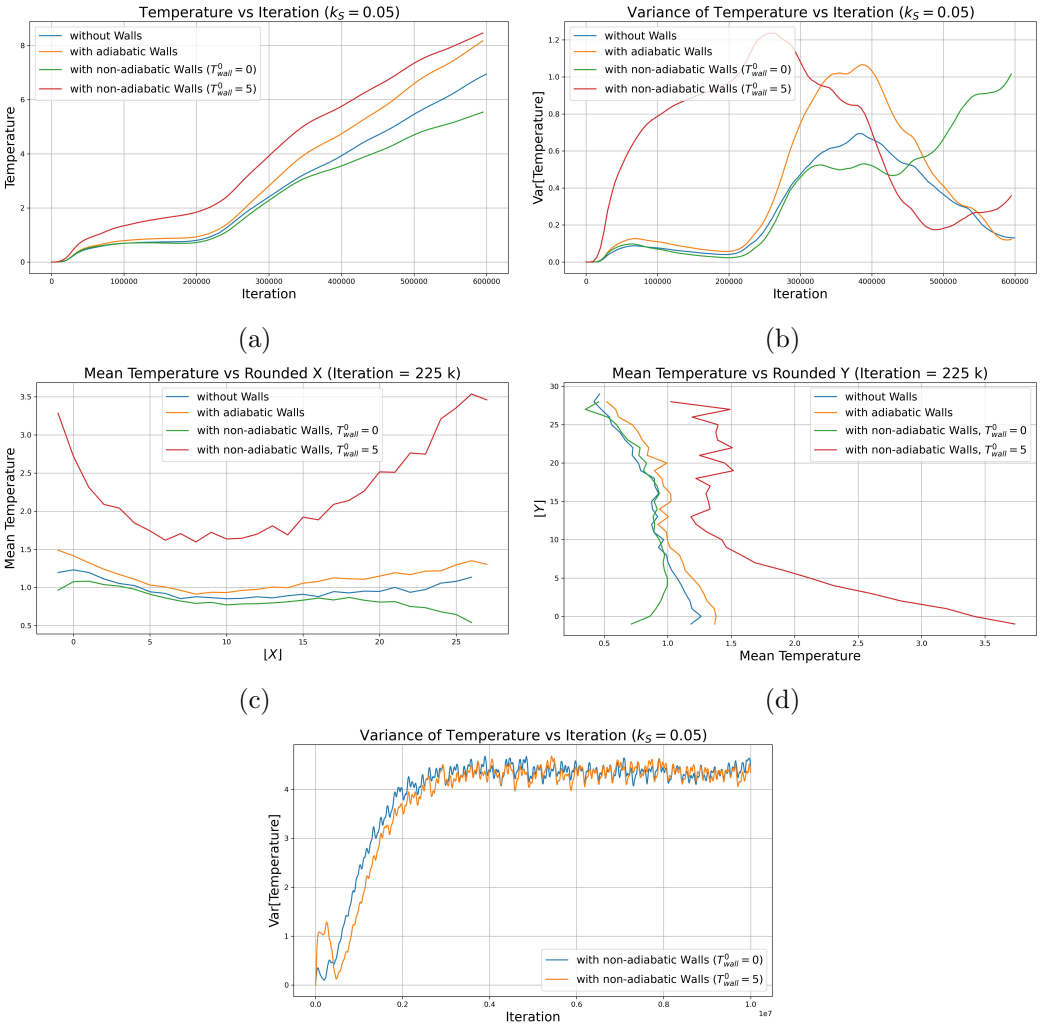


Figure 5.12.: Plots of Heating Square Tumbler Scenario with and without surrounding walls: Evolution of the mean and variance of temperature over time, along with temperature profiles along the x- and y-axes at iteration 225k. Used parameters: $\omega = \frac{\pi}{16}, k_S = 0.05$. Start of rotation is at Iteration 150 k.

by distinguishing the boundary conditions using different `typeId` values. As a result, the solid grains are confined within the domain, whereas the periodic BC allows for continuous upward flow of gas.

The system is initialized with solid particles positioned beneath the gas particles (see Figure 5.13). Initially, downward gravity is applied only to the solid particles (as in a settling scenario) to ensure an isotropic, dense arrangement of the solid particles. Once the settling phase is complete, gas particles are subjected to an upward gravitational force, simulating an upward flow originating from the bottom of the domain. Used parameters are presented in the following:

$$
\begin{aligned}
D_{min} &= (-0.75, -0.75, -0.75), D_{max} = (25.25, 25.75, 25.25) \\
m_{solid} &= 1, r_{solid} = 0.75, g_{down} = 9.8 \\
m_{gas} &= 0.001, r_{gas} = 0.1, g_{up} = 7.5 \cdot 10^{-2} \cdot g_{down} \\
\gamma_{b/br} &= 0, \Delta t = 5 \cdot 10^{-5}
\end{aligned}
$$

All other parameters are the same as those used in the square tumbler scenario.

5.5.2. Simulation Results

Figure 5.13 shows snapshots of the fluidized bed scenario. After the particles are initialized (see Figure 5.13 (a)), the solid particles undergo a settling process (see Figure 5.13 (b)). Following this, the upward gravity is activated for the gas particles, which initiates the expected upward gas flow (see Figure 5.13 (c)). However, due to the dense packing of the solid particles, which is a result of the settling process, the gas flow is unable to break through the solid layer (see Figure 5.13 (d)). Once sufficient pressure has accumulated at the bottom boundary, the densly aligned solid particles begin to misalign, creating gaps between them. Through these gaps, the gas particles start to flow upward, (see Figure 5.13 (e)), leading to the release of gas pressure and an increase in the flow rate (see Figure 5.13 (f)). Finally, Figure 5.13 (g) shows the fluidized state with larger inter-particle distances, resembling the behavior of a fluid or even a gas.

These observations are reflected in the plots in Figure 5.14. Figure 5.14 (a) and (b) depict the initial settling process, followed by the breakthrough (pressure release) and the fluidized state with higher kinetic and rotational energy. Figure 5.14 (c) shows the development of the volumetric flow rate Q_{approx}, which is calculated as:

$$
Q_{approx} = v_{flow} \cdot A_{bottom}
$$

where v_{flow} denotes the flow velocity and A_{bottom} is the cross-sectional area of the bottom wall. To approximate the flow velocity, the average velocity along the y-axis of gas particles positioned in either the bottom sixth or the top sixth of the domain has been computed, i.e.:

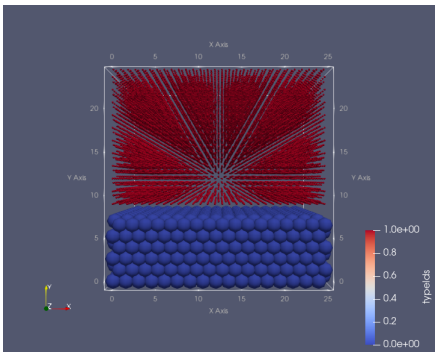
$$
x^y_{gas} < \frac{1}{6} \cdot D^y_{max} \quad || \quad x^y_{gas} > \frac{5}{6} \cdot D^y_{max}
$$

The number of particles considered in the computation of the flow rate, has also been plotted in Figure 5.14 (c). After the settling process, Q_{approx} increases rapidly due to the upward movement of gas particles at the top of the domain (see Figure 5.13 (c)), which, however, drops as expected (see Figure 5.13 (d)). The flow rate rises again due to the breakthrough (see Figure 5.13 (e)) and reaches a plateau in the fluidized state (see Figure 5.13 (g)). Figure 5.14 (d) shows the radial distribution functions throughout the simulation. The probability to find a pair of near solid particles with a distance less than approximately 2 is hightest at the end of the settling process (around Iteration 150k). However, it drops rapidly after the breakthrough ($\approx$ Iteration 200k) and then gradually increases, as the solid particles transition toward the fluidized state, eventually reaching a plateau around Iteration 400k.

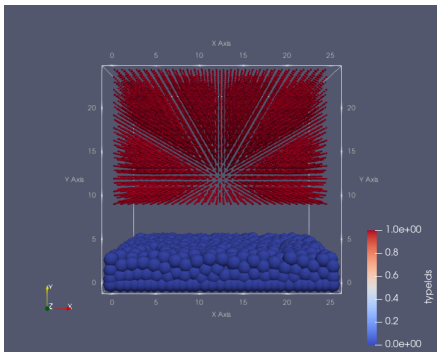
5.5.3. Auto-Tuning Results

Figure 5.15 and 5.16 present the results of auto-tuning. Following trends can be noticed:

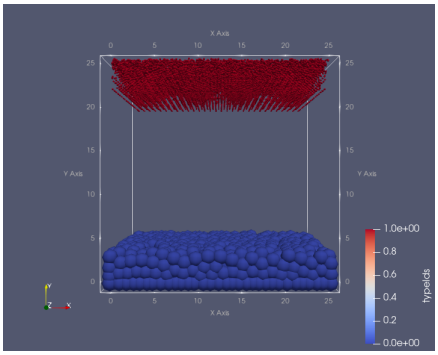
- Figure 5.15 (a) and (b):
  - The force computation time generally correlates with the number of contacts produced by the gas particles as $N_{gas} \gg N_{solid}$. Up until ca. Iteration 150k, the computation time remains relatively low, as the gas particles are stationary and do not produce overlaps yet. Between Iteration 155k to 175k, the computation time peaks due to the accumulation of pressure from the gas particles, which leads to a densely packed configuration with the highest number of contacts. After this point, as the interparticle distances increase and the system reaches a fluidized state (see Figure 5.14 (a) and (d)), the number of contacts levels off.
  - Apart from the interval with the highest density, the data layout AoS dominates. This is likely due to the relatively low number of particles per cell, which results from both the lower density and the small cell size relative to the particle radius or interparticle distance, i.e. `cell-size` $= 1$, $r_{solid} = 0.75$, $d_{gas}^{0,inter} = 0.85$ with $d_{gas}^{0,inter}$ being the initial particle spacing between gas particles, which would allow maximum of approximately five to six particles per cell. In such situations with a lower number of particles per cell, random access in data layout AoS can be more efficient than SoA. However, during the interval with the highest density, the data layout SoA brings more efficiencies, as the number of particles per cell is sufficiently high, and is also chosen by AutoPas (see Figure 5.16 (a)).
  - At around iteration 167k, the particles reach their highest density, with gas particles placed among solid particles (see Figure 5.16). At this point, the algorithmic configuration shifts from Linked Cells to Verlet Lists, using `lc_c01` and `vl_list_iteration` as their respective traversal methods. Detailed explanations of such traversal methods are given in [GSBN21]. Lower computation time with typically more expensive Verlet Lists rebuild iterations are visible on Figure 5.15 (b) between Iteration 167.5k and 170k. Despite of the expensive rebuild iterations, the average computation time of Verlet Lists (with `vl_list_iteration` and AoS) is expected to be lower than the one using Linked Cells (with `lc_c01` and SoA)



