Comp. Part. Mech. (2015) 2:173–196
DOI 10.1007/s40571-015-0047-6

Ultrascale simulations of non-smooth granular dynamics

Tobias Preclik[1] · **Ulrich Rüde**[1]

Received: 31 December 2014 / Revised: 16 May 2015 / Accepted: 18 May 2015 / Published online: 30 May 2015

Abstract This article presents new algorithms for massively parallel granular dynamics simulations on distributed memory architectures using a domain partitioning approach. Collisions are modelled with hard contacts in order to hide their micro-dynamics and thus to extend the time and length scales that can be simulated. The global multi-contact problem is solved using a non-linear block Gauss-Seidel method that is conforming to the subdomain structure. The parallel algorithms employ a sophisticated protocol between processors that delegate algorithmic tasks such as contact treatment and position integration uniquely and robustly to the processors. Communication overhead is minimized through aggressive message aggregation, leading to excellent strong and weak scaling. The robustness and scalability is assessed on three clusters including two peta-scale supercomputers with up to 458,752 processor cores. The simulations can reach unprecedented resolution of up to ten billion (10^{10}) non-spherical particles and contacts.

Keywords Granular dynamics · High performance computing · Non-smooth contact · Parallel computing · Message passing interface

Mathematics Subject Classification 65Y05 · 70F35 · 70F40 · 70E55

✉ Tobias Preclik
tobias.preclik@fau.de

Ulrich Rüde
ulrich.ruede@fau.de

[1] Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91052 Erlangen, Germany

1 Introduction

Granular matter exhibits intriguing behaviours akin to solids, liquids or gases. However, in contrast to those fundamental states of matter, granular matter still cannot be described by a unified model equation homogenizing the dynamics of the individual particles [28]. To date, the rich set of phenomena observed in granular matter, can only be reproduced with simulations that resolve every individual particle. In this paper, we will consider methods where also the spatial extent and geometric shape of the particles can be modelled. Thus in addition to position and translational velocity the orientation and angular velocity of each particle constitute the state variables of the dynamical system. The shapes of the particles can be described for example by geometric primitives, such as spheres or cylinders, with a low-dimensional parameterization. Composite objects can be introduced as a set of primitives that are rigidly glued together. Eventually, even meshes with a higher-dimensional parameterization can be used. In this article the shape of the particles does not change in time, i.e. no agglomeration, fracture or deformation takes place. The rates of change of the state variables are described by the Newton-Euler equations, and the particle interactions are determined by contact models.

Two fundamentally different model types must be distinguished: Soft and hard contacts. Soft contacts allow a local compliance in the contact region, whereas hard contacts forbid penetrations. In the former class the contact forces can be discontinuous in time, leading to non-differentiable but continuous velocities after integration. The differential system can be cast e.g. as an ordinary differential equation with a discontinuous right-hand side or as differential inclusions. However, the resulting differential system is typically extremely stiff if realistic material parameters are employed.

In the latter class, discontinuous forces are not sufficient to accomplish non-penetration of the particles. Instead, impulses are necessary to instantaneously change velocities on collisions or in self-locking configurations if Coulomb friction is present [35]. Stronger mathematical concepts are required to describe the dynamics. For that purpose, Moreau introduced the notion of measure differential inclusions in [29].

Hard contacts are an idealization of reality. The rigidity of contacts has the advantage that the dynamics of the micro-collisions does not have to be resolved in time. However, this also introduces ambiguities: The rigidity has the effect that the force chains along which a particle is supported are no longer unique [31]. If energy is dissipated, this also effects the dynamics. To integrate measure differential inclusions numerically in time, two options exist: In the first approach the integration is performed in subintervals from one impulsive event to the next [11,27]. At each event an instantaneous impact problem must be solved whose solution serves as initial condition of the subsequent integration subinterval. Impact problems can range from simple binary collisions to self-locking configurations and to complicated instantaneous frictional multi-contact problems with simultaneous impacts. The dynamics between events are described by differential inclusions, differential algebraic equations or ordinary differential equations. Predicting the times of the upcoming events correctly is non-trivial in general and handling them in order in parallel is impeding the scalability [27]. In the second approach no efforts are made to detect events, but the contact conditions are only required to be satisfied at discrete points in time. This approach is commonly referred to as a time-stepping method.

This article focuses on the treatment of hard contacts in order to avoid the temporal resolution of micro-collisions and thus the dependence of the time-step length on the stiffness of the contacts. In order to avoid the resolution of events a time-stepping method is employed. This considerably extends the time scales that are accessible to simulations of granular systems with stiff contacts.

To estimate the order of a typical real-life problem size of a granular system, consider an excavator bucket with a capacity of $1\,\mathrm{m}^3$. Assuming sand grains with a diameter of 0.15 mm, and assuming that they are packed with a solid volume fraction of 0.6, the excavator bucket contains in the order of 10^{10} particles. In such a dense packing the number of contacts is in the same order as the number of particles. Only large scale parallel systems with distributed memory can provide enough memory to store the data and provide sufficient computational power to integrate such systems for a relevant simulation time. Consequently a massive parallelization of the numerical method for architectures with distributed memory is absolutely essential.

In the last half decade several approaches were published suggesting parallelizations of the methods integrating the equations of motion of rigid particles in hard contact [17,18,22,30,36,40,41]. The approach put forward in this article builds conceptually on these previous approaches but exceeds them substantially by consistently parallelizing all parts of the code, consistently distributing all simulation data (including the description of the domain partitioning), systematically minimizing the volume of communication and the number of exchanged messages, and relying exclusively on efficient nearest-neighbor communication. The approach described here additionally spares the expensive assembly of system matrices by employing matrix-free computations. All this is accomplished without sacrificing accuracy. The matrix-free implementation allows the direct and straight forward evaluation of wrenches in parallel and thus reduces the amount of communicated data. Furthermore, an exceptionally robust synchronization protocol is defined, which is not susceptible to numerical errors. The excellent parallel scaling behaviour is then demonstrated for dilute and dense test problems in strong- and weak-scaling experiments on three clusters with fundamentally different interconnect networks. Among the test machines are the peta-scale supercomputers SuperMUC and Juqueen, as they will be described in Sect. 7.3. The results show that given a sufficient computational intensity of the granular setup and an adequate processor interconnect, a few hundred particles per process are already enough to obtain satisfactory scaling even on millions of processes.

In Sect. 2 of this paper the underlying differential equations and the time-continuous formulation of the hard contact models are formulated. Sect. 3 proposes a discretization scheme and discrete constraints for the hard contact model. The problem of reducing the number of contacts in the system for efficiency reasons is addressed in Sect. 4. Subsequently, an improved numerical method for solving multi-contact problems in parallel is introduced in Sect. 5 before turning to the design of the parallelization in Sect. 6. The scalability of the parallelization is then demonstrated in Sect. 7 by means of dilute and dense setups on three different clusters. Finally, the algorithms and results are compared to previous work by other authors in Sect. 8 before summarizing in Sect. 9.

2 Continuous dynamical system

The Newton-Euler equations for a system with ν_b particles are [24]

$$\begin{pmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\boldsymbol{\varphi}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}(t) \\ \mathbf{Q}(\boldsymbol{\varphi}(t))\boldsymbol{\omega}(t) \end{pmatrix},$$

$$\mathbf{M}(\boldsymbol{\varphi}(t)) \begin{pmatrix} \dot{\boldsymbol{v}}(t) \\ \dot{\boldsymbol{\omega}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{s}(t), t) \\ \boldsymbol{\tau}(\boldsymbol{s}(t), t) - \boldsymbol{\omega}(t) \times \mathbf{I}(\boldsymbol{\varphi}(t))\boldsymbol{\omega}(t) \end{pmatrix},$$

where the positions $\boldsymbol{x}(t) \in \mathbb{R}^{3\nu_b}$, the rotations $\boldsymbol{\varphi}(t) \in \mathbb{R}^{4\nu_b}$, translational velocities $\boldsymbol{v}(t) \in \mathbb{R}^{3\nu_b}$, and angular velocities $\boldsymbol{\omega}(t) \in \mathbb{R}^{3\nu_b}$ are the state variables at time t.

Different parameterizations exist for the rotations, but quaternions having four real components are the parameterization of choice here. Independent of the parameterization, the derivatives of the rotation components can be expressed in terms of a matrix-vector product between a block-diagonal matrix and the angular velocities [10]. If the rotation of particle i is described by the quaternion $q_w + q_x\mathrm{i} + q_y\mathrm{j} + q_z\mathrm{k} \in \mathbb{H}$ then, according to [10], the i-th diagonal block of $\mathbf{Q}(\boldsymbol{\varphi}(t))$ is

$$\mathbf{Q}_{ii}(\boldsymbol{\varphi}_i(t)) = \frac{1}{2}\begin{bmatrix} -q_x & -q_y & -q_z \\ q_w & q_z & -q_y \\ -q_z & q_w & q_x \\ q_y & -q_x & q_w \end{bmatrix}.$$

Each particle has an associated body frame whose origin coincides with the body's center of mass and whose axes are initially aligned with the axes of the observational frame. The body frame is rigidly attached to the body and translates and rotates with it. All of the state variables and other quantities are expressed in the observational frame unless noted otherwise. Furthermore, the matrix

$$\mathbf{M}(\boldsymbol{\varphi}(t)) = \begin{bmatrix} \underset{i=1..\nu_b}{\mathrm{diag}}\ m_i\mathbf{1} & \\ & \underset{i=1..\nu_b}{\mathrm{diag}}\ \mathbf{I}_{ii}(\boldsymbol{\varphi}_i(t)) \end{bmatrix}$$

is the block-diagonal mass matrix, where $\mathbf{1}$ denotes the 3×3 identity matrix. The mass matrix contains the constant particle masses m_i and the particles' inertia matrices $\mathbf{I}_{ii}(\boldsymbol{\varphi}_i(t))$ about the particles' centers of mass. The latter can be calculated by similarity transformations from the constant body frame inertia matrices $\mathbf{I}^0_{ii}$. If the body frames are attached such that they coincide with the principal axes of their particles, then the body frame inertia matrices are diagonal, and floating-point operations as well as memory can be saved. The lower-right block of the mass matrix corresponds to the matrix $\mathbf{I}(\boldsymbol{\varphi}(t))$. $\boldsymbol{f}(\boldsymbol{s}(t), t)$ and $\boldsymbol{\tau}(\boldsymbol{s}(t), t)$ are the total forces and torques (together they are referred to as wrenches) acting at the particles' centers of mass. Both may depend on any of the state variables $\boldsymbol{s}(t)$ of the system and time t. The wrench contributions from contact reactions are summed up with external forces $\boldsymbol{f}_{ext}$ and torques $\boldsymbol{\tau}_{ext}$ such as fictitious forces from non-inertial reference frames.

Let $\boldsymbol{\lambda}_j(t) \in \mathbb{R}^3$ be the contact reaction of a contact $j \in \mathcal{C}$, where $\mathcal{C} = \{1 \,..\, \nu_c\}$ is the set of potential contact indices. Let $(j_1, j_2) \in \mathcal{B}^2$ be the index pair of both particles involved in the contact j, where $\mathcal{B} = \{1 \,..\, \nu_b\}$ is the set of body indices. Let $\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) \in \mathbb{R}^3$ be the location of contact j, then the wrench on body i is

$$\begin{pmatrix} \boldsymbol{f}_i(\boldsymbol{s}(t), t) \\ \boldsymbol{\tau}_i(\boldsymbol{s}(t), t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{i,ext}(\boldsymbol{s}(t), t) \\ \boldsymbol{\tau}_{i,ext}(\boldsymbol{s}(t), t) \end{pmatrix} + \underbrace{\sum_{\substack{j\in\mathcal{C} \\ j_1=i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) - \boldsymbol{x}_i(t))^\times \end{bmatrix} \boldsymbol{\lambda}_j(t) - \sum_{\substack{j\in\mathcal{C} \\ j_2=i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) - \boldsymbol{x}_i(t))^\times \end{bmatrix} \boldsymbol{\lambda}_j(t)}_{\text{wrench contributions}}, \tag{1}$$

where $(\cdot)^\times$ is a matrix, which when multiplied to a vector corresponds to the cross product between its operand $(\cdot)$ and the vector.

In contrast to soft contact models, the contact reactions in hard contact models cannot be explicitly expressed as a function of the state variables but are defined implicitly, e.g. by implicit non-linear functions [23], complementarity constraints [1,3], or inclusions [38]. In any case, the constraints distinguish between reactions in the directions normal to the contact surfaces and reactions in the tangential planes of the contact surfaces. The former are used to formulate the non-penetration constraints, and the latter are used to formulate the friction constraints. For that reason, each contact j is associated with a contact frame, where the axis $\boldsymbol{n}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) \in \mathbb{R}^3$ points along the direction normal to the contact surface, and the other two axes $\boldsymbol{t}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) \in \mathbb{R}^3$ and $\boldsymbol{o}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) \in \mathbb{R}^3$ span the tangential plane of the contact.