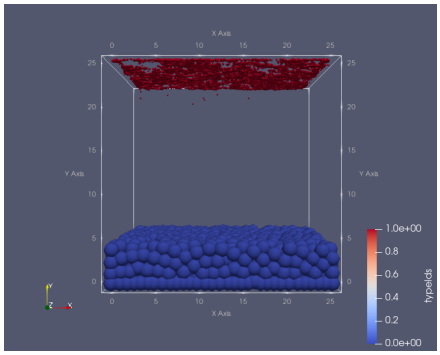
(a) Iteration 0



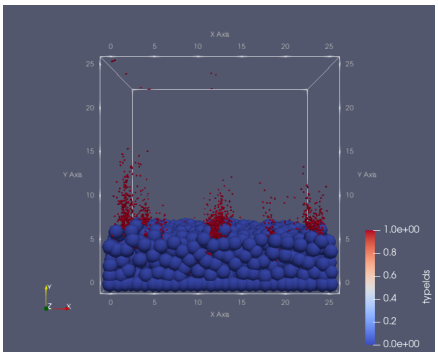
(b) Iteration 150 k



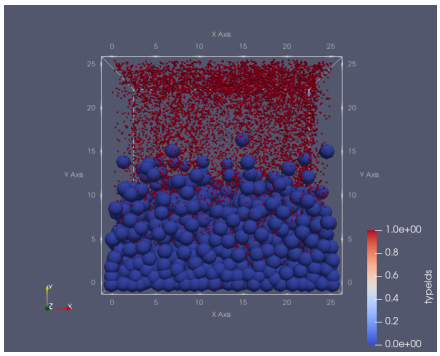
(c) Iteration 154 k



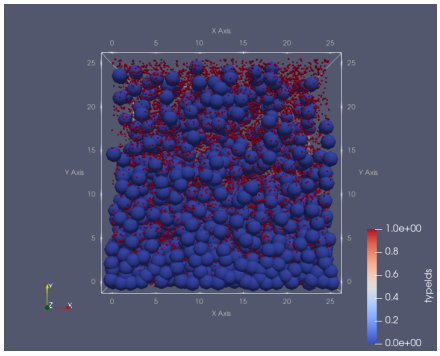
(d) Iteration 160 k



(e) Iteration 170 k



(f) Iteration 180 k



(g) Iteration 230 k

Figure 5.13.: Snapshots of Fluidized Bed Scenario: Gas and solid particles are colored red and blue, respectively. (a) Initial State, (b) Settled State, (c) Accelerated Gas, (d) Accumulated Pressure At Bottom, (e) Release of Pressure, (f) Fluidized State

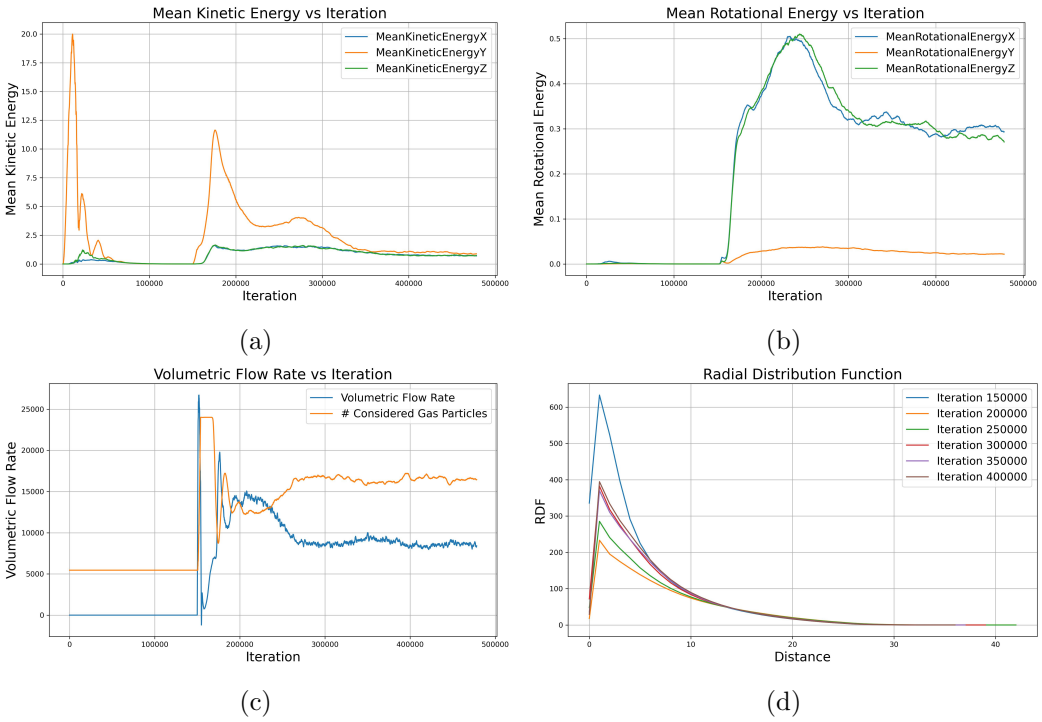


Figure 5.14.: Plots of the Fluidized Bed Scenario: Evolution of the mean kinetic and rotational energy, volumetric flow rate, and radial distribution functions throughout the simulation

(see Figure 5.15 (c)):

$$
\begin{aligned}
t_{vl}^{rebuild} &\approx 137 \cdot 10^6 \text{ ns}, \quad t_{vl}^{regular} \approx 101 \cdot 10^5 \text{ ns}, \\
t_{vl}^{average} &= \frac{t_{vl}^{rebuild} + t_{vl}^{regular} \cdot (\texttt{rebuild\_period} - 1)}{\texttt{rebuild\_period}} \\
&= 228 \cdot 10^5 \text{ ns} \quad \text{with } \texttt{rebuild\_period} = 10, \\
t_{lc} &\approx 279 \cdot 10^5 \text{ ns}.
\end{aligned}
$$

Here, these values are obtained from the outputs of `TuningDataLogger`. Refer to [GSBN21] for further information about loggers in AutoPas.

- One possible explanation for the higher efficiency of Verlet Lists is the transition to a quasi-homogeneous state (see [NGM+24]). Due to the high density, the number of particles per unit volume becomes relatively uniform. As a result, the benefit of the higher hit rate of Verlet Lists outweighs the cost of the expensive rebuild iterations.

- Figure 5.15 (d):
  - In general, AoS exhibits faster computation times, as indicated by AutoPas.
  - An exception occurs around iteration 159k, where SoA slightly outperforms AoS.

5.6. Strain-Stress

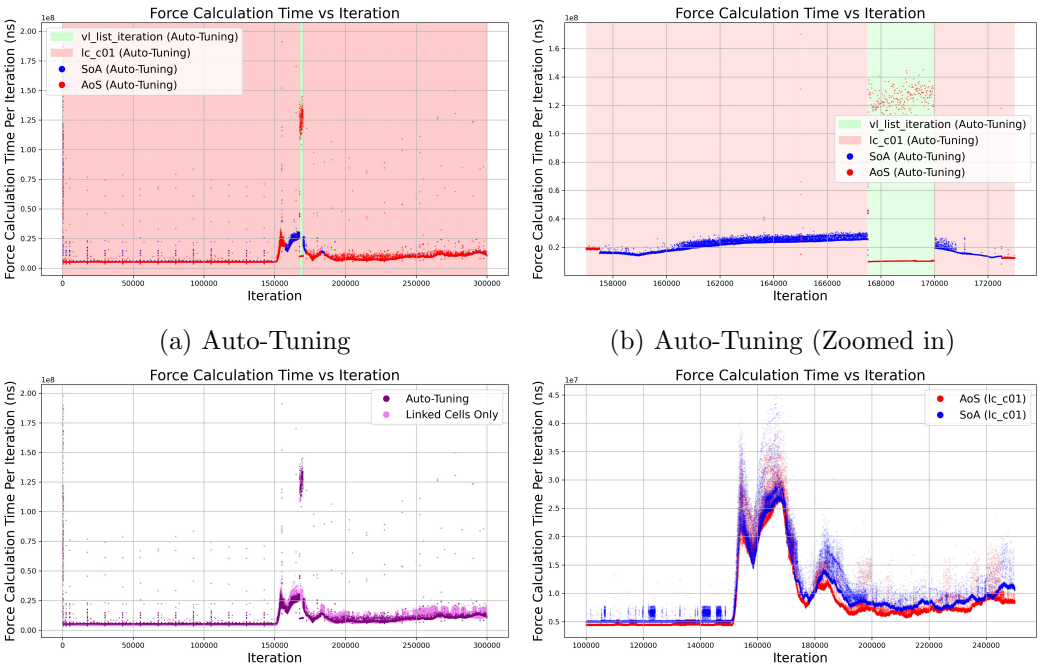
The strain-stress scenario discusses the reaction of the particles if they are opposed to strain, i.e. compression. This scenario takes reference of the parameters and results in [Lud08b].

5.6.1. Scenario Description

This simulation models dynamics of $N = 1845$ particles, each with a radius $r = 0.5$, mass $m = 1$. The particles are initially arranged in a closely packed diagonal formation within a cubic domain D, where the minimum corner is at $D_{min} = (0, 0, 0)$ and the maximum corner is at $D_{max} = (15.5, 15.5, 5.5)$.

The upper wall of the domain, which is parallel to the xz-plane, moves slowly downward according to a predefined maximum strain. This downward movement is modeled using a cosine function, which reaches the maximum strain after its half-period. Once the maximum strain is reached, the position of the upper wall remains fixed for the rest of the simulation. In contrast, the right wall, which is parallel to the yz-plane, moves in response to the balance of forces acting on it from both inside and outside the domain. The external force is specified by a predefined pressure applied to the right wall, which is opposed by the repulsive forces exerted by the particles within the domain. The front wall along the z-axis remains fixed throughout the simulation. This experimental setup is illustrated in Figure 5.17 and the boundary conditions described above are formulated as follows [Lud08b]:

$$
y(t) = \begin{cases} y_f + \frac{y_0 - y_f}{2}(1 + \cos(\omega t)) & \text{for } t \in [t_0, \frac{T}{2}] \\ y_f & \text{for } t > \frac{T}{2} \end{cases} \tag{5.2}
$$



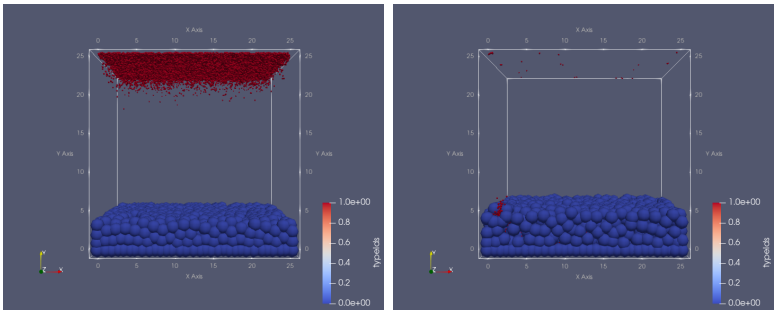
(a) Auto-Tuning

(b) Auto-Tuning (Zoomed in)

(c) Auto-Tuning vs Linked Cells Only

(d) AoS vs SoA of Linked Cells (Zoomed in)

Figure 5.15.: Auto-tuning Results of Fluidized Bed Scenario. (a) Auto-tuning results: The colored vertical lines in the background indicate the selected traversal method by auto-tuning, while the color of the scatter points represents the chosen data layout (AoS or SoA) determined by auto-tuning. (b) Zoomed-in version of Plot (a) focusing on the interval with configurational changes. (c) Comparison of Auto-Tuning and fixed configuration (`lc_c01`) (d) Zoomed-in comparison of AoS and SoA with fixed configuration (`lc_c01`). Relevant tuning parameters are same as in Figure 5.7



(a) First Change to SoA

(b) Change to Verlet Lists

Figure 5.16.: Snapshots of Fluidized Bed Scenario at iterations of configuration changes. (a) First change from AoS to SoA around Iteration 156k (b) Change from Linked Cells to Verlet Lists around Iteration 168k

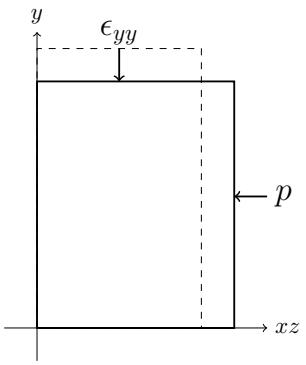


Figure 5.17.: Setup of strain-stress scenario: strain-controlled top wall and stress-controlled right wall
Source: [Lud08b]

where $y(t)$ represents the size of the domain along the y-axis at time t. y_0 and y_f denote the initial and final sizes, respectively, while ω and T represent the rate of deformation and the period, respectively, with the following relation:

$$\omega = 2\pi f, \quad T = \frac{1}{f} \tag{5.3}$$

The front wall remains unchanged:

$$z(t) = z_0 \tag{5.4}$$

The movement of the right wall is controlled as follows:

$$\ddot{x}_{wall} = \frac{1}{m_{wall}} \cdot (F_{inside} - p \cdot A_{wall}) \tag{5.5}$$

$$\|\Delta x_{max}\| = k_r \cdot r, \quad k_r \in (0, 1) \tag{5.6}$$

$$\|\Delta x_0\| = \|\frac{k_x}{2} \cdot \ddot{x}_{wall} \cdot (\Delta t)^2\| \tag{5.7}$$

$$\Delta x = sign(\ddot{x}_{wall}) \cdot \min(\|\Delta x_0\|, \|\Delta x_{max}\|) \tag{5.8}$$

The term m_{wall} represents an imaginary mass of the right wall, introduced to dampen fast oscillations of the wall. F_{inside} is the total force exerted on the right wall by the particles within the domain, p denotes the predefined pressure, and A_{wall} refers to the current area of the right wall. With these values, the acceleration of the wall, i.e. $\ddot{x}_{wall}$, is computed (see Eq. 5.5), which is used to compute the movement of the wall, i.e. Δx, via the following displacement-time-relationship:

$$\Delta x = v\Delta t + \frac{1}{2}a(\Delta t)^2 + O((\Delta t)^3) \tag{5.9}$$

As the velocity information, i.e. v, is not given, the constant k_x is added control the underestimated Δx.

In summary, the motion of the right wall is governed by the difference between the internal forces (from within the domain) and external forces (acting on the wall). However, excessively large movements of the wall within a single timestep can cause particles near the wall to escape the domain, which is undesirable. To prevent this, the maximum allowable movement of the wall, denoted as $\|\Delta x_{max}\|$, is defined as a proportion of the particle radius. Consequently, the resulting Δx is computed as the minimum of the maximal allowable movement and the computed movement, while preserving the direction of motion using $sign(\ddot{x}_{wall})$.

Reflective Boundary and F_{inside}

As described in the scenario, the entire domain is confined by reflective boundaries to ensure that all particles remain within the domain. To achieve this, repulsive forces based on the Lennard-Jones 12-6 potential are used by default to push particles near the boundaries back into the domain. Using normal forces from Sec. 2.4 would be unsuitable because these forces require a positive overlap with the boundary to become active, which could allow particles to cross the boundary—an undesirable outcome. Consequently, F_{inside} i.e. the sum of the forces exerted by particles on the wall, is approximated by the sum of the Lennard-Jones repulsive forces on the right boundary, with the zero-crossing set to match the radius of the particles, i.e., $\sigma_{LJ} = r$. However, since the Lennard-Jones (LJ) potential generally produces a much stronger repulsive force than the normal contact forces for distances between particles and walls less than sigma, i.e. $d < \sigma_{LJ}$, this approach does not perfectly control the boundary condition with appropriate scaling. This issue should be further addressed in the parameter settings.

Parameter Scaling

The reference paper [Lud08b] provides the parameters used in their simulation, expressed in standard units. This information can be utilized to appropriately scale the parameters.

The reference parameters are:

$$r \approx 10^{-3}\text{m}, \quad k^n = 10^5 \frac{\text{N}}{\text{m}}, \quad p \in (20, 40, ...) \tag{5.10}$$

Our parameters, i.e. $\tilde{m}, \tilde{r}, \tilde{k}$, can be used to compute scalars:

$$\begin{aligned} r^* = \frac{r}{\tilde{r}} = \frac{10^{-3}\,\text{m}}{0.5} = 2 \cdot 10^{-3}\,\text{m}, \quad k^* = \frac{k}{\tilde{k}} = \frac{10^5\,\frac{\text{N}}{\text{m}}}{100} = 10^3 \frac{\text{N}}{\text{m}} = 10^3 \frac{\text{kg}}{\text{s}^2} \\ \Rightarrow p^* = \frac{k^*}{r^*} = 5 \cdot 10^5 \frac{\text{N}}{\text{m}^2} \quad \text{assuming Pa as pressure unit.} \end{aligned} \tag{5.11}$$

The scaled pressure $\tilde{p} = \frac{p}{p^*} \approx 4 \cdot 10^5$ can be further increased to about 10^4 considering higher repulsive forces of LJ potential.

5.6.2. Simulation Results

The results of this scenario are illustrated in Figure 5.18, which allow following observations and realizations:

- The domain box exhibits a spring-like behavior as a whole, which aligns with the underlying linear spring model of the discrete particles.
  - Development of density $v = \frac{N \cdot \pi \cdot r^2}{V}$ (see Figure 5.18 (b)) suggests initial compression (increasing v) followed by strong dilation (decreasing v, eventually approaching a quasi-steady-state (decreasing $\frac{dv}{dt}$)).
  - This trend is further reflected in the evolution of volumetric strain $\epsilon_V = \frac{\Delta V}{V}$. Initially, ϵ_V decreases to below zero, indicating compression, followed by a significant increase until it slowly plateaus.
  - Similar trend is also visible on the development of domain size along x-axis (see Figure 5.18 (a)).
- Static stress, which describes kinetic energy of particles, is greatest along the direction of dilation (x-axis).
- Dynamic stress, which summarizes active contact forces, is greatest along the direction of active strain (y-axis).

Further details of formulas and stress calculation are given in [Lud08b]. Moreover, details of these results differ from the reference experiments in [Lud08b], presumably due to the mentioned scaling issue (see Sec. 5.6.1). However, the overall trend of initial compression followed by dilation is consistent with the reference results in [Lud08b].

5.7. Alignment of Non-spherical Particles

This scenario discusses the alignment of non-spherical particles if they are subjected to some background flow.

5.7.1. Scenario Description

Over the domain D, small $N_{gas} = 2750$ gas particles are uniformly distributed. A global force is applied in the positive x-direction for these gas particles. However, a velocity cap is enforced to prevent these particles from undergoing endless acceleration. Alongside, larger $N_{solid} = 50$ solid particles with a square shape, as shown in Figure 4.4, are also uniformly distributed. The normal vectors of these solid particles are aligned along the x-axis, i.e. their flat sides face the yz-plane. Unlike the gas particles, no global force is applied to these solid particles.

The domain boundaries along the x-axis employ periodic BC, while reflective BCs are applied to other walls. During the simulation, the solid particles are expected to naturally align themselves in the direction that minimizes the collision rate with the moving gas particles. Specifically, the x-component of the normal vectors of the solid particles is expected to decrease. Used parameters follow:

$$\begin{aligned} D_{min} = (0, 0, -2.5), D_{max} = (52.5, 20.0, 12.5), \\ m_{solid,subsphere} = 0.5, r_{solid,subsphere} = 0.75 \\ m_{gas} = 0.3, r_{gas} = 0.2 \\ \gamma_{b/br} = 0.25, \Delta t = 5 \cdot 10^{-5} \end{aligned}$$

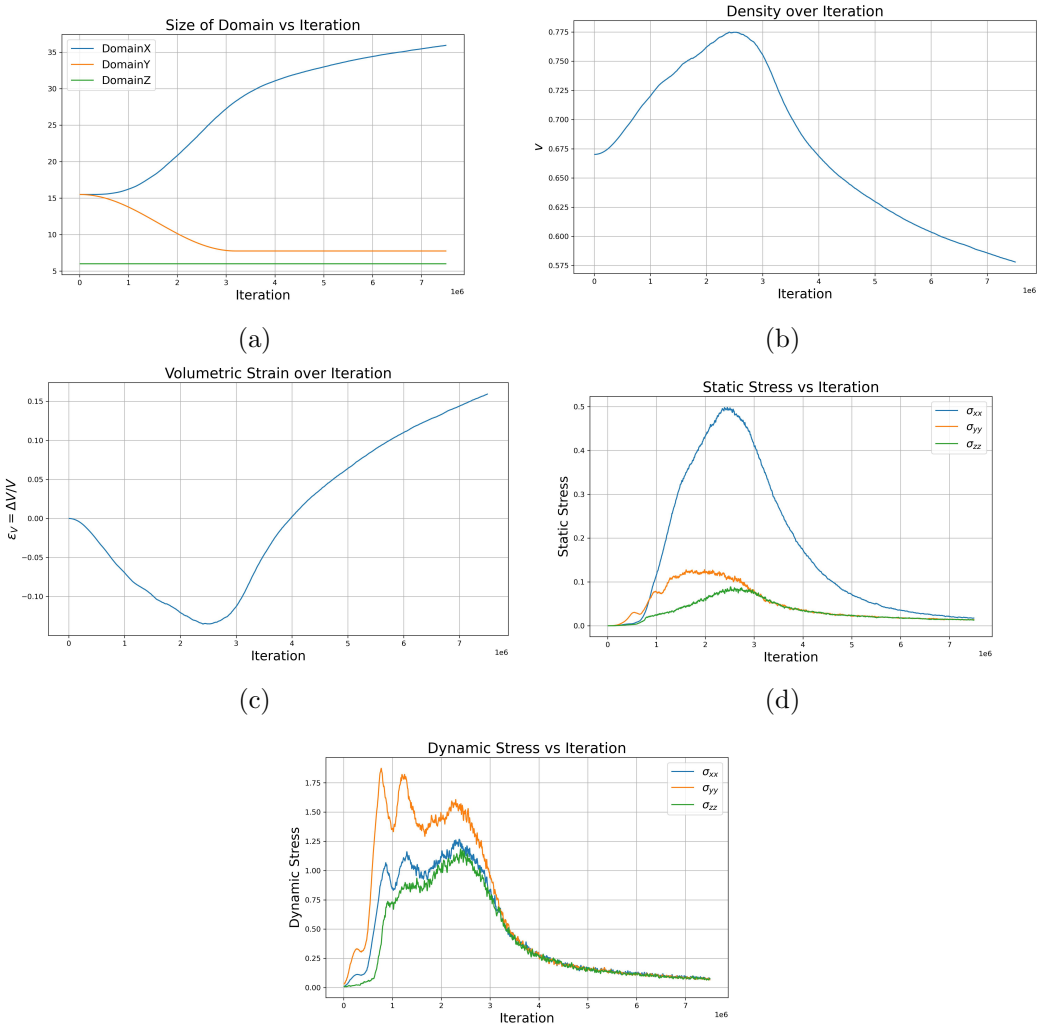


Figure 5.18.: Plots of strain-stress scenario: Evolution of domain size along each axis, density, volumetric strain, as well as static and dynamic stress throughout the simulation.

5.7.2. Simulation Results

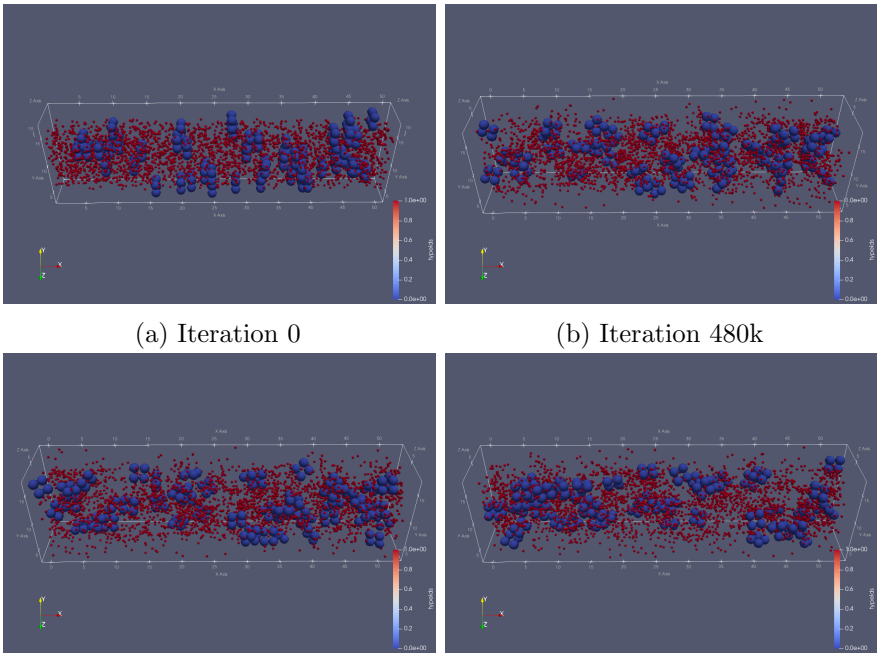
Figure 5.19 and 5.20 present the simulation results of this scenario. Comparing Figure 5.19 (a) and (d), it is noticable, that solid particles have rotated in such a way, that their flat surfaces face not the yz-plane anymore, but rather the xz-plane.