Let $\mathcal{S}_i$ be the set of points in the observational frame defining the shape of particle i, and let $f_i(\boldsymbol{x}_i(t), \boldsymbol{\varphi}_i(t), \boldsymbol{y}) \in \mathbb{R}$ be the associated signed distance function for a point $\boldsymbol{y}$ in the observational frame. The signed distance function shall be negative in the interior of $\mathcal{S}_i$. Assuming that all particles are (strictly) convex with sufficiently smooth boundaries, then for a pair of particles (j_1, j_2) involved in a contact j, the contact location $\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t))$ is defined by the optimization problem

$$\begin{aligned} \hat{\boldsymbol{x}}_j(t) &:= \hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) \\ &= \underset{f_{j_2}(\boldsymbol{x}_{j_2}(t), \boldsymbol{\varphi}_{j_2}(t), \boldsymbol{y}) \le 0}{\arg\min} f_{j_1}(\boldsymbol{x}_{j_1}(t), \boldsymbol{\varphi}_{j_1}(t), \boldsymbol{y}), \end{aligned} \tag{2}$$

with associated contact normal

$$\boldsymbol{n}_j(t) := \boldsymbol{n}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) = \nabla_{\boldsymbol{y}} f_{j_2}(\boldsymbol{x}_{j_2}(t), \boldsymbol{\varphi}_{j_2}(t), \hat{\boldsymbol{x}}_j(t)),$$

pointing outwards with respect to $\mathcal{S}_{j_2}$ and associated signed contact distance

$$\xi_j(t) := \xi_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) = f_{j_1}(\boldsymbol{x}_{j_1}(t), \boldsymbol{\varphi}_{j_1}(t), \hat{\boldsymbol{x}}_j(t))$$

which is negative in the case of penetrations.

For convex particles each pair of bodies results in a potential contact, and thus the total number of contacts ν_c is limited by $\frac{1}{2}\nu_b(\nu_b - 1)$. Non-convex objects e.g. can be implemented as composite objects of convex particles. By convention a positive reaction in normal direction is repulsive, and thus

the contact reaction $\boldsymbol{\lambda}_j(t)$ acts positively on particle j_1 and negatively on j_2, thus explaining the signs in (1). By applying the opposite reactions at the same point in the observational frame, not only the linear momentum can be conserved but also the angular momentum of the system. Conservation of energy can only hold if the contact model does not include dissipative effects. Hard-contact models require the Signorini condition to hold. Written as a complementarity condition for a contact j, it reads

$$\xi_j(t) \geq 0 \perp \lambda_{j,n}(t) \geq 0,$$

where $\lambda_{j,n}(t) = \boldsymbol{n}_j(t)^{\mathrm{T}}\boldsymbol{\lambda}_j(t)$. The signed contact distance is required to be non-negative, resulting in a non-penetration constraint. The contact reaction in direction of the contact normal is also required to be non-negative, resulting in non-adhesive contact reactions. Furthermore, both quantities must be complementary, meaning that either of them must be equal to zero. This effects that the contact reaction can only be non-zero if the contact is closed.

However, the Signorini condition does not determine the contact reaction force if the contact is closed. In that case the non-penetration constraint on the velocity level,

$$\dot{\xi}_j^+(t) \geq 0 \perp \lambda_{j,n}(t) \geq 0,$$

must be added to the system, where $\dot{\xi}_j^+$ is the right derivative of the signed contact distance with respect to time. The constraint allows the contact to break only if no reaction force is present and otherwise forces $\dot{\xi}_j^+(t) = 0$. In the latter case the reaction force is still not fixed. The non-penetration constraint on the acceleration level,

$$\ddot{\xi}_j^+(t) \geq 0 \perp \lambda_{j,n}(t) \geq 0,$$

then determines the force also if $\ddot{\xi}_j^+(t) = 0$.

When considering impacts, a non-penetration constraint for the reaction impulse in the direction normal to the contact surface must be formulated, and, if the contact is closed, an additional constraint modelling the impact dynamics, such as Newton's impact law, must be added. The coefficient of restitution in Newton's impact law can then be used to control the amount of energy that is dissipated in the collisions. This is analogous to damping elements in soft contact models.

These non-penetration conditions can be complemented by a friction condition. The most prominent model for dry frictional contact is the Coulomb model which restricts the relative contact velocity in the tangential plane of the contact. The relative contact velocity for a pair of particles (j_1, j_2) involved in a contact j is

$$\begin{aligned}\boldsymbol{\delta v}_j^+(\boldsymbol{s}(t)) = \boldsymbol{v}_{j_1}^+(t) + \boldsymbol{\omega}_{j_1}^+(t) \times (\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) - \boldsymbol{x}_{j_1}(t)) \\ - \boldsymbol{v}_{j_2}^+(t) - \boldsymbol{\omega}_{j_2}^+(t) \times (\hat{\boldsymbol{x}}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t)) - \boldsymbol{x}_{j_2}(t)).\end{aligned}$$

Let

$$\boldsymbol{\delta v}_{j,to}^+(t) := \boldsymbol{\delta v}_{j,to}^+(\boldsymbol{s}(t)) = \begin{pmatrix} \boldsymbol{t}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t))^{\mathrm{T}}\boldsymbol{\delta v}_j^+(\boldsymbol{s}(t)) \\ \boldsymbol{o}_j(\boldsymbol{x}(t), \boldsymbol{\varphi}(t))^{\mathrm{T}}\boldsymbol{\delta v}_j^+(\boldsymbol{s}(t)) \end{pmatrix}$$

be the relative contact velocity in the tangential plane after application of the contact impulses, then the Coulomb conditions for a non-impulsive point in time t are

$$\begin{aligned}&\|\boldsymbol{\lambda}_{j,to}(t)\|_2 \leq \mu_j \lambda_{j,n}(t) \text{ and} \\ &\|\boldsymbol{\delta v}_{j,to}^+(t)\|_2 \boldsymbol{\lambda}_{j,to}(t) = -\mu_j \lambda_{j,n}(t) \boldsymbol{\delta v}_{j,to}^+(t).\end{aligned}$$

This inequality limits the magnitude of the friction force to the product of the coefficient of friction μ_j times the reaction force in direction of the contact normal $\lambda_{j,n}(t)$. If the magnitude of the friction force reaches the limiting value, then the friction conditions can only be fulfilled if the friction force directly opposes the tangential relative contact velocity. If instead the magnitude of the friction force is below the limit, then the equation can only be fulfilled if $\|\boldsymbol{\delta v}_{j,to}^+(t)\|_2 = 0$. However, in this case these conditions must be supplemented by the constraint

$$\|\dot{\boldsymbol{\delta v}}_{j,to}^+(t)\|_2 \boldsymbol{\lambda}_{j,to}(t) = -\mu_j \lambda_{j,n}(t) \dot{\boldsymbol{\delta v}}_{j,to}^+(t)$$

on acceleration level in order to determine the friction force. Likewise constraints for the friction impulse are necessary. At this point we refrain from formulating the measure differential inclusion in further detail since it would not contribute information essential to the remaining paper which only deals with the discrete-time system.

3 Discrete dynamical system

In simulations of granular matter impulsive reactions are abundant. Higher-order integrators for time-stepping schemes are still subject to active research [33]. In particular, discontinuities pose problems for these integrators. Hence, the continuous dynamical system is discretized in the following with an integrator of order one, resembling the semi-implicit Euler method and similar to the one suggested in [2].

Let $\boldsymbol{s}$, $\boldsymbol{x}$, $\boldsymbol{\varphi}$, $\boldsymbol{v}$ and $\boldsymbol{\omega}$ denote the given discrete-time state variables at time t and $\boldsymbol{\lambda}$ the contact reactions at time t. Then the state variables at time $t + \delta t$ are functions depending on the contact reactions: $\boldsymbol{s}'(\boldsymbol{\lambda})$, $\boldsymbol{x}'(\boldsymbol{\lambda})$, $\boldsymbol{\varphi}'(\boldsymbol{\lambda})$, $\boldsymbol{v}'(\boldsymbol{\lambda})$ and $\boldsymbol{\omega}'(\boldsymbol{\lambda})$. The discrete-time Newton-Euler equations integrated by the proposed scheme are

$$\begin{aligned}\begin{pmatrix} \boldsymbol{x}'(\boldsymbol{\lambda}) \\ \boldsymbol{\varphi}'(\boldsymbol{\lambda}) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\varphi} \end{pmatrix} + \delta t \begin{pmatrix} \boldsymbol{v}'(\boldsymbol{\lambda}) \\ \mathbf{Q}(\boldsymbol{\varphi})\boldsymbol{\omega}'(\boldsymbol{\lambda}) \end{pmatrix}, \\ \begin{pmatrix} \boldsymbol{v}'(\boldsymbol{\lambda}) \\ \boldsymbol{\omega}'(\boldsymbol{\lambda}) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{pmatrix} + \delta t \mathbf{M}(\boldsymbol{\varphi})^{-1} \begin{pmatrix} \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{\lambda}, t) \\ \boldsymbol{\tau}(\boldsymbol{s}, \boldsymbol{\lambda}, t) - \boldsymbol{\omega} \times \mathbf{I}(\boldsymbol{\varphi})\boldsymbol{\omega} \end{pmatrix}.\end{aligned} \tag{3}$$

Positions and orientations at time $t+\delta t$ appear exclusively on the left-hand side of the position and orientation integration. Velocities at time $t+\delta t$ appear on the left-hand side of the velocity integration and additionally in the integration of positions and orientations. The numerical integration of the quaternion has the effect that the quaternion gradually looses its unit length. This deficiency can be compensated by renormalizing the quaternions after each integration.

Instead of discretizing each of the five intermittently active continuous-time complementarity constraints, the Signorini condition is only required to hold at the end of each time step. This has the effect that impulsive reactions are no longer necessary to satisfy the condition since the condition is no longer required to be fulfilled instantaneously. Furthermore, the signed distance function gets linearized, resulting in

$$\xi_j(t+\delta t) = \xi_j(t) + \delta t \dot{\xi}_j(t) + \mathcal{O}(\delta t^2),$$

where the time derivative of the signed contact distance can be determined to be

$$\dot{\xi}_j(t) = \boldsymbol{n}_j(t)^{\mathrm{T}} \boldsymbol{\delta v}_j^+(\boldsymbol{s}(t))$$

under the assumption that the contact point $\hat{\boldsymbol{x}}_j(t)$ translates and rotates in accordance with body j_2, such that

$$\dot{\hat{\boldsymbol{x}}}_j(t) = \boldsymbol{v}_{j_2}(t) + \boldsymbol{\omega}_{j_2}(t) \times (\hat{\boldsymbol{x}}_j(t) - \boldsymbol{x}_{j_2}(t)).$$

Let the time-discrete relative contact velocity be

$$\begin{aligned}\boldsymbol{\delta v}'_j(\boldsymbol{\lambda}) =& \boldsymbol{v}'_{j_1}(\boldsymbol{\lambda}) + \boldsymbol{\omega}'_{j_1}(\boldsymbol{\lambda}) \times (\hat{\boldsymbol{x}}_j - \boldsymbol{x}_{j_1}) \\ & - \boldsymbol{v}'_{j_2}(\boldsymbol{\lambda}) - \boldsymbol{\omega}'_{j_2}(\boldsymbol{\lambda}) \times (\hat{\boldsymbol{x}}_j - \boldsymbol{x}_{j_2}),\end{aligned}$$

where the velocities are discretized implicitly. The discrete non-penetration constraint then is

$$\frac{\xi_j}{\delta t} + \boldsymbol{n}_j^{\mathrm{T}} \boldsymbol{\delta v}'_j(\boldsymbol{\lambda}) \geq 0 \perp \lambda_{j,n} \geq 0. \tag{4}$$

The term $\frac{\xi_j}{\delta t}$ acts as an error correction term if penetrations are present ($\xi_j < 0$). In that case it can be scaled down to avoid introducing an excessive amount of energy. If no numerical error is present, the contact is inelastic and thus corresponds to a hard contact model with Newton's impact law, where the coefficient of restitution is 0. The frictional constraints translate into

$$\begin{aligned}&\|\boldsymbol{\lambda}_{j,to}\|_2 \leq \mu_j \lambda_{j,n} \text{ and} \\ &\|\boldsymbol{\delta v}'_{j,to}(\boldsymbol{\lambda})\|_2 \boldsymbol{\lambda}_{j,to} = -\mu_j \lambda_{j,n} \boldsymbol{\delta v}'_{j,to}(\boldsymbol{\lambda}).\end{aligned} \tag{5}$$

Let $\boldsymbol{F}_j(\boldsymbol{\lambda}) = \boldsymbol{0}$ denote a non-linear system of equations equivalent to the constraints from (4) and (5) of a single contact j, and let $\boldsymbol{F}(\boldsymbol{\lambda})$ denote the collection of all $\boldsymbol{F}_j(\boldsymbol{\lambda})$. Neither $\boldsymbol{F}(\boldsymbol{\lambda}) = \boldsymbol{0}$ nor $\boldsymbol{F}_j(\boldsymbol{\lambda}) = \boldsymbol{0}$ for given $\boldsymbol{\lambda}_{\bar{j}}$ have unique solutions. Let $\boldsymbol{F}_j^{-1}(\boldsymbol{0}, \boldsymbol{\lambda}_{\bar{j}})$ be a possible solution of the one-contact problem of contact j, given the contact reactions $\boldsymbol{\lambda}_{\bar{j}}$ of all other contacts $\bar{j}$.