This observation is also reflected in the plots of Figure 5.20. Figure 5.20 (a) suggests decreasing $\|n_x\|$, while $\|n_y\|$ and $\|n_z\|$ remain relatively large. However, contrary to our initial expectation, $\|n_x\|$ does not approach zero, but instead plateaus at a non-zero value. One possible explanation for this is that, over the course of the simulation, the solid and gas particles undergo spatial segregation. As the particles exhibit minimal movement along y- or z-axis (see Figure 5.20 (c) (d)) and no mechanism exits to fill empty space (in contrast to gas dynamics), the solid and gas particles naturally separate from each other over time. As a result, only a few collisions occur between them toward the end, which is insufficient to further reduce $\|n_x\|$.

Moreover, the variance of $\|n_x\|$ is over the simulation lower than $\|n_y\|$ and $\|n_z\|$ (see Figure 5.20 (b)), which aligns with our expectations. When a solid particle is oriented along the x-axis (i.e. its normal vector is parallel to the yz-plane), it can freely rotate around the x-axis, i.e. parallel to the yz-plane, and it would not increase collision rate with gas particles. Therefore, the variance of $\|n_y\|$ and $\|n_z\|$ is expected to be relatively large.

Regarding the Figure 5.20 (c) and (d), the movement of gas particles along the x-axis is nearly constant at its velocity cap over time. In contrast, the movement of solid particles decreases over time, primarily due to the positive background friction coefficient $\gamma_{b/br}$, as well as the decreasing number of collisions with gas particles, which, in contrast, continue to move at a constant speed.

Lastly, to better address the above-mentioned issue of spatial separation in future simulations, one potential approach is to extend the reach of the repulsive forces from the reflective boundaries. Currently, the Lennard-Jones potential is used to model these reflective forces. However, employing an alternative function with a gentler slope and a larger zero-crossing could better maintain the particles near the center of the domain and prevent spatial separation.



(a) Iteration 0 (b) Iteration 480k

(c) Iteration 960k (d) Iteration 1920k

Figure 5.19.: Snapshots of Alignment of non-spherical particles scenario. Gas and solid particles are colored red and blue, respectively.

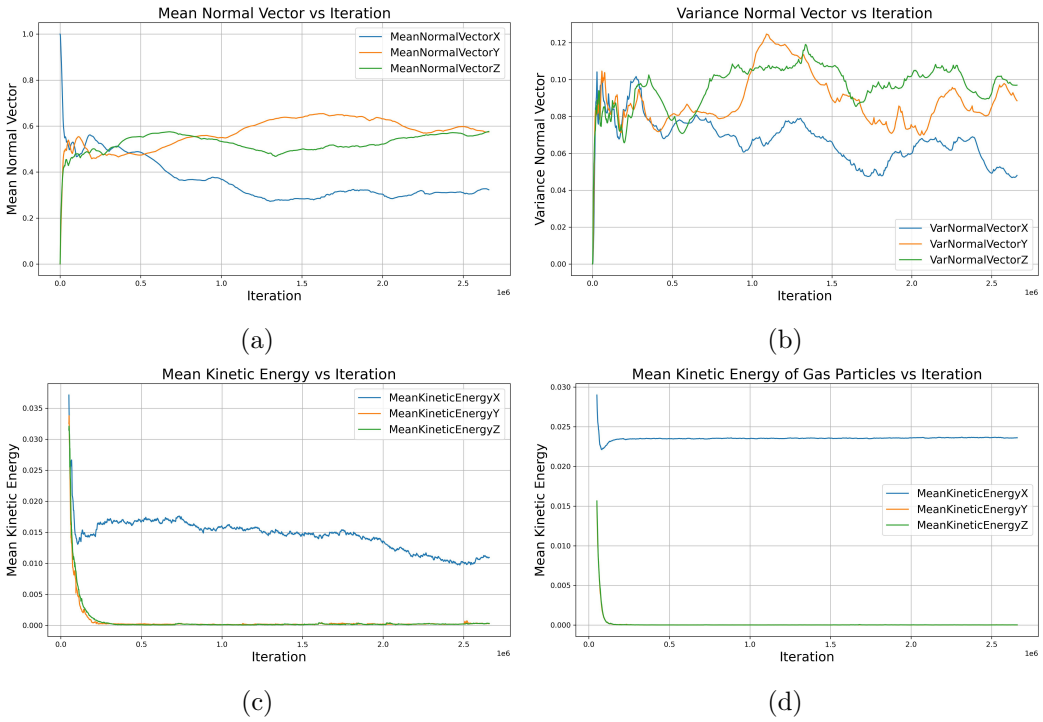


Figure 5.20.: Plots of Alignment of non-spherical particles scenario. (a) Mean of absolute normal vector, i.e. $\mathbf{n_{abs}} := (\|n_x\|, \|n_y\|, \|n_z\|)^T$ (b) Variance w.r.t. $\mathbf{n_{abs}}$. Plots (c) and (d) display mean kinetic energy of solid and gas particles, respectively. For Plots (c) and (d), the data before Iteration 50k is truncated for better visualization, as both types of particles exhibit chaotic movement (high kinetic energy) directly after their initialization due to the initial overlaps between them.

6. Future Work

The models implemented in md-flexible as part of this work already yield promising results and capture physical phenomena relatively well. However, there is room for further enhancement to achieve more accurate simulations. Below are some ideas for improvements and interesting topics for future exploration.

- **Addition of heat dissipation model:**
  A model for heat dissipation could be introduced to balance the heat generated in the heating square tumbler simulations from Sec. 5.4. This heat dissipation would account for the transfer of heat from the heated particles and wall elements to the air inside the tumbler. Such a model can be found in [HRK+24].
- **Framework for generation of objects with smooth surfaces:**
  A framework for generating boundary objects with smooth surfaces, such as flat walls, could be implemented in md-flexible to improve simulation accuracy and user friendliness. While walls composed of small overlapping spherical particles, as used in the square tumbler scenarios from Sec. 5.3, can approximate flat walls, they suffer from computational inefficiencies when there are too many wall particles and lack accuracy when there are too few as the surface would then not be flat. Moreover, such a framework would also simplify the setup of simulations for scenarios such as square tumbler with walls, strain-stress from Sec. 5.6, or hopper in Figure 1.1, if it would offer ways to easily generate such boundary objects. In a similar context, the framework could also support the creation of circular walls, which would enable the simulation of a circular tumbler—widely used in granular processes such as mixing, granulation, and coating [EKB+21]—instead of a square one.
- **CFD-DEM coupling:**
  AutoPas could be further coupled with a computational fluid dynamics (CFD) solver to enhance the accuracy of simulations involving granular materials subjected to gas or liquid flows. Scenarios from Sec. 5.5 and Sec. 5.7 simulate gas flows using small DEM-applied granular particles. However, this approach does not fully capture gas-specific behaviors, such as the flow from high to low pressure. Similar limitations apply to liquid flows simulated with DEM. By coupling AutoPas with a CFD solver, more accurate and realistic results could be achieved, capturing these dynamic gas and fluid behaviors. For further details, refer e.g. to [GSGZ+20].
- **More complex modeling of non-spherical particle shapes:**
  The multi-sphere method from Sec. 2.6 provides promising approximations. However, it would suffer from computational inefficiencies when more sub-spheres are needed to mimic the particle shape. To address this, more sophisticated models, such as polyhedral models or spline-based approximations, can be employed, which include their own specialized contact detection methods. For further details, refer to [ZYL+16] and [LCGR20].

- **Hierarchical grids for polydisperse particles:**
  The Linked Cells algorithm requires the cell size to be greater or equal to the diameters of all particles (see Sec. 3.1). Therefore, in simulation scenarios involving polydisperse particles with a wide size range, such as the fluidized bed from Sec. 5.5, the cell size may become too large relative to the diameters of smaller particles, leading to a lower hit rate and reduced contact detection efficiency. One potential solution is to introduce hierarchical grids with varying cell sizes. These multi-level grids could better accommodate particles of different sizes, improving the contact detection efficiency. Fur further details, refer to [CZ24].

7. Conclusion

To summarize, this thesis introduced Discrete Element Method (DEM) and extended the MD simulator, md-flexible, to simulate granular particles using AutoPas. This thesis first explored the theoretical foundations of DEM, including contact force and heat models. Next, These theoretical models were implemented in md-flexible and employed for DEM simulations of scenarios such as square tumbler and fluidized bed. The simulation results highlighted substantial benefits of AutoPas's auto-tuning capabilities in optimizing the simulation runtime.

In conclusion, DEM demonstrates significant potential as an application area for AutoPas, benefiting from relatively straightforward implementation of forces using a custom functor class and optimized algorithmic configurations during the simulation through auto-tuning. This potential makes the idea compelling to further extend the DEM simulator, implemented in this thesis, with the goal of enhancing simulation accuracy while continuing to leverage the benefits of auto-tuning.

A. Appendix

A.1. Contact Force Laws

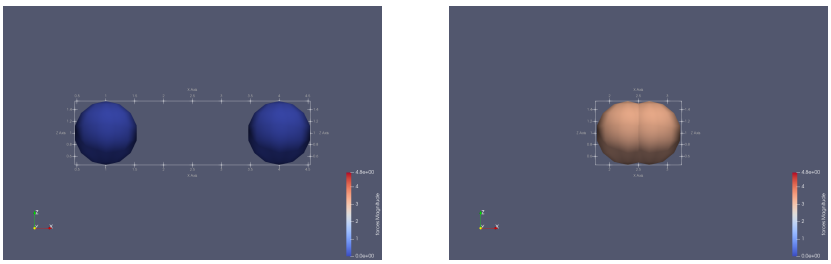
A.1.1. Linear normal contact model

This section contains experiments of the linear normal contact model from Sec. 2.4.1 with different values of γ_n. Furthermore, for better understanding of the experimental results, this chapter also derives the differential equation of the damped harmonic oscillator.

Experiments with varying γ_n

Figure A.1 and Figure A.2 show a simple simulation to verify the effect of the linear normal contact force. Figure A.2 (a) and (b) show the repulsive effect of the force when the overlap is positive. As a consequence, the particles i and j move apart from each other in the normal direction, which can be observed in all cases with varying γ_n. However, the variations in the normal viscosity coefficient γ_n show interesting effects:

- Figure A.2 (b) suggests, cases with higher γ_n experience lower maximum of overlap, i.e. $\max(\delta)$, implying higher repulsion and deceleration.
- In Figure A.2 (a), cases with greater γ_n show bigger differences between their velocities before (v_{before}) and after (v_{after}) the collision with decreasing ratio of $\frac{v_{after}}{v_{before}}$ for higher γ_n.
- In Figure A.2 (c), cases with higher γ_n show longer contact duration.
- In Figure A.2 (c), cases with high $\gamma_n \in \{0.5, 1, 5\}$ experience negative / adhesive force at the end of their particle contacts (ca. Iteration $\in$ (12500, 13000) for $\gamma_n = 1$).



(a) Initial State (b) In Contact

Figure A.1.: A simple simulation of linear normal contact force with a collision of two particles i (right), j (left), color scaled by $\|f_{ji}^n\|$. Simulation parameters of the initial state: $r_i = r_j = 0.5$, $v_{ji}^n = 6$. (a) Initial state of the particles. (b) State of the particles in contact.

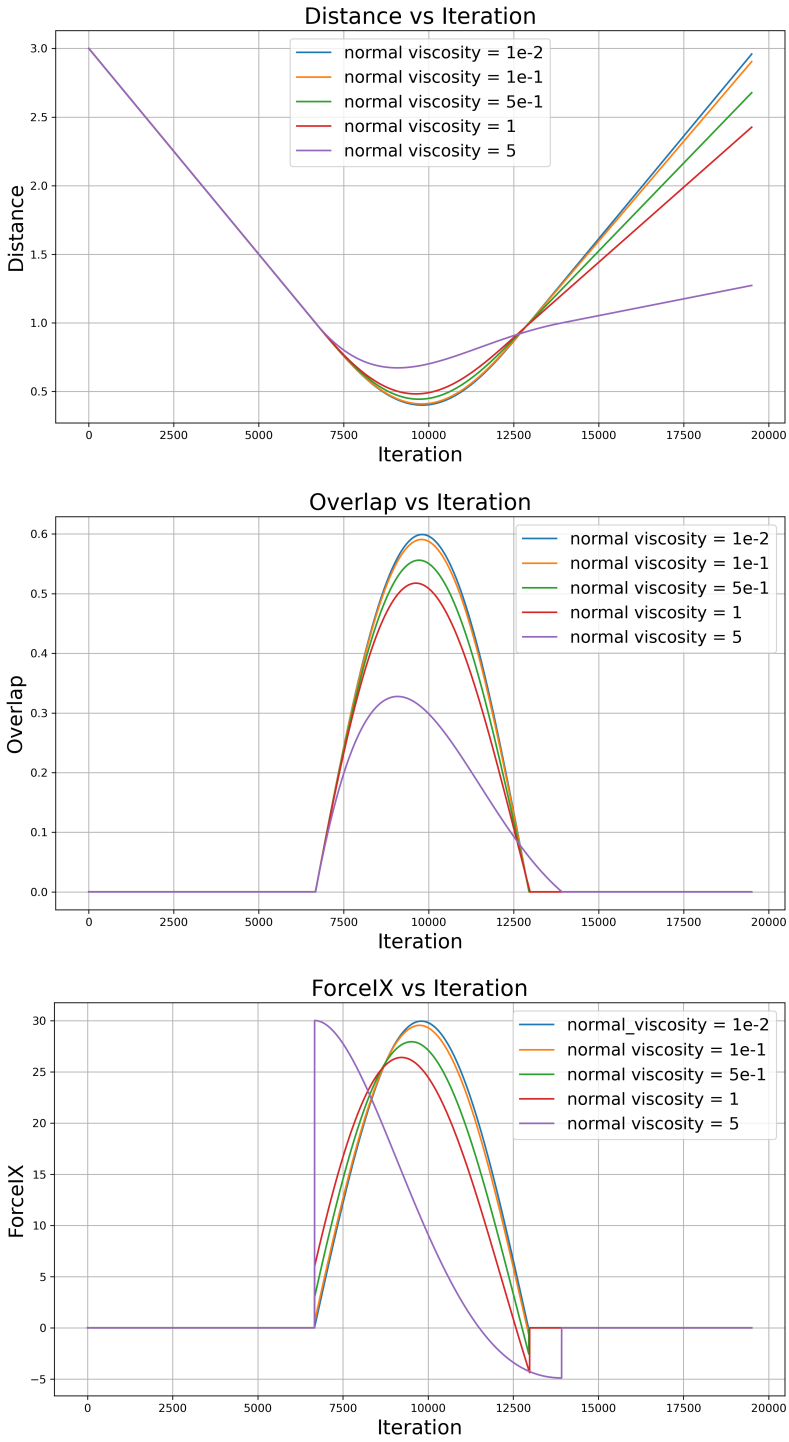
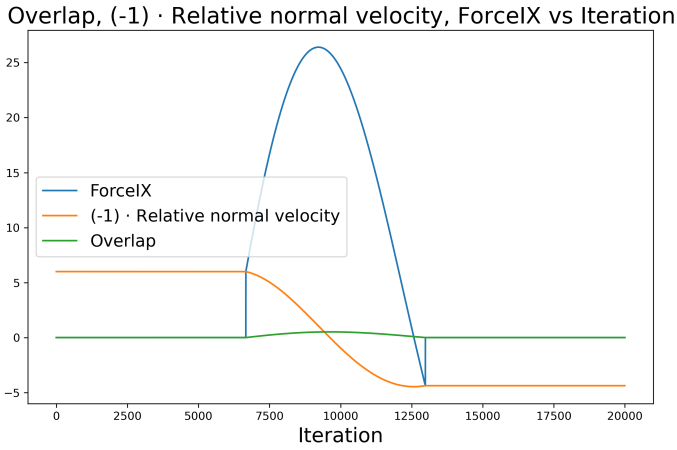
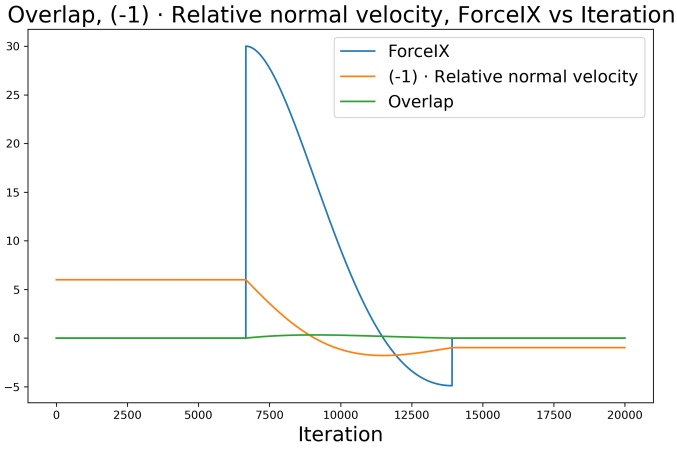


Figure A.2.: Development of distance ($\|x_i - x_j\|_2$), overlap (δ), and force (f_{ji}^n over time in scenario described in Figure A.1. Simulation parameters: $k^n = 50$, $\gamma_n \in \{0.01, 0.1, 0.5, 1, 5\}$, $\Delta t = 5 \cdot 10^{-5}$, number of total iterations = 20000 (a) Distance between particles i and j vs Iteration. (b) Overlap δ vs Iteration. (c) Linear normal force f_{ji}^n vs Iteration .



(a) $\gamma_n = 1$



(b) $\gamma_n = 5$

Figure A.3.: Development of overlap (δ), (-1) · relative normal velocity ($-v_{ji}^n$), force (f_{ji}^n) over time in scenario described in Figure A.1. Same parameter setting as in Figure A.2.

These effects arise as the dissipative force $f^n_{dissipative} = -\gamma_n \cdot v^n_{ji}$ becomes increasingly dominant due to higher γ_n, as Figure A.3 suggests. At the point of contact where v_i changes its sign due to the repulsive normal force, the sign of relative velocity v^n_{ij} changes too. As the term $-v^n_{ji}$ becomes negative, this can negatively affect the force and make it from repulsive to adhesive, if it is supported with high γ_n. This (negative) pulling force serves as an explanation for the higher deceleration effect for cases with higher γ_n. Plus, great $f^n_{dissipative} = -\gamma_n \cdot v^n_{ji}$ also explains the abrupt increase of force f^n_{ji} at the beginning of the contact of the particles ($\delta \approx 0$).

Damped harmonic oscillator

The explanations in A.1.1 are further supported by the differential equation of the damped harmonic oscillator with the oscillation frequency ω_0 and the effective viscosity η [Lud98] :

$$\ddot{\delta} + 2\eta\dot{\delta} + \omega_0^2\delta = 0 \tag{A.1}$$

with the solution:

$$\delta(t) = e^{-\eta t}\frac{v_0}{\omega} sin(\omega t) \tag{A.2}$$

In the following, this differential equation of the damped harmonic oscillator is derived and solved.

First, the overlap between spherical particles i and j is computed as:

$$\begin{aligned}
\delta_{ji} &= (r_i + r_j) - \|\mathbf{x_i} - \mathbf{x_j}\|_2 \\
&= (r_i + r_j) - \frac{(\|\mathbf{x_i} - \mathbf{x_j}\|_2)^2}{\|\mathbf{x_i} - \mathbf{x_j}\|_2} \\
&= (r_i + r_j) - \frac{1}{\|\mathbf{x_i} - \mathbf{x_j}\|_2} \cdot (\mathbf{x_i} - \mathbf{x_j}) \cdot (\mathbf{x_i} - \mathbf{x_j}) \\
&= (r_i + r_j) - (\mathbf{x_i} - \mathbf{x_j}) \cdot \mathbf{n_{ji}}.
\end{aligned} \tag{A.3}$$

Using Eq. A.3 to express $f^n_{dissipative}$ gives:

$$\begin{aligned}
f^n_{dissipative} &= -\gamma_n v^n_{ji} \\
&= -\gamma_n \cdot (\mathbf{v_i} - \mathbf{v_j}) \cdot \mathbf{n_{ji}} \\
&= -\gamma_n \cdot \frac{\mathbf{x_i} - \mathbf{x_j}}{\|\mathbf{x_i} - \mathbf{x_j}\|_2} \cdot \frac{d}{dt}(\mathbf{x_i} - \mathbf{x_j}) \\
&= \gamma_n \cdot \left(-\frac{d}{dt}\|\mathbf{x_i} - \mathbf{x_j}\|_2\right) \\
&= \gamma_n \cdot \frac{d\delta_{ji}}{dt} \\
&= \gamma_n \cdot \dot{\delta}.
\end{aligned} \tag{A.4}$$

Computing the second derivative of δ yields:

$$\begin{aligned}
\ddot{\delta} &= -\frac{d}{dt}\big((\mathbf{v_i} - \mathbf{v_j}) \cdot \mathbf{n_{ji}}\big) \\
&= -(\ddot{\mathbf{x}}_\mathbf{i} - \ddot{\mathbf{x}}_\mathbf{j}) \cdot \mathbf{n_{ji}}, \quad \text{using} \quad \mathbf{n_{ji}} = \text{const} \quad \text{during contact} \\
&= -\left(\frac{f_i^n}{m_i} - \frac{f_j^n}{m_j}\right), \quad \text{with} \quad f_i^n = m_i \cdot (\ddot{\mathbf{x}}_\mathbf{i} \cdot \mathbf{n_{ji}}) \\
&= -\left(\frac{f_i^n}{m_i} + \frac{f_i^n}{m_j}\right), \quad \text{with} \quad f_j^n = f_i^n \quad \text{(Newton's 3rd Law)} \\
&= -\frac{f_i^n}{m_{ij}}, \quad \text{with} \quad m_{ij} = \frac{m_i \cdot m_j}{(m_i + m_j)}.
\end{aligned} \tag{A.5}$$

Inserting Eqs. A.4 and 2.10 into A.5 results in:

$$\begin{aligned}
\ddot{\delta} + \frac{f_i^n}{m_{ij}} &= \ddot{\delta} + \frac{f_{elastic}^n + f_{dissipative}^n}{m_{ij}} \\
&= \ddot{\delta} + \frac{1}{m_{ij}} \cdot \left(k^n \cdot \delta + \gamma_n \cdot \dot{\delta}\right) \\
&= \ddot{\delta} + \frac{\gamma_n}{m_{ij}} \cdot \dot{\delta} + \frac{k^n}{m_{ij}} \cdot \delta \\
&= \ddot{\delta} + \frac{\gamma_n}{m_{ij}} \cdot \dot{\delta} + \frac{k^n}{m_{ij}} \cdot \delta \\
&= \ddot{\delta} + 2\eta\dot{\delta} + \omega_0^2\delta = 0, \quad \text{with} \quad \omega_0 = \sqrt{\frac{k^n}{m_{ij}}}, \quad \eta = \frac{\gamma_n}{2m_{ij}}.
\end{aligned} \tag{A.6}$$

Eq. A.6 describes the differential equation of damped harmonic oscillator with the oscillation frequency ω_0 and the effective viscosity η [Lud98]. Various methods can be applied here to solve this differential equation. One of it proceeds as following.