A detailed discussion of solution algorithms for one-contact problems is out of the scope of this article. However, splitting methods, where non-penetration and friction constraints are solved separately, are prone to slow convergence or cycling. In [7] Bonnefon et al. solve the one-contact problem by finding the root of a quartic polynomial. Numerous other approaches exist for modified friction laws, notably those where the friction cone is approximated by a polyhedral cone and solution algorithms for linear complementarity problems can be used [1,32]. In any case the algorithm of choice should be as robust as possible in order to successfully resolve ν_c contacts per iteration and time step. In this article we will demonstrate that ν_c can be in the order of 10^{10}.

4 Contact detection

The contact problem $\boldsymbol{F}(\boldsymbol{\lambda}) = \boldsymbol{0}$ constitutes $\mathcal{O}(\nu_b^2)$ non-linear equations. Thus, already the setup of the contact problem would not run in linear time, much less the solution algorithm even if it were optimal. The contact constraints of a contact j can be removed from the system without altering the result if the contact is known to stay open ($\boldsymbol{\lambda}_j = \boldsymbol{0}$) within the current time step. Let

$$\mathcal{S}_i(t) = \left\{ \boldsymbol{y} \in \mathbb{R}^3 \;\middle|\; f_i(\boldsymbol{x}_i(t), \boldsymbol{\varphi}_i(t), \boldsymbol{y}) \leq 0 \right\}$$

be the set of points in space corresponding to the rotated and translated shape of particle i at time t and let

$$\mathcal{H}_i(t) = \mathcal{S}_i(t) + \left\{ \boldsymbol{y} \in \mathbb{R}^3 \;\middle|\; \|\boldsymbol{y}\|_2 \leq h_i(t) \right\}$$

be an intersection hull that spherically expands the particle shape by the radius $h_i(t) > 0$. If $h_i(t)$ is chosen large enough then an algorithm finding intersections between the hulls can detect all contacts that can potentially become active in the current time step. A possible choice for the expansion radius is

$$h_i(t) = \delta t (\|\boldsymbol{v}_i(t)\|_2 + \|\boldsymbol{\omega}_i(t)\|_2 \bar{r}_i) + \tau, \tag{6}$$

where $\bar{r}_i = \max_{\boldsymbol{y} \in \mathcal{S}_i(0)} \|\boldsymbol{y}\|_2$ is the bounding radius of particle i, and τ is a safety margin. The safety margin becomes necessary since an explicit Euler step is underlying the derivation of (6). In practice, the usage of intersection hulls reduces the number of contacts considerably. E.g., monodisperse spherical particles can have at most 12 contacts per particle if the expansion radii are small enough [34], resulting in $\mathcal{O}(\nu_b)$ potential contacts.

Broad-phase contact detection algorithms aim to limit the particle pairs that are possible candidates for contacts to as few as possible. To this end they use e.g. spatial partitioning or they exploit the temporal coherence of the particle positions [9]. The candidate pairs are then checked in detail in the narrow-phase contact detection, where (2) is solved for each pair, leading to the contact location $\hat{\boldsymbol{x}}_j$, normal $\boldsymbol{n}_j$ and signed distance ξ_j for a contact j.

To solve (2) for non-overlapping particles, the Gilbert-Johnson–Keerthi (GJK) algorithm can be used [4,13]. For overlapping particle shapes the expanding polytope algorithm (EPA) computes approximate solutions [5]. For simple geometric primitives like spheres, the optimization problem can be solved analytically. The indices of all contacts found that way form the set of potential contacts $\mathcal{C} = \{1 \,..\, \nu_c\}$ at time t. Let $\boldsymbol{F}(\boldsymbol{\lambda}) = \boldsymbol{0}$ from now on denote the contact problem where all contact conditions and contact reactions whose indices are not part of $\mathcal{C}$ have been filtered out.

5 Numerical solution algorithms

To solve the multi-contact problem, when suitable solution algorithms for the one-contact problems $\boldsymbol{F}_j^{-1}$ are given, a non-linear block Gauss-Seidel (NBGS) can be used as propagated by the non-smooth contact dynamics (NSCD) method [20]. Unfortunately, the Gauss-Seidel algorithm cannot be efficiently executed in parallel for irregular data dependencies as they appear in contact problems [22].

As an alternative, a more general variant is proposed here, accommodating the subdomain structure that will arise in the domain partitioning. Therefore, each contact $j \in \mathcal{C}$ is associated with a subdomain number $s_c(j) \in \mathcal{P}$, where $\mathcal{P} = \{1 \,..\, \nu_p\}$ is the set of subdomain indices for ν_p subdomains. Algorithm 1 presents pseudo-code for the subdomain NBGS with the relaxation parameter $\omega > 0$. The initial solution is chosen to be zero, however, any other initialization can be used, in particular contact reactions from the previous time step.

The algorithm is of iterative nature and needs an appropriate stopping criterion to terminate. Note that the choice of the stopping criterion for parallel executions of the algorithm is not different from serial executions [22,36]. In each iteration k a sweep over all contacts is performed, where each contact j is relaxed, given an approximation of all other contact reactions $\tilde{\boldsymbol{\lambda}}^{(k,j)}$. In the subdomain NBGS, the approximation of contact reaction l is taken from the current iteration if it was already relaxed ($l < j$) and if it is associated with the same subdomain as the contact j to be relaxed ($s_c(l) = s_c(j)$). In all other cases, the approximation is taken from the previous iteration. The contact reaction $\boldsymbol{\lambda}_j^{(k+1)}$ is then a weighted mean between the previous approximation and the relaxation result. If all contacts are associated

1: $k \leftarrow 0$
2: $\boldsymbol{\lambda}^{(k)} \leftarrow \boldsymbol{0}$
3: **while** convergence criterion not met **do**
4: **for** $j \leftarrow 1$ **to** ν_c **do**
5: **for** $l \in \mathcal{C}$ **do**
6: $\tilde{\boldsymbol{\lambda}}_l^{(k,j)} \leftarrow \begin{cases} \boldsymbol{\lambda}_l^{(k+1)} & \text{if } l < j \wedge s_c(l) = s_c(j) \\ \boldsymbol{\lambda}_l^{(k)} & \text{else} \end{cases}$
7: **end for**
8: $\boldsymbol{y} \leftarrow \boldsymbol{F}_j^{-1}(\boldsymbol{0}, \tilde{\boldsymbol{\lambda}}_j^{(k,j)})$
9: $\boldsymbol{\lambda}_j^{(k+1)} \leftarrow \omega \boldsymbol{y} + (1-\omega)\boldsymbol{\lambda}_j^{(k)}$
10: **end for**
11: $k \leftarrow k+1$
12: **end while**

Algorithm 1: The subdomain NBGS method with relaxation parameter ω.

with the same subdomain and $\omega = 1$ then Algorithm 1 corresponds to a classic NBGS. If each contact is associated to a different subdomain then Algorithm 1 corresponds to a non-linear block Jacobi (NBJ) with relaxation parameter ω.

It is well known that Gauss-Seidel and Jacobi methods of equations do not scale linearly with the number of unknowns for many problems of practical interest. This means that the number of iterations needed to obtain a given error bound increases with the number of unknowns in the system. However, this effect is most severe only when large global systems must be solved. In many cases of interest in granular dynamics, the global system effectively splits into many small systems that correspond to the clusters of objects in contact. Therefore, and in particular when a good initial guess is available from previous time steps, often a moderate number of iterations is sufficient to obtain a satisfactory accuracy. To compensate for effects due to a variable number of iterations in the scaling experiments in Sect. 7, the iterative solver is stopped there after a constant number of iterations.

The subsequent section explains how the subdomain NBGS algorithm presented in this section can be implemented and efficiently executed in parallel on machines with distributed memory. Under the assumption that the algorithm converges, the contact reactions thus obtained in a parallel execution of the NBGS solve the mathematical statement of the problem from Sect. 3 just as if the algorithm is executed serially. However, as mentioned briefly in Sect. 3, the discrete systems have a non-unique solution, as it is inherently caused by the hard contact model. Hence, solutions obtained by the NBJ, the classic NBGS and the subdomain NBGS can differ. However, this is simply an effect of an insufficient regularization of the hard contact problem, it is not caused by the parallelization per se.

6 Parallelization design

Sect. 6.1 introduces the domain partitioning approach. Sect. 6.2 then discusses requirements that must be met in

order to be able to treat all contacts exactly once in parallel. Sect. 6.3 explains a special technique to reduce the data dependencies to other processes. To this end, *accumulator* and *correction* variables will be introduced. In Sect. 6.4 conditions are discussed under which the set of communication partners can be reduced to the nearest neighbors. Time-integration and the subsequent necessity of synchronization are addressed in Sect. 6.5 before summarizing the time-stepping procedure in Sect. 6.6.

6.1 Domain partitioning

Under the assumption that no contacts are present, there exists no coupling between the data of any two particles, and the problem becomes embarrassingly parallel: Each process integrates $\lfloor \frac{v_b}{v_p} \rfloor$ or $\lceil \frac{v_b}{v_p} \rceil$ particles. Let $s_b(i) \in \mathcal{P}$ determine the process responsible for the time-integration of particle i as of now referred to as the parent process. All data associated with this particle, that is the state variables (position, orientation, velocities) and constants (mass, body frame inertia matrix, shape parameters), are instantiated only at the parent process in order to distribute the total memory load. However, contacts or short-range potentials introduce data dependencies to particles that in general are not instantiated on the local process nor on a process close to the local one, rendering a proper scaling impossible. A domain partitioning approach alleviates this problem.

Let Ω denote the computational domain within which all particles are located and $\Omega_p \subseteq \Omega, p \in \mathcal{P}$, a family of disjoint subdomains into which the domain shall be partitioned. In this context, subdomain boundaries are assigned to exactly one process. One process shall be executed per subdomain. The number of processes can e.g. correspond to the number of compute nodes in a hybrid parallelization or to the total number of cores or even threads in a homogeneous parallelization. In the domain partitioning approach the integration of a particle whose center of mass $\boldsymbol{x}_i$ is located in a subdomain Ω_p at time t is calculated by process p. That way data dependencies typically pertain the local or neighboring subdomains since they are considered to be of short range. Let $s_b(i)$ be adapted accordingly. Special care is required when associating a particle to a subdomain whose center of mass is located on or near subdomain interfaces. Especially, periodic boundary conditions can complicate the association process since the finite precision of floating-point arithmetics does in general not allow a consistent parametric description of subdomains across periodic boundaries. Sect. 6.5 below explains how the synchronization protocol can be used to realize a reliable association.

The domain partitioning should be chosen such that approximately an equal number of particles is located initially in each subdomain and that this is sustained over the course of the simulation in order to balance the computational load which is directly proportional to the number of particles. Particles now migrate between processes if their positions change the subdomain. Migration can lead to severe load imbalances that may need to be addressed by dynamically repartitioning the domain. Such load-balancing techniques are beyond the scope of this article.

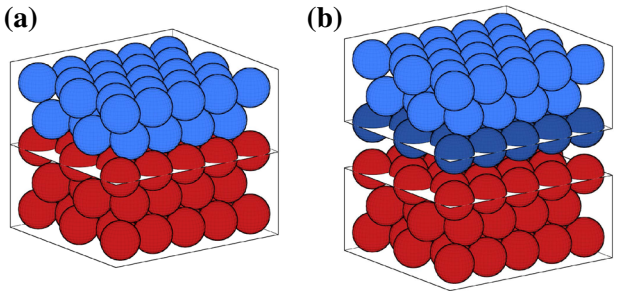

Fig. 1 A hexagonal close packing of *spheres* inside a *box-shaped* domain. The domain is partitioned into two subdomains **a** *Red* particles are associated with the bottom process, *blue* particles are associated with the top process. **b** The view of each process is illustrated separately: The top process instantiates additional shadow copies. (Color figure online)

6.2 Shadow copies

A pure local instantiation of particles has the effect that contacts cannot be detected between particles that are not located on the same process. A process can detect a contact if both particles involved in the contact are instantiated on that process. In order to guarantee that at least one process can detect a contact, the condition that a contact j must be detected by all processes whose subdomains intersect with the hull intersection $\mathcal{H}_{j_1} \cap \mathcal{H}_{j_2}$ is sufficient if the intersection of the hull intersection and the domain is non-empty. This condition can be fulfilled by the following requirement:

Requirement 1 *A particle i must be instantiated not only on the parent process but also on all processes whose subdomains intersect with the particle's hull.*

These additional instantiations shall be termed *shadow copies* in the following. As passive copies of the original data structures, they must be kept in synchronization with the original instantiation on the parent process. Figure 1 illustrates the concept. A hexagonal close packing of spheres confined by lateral walls is shown. The illustration is simplified in the sense that no intersection hulls are considered. The box-shaped domain is split into an upper and a lower subdomain. Each subdomain is associated to one process. The red particles are associated to the bottom process according to the positions, whereas the blue particles are associated to the top process. In Fig. 1b the view of the bottom and top process is

split. Since the upper-most layer of red particles extends into the subdomain of the top process, the top process must instantiate synchronized copies of these particles. These shadow copies are marked by using a darker shading in the figure.

In order to determine which process will be responsible for treating the contact a rule is needed. Ideally this does not require additional communication. Here, the statement that a process is responsible for *treating* a contact refers to the responsibility of the process for executing the relaxation of the respective contact in Algorithm 1. The typical choice for this rule requires that the process whose subdomain contains the point of contact is put in charge to treat the contact [36].