Reformulating Eq. A.6 yields:

$$\ddot{\delta} = -2\eta\dot{\delta} - \omega_0^2\delta. \tag{A.7}$$

where term of the multiple derivatives contains the original overlap function itself. This leads to the assumption of the structure of $\delta(t) = e^{\lambda \cdot t}$ and following consequences:

$$\dot{\delta} = \lambda \cdot e^{\lambda \cdot t}, \quad \ddot{\delta} = \lambda^2 \cdot e^{\lambda \cdot t} \tag{A.8}$$

Substituting Eq. A.8 into A.6 gives the characteristic equation:

$$\lambda^2 + 2\eta\lambda + \omega_0^2 = 0, \tag{A.9}$$

$$\begin{aligned}
\lambda_{1/2} &= -\eta \pm \sqrt{\eta^2 - \omega_0^2}, \\
&= -\eta \pm i\sqrt{\omega_0^2 - \eta^2}, \\
&= -\eta \pm i\omega, \quad \text{with} \quad \omega = \sqrt{\omega_0^2 - \eta^2},
\end{aligned} \tag{A.10}$$

which yields complex roots $\lambda_{1/2}$ as $\eta^2 < \omega_0^2$ usually holds (also for all scenarios from Figures A.1, A.2, and A.3). Applying the general solution gives:

$$\begin{aligned}\delta(t) &= C_1 \cdot e^{\lambda_1 t} + C_2 \cdot e^{\lambda_2 t} \\ &= e^{-\eta t} \cdot \left[C_1 \cdot e^{i\omega t} + C_2 \cdot e^{-\omega t}\right] \\ &= e^{-\eta t} \cdot [(C_1 + C_2)cos(\omega t) + i(C_1 - C_2)sin(\omega t)] \\ &= e^{-\eta t} \cdot i(C_1 - C_2)sin(\omega t) \\ &= e^{-\eta t} \cdot C \cdot sin(\omega t), \quad \text{with} \quad C = i(C_1 - C_2)\end{aligned} \tag{A.11}$$

using the equalities $e^{i\omega t} = cos(\omega t) + isin(\omega t)$, $e^{-i\omega t} = cos(\omega t) - isin(\omega t)$ and assuming zero overlap at the beginning of contact ($\delta(0) = (C_1 + C_2) = 0$).

The derivative of $\delta(t)$ is:

$$\begin{aligned}\dot{\delta}(t) &= \frac{d}{dt}(d^{-\eta t}Csin(\omega t)) \\ &= -\eta e^{-\eta t}Csin(\omega t) + e^{-\eta t}C\omega cos(\omega t) \\ &= C \cdot e^{-\eta t} \cdot (-\eta sin(\omega t) + \omega cos(\omega t))\end{aligned} \tag{A.12}$$

Assuming $\dot{\delta}(0) = v_0$ yields:

$$\begin{aligned}\dot{\delta}(0) = C \cdot \omega &= v_0, \\ C &= \frac{v_0}{\omega}, \\ \delta(t) &= e^{-\eta t}\frac{v_0}{\omega}sin(\omega t).\end{aligned} \tag{A.13}$$

which matches the result shown in [Lud98]. This solution in Eq. A.13 represents a sinusoidal oscillation with its amplitude decaying exponentially due to the damping term $e^{-\eta t}$, which is plotted in Figure A.4. This solution of the differential equation allows us to compute further interesting values for the simulation, which are presented in the following.

Contact duration

The contact of colliding particles only lasts for the first half period of oscillation in Figure A.4, as the contact ends as soon as $\delta(t) < 0$. The contact duration of particles can be approximated with the help of the solution of the differential equation:

$$t_{contact} = \frac{\pi}{\omega}. \tag{A.14}$$

This emphasizes the importance of selecting a sufficiently small $\Delta t \ll t_{\text{contact}}$ to perform well-approximated integration steps. With an appropriate Δt, Figure A.2 (b) matches the function graph of Figure A.4 well up to the truncation before and after the half period of oscillation.

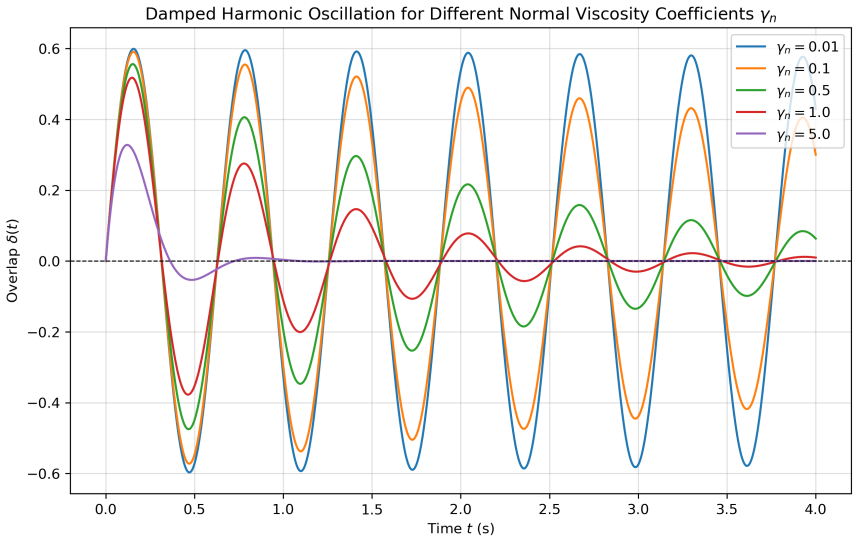


Figure A.4.: Damped Harmonic Oscillation of Eq. A.1 with parameter settings from Figure A.1, A.2, $\gamma_n \in \{0.01, 0.1, 0.5, 1.0, 5.0\}$

Restitution coefficient

Moreover, the expected change in relative normal velocity before and after the collision of two particles (i and j) can be explored by looking into the so-called restitution coefficient e_n [Lud98]:

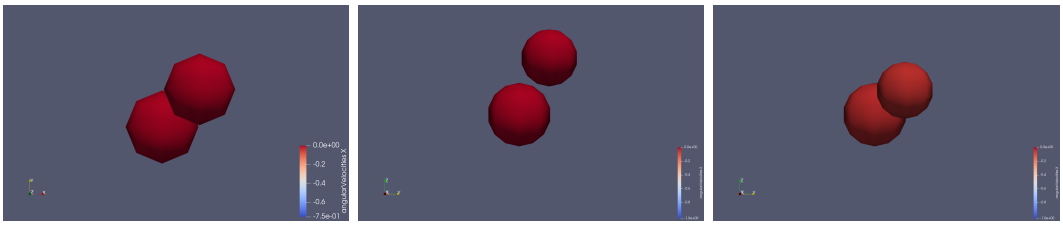
$$\begin{aligned} e_n &= -\frac{(\mathbf{v_i^{after}} - \mathbf{v_j^{after}}) \cdot \mathbf{n}}{(\mathbf{v_i^{before}} - \mathbf{v_j^{before}}) \cdot \mathbf{n}} \\ &= -\frac{\mathbf{v_i^{after}}\mathbf{n}}{\mathbf{v_i^{before}} \cdot \mathbf{n}}, \quad \text{assuming} \quad \mathbf{v_i} = -\mathbf{v_j} \\ &= -\frac{\dot{\delta}(\pi/\omega)}{\dot{\delta}(0)} \\ &= -\frac{-v_0 \cdot e^{-(\pi\eta)/\omega}}{v_0} \\ &= e^{-(\pi\eta)/\omega} \end{aligned} \tag{A.15}$$

The restitution coefficient $e_n^{empirical}$ computed by empirical results from Figure A.3 match well with this derived theoretical $e_n^{theoretical}$:

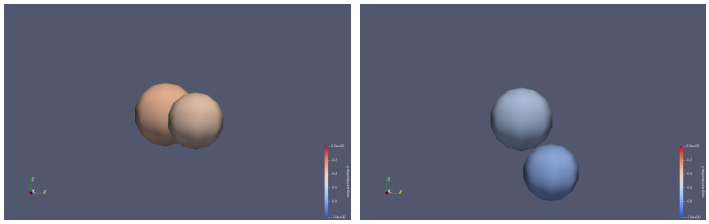
$$\text{Empirical result for } \gamma_n = 1, v_{ji}^{before} = -6: \quad v_{ji}^{after} = 4.37521.$$

$$\Rightarrow e_n^{empirical} = -\frac{4.37521}{-6} \approx 0.72920$$

$$\text{Theoretical result for } \gamma_n = 1:$$



(a) Before Contact from above, Iteration 125 (b) Before Contact from side, Iteration 125 (c) In Contact from side, Iteration 625



(d) In Contact from side, Iteration 1225 (e) In Contact from side, Iteration 2000

Figure A.5.: Simple simulation of (sliding) tangential frictional torque $\mathbf{q^t}$ with only the frictional force f^t activated. The smaller particle i falls down with negative initial velocity in z-direction, the bigger particle j, placed diagonally below the particle i at the beginning, causes frictional torque with the other particle. Simulation parameters are set in the following way: $(x_i^0, y_i^0, z_i^0) = (2.2, 2.2, 3.5), (x_j^0, y_j^0, z_j^0) = (1, 1, 1), k^n = 5, \gamma_n = 5 \cdot 10^{-5}, \mu_d = 1, r_i = 1.0, r_j = 1.2, \omega_\mathbf{i}^\mathbf{0} = \omega_\mathbf{j}^\mathbf{0} = \mathbf{0}, \Delta t = 5 \cdot 10^{-4}$. Color is scaled by ω_z.

$$\Rightarrow e_n^{\text{theoretical}} = e^{(\pi\eta)/\omega} = e^{-(\pi\cdot 1)/\sqrt{99}} \approx 0.72925$$

$$\text{Empirical result for } \gamma_n = 5, v_{ji}^{before} = -6: \quad v_{ji}^{after} = 0.97760.$$

$$\Rightarrow e_n^{empirical} = -\frac{0.97760}{-6} \approx 0.16293$$

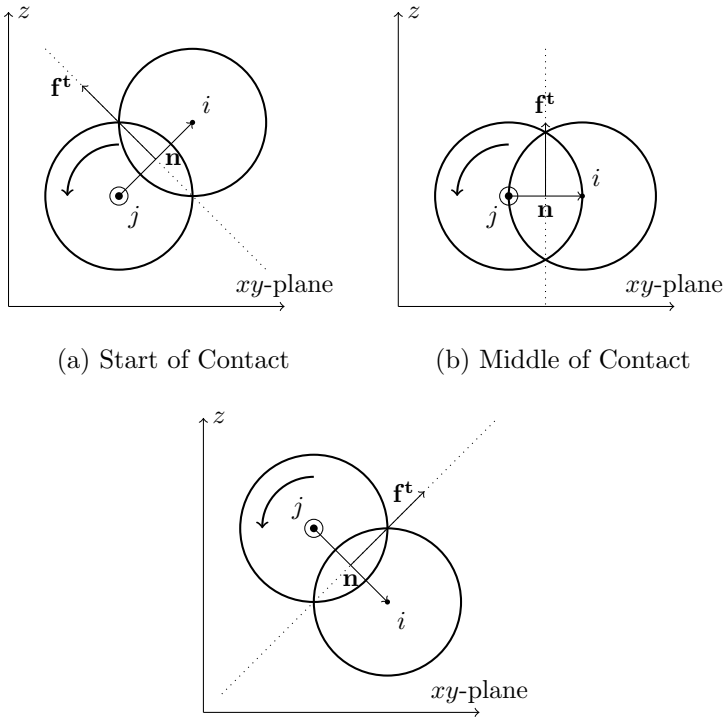
$$\text{Theoretical result for } \gamma_n = 5:$$

$$\Rightarrow e_n^{\text{theoretical}} = e^{(\pi\eta)/\omega} = e^{-(\pi\cdot 5)/\sqrt{75}} \approx 0.16303$$

Furthermore, the maximum overlap δ_{max} at t_{max} can additionally be computed by setting:

$$\dot{\delta}(t_{max}) = 0$$

and searching for smallest $t_{max} > 0$ satisfying the condition.