However, this seemingly natural rule only works if the process whose subdomain contains the point is able to detect the contact. Unfortunately, this is only guaranteed if the point of contact is located within the hull intersection. Also, if the point of contact is located outside of the domain Ω, then no process would be responsible to treat it.

A more intricate drawback of this approach is that it can fail in case of periodic boundary conditions: If the contact point is located near the periodic boundary, the periodic image of the contact point will be detected at the other end of the simulation box. Due to the shifted position of the contact point image and the limited numerical precision, the processes can no longer consistently determine which process gets to treat the contact.

A more robust rule can be established by fulfilling the following requirement:

Requirement 2 *All shadow copy holders of a particle maintain a complete list of all other shadow copy holders and the parent process of that particle.*

Then each process detecting a contact can determine the list of all processes detecting that very same contact, which is the list of all processes with an instantiation of both particles involved in the contact. This list is exactly the same on all processes detecting the contact and is not prone to numerical errors. The rule can then e.g. appoint the detecting process with smallest rank to treat the contact. In order to enhance the locality of the contact treatment, the rule should favor the particle parents if they are among the contact witnesses. Any such rule defines a partitioning of the contact set $\mathcal{C}$. Let $\mathcal{C}_p$ be the set of all contacts treated by process $p \in \mathcal{P}$. Then process p instantiates all contacts $j \in \mathcal{C}_p$.

Figure 2 illustrates the association of contacts to processes on the basis of the hexagonal close packing. In Fig. 2a the contacts detected by each process are shown. The blue contacts are only detected on the top process and the red contacts are only detected on the bottom process. The white contacts are detected on both processes. Requirement 2 enables the processes to consistently associate these contacts to the processes. In Fig. 2b the ambiguity is resolved and the red and blue contacts form a partitioning of all contacts. If the bottom process has the smaller rank, the smallest rank rule from above would result in such a contact partitioning.

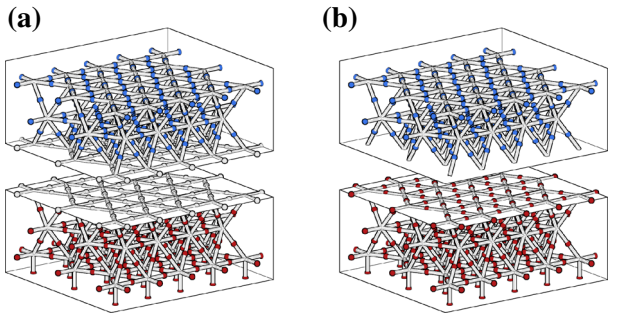

Fig. 2 The contact network of the *hexagonal* close packing. The view is separated for each process **a** *Red* and *blue* contact nodes are unambiguously associated with the *bottom* and *top* process respectively. White contact nodes are detected redundantly **b** All redundant contacts are removed and the *red* and *blue* contact nodes form a partitioning of all contacts. (Color figure online)

6.3 Accumulator and correction variables

The contact relaxations in Algorithm 1 exhibit sums with non-local data dependencies. In the following, the redundant evaluation of these sums is prevented by introducing accumulator variables and the non-local data dependencies are reduced by introducing correction variables.

The relaxation of a contact j depends on the data of the state variables of both particles (j_1, j_2) involved in the contact, their constants and shape parameters, as can be seen by inspecting (4), (5) and the definitions of the terms appearing therein. All of these quantities are instantiated on the detecting process, either as a shadow copy or as an original instance. The contact variables of contact j (location, signed distance and the contact frame) are also required. They are available on the detecting process since they result from the positions, orientations, and the shape parameters of the particles (j_1, j_2) in the contact detection. Furthermore, the force and torque terms from (1) acting on these particles additionally depend on the locations $\hat{\boldsymbol{x}}_l$ and reaction approximations $\tilde{\boldsymbol{\lambda}}_l^{(k,j)}$ of all other contacts l involving one of the particles (j_1, j_2). Neither the locations nor the reaction approximations of these contacts are necessarily available on the process treating contact j. To rectify this deficiency, one can introduce contact shadow copies so that location and reaction approximation can be mirrored at every instantiation of both particles involved in the contact. However, the organisational overhead of contact shadow copies can be circumvented. It is not necessary that the process treating the contact evaluates all the wrench contributions to the particles involved in the contact. Instead, parts of the wrench contribution sum can be evaluated on the processes actually treating the remote contacts and can subsequently be communicated:

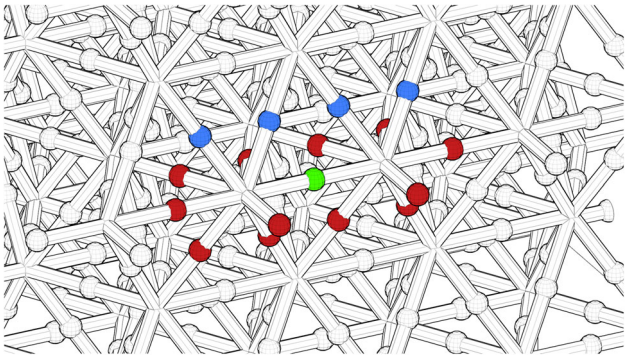

Fig. 3 The contact network of the *hexagonal* close packing. In order to relax the *green* contact on the *bottom* process, the contact reactions of the *red* contact nodes are needed from the *bottom* process and the contact reactions of the *blue* contact nodes are needed from the *top* process. (Color figure online)

$$\begin{aligned}\begin{pmatrix} f_i(\boldsymbol{\lambda}) \\ \boldsymbol{\tau}_i(\boldsymbol{\lambda}) \end{pmatrix} &= \begin{pmatrix} f_{i,ext} \\ \boldsymbol{\tau}_{i,ext} \end{pmatrix} + \sum_{\substack{l \in \mathcal{C} \\ l_1 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} \boldsymbol{\lambda}_l - \sum_{\substack{l \in \mathcal{C} \\ l_2 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} \boldsymbol{\lambda}_l \\ &= \begin{pmatrix} f_{i,ext} \\ \boldsymbol{\tau}_{i,ext} \end{pmatrix} + \sum_{p \in \mathcal{P}} \underbrace{\left(\sum_{\substack{l \in \mathcal{C}_p \\ l_1 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} \boldsymbol{\lambda}_l - \sum_{\substack{l \in \mathcal{C}_p \\ l_2 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} \boldsymbol{\lambda}_l \right)}_{\text{wrench contribution } (f_{i,p} \ \boldsymbol{\tau}_{i,p})^{\mathrm{T}} \text{ to particle } i \text{ from process } p}\end{aligned}$$

Figure 3 illustrates the data dependencies in the relaxation of a contact on the basis of the contact network in the hexagonal close packing. When relaxing the green contact, all contact reactions acting on the two contacting particles (j_1, j_2) are required in order to perform the relaxation. Let j_1 be the particle left of the green contact and j_2 be the particle right of the green contact. The total wrench on j_1 is the sum of the wrench contribution on the particle from the top process and the bottom process. The wrench contribution from the top process consists of two contact reactions marked in blue and the wrench contribution from the bottom process consists of six contact reactions marked in red. The same applies to particle j_2.

The total wrench on particle $i \in \{j_1, j_2\}$ can also be expressed in terms of the total wrench on particle i at the beginning of iteration k:

$$\begin{aligned}\begin{pmatrix} f_i(\boldsymbol{\lambda}) \\ \boldsymbol{\tau}_i(\boldsymbol{\lambda}) \end{pmatrix} = \begin{pmatrix} f_i(\boldsymbol{\lambda}^{(k)}) \\ \boldsymbol{\tau}_i(\boldsymbol{\lambda}^{(k)}) \end{pmatrix} + \sum_{p \in \mathcal{P}} \Bigg(&\sum_{\substack{l \in \mathcal{C}_p \\ l_1 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} (\boldsymbol{\lambda}_l - \boldsymbol{\lambda}_l^{(k)}) \\ &- \sum_{\substack{l \in \mathcal{C}_p \\ l_2 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} (\boldsymbol{\lambda}_l - \boldsymbol{\lambda}_l^{(k)}) \Bigg)\end{aligned}$$

When relaxing the contact j in iteration k of the subdomain NBGS, the wrench on particle $i \in \{j_1, j_2\}$ is evaluated with the reaction approximation $\tilde{\boldsymbol{\lambda}}^{(k,j)}$ as parameter. Since the subdomain NBGS respects the subdomain association of the contacts, the remote wrench contributions to particle i cancel out, and just the total wrench on particle i from the last iteration is needed in addition to corrections stemming from contacts that were already relaxed by the same process.

$$\begin{aligned}\begin{pmatrix} f_i(\tilde{\boldsymbol{\lambda}}^{(k,j)}) \\ \boldsymbol{\tau}_i(\tilde{\boldsymbol{\lambda}}^{(k,j)}) \end{pmatrix} = \begin{pmatrix} f_i(\boldsymbol{\lambda}^{(k)}) \\ \boldsymbol{\tau}_i(\boldsymbol{\lambda}^{(k)}) \end{pmatrix} &+ \sum_{\substack{l \in \mathcal{C}_{s_c(j)} \\ l < j \\ l_1 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} (\boldsymbol{\lambda}_l^{(k+1)} - \boldsymbol{\lambda}_l^{(k)}) \\ &- \sum_{\substack{l \in \mathcal{C}_{s_c(j)} \\ l < j \\ l_2 = i}} \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_l - \boldsymbol{x}_i)^\times \end{bmatrix} (\boldsymbol{\lambda}_l^{(k+1)} - \boldsymbol{\lambda}_l^{(k)})\end{aligned}$$

Our implementation instantiates variables on process p for the reaction approximations $\boldsymbol{\lambda}^{[p]} \in \mathbb{R}^{3|\mathcal{C}_p|}$ of all contacts treated by process p. Any updates to the reaction approximations occur in place. Furthermore, an implementation can instantiate accumulator variables $\boldsymbol{f}^{[p]}, \boldsymbol{\tau}^{[p]} \in \mathbb{R}^{3|\mathcal{B}_p|}$ on process p for the wrenches from the last iteration of all instantiated particles (shadow copies and original instances), where $\mathcal{B}_p$ contains the indices of all shadow copies and original instances instantiated on process p. This set is partitioned into $\mathcal{B}_{p,local}$ and $\mathcal{B}_{p,shadow}$, containing the indices of the original instances and the shadow copies respectively.

Instead of evaluating the wrench contribution sums each time when calculating the total wrench on particle i anew, the contributions can be accumulated as the contacts are relaxed. For that purpose, the correction variables $\delta\boldsymbol{f}^{[p]} \in \mathbb{R}^{3|\mathcal{B}_p|}$ and $\delta\boldsymbol{\tau}^{[p]} \in \mathbb{R}^{3|\mathcal{B}_p|}$ can be instantiated. Then, after line 9 of Algorithm 1, these wrench corrections can be updated by assigning

$$\begin{aligned}\begin{pmatrix} \delta\boldsymbol{f}_{j_1}^{[s_c(j)]} \\ \delta\boldsymbol{\tau}_{j_1}^{[s_c(j)]} \end{pmatrix} &\leftarrow \begin{pmatrix} \delta\boldsymbol{f}_{j_1}^{[s_c(j)]} \\ \delta\boldsymbol{\tau}_{j_1}^{[s_c(j)]} \end{pmatrix} + \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_j - \boldsymbol{x}_{j_1})^\times \end{bmatrix} (\boldsymbol{\lambda}_j^{(k+1)} - \boldsymbol{\lambda}_j^{(k)}), \\ \begin{pmatrix} \delta\boldsymbol{f}_{j_2}^{[s_c(j)]} \\ \delta\boldsymbol{\tau}_{j_2}^{[s_c(j)]} \end{pmatrix} &\leftarrow \begin{pmatrix} \delta\boldsymbol{f}_{j_2}^{[s_c(j)]} \\ \delta\boldsymbol{\tau}_{j_2}^{[s_c(j)]} \end{pmatrix} - \begin{bmatrix} \mathbf{1} \\ (\hat{\boldsymbol{x}}_j - \boldsymbol{x}_{j_2})^\times \end{bmatrix} (\boldsymbol{\lambda}_j^{(k+1)} - \boldsymbol{\lambda}_j^{(k)}).\end{aligned}$$

The evaluation of the total wrench on particle i in line 8 of Algorithm 1 when relaxing contact j in iteration k becomes

$$\begin{pmatrix} f_i(\tilde{\boldsymbol{\lambda}}^{(k,j)}) \\ \boldsymbol{\tau}_i(\tilde{\boldsymbol{\lambda}}^{(k,j)}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^{[s_c(j)]} \\ \boldsymbol{\tau}_i^{[s_c(j)]} \end{pmatrix} + \begin{pmatrix} \delta\boldsymbol{f}_i^{[s_c(j)]} \\ \delta\boldsymbol{\tau}_i^{[s_c(j)]} \end{pmatrix},$$

that is the sum of the accumulator and the correction variables.