(a) Start of Contact

(b) Middle of Contact

(c) End of Contact

Figure A.6.: Diagrams of snapshots in the simulation of Figure A.5 with only approximate radius and vector scaling due to simplicity reasons. During the whole contact between the particles i and j, the frictional torque on particle j, i.e. $\mathbf{q_j^t}$ is directed outward from the figure. The resulting direction of angular velocity ω_j is indicated with curved arrows. As the frictional torques on particle i is parallel to the one on particle j, i.e. $\mathbf{q_i^t} \parallel \mathbf{q_j^t}$ (see Eq. 2.17), they are both directed outward from the figure and are parallel to the xy-plane.

A.2. Tangential torque

This section provides further analysis of the simulation results in Figure 2.6 and verifies the emprical values gained from the simulation (see Figure A.5). Following observations can be made first:

- Magnitudes of tangential forces and torques are proportional to overlap δ due to $f^t \propto f^n \propto \delta$
- Due to the third law of Newton, the relation $\mathbf{f_i^t} = -\mathbf{f_j^t}$ is visible on the plots of frictional forces.
- The plots of frictional torques on particle i and j confirm the parallel and radius-dependent torques (see Eq. 2.17).

Furthermore, the radius-dependency of torques and their effect on angular velocities can be verified for empirical values from the simulation. First, the corrected radii at the highest overlap, i.e. $\delta_{max} \approx 0.52$, can be approximated as following:

$$\alpha_i = r_i - \frac{\delta_{max}}{2} \approx 0.74$$
$$\alpha_j = r_j - \frac{\delta_{max}}{2} \approx 0.94$$

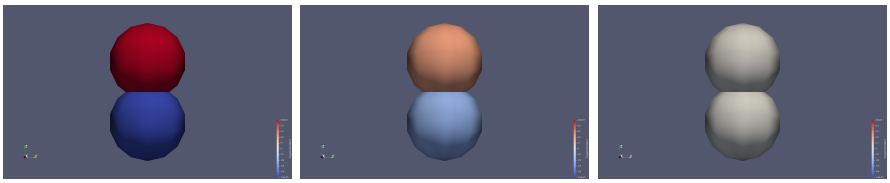
The scalar computed with these approximated corrected radii match the empirical torque values at the highest overlap:

$$\begin{aligned}\mathbf{q^t_{j,empirical,max}} &= (-0.87, 0.87, 0)^T \\ &\approx (-0.864, 0.864, 0)^T \\ &= \left(\frac{0.94}{0.74}\right) \cdot (-0.68, 0.68, 0)^T \\ &= \left(\frac{\alpha_j}{\alpha_i}\right) \cdot \mathbf{q^t_{i,empirical,max}}\end{aligned}$$

Moreover, the frictional torque in z-direction remains zero during the whole contact, i.e. $q_z^t = 0$, as the produced torque $\mathbf{q^t}$ is parallel to the xy-plane (see Figure A.6). The plots of angular velocities also match the plots of frictional torques due to their following relation:

$$\mathbf{q^t} = I \cdot \frac{d\omega}{dt} \tag{A.16}$$

with I as the moment of inertia, i.e. the torques are proportional to the time derivative of angular velocities. Due to unequal torque values, the two particles i and j have different angular velocities after the collision, i.e. $\omega_{\mathbf{after}}$, which can be approximated for verification:



(a) Initial State (b) Iteration 120 (c) Iteration 250

Figure A.7.: A simple simulation of (dynamic) rolling torque $\mathbf{q^r}$ with other forces and torques deactivated. The two particles i (upper) and j (lower) have anti-parallel initial angular velocities and zero translational velocities. The activated rolling torque $\mathbf{q^r}$ acts against their rolling motion and reduces their angular velocities to zero. Simulation parameters are set in the following way: $(x_i, y_i, z_i) = (1, 1, 2.75), (x_j, y_j, z_j) = (1, 1, 1), k^n = 5, \gamma_n = 5 \cdot 10^{-5}, \mu_r = 5, r_i = r_j = 1, \omega_\mathbf{i}^\mathbf{0} = (0.5, 0.5, 0.5), \omega_\mathbf{j}^\mathbf{0} = (-0.5, -0.5, -0.5), \Delta t = 5 \cdot 10^{-4}$. Color is scaled by ω_x

$$
\begin{aligned}
\omega_\mathbf{j}^\text{after} &= \omega_\mathbf{j}^\text{before} + \frac{1}{I_j} \cdot \int_C \mathbf{q_j^t}\, dt \quad \text{using Eq. 2.17 and } I_\text{sphere} \qquad \text{(A.17)} \\
&= \frac{1}{0.4 \cdot m_j \cdot r_j^2} \cdot \frac{\alpha_j}{\alpha_i} \cdot \int_C \mathbf{q_i^t}\, dt \\
\Rightarrow \frac{\omega_\mathbf{j}^\text{after}}{\omega_\mathbf{i}^\text{after}} &= \frac{r_i^2}{r_j^2} \cdot \frac{\alpha_j}{\alpha_i} \\
&\approx \frac{r_i^2}{r_j^2} \cdot \frac{r_j - \frac{0.5 \cdot \delta_\text{max}}{2}}{r_i - \frac{0.5 \cdot \delta_\text{max}}{2}} \approx 0.854 \\
\omega_{\mathbf{j},\text{empirical}}^\text{after} &= (-0.655, 0.655, 0)^T \\
&\approx \frac{\omega_\mathbf{j}^\text{after}}{\omega_\mathbf{i}^\text{after}} \cdot \omega_{\mathbf{i},\text{empirical}}^\text{after} \\
&= 0.854 \cdot (-0.753, 0.753, 0)^T \\
&= (-0.643, 0.643, 0)^T
\end{aligned}
$$

For simplicity reasons, this approximation assumes constant corrected radii $\alpha_{i/j}$ that in reality depends on the overlap δ and changes during the contact.

A.3. Rolling torque

This section shows results of a simple simulation of rolling torque (only with dynamic rolling frictional "force") to verify the expected behavior of the torque. Following observations and realizations can be made from results in Figure A.7 with its according plots in Figure A.8.

- The snapshots in Figure A.7 suggest the effect of the rolling torque, which opposes the rolling motion in the tangential plane of the two contacting particles, gradually reducing the magnitude of the rolling velocity $\|\mathbf{v_r}\|_2$ to zero. In this case, the rolling

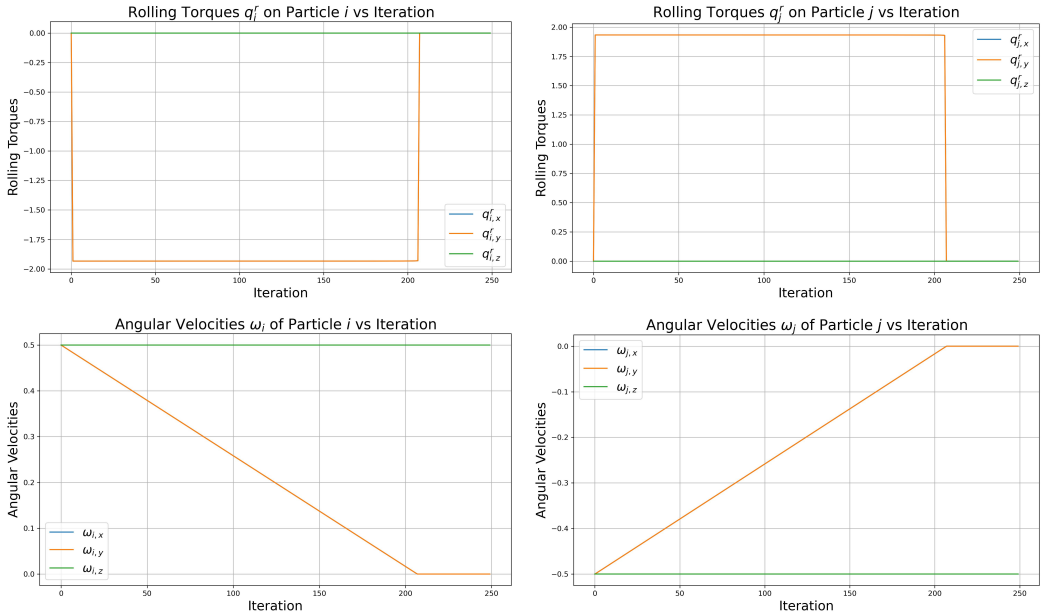


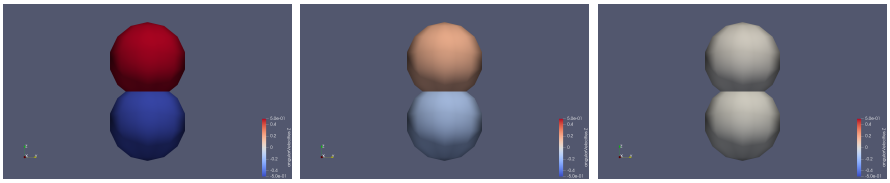
Figure A.8.: Development of rolling torque $\mathbf{q^r}$ and angular velocities ω in the simulation of Figure A.7. The lines for q_x^r and q_y^r, as well as those for ω_x and ω_y overlap due to symmetry.

torque reduces the angular velocities along the xy-plane, i.e. ω_x and ω_y, eventually to zeros.

- The constant rolling torques $\mathbf{q^r_{i/j}}$ can be explained with its constant factors (see Eq. 2.22), i.e. the reduced radius α_{ij}, the normal unit vector $\mathbf{n}$, and the (dynamic) rolling "force" $\mathbf{f^r}$. The reduced radius α_{ij} and the normal unit vector $\mathbf{n}$ remain constant due to the constant overlap δ and the fixed positions of the particles i and j. The rolling "force" $\mathbf{f^r}$ stays constant due to its constant terms (see Eq. 2.21), i.e. μ_r, f^n, and $\mathbf{r}$. Moreover, in this scenario, the rolling unit vector $\mathbf{r}$ stays constant as the angular velocities in the affected directions are decelerated by the same amount. This can be observed in plots of angular velocities in Figure A.8, in which the lines for ω_x and ω_y overlap.

A.4. Torsion torque

This section shows a simple simulation for torsion resistance to verify its correctness in the implementation and explore it further (see Figure A.9, A.10). Similar to the simulation in Sec. 2.4.2, only the dynamic case is activated here. The plots show similar developments of torques and angular velocities as for rolling torques in Figure A.8 due to the structural similarities of the formulas. The resulting torques are in both cases dependent only on constant factors until the torques are reduced to zero. Moreover, the relation between the projected final angular velocities and the projected average of initial velocities agrees with these simulation results:



(a) Initial State (b) Iteration 100 (c) Iteration 200

Figure A.9.: A simple simulation of (dynamic) torsion torque $\mathbf{q^o}$ with other forces and torques deactivated. The two particles have different initial angular velocities, especially along their normal direction, i.e. z-axis, and zero translational velocities. The activated torsion torque $\mathbf{q^o}$ reduces the difference between their angular velocities, i.e. $\Delta\omega$, such that $\omega_{\mathbf{i/j}} \Rightarrow \mathbf{0}$. Simulation parameters are set as in Figure A.7 with the only following change: $\mu_o = 5$. Color is scaled by ω_z

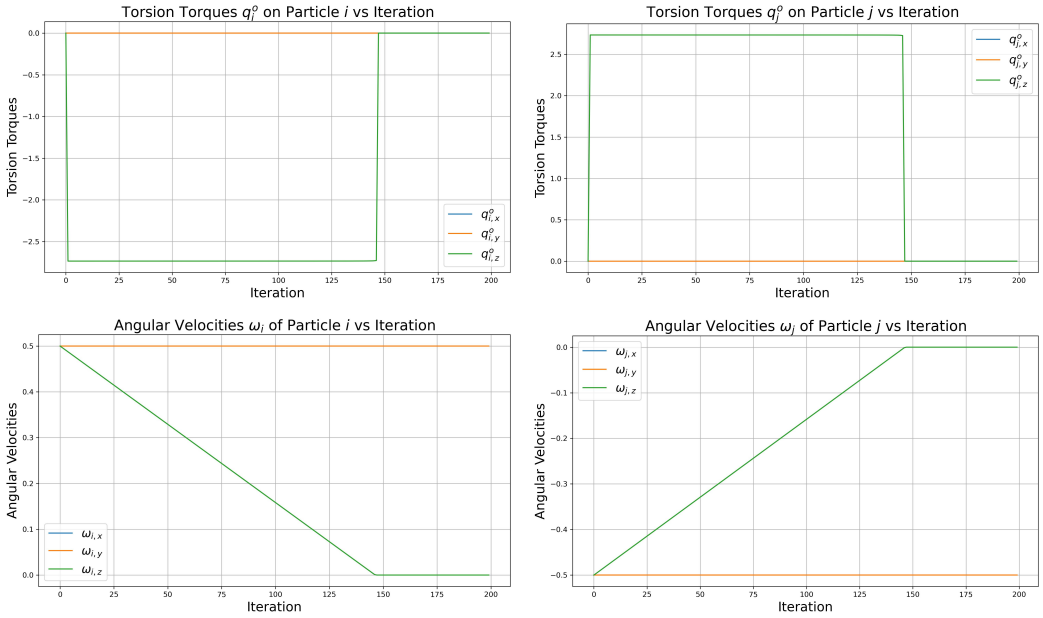
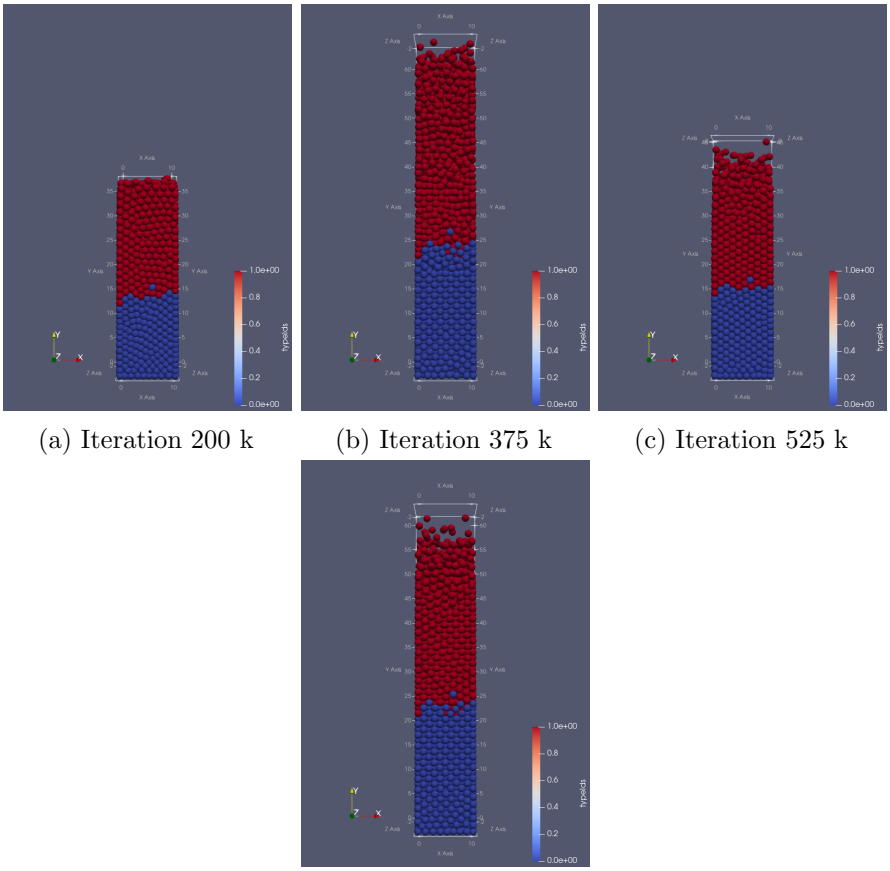


Figure A.10.: Development of (dynamic) torsion torque $\mathbf{q^o}$ and angular velocities ω in the simulation of Figure A.9.

$$
\begin{aligned}
\mathbf{n} = \hat{\mathbf{z}} &= (0,0,1)^T \\
\omega_{\mathbf{i}}^{\mathbf{after}} &= (0.5,0.5,0)^T = -\omega_{\mathbf{j}}^{\mathbf{after}} \\
\omega_{\mathbf{i}}^{\mathbf{before}} &= (0.5,0.5,0.5)^T = -\omega_{\mathbf{j}}^{\mathbf{before}} \\
\mathbf{n}\cdot\omega_{\mathbf{i}}^{\mathbf{after}} = \mathbf{n}\cdot\omega_{\mathbf{j}}^{\mathbf{after}} &= \mathbf{n}\cdot\frac{\omega_{\mathbf{i}}^{\mathbf{before}} + \omega_{\mathbf{j}}^{\mathbf{before}}}{2} = 0
\end{aligned}
$$

A.5. Simulation Results

A.5.1. Thermal Equilibrium of Stacked Particles



(a) Iteration 200 k (b) Iteration 375 k (c) Iteration 525 k

(d) Iteration 675 k

Figure A.11.: Snapshots of Thermal Equilibrium of Stacked Particles Scenario at iterations where the kinetic energy, shown in Figure 5.4, approximately reaches local minima. Color is scaled by `typeId`.

List of Figures

Bibliography

[BAC+14] Josephine Boac, Kingsly Ambrose, Mark Casada, Ronaldo Maghirang, and Dirk Maier. Applications of discrete element method in modeling of grain postharvest operations. *Food Engineering Reviews*, 6, 08 2014.

[BK04] Katalin Bagi and Matthew Kuhn. A definition of particle rolling in a granular assembly in terms of particle translations and rotations. *Journal of Applied Mechanics-transactions of The Asme - J APPL MECH*, 71, 07 2004.

[CKK14] Ray Cocco, Sb Karri, and Ted Knowlton. Introduction to fluidization. *Chemical Engineering Progress*, 110:21–29, 11 2014.

[Cle99] Paul Cleary. The effect of particle shape on hopper discharge. *Second International Conference on CFD in the Minerals and Process Industries*, 01 1999.

[CMT06] Bodhisattwa Chaudhuri, Fernando J. Muzzio, and M. Silvina Tomassone. Modeling of heat transfer in granular flow in rotating vessels. *Chemical Engineering Science*, 61(19):6348–6360, 2006.

[CZ24] Jun Chen and Jingxin Zhang. A hierarchical linked cell method based on scale-proportional multi-level dem grids. *Powder Technology*, 448:120296, 2024.

[EKB+21] Safae Elmisaoui, Lhachmi Khamar, Saad Benjelloun, Mohamed Khamar, and Jean-Michel Ghidaglia. Numerical study of fertilizer granules dynamics within rotary drum granulator. In Metin Türkay and Rafiqul Gani, editors, *31st European Symposium on Computer Aided Process Engineering*, volume 50 of *Computer Aided Chemical Engineering*, pages 327–332. Elsevier, 2021.

[FAKR99] J. F. Favier, M. H. Abbaspour-Fard, M. Kremmer, and A. O. Raji. Shape representation of axi-symmetrical, non-spherical particles in discrete element simulation using multi-element model particles. *Engineering Computations*, 16(4):467–480, 2025/02/05 1999.

[FLS21] Marc Fransen, Matthijs Langelaar, and Dingena Schott. Application of dem-based metamodels in bulk handling equipment design: Methodology and dem case study. *Powder Technology*, 393, 07 2021.

[GDK21] Yijie Gao, Giovanni De Simone, and Maya Koorapaty. Calibration and verification of dem parameters for the quantitative simulation of pharmaceutical powder compression process. *Powder Technology*, 378:160–171, 2021.

[GSBN21] Fabio Gratl, Steffen Seckler, Hans-Joachim Bungartz, and Philipp Neumann. N ways to simulate short-range particle systems: Automated algorithm selection with the node-level library autopas. *Computer Physics Communications*, 273:108262, 12 2021.

[GSGZ+20] Shahab Golshan, Rahmat Sotudeh-Gharebagh, Reza Zarghami, Navid Mostoufi, Bruno Blais, and J.A.M. Kuipers. Review and implementation of cfd-dem applied to chemical process systems. *Chemical Engineering Science*, 221:115646, 2020.

[GSL+16] Ray Grout, Hariswaran Sitaraman, Peiyuan Liu, Timothy Brown, William Fullmer, Thomas Hauser, and Christine Hrenya. Comprehensive benchmark suite for simulation of particle laden flows using the discrete element method with performance profiles from the multiphase flow with interface exchanges (mfix) code, 2016.

[GST+19a] Fabio Alexander Gratl, Steffen Seckler, Nikola Tchipev, Hans-Joachim Bungartz, and Philipp Neumann. Autopas: Auto-tuning for particle simulations. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 748–757, 2019.

[GST+19b] Fabio Alexander Gratl, Steffen Seckler, Nikola Tchipev, Hans-Joachim Bungartz, and Philipp Neumann. Autopas: Auto-tuning for particle simulations. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 748–757, 2019.

[HRK+24] Jiawei Hu, Rafael L. Rangel, Francisco Kisuka, Ling Zhang, Shaowu Yin, and Chuan-Yu Wu. Dem analysis of heat generation and transfer during granular flow in a rotating drum. *Chemical Engineering Journal*, 499:155945, 2024.

[KGK+12] Christoph Kloss, Christoph Goniva, Alice König, Stefan Amberger, and Stefan Pirker. Models, algorithms and validation for opensource dem and cfd-dem. *Progress in Computational Fluid Dynamics*, 12:140 – 152, 06 2012.

[LCGR20] Shiwen Liu, Feiguo Chen, Wei Ge, and Philippe Ricoux. Nurbs-based dem for non-spherical particles. *Particuology*, 49:65–76, 2020.

[Lud98] S. Luding. *Collisions & Contacts between Two Particles*, pages 285–304. Springer Netherlands, Dordrecht, 1998.

[Lud08a] Stefan Luding. Cohesive, frictional powders: contact models for tension. *Granular Matter*, 10(4):235–246, 2008.

[Lud08b] Stefan Luding. Introduction to discrete element methods: Basics of contact force models and how to perform the micro-macro transition to continuum theory. In *Discrete Element Methods*, volume 10, pages 1–43. TS, CTW, UTwente, Enschede, Netherlands, 2008. Available online: `s.luding@utwente.nl`.

[NGM+24] Samuel James Newcome, Fabio Alexander Gratl, Markus Muehlhaeusser, Philipp Neumann, and Hans-Joachim Bungartz. Autopas: Dynamic algorithm selection

in molecular dynamics for optimal time and energy. In *SIAM Conference on Parallel Processing (PP24)*. SIAM, Mar 2024.

[NGNB23] Samuel James Newcome, Fabio Alexander Gratl, Philipp Neumann, and Hans-Joachim Bungartz. Towards auto-tuning multi-site molecular dynamics simulations with autopas. *Journal of Computational and Applied Mathematics*, 433:115278, Dec 2023.

[RK10] Dmitri Rozmanov and Peter G. Kusalik. Robust rotational-velocity-verlet integration methods. *Phys. Rev. E*, 81:056706, May 2010.

[Roy01] David Roylance. Engineering viscoelasticity. 2139, 01 2001.

[WCNT23] Kimiaki Washino, Ei L. Chan, Yukiko Nishida, and Takuya Tsuji. Coarse grained dem simulation of non-spherical and poly-dispersed particles using scaled-up particle (sup) model. *Powder Technology*, 426:118676, 2023.

[ZYL+16] Wenqi Zhong, Aibing Yu, Xuejiao Liu, Zhenbo Tong, and Hao Zhang. Dem/cfd-dem modelling of non-spherical particulate systems: Theoretical developments and applications. *Powder Technology*, 302:108–152, 2016.