At the end of each iteration the wrench corrections for each body must be reduced and added to the accumulated wrench from the last iteration. This can be performed in two message exchanges. In the first message exchange each process sends the wrench correction of each shadow copy to its parent process. Then each process sums up for each original instance all wrench corrections obtained from the

shadow copy holders, its own wrench correction, and the original instance's accumulated wrench. Subsequently, the updated accumulated wrench of each original instance is sent to the shadow copy holders in a second message-exchange communication step. The wrench corrections are then reset everywhere.

The accumulated wrenches $\boldsymbol{f}^{[p]}$, $\boldsymbol{\tau}^{[p]}$ are initialized on each process p before line 3 in Algorithm 1 to

$$\begin{pmatrix} \boldsymbol{f}_i^{[p]} \\ \boldsymbol{\tau}_i^{[p]} \end{pmatrix} \leftarrow \begin{pmatrix} \boldsymbol{f}_{i,ext} \\ \boldsymbol{\tau}_{i,ext} \end{pmatrix} \quad \forall i \in \mathcal{B}_p$$

unless the initial solution is chosen to be non-zero. The wrench corrections are initially set to $\mathbf{0}$. If the external forces and torques are not known on each process or are scattered among the processes having instantiated the particles, the initialization requires another two message exchanges, as they are necessary at the end of each iteration.

An alternative to storing accumulated wrenches and wrench corrections is to store accumulated velocities and velocity corrections. In that case, a process p instantiates variables $\boldsymbol{v}^{[p]}$, $\boldsymbol{\omega}^{[p]}$, $\delta\boldsymbol{v}^{[p]}$, $\delta\boldsymbol{\omega}^{[p]} \in \mathbb{R}^{3|\mathcal{B}_p|}$. The accumulated velocities are set to $\boldsymbol{v}_i'(\boldsymbol{\lambda}^{(k)})$ and $\boldsymbol{\omega}_i'(\boldsymbol{\lambda}^{(k)})$ for all $i \in \mathcal{B}_p$ in each iteration. They are initialized and updated accordingly. The velocity corrections are initialized and updated analogously to the wrench corrections. Hereby, the velocity variables can be updated in place. In the classic NBGS, no wrench or velocity correction variables would be necessary, but the corrections could be added to the velocity variables right away which is similar to the approach suggested by Tasora et al. in [39].

6.4 Nearest-neighbor communication

In the following we describe how the strict locality of particle interactions can be used to optimize the parallel communication and synchronization by exchanging messages only between nearest neighbors. So far the shadow copies can be present on any process, and the corrections in the summation over wrench or velocity corrections can originate from a long list of processes. However, by requiring that the particle hulls do not extend past any neighboring subdomains, all message exchanges can be reduced to nearest-neighbor communications. Let

$$\mathcal{N}_p = \left\{ q \in \mathcal{P} \setminus \{p\} \middle| \inf\{\|\boldsymbol{y}_p - \boldsymbol{y}_q\|_2 | \boldsymbol{y}_p \in \Omega_p, \boldsymbol{y}_q \in \Omega_q\} = 0 \right\}$$

be the set of process indices in direct neighborhood of process p's subdomain, and let

$$l_{dd} = \min_{p \in \mathcal{P}} \inf \left\{ \|\boldsymbol{y}_p - \boldsymbol{y}_q\|_2 \,\middle|\, \boldsymbol{y}_p \in \Omega_p, \boldsymbol{y}_q \in \bigcup_{q \in \mathcal{P} \setminus (\mathcal{N}_p \cup \{p\})} \Omega_q \right\},$$

be the shortest distance from a point inside a subdomain to a non-nearest neighbor. Then the condition

$$\bar{r}_i + \|\boldsymbol{v}_i(t)\|_2 \delta t + \tau < l_{dd} \quad \forall i \in \mathcal{B} \tag{7}$$

ensures in the first approximation that no hull extends past neighboring subdomains. This immediately defines a hard upper limit of $\bar{r}_i < l_{dd} - \tau$ for the bounding radius and thus for the size of all objects. Furthermore, given the particle shapes, velocities, and safety margins, the condition defines an upper limit for the time-step length. The introduction of condition (7) entails that on a process p only the description of the subdomains within $\Omega_p + \left\{ \boldsymbol{y} \in \mathbb{R}^3 \middle| \|\boldsymbol{y}\|_2 \leq l_{dd} \right\}$ needs to be available. Thus the description of non-nearest-neighbor subdomains can be dispensed with and the description of nearest-neighbor subdomains does not have to be correct outside of the spherical expansion of Ω_p with expansion radius l_{dd}. This permits a *localized* description of the domain partitioning on each process, describing the surrounding subdomains only.

Typically, the size limit stemming from (7) is not a problem for the particles of the granular matter themselves, but very well for boundaries or mechanical parts the granular matter interacts with. However, the number of such large bodies is in many applications of practical interest significantly smaller than the number of small-sized particles, suggesting that they can be treated globally. Let $\mathcal{B}_{global}$ be the set of all body indices exceeding the size limit. These bodies will be referred to as being global in the following. All associated state variables and constants shall be instantiated on all processes and initialized equally. The time-integration of these global bodies then can be performed by all processes equally. If a global body i has infinite inertia ($m_i = \infty$ and $\mathbf{I}_{ii}^0 = \infty\mathbf{1}$), such as a stationary wall or a non-stationary vibrating plate, the body velocities are constant, and no wrenches need to be communicated. Global bodies having a finite inertia can be treated by executing an all-reduce communication primitive whenever reducing the wrench or velocity corrections of the small-sized bodies. Instead of only involving neighboring processes, the all-reduce operation sums up the corrections for each global body with finite inertia from all processes and broadcasts the result, not requiring any domain partitioning information.

6.5 Time-integration and synchronization protocol

Having solved the contact problem $\boldsymbol{F}(\boldsymbol{\lambda}) = \mathbf{0}$ by Algorithm 1, the time-integration defined in (3) needs to be performed. If the NBGS implementation uses velocity accumulators, the integrated velocities are at hand after the final communication of the velocity corrections. If instead the NBGS implementation uses wrench accumulators, the wrenches are at hand,

and the velocities of all local bodies can be updated immediately.

Subsequently, the time-integration of the positions can take place. Updating a body's position or orientation effects that the list of shadow copy holders changes since the intersection hull possibly intersects with different subdomains. Also, the body's center of mass can move out of the parent's subdomain. In order to restore the fulfillment of the requirements 1 and 2, a process must determine the new list of shadow copy holders and the new parent process for each local body after the position update. Shadow copy holders must be informed when such shadow copies become obsolete and must be removed. Analogously, processes must be notified when new shadow copies must be added to their state. In this case copies of the corresponding state variables, constants, list of shadow copy holder indices, and index of the parent process must be transmitted.

All other shadow copy holders must obtain the new state variables, list of shadow copy holder indices, and index of the parent process. Hereby, the condition from (7) guarantees that all communication partners are neighbors. All information can be propagated in a single aggregated nearest-neighbor message-exchange. The information should be communicated explicitly and should not be derived implicitly, in order to avoid inconsistencies. This is essential to guarantee that the responsibility of a process to treat a contact can always be determined as well as the responsibility to perform the time integration.

Our implementation of the synchronization protocol makes use of separate container data structures for storing shadow copies and original instances in order to be able to enumerate these different types of bodies with good performance. Both containers support efficient insertion, deletion and lookup operations for handling the fluctuations and updates of the particles efficiently. Furthermore, the determination of the new list of shadow copy holders involves intersection tests between intersection hulls of local bodies and neighboring subdomains as requirement 1 explains in Sect. 6.2. However, determining the minimal set of shadow copy holders is not necessary. Any type of bounding volumes can be used to ease intersection testing. In particular bounding spheres either with tightly fitting bounding radii $\overline{r}_i + h_i(t)$ or even with an overall bounding radius $\max_{i \in \mathcal{B}} \overline{r}_i + h_i(t)$ as proposed by Shojaaee et al. in [36] are canonical. Concerning the geometry of the subdomains at least the subdomain closures can be used for intersection testing. In our implementation we chose to determine almost minimal sets of shadow copy holders by testing the intersections of the actual hull geometries of the particles with the closures of the subdomains. This reduces the number of shadow copies and thus the overall communication volume in exchange for more expensive intersection tests.

```
1: procedure SIMULATETIMESTEP
2:     C_{p,bp} = BROADPHASECOLLISIONDETECTION
3:     C_{p,np} = NARROWPHASECOLLISIONDETECTION(C_{p,bp})
4:     C_p = FILTERCONTACTS(C_{p,np})
5:     INITIALIZEACCUMULATORANDCORRECTIONVARIABLES
6:     k ← 0
7:     λ^[p] ← 0
8:     while convergence criterion not met do
9:         for j ← 1 to ν_c ∧ j ∈ C_p do
10:            λ_j^[p] ← ωF_j^{-1}(0, λ_j̄^[p]) + (1 − ω)λ_j^[p]
11:        end for
12:        REDUCECORRECTIONS
13:        k ← k + 1
14:    end while
15:    INTEGRATESTATEVARIABLES
16:    SYNCHRONIZE
17: end procedure
```

Algorithm 2: A single time step of the simulation on process p.

6.6 Summary

Algorithm 2 summarizes the steps that must be executed on a process p when time-integrating the system for a single time step δt in parallel. The algorithm requires that all shadow copies are instantiated on all subdomains their hull intersects with. Furthermore, the shadow copies must be in synchronized state with the original instance, and the global bodies must also be in sync to each other. The positions of all local bodies must be located within the local subdomain. The time step proceeds by executing the broad-phase contact detection which uses the positions, orientations, shapes, hull expansion radii, and possibly information from previous time steps, in order to determine a set of contact candidates (body pairs) $\mathcal{C}_{p,bp}$ on process p in near-linear time.

Then, in the narrow-phase contact detection, for all candidates the contact location, associated contact frame, and signed contact distance is determined if the hulls actually intersect. Finally, this set of detected contacts $\mathcal{C}_{p,np}$ must be filtered according to one of the rules presented above, resulting in $\mathcal{C}_p$, the set of contacts to be treated by process p. Before entering the iteration of the subdomain NBGS, the accumulator, correction, and contact reaction variables must be initialized. The initialization of the accumulator variables requires an additional reduction step if the external forces or torques cannot be readily evaluated on all processes.

Each iteration of the subdomain NBGS on process p involves a sweep over all contacts to be treated by the process. The contacts are relaxed by a suitable one-contact solver. The $\bar{j}$ indexing indicates that such a solver typically needs to evaluate the relative contact velocity under the assumption that no reaction acts at the contact j. This can be achieved by subtracting out the corresponding part from the accumulator variables. The weighted relaxation result is then stored in

place. The update of the wrench or velocity correction variables is not explicitly listed. After the sweep the wrench or velocity corrections are sent to the respective parent process and summed up per body including the accumulator variables. Then the accumulator variables are redistributed to the respective shadow copy holders in a second message-exchange step.

After a fixed number of iterations or when a prescribed convergence criterion is met, the time step proceeds by executing the time-integration for each local body. The changes of the state variables must then be synchronized in a final message-exchange step, after which the preconditions of the next time step are met. Any user intervention taking place between two time steps needs to adhere to these requirements.

7 Experimental validation of scalability

This section aims to assess the scalability of the parallel algorithms and data strcutures that were presented in Sect. 6. The methods have been implemented in the open-source software framework *pe* for massively parallel simulations of rigid bodies [18,19]. The implementation is based on velocity accumulators and corrections, as introduced in Sect. 6.3. The accumulator initialization performs an additional initial correction reduction step in all experiments.

In Sect. 7.1 the idea behind weak- and strong-scaling experiments is explained before presenting the test problems for which those experiments are executed in Sect. 7.2. The scaling experiments are performed on three clusters whose properties are summarized and compared in Sect. 7.3. Sect. 7.4 points out the fundamental differences in the scalability requirements of the two test problems. Finally, in Sect. 7.5 the weak-scaling and in Sect. 7.6 the strong-scaling results are presented for each test problem and cluster.

7.1 Weak and strong scalability

To demonstrate the scalability of the algorithms and their implementation, we perform weak-scaling experiments, where the problem size is chosen directly proportional to the number of processes and such that the load per process stays constant. Thus, if ideal scaling is achieved, the time to solution will stay constant. Let t_p be the time to solution on p processes, then the parallel efficiency $e_{p,ws}$ in a weak-scaling experiment is defined to be

$$e_{p,ws} = \frac{t_1}{t_p}.$$

In strong-scaling experiments, in contrast, the problem size is kept constant, effecting a decreasing work load per process when increasing the number of processes. Thus, ideally the time to solution on p processes should be reduced by p in comparison to the time to solution on a single process. The speedup s_p on p processes is defined to be

$$s_p = \frac{t_1}{t_p}.$$

The parallel efficiency $e_{p,ss}$ in a strong-scaling experiment is then the fraction of the ideal speedup actually achieved

$$e_{p,ss} = \frac{s_p}{p} = \frac{t_1}{pt_p}.$$

Sometimes speedup and parallel efficiency are also stated with respect to a different baseline, that is, a single central processing unit (CPU) or a single node rather than a single hardware thread or core—the principle remains the same. The parallel efficiency in a weak- and strong-scaling context is a simple performance metric that will serve in the following to assess the quality of the parallelization.

7.2 Test problems

The scalability of the parallelization algorithm as it is implemented in the *pe* framework is validated based on two fundamentally different families of test problems. Sect. 7.2.1 describes a family of dilute granular gas setups whereas Sect. 7.2.2 describes a family of hexagonal close packings of spheres corresponding to structured and dense setups. We chose these setups because their demands towards the implementation vary considerably. This will be analyzed in detail in Sect. 7.4.

7.2.1 Granular gas

Granular material attains a gaseous state when sufficient energy is brought into the system, for example by vibration. Consequently, granular gases feature a low solid volume fraction and are dominated by binary collisions. When the energy supply ceases, the system cools down due to dissipation in the collisions. Granular gases are not only observed in laboratory experiments, but appear naturally for example in planetary rings [37] and in technical applications such as granular dampers [21]. These systems in general exhibit interesting effects like the inelastic collapse [26] or other clustering effects as they e.g. can be observed in the Maxwell-demon experiment [43].

As initial conditions, a rectangular domain with confining walls is chosen. The domain contains a prescribed number of non-spherical particles arranged in a Cartesian grid. Random initial translational velocities are assigned to the particles. All translational velocity components vary uniformly between

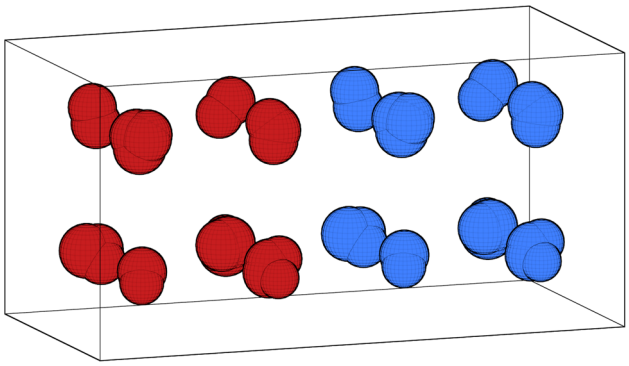

Fig. 4 The granular gas test problem with two processes and $2 \times 2 \times 2$ particles per process

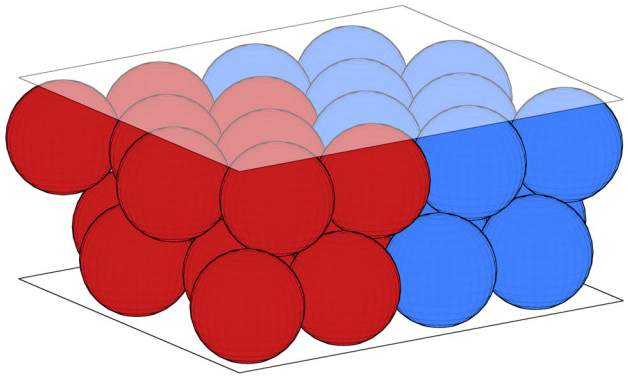

Fig. 5 The *hexagonal* close packing test problem with two processes and two layers of *spherical* particles

−0.2m/s and 0.2m/s. The particles have no initial rotational velocities. The particles are composed of two to four spheres of varying radius, which are arranged at the boundary of the particles' bounding spheres. The particle's bounding sphere has a diameter of 1 cm causing the spherical components of the particle to overlap. The number of spheres per particle varies uniformly and also the radii of the spheres vary uniformly in the range [0.6 cm, 0.8 cm]. Figure 4 illustrates the initial setup with two processes and $2 \times 2 \times 2$ particles per process.

The distance between the centers of two granular particles along each spatial dimension is 1.1 cm, amounting to a solid volume fraction of 23 % on average. In [18] almost the same family of setups served as a scalability test problem. However, there the granular gases had a solid volume fraction of 3.8 % on average. In order to test a higher collision frequency, a denser granular gas was chosen here. The system is simulated for $\frac{1}{10}$ s, and the time step is kept constant at 100 µs, resulting in 1000 time steps in total. Since the contacts are dissipative and no energy is added, the system is quickly cooling down. The coefficient of friction is 0.1 for any contact whether it is a contact between a pair of particles or a contact between a particle and a confining wall.

For this test problem, the subdomain NBGS solver requires a slight underrelaxation in order to prevent divergence. Using an underrelaxation parameter of 0.75 produces good results. For binary collisions, a single iteration of the solver would suffice, but because particles cluster due to the inelastic contacts, more iterations are required. This could be determined by a dynamic stopping criterion, but in the scenario presented here it was found to be more efficient to perform a fixed number of 10 iterations.

For particle simulations, the work load strongly depends on the number of particles and contacts. For the weak-scaling experiments, each process is responsible for a rectangular subdomain, initially containing a fixed number of particles arranged in a Cartesian grid. For the strong-scaling experiments, the total number of particles in x-, y-, and z-dimension should be divisible by the number of processes in x-, y-, and z-dimension that is used in the experiment. With this arrangement the initial load is perfectly balanced. Statistically, the load, that is the number of particles and contacts per subdomain, remains balanced if the subdomains are large enough, and clustering effects have not yet progressed too far. In the simulation performed here, the duration of the simulation was chosen such that the load remains well balanced throughout.

7.2.2 Hexagonal close packing of spheres

The next setup aims to assess the scalability of the parallelization for a dense granular scenario. To demonstrate the scalability, it should be easy and efficient to generate the initial setup for arbitrary problem sizes, and a good load balance should be possible for a longer period of time. Hence, a hexagonal close packing of equal spheres was chosen, for which simple formulas for the position of the spheres are available. The packing density is known to be $\frac{\pi}{3\sqrt{2}} \approx 74.0\,\%$. According to the Kepler conjecture, a hexagonal close packing is the densest possible packing of spheres. To avoid load imbalances at the boundaries, a domain is chosen that is periodic in the x- and y-dimension. In z-dimension the packing is confined by walls that are in direct contact with the spheres on both sides. Assuming an even number of particles in y-direction, the number of contacts is permanently $n_x n_y (6n_z - 1)$ for $n_x \times n_y \times n_z$ particles. The domain is decomposed in x- and y-dimensions only. The objects are subject to gravity. However, the direction of gravity is tilted in the x-z-plane such that the setup corresponds to a ramp inclined by 30° including a lid. The basic setup is illustrated in Fig. 5 on the basis of two processes and two layers of spherical particles. The magnitude of gravity is $9.81\,\mathrm{m/s^2}$. The time step is set to 10 µs and remains constant. The radii of the particles are 1 mm, and their density is $2.65\,\mathrm{g/cm^3}$. All particles are set to an initial downhill velocity of 10 cm/s. The coefficient of friction is 0.85 for contacts between pairs of

Table 1 The test machines used for performing the weak- and strong-scaling experiments

Cluster name	Emmy	SuperMUC	Juqueen
Computing centre	Regional computing centre in Erlangen (RRZE), Germany	Leibniz supercomputing centre (LRZ), Germany	Jülich supercomputing centre (JSC), Germany
Best TOP 500 ranking	–	4th (June 2012)	5th (November 2012)
Peak performance in PFlop/s	0.23	3.2	5.9
Number of nodes	560	9216	28,672
Number of sockets	2	2	1
Name of CPU	Intel Xeon E5-2660 v2	Intel Xeon E5-2680	IBM PowerPC A2
Clock rate in GHz	2.2	2.7	1.6
Number of cores per CPU	10	8	16
Number of threads per core	2	2	4
Total RAM in TiB	35	288	448
Interconnection fabric	Infiniband QDR	Infiniband QDR/ Infiniband FDR 10	BlueGene/Q
Network topology	Non-blocking tree	Non-blocking tree/ 4:1 pruned tree	5D torus

particles and contacts between particles and walls. The high coefficient of friction causes a slip-stick transition shortly after the simulation begins. As in the granular gas setups the subdomain NBGS uses an underrelaxation of 0.75. The solver unconditionally performs 100 iterations in each time step. This intentionally disregards that the iterative solver converges faster for smaller problems. A multigrid solver could possibly remedy the dependence on the problem size, but the successful construction of such a solver needs substantial further research.

7.3 Test machines

In the following all test machines will be introduced. Table 1 summarizes their basic technical characteristics. The Emmy cluster is located at the Regional Computing Centre in Erlangen (RRZE) in Germany which is associated to the Friedrich-Alexander-Universität Erlangen-Nürnberg. The cluster comprises 560 compute nodes. Each node has a dual-socket board equipped with two Xeon E5-2660 v2 processors. Each processor has 10 cores clocked at 2.2 GHz. The processors offer 2-way simultaneous multithreading (SMT). The peak performance of the cluster is 0.23 PFlop/s. Each node is equipped with 64 GiB of random access memory (RAM). The cluster features a fully non-blocking Infiniband interconnect with quad data rate (QDR) and 4× link aggregation, resulting in a bandwidth of 40 Gbit/s per link and direction. In all experiments on the Emmy cluster, each core is associated with a subdomain since preliminary tests showed that we could not take advantage of the SMT features by associating each hardware thread with a subdomain. The Emmy cluster has the smallest peak performance among the test machines and was never among the 500 world's largest commercially available supercomputers. However, it is the machine with the largest non-blocking tree network topology and it has the largest amount of RAM per core.

The second test machine is the SuperMUC supercomputer which is located at the Leibniz Supercomputing Centre (LRZ) in Germany and was best ranked on the 4th place of the TOP 500 list in June 2012. The cluster is subdivided into multiple islands. The majority of the compute power is contributed by the 18 thin-node islands. Each thin-node island consists of 512 compute nodes (excluding four additional spare nodes) connected to a fully non-blocking 648 port FDR10 Infiniband switch with 4× link aggregation, resulting in a bandwidth of 40 Gbit/s per link and direction. Though QDR and FDR10 use the same signaling rate, the effective data rate of FDR10 is more than 20 % higher since it uses a more efficient encoding of the transmitted data. The islands' switches are each connected via 126 links to 126 spine switches. This results in a blocking switch-topology. Thus, if e.g. all nodes within an island send to nodes located in another island, then the 512 nodes have to share 126 links to the spine switches, effecting that the bandwidth is roughly one quarter of the bandwidth that would be available in an overall non-blocking switch-topology. Each (thin) compute node has two sockets, each equipped with an Intel Xeon E5-2680 processor having 8 cores clocked at 2.7 GHz. The processors support 2-way SMT. In the following, as in the case of the Emmy cluster, each core is associated with a single subdomain. The peak performance of the cluster is stated to be 3.2 Pflop/s. Each node offers 32 GiB of RAM, summing up to 288 TiB in total. The SuperMUC supercomputer has an interesting blocking tree network-topology and the processors with the highest clock rate among the processors in the test machines.

The third test machine is the Juqueen supercomputer which is located at the Jülich Supercomputing Centre (JSC)

Table 2 Summary of the domain partitionings used on all test clusters

1D	nodes	$\frac{1}{20}$	$\frac{2}{20}$	$\frac{4}{20}$	$\frac{8}{20}$	$\frac{10}{20}$	$\frac{16}{20}$	1	2	4	8	16	32	64	128	256	512
	p_x	1	2	4	8	10	16	20	40	80	160	320	640	1 280	2 560	5 120	10 240
		⟵ weak-scaling granular gas ⟶															
2D	nodes	$\frac{4}{20}$	$\frac{8}{20}$	$\frac{10}{20}$	$\frac{16}{20}$	1	2	4	8	16	32	64	128	256	512		
	p_x	2	4	5	4	5	8	10	16	20	32	40	64	80	128		
	p_y	2	2	2	4	4	5	8	10	16	20	32	40	64	80		
		⟵ weak-scaling granular gas ⟶															
		⟵ weak-scaling hexagonal close packing ⟶															
		⟵ strong-scaling hexagonal close packing ⟶															
3D	nodes	$\frac{8}{20}$	$\frac{16}{20}$	1	2	4	8	16	32	64	128	256	512				
	p_x	2	4	5	5	5	8	8	10	16	16	20	32				
	p_y	2	2	2	4	4	5	8	8	10	16	16	20				
	p_z	2	2	2	2	4	4	5	8	8	10	16	16				
		⟵ weak-scaling granular gas ⟶															
		⟵ strong-scaling granular gas ⟶															

(a) Domain partitionings used on the Emmy cluster

2D	nodes	1	2	4	8	16	32	64	128	256	512	1 024	2 048	4 096	8 192	16 384	28 672
	p_x	8	16	16	32	32	64	64	128	128	256	256	512	512	1 024	1 024	1 024
	p_y	8	8	16	16	32	32	64	64	128	128	256	256	512	512	1 024	1 792
		⟵ weak-scaling hexagonal close packing ⟶															
		⟵ strong-scaling hexagonal close packing ⟶															
3D	nodes	1	2	4	8	16	32	64	128	256	512	1 024	2 048	4 096	8 192	16 384	28 672
	p_x	4	8	8	8	16	16	16	32	32	32	64	64	64	128	128	128
	p_y	4	4	8	8	8	16	16	16	32	32	32	64	64	64	128	128
	p_z	4	4	4	8	8	8	16	16	16	32	32	32	64	64	64	112
		⟵ weak-scaling granular gas ⟶															
		⟵ strong-scaling granular gas ⟶															

(b) Domain partitionings used on the Juqueen supercomputer

3D	nodes	1	2	4	8	16	32	64	128	256	512	1 024	2 048	4 096	8 192
	p_x	4	4	4	8	8	8	16	16	16	32	32	32	64	64
	p_y	2	4	4	4	8	8	8	16	16	16	32	32	32	64
	p_z	2	2	4	4	4	8	8	8	16	16	16	32	32	32
		⟵ weak-scaling granular gas ⟶													
		⟵ strong-scaling granular gas ⟶													

(c) Domain partitionings used on the SuperMUC supercomputer

in Germany and was best ranked on the 5th place of the TOP 500 list in November 2012. The cluster is a BlueGene/Q system with 28,672 compute nodes since 2013 [14,42]. Each node features a single IBM PowerPC A2 processor having 18 cores clocked at 1.6 GHz, where only 16 cores are available for computing. The processors support 4-way SMT. The Juqueen supercomputer is the only machine, where we associate each hardware thread with a subdomain in the scaling experiments. The machine's peak performance is 5.9 PFlop/s. Each node offers 16 GiB of RAM, summing up to 448 TiB in total. The interconnect fabric is a 5D torus network featuring a bandwidth of 16 Gbit/s per link and direction [8]. The Juqueen supercomputer is the machine with the highest peak performance, the largest number of cores and threads and the only machine among our test machines with a torus interconnect.

Table 2 presents a summary of the domain partitionings used for the scaling experiments on the various clusters. The number of nodes are always a power of two except when using the whole machine or when performing intra-node scalings. The intra-node scaling behaviour is analyzed by means of weak-scaling experiments choosing the granular gas as a test problem and the Emmy cluster as a test machine. The influence of the number of dimensions in which the domain is partitioned is also only analyzed for this configuration. All further scaling tests of the granular

gas scenario use three-dimensional domain partitionings. All inter-node weak-scaling experiments start with a single node and extend to the full machine where possible. The experiments on the SuperMUC supercomputer were obtained at the Extreme Scaling Workshop in July 2013 at the LRZ, where at most 16 islands corresponding to 8192 nodes were available. All strong-scaling experiments start on a single node except on the Juqueen supercomputer, where we chose to start at 32 nodes which is the minimum allocation unit in the batch system on Juqueen. The experiments extend to a number of nodes where a notable efficiency degradation is observed. Since the results on the SuperMUC were obtained well before the other experiments, no scaling experiments with the hexagonal close packing scenario are available.

7.4 Time-step profiles

In this section we clarify how much time is spent in the various phases of the time-step procedure and how this time changes in a weak scaling depending on the test problem. Figure 6 breaks down the wall-clock times of various time step components in two-level pie charts for the granular gas scenario. The times are averaged over all time steps and processes. The dark blue section corresponds to the fraction of the time in a time step used for detecting and filtering contacts. The orange section corresponds to the time used for initializing the velocity accumulators and corrections. The time to relax the contacts is indicated by the yellow time slice. It includes the contact sweeps for all 10 iterations without the correction reductions. The time used by all correction reductions is shown in the green section which includes the reductions for each iteration and the reduction after the initialization. The time slice is split up on the second level in the time used for assembling, exchanging, and processing the first correction reduction message (dark green section) and

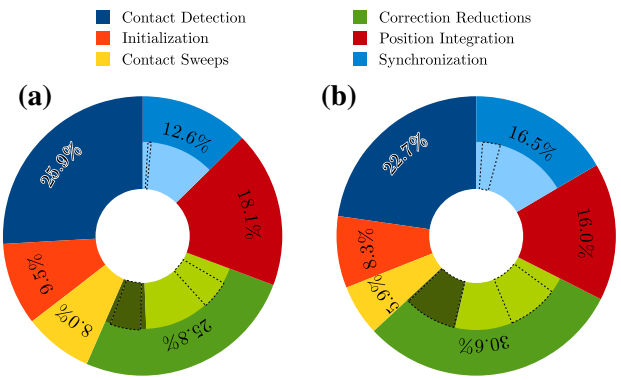

Fig. 6 The time-step profiles for two weak-scaling executions of the granular gas on the Emmy cluster with 25^3 particles per process **a** Time-step profile of the granular gas executed with $5 \times 2 \times 2 = 20$ processes on a single node **b** Time-step profile of the granular gas executed with $8 \times 8 \times 5 = 320$ processes on 16 nodes. (Color figure online)

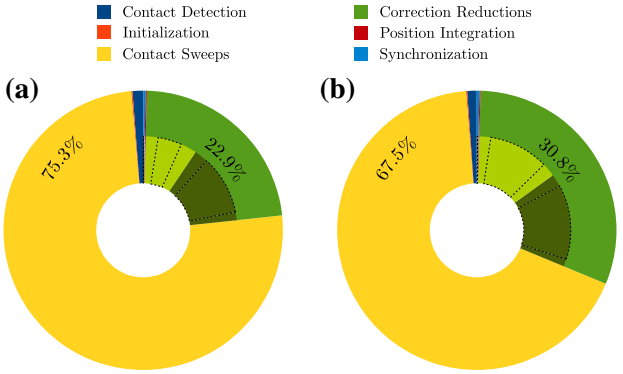

Fig. 7 The time-step profiles for two weak-scaling executions of the *hexagonal* close packing scenario on the Emmy cluster with 10^3 particles per process **a** Time-step profile of the *hexagonal* close packing scenario executed with $5 \times 2 \times 2 = 20$ processes on a single node **b** Time-step profile of the hexagonal close packing scenario executed with $8 \times 8 \times 5 = 320$ processes on 16 nodes. (Color figure online)

the time used for assembling, exchanging, and processing the second correction reduction message (light green section). The time slices are depicted counterclockwise in the given order. The message-exchange communications have a dotted border to distinguish them from the rest. A single message-exchange communication time measurement started, when sending the first message buffer to the neighbors, and ended, when having received the last message buffer from the neighbors. The dark red section corresponds to the time used by the time-integration of the positions, and the final blue section indicates the time used by the position synchronization. The latter is split up into assembling, exchanging, and processing of the message in the inner ring. The message-exchange communication is highlighted by the dashed border again. The first pie chart in Fig. 6a corresponds to the time-step profile of an execution in the weak-scaling experiment with the three-dimensional domain partitioning $5 \times 2 \times 2$ on a single node of the Emmy cluster. Figure 6b shows the time-step profile of an execution in the weak-scaling experiment with the three-dimensional domain partitioning $8 \times 8 \times 5$ on 16 nodes. The two time slices involving communication need more time in comparison to Fig. 6a, especially the framed slices on the second level which amount to the communication. The wall-clock time for the components involving no communication was roughly the same in both runs. The enlarged synchronization time-slices in Fig. 6b then approximately amount to the increased time-step duration on 16 nodes. Overall, computations in the time step of this granular gas scenario prevail. But since the collision frequency is low, the 10 contact sweeps, marked by the yellow and green sections, are dominated by communication.

Figure 7 presents time-step profiles for two weak-scaling executions of the hexagonal close packing scenario. The time-step profiles use the same color coding as in Fig. 6. In

Table 3 Summary of the test problem parameters used for the weak-scaling experiments

	Granular gas			Hexagonal close packing	
	Emmy	Juqueen	SuperMUC	Emmy	Juqueen
Number of particles per process	25^3	10^3	10^3	10^3	10^3
Number of time steps	1000	1000	10,000	1000	100
Maximum number of particles	1.6×10^8	1.8×10^9	1.3×10^8	1.0×10^7	1.8×10^9
Initial number of contacts	0	0	0	6.0×10^7	1.1×10^{10}
Solid volume fraction (%)	23	23	3.8	74	74

contrast to the time-step profiles of the granular gas scenario, the time step is dominated by the 100 contact sweeps (yellow section) and the 100 correction reductions (green section). Contact detection, position integration, and synchronization play a negligible role. In Fig. 7a the time-step profile of a weak-scaling execution with again 20 processes on a single node of the Emmy cluster is presented, whereas in Fig. 7b the time-step profile of a weak-scaling execution with again 320 processes on 16 nodes is shown. The wall-clock time spent in the contact sweep was roughly the same in both executions, hence the increased communication costs are mainly responsible for the larger time slice of the correction reduction.

The time-step profiles showed that for the dilute granular gas scenario the time spent in the various time-step components is well balanced and the time spent in the communication routines moderately increases as the problem size is increased. For the hexagonal close packings most of the time is spent in the contact sweeps and the reduction of the velocity corrections. Components such as the position integration and the final synchronization play a negligible role due to the higher number of iterations in comparison to the granular gas scenario.

7.5 Weak-scaling results

In the following subsections the weak-scaling results for both test problems on the clusters are presented. Table 3 presents an overview of the employed parameters. The experiments differ in terms of the number of particles generated per process depending on the amount of memory available. In order to control the overall wall-clock time, the number of time steps performed varies between 100 and 10,000. All wall-clock times presented in the following subsections correspond to the average wall-clock time needed to perform a single time step per 1000 particles facilitating the comparison of the charts. The wall-clock times exclude the time needed to setup the systems and generate the simulation output. The scaling experiments of the granular gas scenario on SuperMUC differs from the other granular gas experiments in that the gas is considerably more dilute and a longer period of time is simulated.

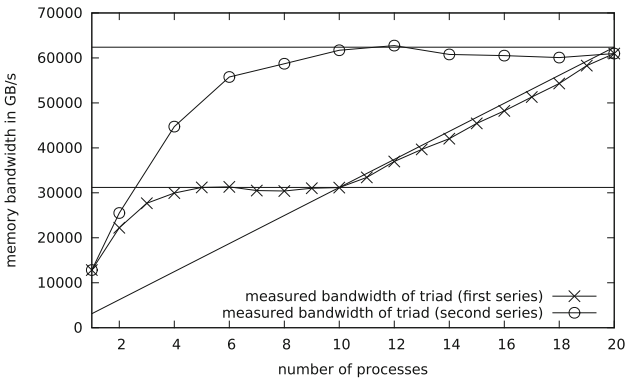

Fig. 8 Measured bandwidth of the triad in the stream benchmark computed with a varying number of cores on a single node of the Emmy cluster

7.5.1 Granular gas

First, we pay special attention to the intra-node weak-scaling before turning to the inter-node weak-scaling since the former is subject to the non-linear scaling behaviour of the memory bandwidth. As a test problem we chose the granular gas scenario and as the test machine the Emmy cluster. A single node in the cluster is equipped with two processors each one having a single on-chip memory controller. The total memory bandwidth available to both sockets is exactly twice the bandwidth of a single socket. However, for a single socket a simple stream benchmark [25] reveals that the memory architecture is designed such that for x cores more than $\frac{1}{x}$ of the socket's total memory bandwidth is available. Figure 8 plots the measured memory bandwidth of computations of the triad as defined in the stream benchmark. The computations are performed by a varying number of cores in parallel. The first series of measurements minimizes the number of sockets in use meaning that the processor affinities are adjusted such that in measurements with less or equal to 10 processes all share the same memory controller. In the second series of measurements, the processes are pinned to the sockets alternately such that $2x$ processes have twice the bandwidth at their disposal as x processes in the first series. Indeed, the lower-left part of the first series' graph very well matches the second series' graph with proper scal-

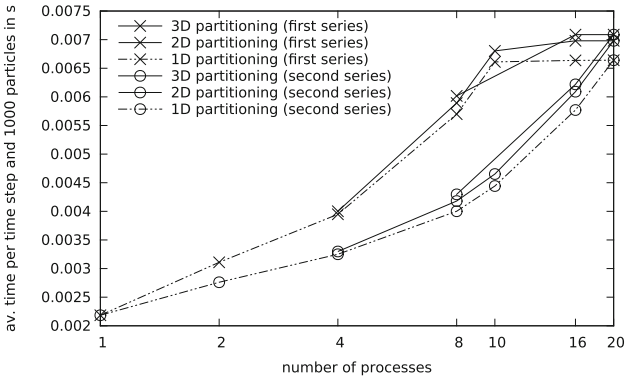

Fig. 9 Intra-node weak-scaling graphs for a granular gas on the Emmy cluster

ing. The measured bandwidth in the first series increases for an increasing number of processes until the available memory bandwidth of the first memory controller is saturated. Measurements with more than 10 processes start to make use of the second memory controller and the measured bandwidth continues to increase linearly.

An analogous behavior can be observed in the intra-node weak-scaling graphs. Figure 9 plots the average wall-clock time needed for a single time step and 1000 particles. In the first series of executions again the pinning strategy minimizing the number of sockets in use is employed. The average wall-clock time needed per time step increases considerably for executions with up to 10 processes. However, beyond that point the weak-scaling graph continues almost ideally. The second series of executions uses the pinning strategy as before for minimizing the maximum number of processes per socket. This means that executions with $2x$ processes in the second series have twice the bandwidth at their disposal as the executions with x processes in the first series. The graphs of the second series show that the wall-clock times needed per time step on $2x$ processes indeed closely match the wall-clock times on x processes in the first series. This indicates that our implementation is limited by the available memory bandwidth.

The figure also distinguishes between weak-scaling graphs with one-, two-, and three-dimensional domain partitionings since their communication volumes differ. Higher-dimensional non-periodic domain partitionings have typically a higher communication volume in comparison to lower dimensional non-periodic domain partitionings with the same number of processes, due to the larger area of the interfaces between the subdomains. The plotted timings for the one-dimensional domain partitionings are indeed consistently slightly better than the timings for two-dimensional domain partitionings, which are in turn slightly better than the timings for three-dimensional domain partitionings.

Even though the intra-node weak-scaling results reveal an underperforming parallel efficiency between 30.8 and 32.9 %

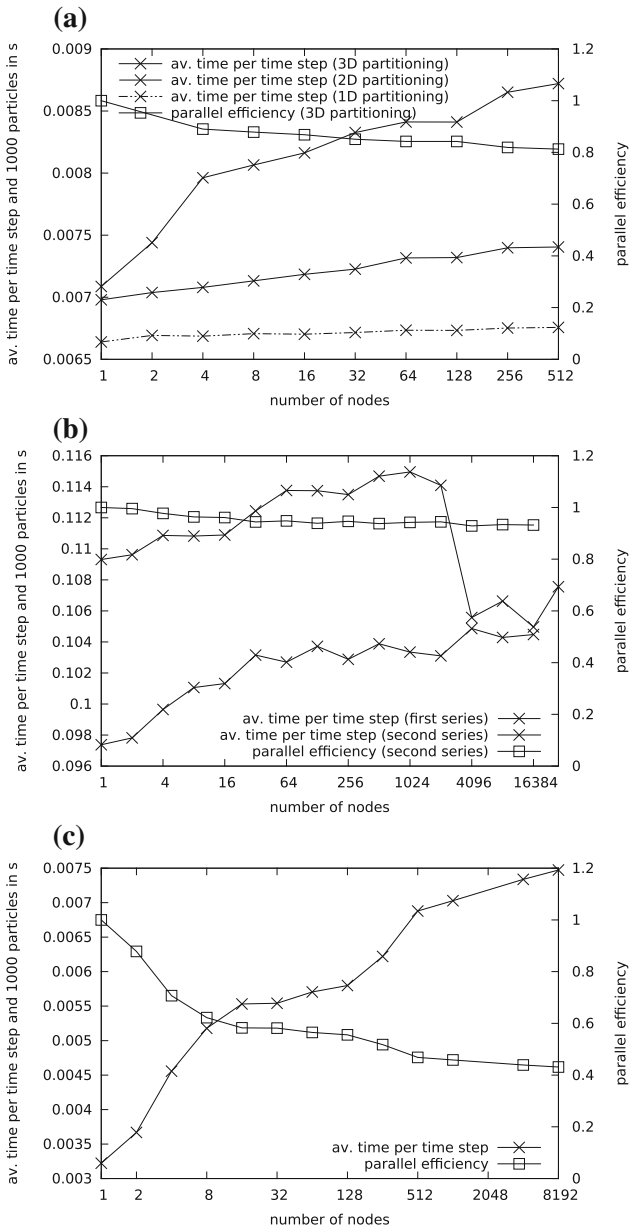

Fig. 10 Inter-node weak-scaling graphs for a granular gas on all test machines **a** Weak-scaling graph on the Emmy cluster **b** Weak-scaling graph on the Juqueen supercomputer **c** Weak-scaling graph on the SuperMUC supercomputer

when computing on all cores of an Emmy node, the correlation with the measured memory bandwidth of a triad suggests that a good intra-node scaling can be expected as long as the available bandwidth scales. With corresponding pinning, this is the case as off the first full socket on the Emmy cluster.

Figure 10a extends the weak-scaling experiment to almost the full Emmy cluster for one-, two-, and three-dimensional domain partitionings. The scaling experiment for the one-dimensional domain partitionings performs best and achieves on 512 nodes a parallel efficiency of 98.3 % with respect to the

single-node performance. The time measurements for two-dimensional domain partitionings are consistently slower, but the parallel efficiency does not drop below 89.7 %. The time measurements for three-dimensional domain partitionings come in last, and the parallel efficiency goes down to 76.1 % for 512 nodes. This behaviour can be explained by the differences in the communication volumes of one-, two-, and three-dimensional domain partitionings. The results attest that the problem can be efficiently scaled (almost) up to the full machine if the load per process is sufficiently large.

Figure 10b shows the results of the inter-node weak-scaling experiments on the Juqueen supercomputer. The scaling experiments are only performed with the more demanding three-dimensional domain partitionings. In the first series of measurements the average wall-clock time per time step increases as expected up to 2048 nodes. But then the average time-step duration for setups with 4096 nodes and beyond is significantly shorter than the average time-step duration with fewer nodes. The time steps are even computed faster than on a single node, where no inter-node communication takes place at all. Assuming that intra-node communication is faster than inter-node communication, this is a puzzling result. In fact, it turns out that the intra-node communication is responsible for the behaviour: The default mechanism for intra-node communication is via shared memory on the Juqueen. In the second series of measurements we disallow the usage of shared memory for intra-node communication. This results in the measurements that are consistently faster than the measurements from the first series, and the parallel efficiency is now essentially monotonically decreasing with an excellent parallel efficiency of at least 92.9 %.

The reason why the measured times in the first series become shorter for 4096 nodes and more is revealed when considering how the processes get mapped to the hardware. The default mapping on Juqueen is ABCDET, where the letters A to E stand for the five dimensions of the torus network, and T stands for the hardware thread within each node. The six-dimensional coordinates are then mapped to the MPI ranks in a row-major order, that is, the last dimension increases fastest. The T coordinate is limited by the number of processes per node, which is 64 for the above measurements. Upon creation of a three-dimensional communicator, the three dimensions of the domain partitioning are mapped also in row-major order. If the number of processes in z-dimension is less than the number of processes per node, this has the effect that a two-dimensional or even three-dimensional section of the domain partitioning is mapped to a single node. However, if the number of processes in z-dimension is larger or equal to the number of processes per node, only a one-dimensional section of the domain partitioning is mapped to a single node. A one-dimensional section of the domain partitioning performs considerably less intra-node communication than a two- or three-dimensional section of the domain partitioning. This matches exactly the situation for 2048 and 4096 nodes. For 2048 nodes, a two-dimensional section $1 \times 2 \times 32$ of the domain partitioning $64 \times 64 \times 32$ is mapped to each node, and for 4096 nodes a one-dimensional section $1 \times 1 \times 64$ of the domain partitioning $64 \times 64 \times 64$ is mapped to each node. To substantiate this claim, we confirmed that the performance jump occurs when the last dimension of the domain partitioning reaches the number of processes per node, also when using 16 and 32 processes per node.

Figure 10c presents the weak-scaling results on the SuperMUC supercomputer. The setup differs from the granular gas scenario presented in Sect. 7.2.1 in that it is more dilute. The distance between the centers of two granular particles along each spatial dimension is 2cm, amounting to a solid volume fraction of 3.8 % and consequently to fewer collisions. As on the Juqueen supercomputer only three-dimensional domain partitionings are used. All runs on up to 512 nodes were running within a single island. The run on 1024 nodes also used the minimum number of 2 islands. The run on 4096 nodes used nodes from 9 islands, and the run on 8192 nodes used nodes from 17 islands, that is both runs used one island more than required. The graph shows that most of the performance is lost in runs on up to 512 nodes. In these runs only the non-blocking intra-island communication is utilised. Thus this part of the setup is very similar to the Emmy cluster since it also has dual-socket nodes with Intel Xeon E5 processors and a non-blocking tree Infiniband network. Nevertheless, the intra-island scaling results are distinctly worse. The reasons for these differences were not yet fully investigated. However, the scaling behaviour beyond a single island can be considered satisfactory, featuring a parallel efficiency of 73.8 % with respect to a single island. A possible explanation of the underperforming intra-node scaling behaviour could be that during the tests some of the Infiniband links were degraded to QDR, which was a known problem at the time the extreme-scaling workshop took place. The communication routines then need $\frac{5 \cdot 64}{4 \cdot 66} \approx 1.21$ times longer to complete. This could also explain the high variability of the runs' wall-clock times.

Subsequently, a second series of measurements is performed with 60^3 non-spherical particles per process. The scaling behaviour is comparable to the scaling behaviour observed in Fig. 10c. However, the largest weak-scaling run simulated $28,311,552,000 \approx 2.8 \cdot 10^{10}$ non-spherical particles—possibly a record-breaking number for non-smooth contact dynamics.

7.5.2 Hexagonal close packings of spheres

Figure 11a shows the average wall-clock time needed for a single time step in the hexagonal close packing test on the

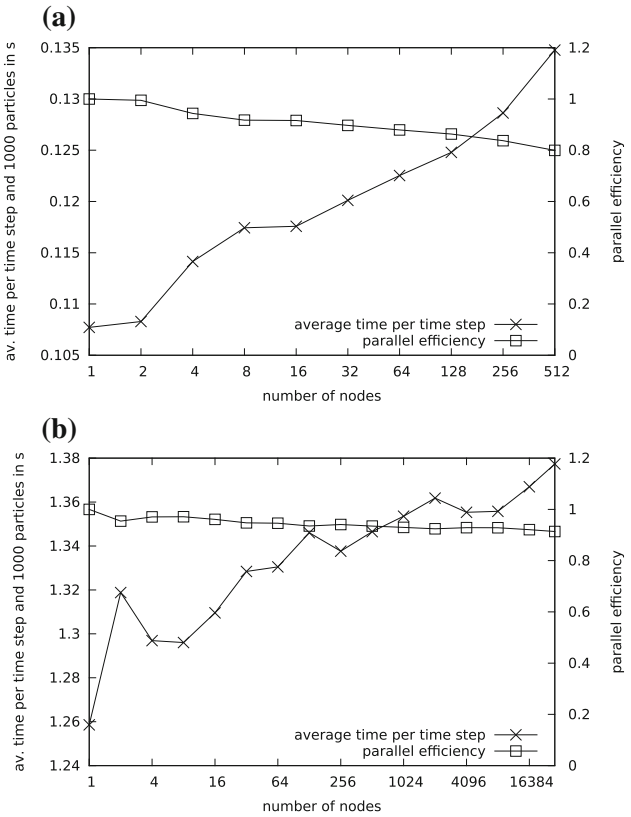

Fig. 11 Inter-node weak-scaling graphs for *hexagonal* close packings of *spheres* **a** Weak-scaling graph on the Emmy cluster **b** Weak-scaling graph on the Juqueen supercomputer

Emmy cluster. The parallel efficiency with respect to a single node remains above 79.9 % for all executions. This is slightly better than the parallel efficiency of 76.1 % for the granular gas.

The weak-scaling results of the hexagonal close packing scenario on the Juqueen supercomputer are presented in Fig. 11b. The parallel efficiency with respect to a single node stays above 91.4 % for all measurements. This result is almost as good as the 92.9 % parallel efficiency in the scaling experiments of the granular gas. The largest execution exercises $1024 \times 1792 \times 1 = 1{,}835{,}008$ processes on all 28,672 nodes of the machine, where $10{,}240 \times 17,\ 920 \times 10 = 1{,}835{,}008{,}000$ particles are spawned, in total leading to $10{,}826{,}547{,}200 \approx 1.1 \cdot 10^{10}$ contacts – again a possibly record-breaking number for non-smooth contact dynamics.

7.6 Strong-scaling results

In the following subsections the strong-scaling results for both test problems on the clusters are presented. Table 4 gives an overview of the employed parameters. The experiments differ in terms of the number of particles generated in total and the number of time steps used for averaging. As in the weak-scaling experiments the granular gas scenario on SuperMUC is more dilute than on the other machines.

7.6.1 Granular gas

Figure 12 presents the strong-scaling results of the granular gas scenario on all clusters. The strong-scaling graph on the Emmy cluster is presented in Fig. 12a. A total of $320 \times 160 \times 160 = 8{,}192{,}000$ particles is used, leading to at most $64 \times 80 \times 80 = 409{,}600$ particles per process on a single node and at least $10 \times 8 \times 10 = 800$ particles per process on 512 nodes. The speedup is ideal for up to 64 nodes and then the simulation becomes gradually less efficient. Some time measurements exceed the optimal speedup, which can happen for example if the problem becomes small enough to fit into one of the caches. In conclusion, the scaling experiments for this dilute setup on the Emmy cluster suggest that one obtains a satisfactory parallel efficiency on the whole cluster, as long as several thousand particles are allocated to each process.

Figure 12b presents the results of the strong-scaling experiments on the Juqueen supercomputer for the granular gas. The total number of particles was 32,768,000 particles. In the execution on 32 nodes each of the $16 \times 16 \times 8 = 2048$ processes initially had $20 \times 20 \times 40 = 16{,}000$ non-spherical particles, and in the execution on 4096 nodes each of the $64 \times 64 \times 64 = 262{,}144$ processes spawned $5 \times 5 \times 5 = 125$ particles. The parallel efficiency is plotted with respect to 32 nodes and stays above 80.7 % for up to 1024 nodes and 500 particles per process before rapidly decreasing. On 4096 nodes the efficiency is at 55.4 %. The weak- and strong-scaling results are both better in comparison to the Emmy

Table 4 Summary of the test problem parameters used for the strong-scaling experiments

	Granular gas			Hexagonal close packing	
	Emmy	Juqueen	SuperMUC	Emmy	Juqueen
Number of particles	320 × 160 × 160	320 × 320 × 320	128 × 128 × 128	1280 × 640 × 10	2048 × 2048 × 10
Number of time steps	1000	1000	100	50	20
Solid volume fraction (%)	23	23	3.8	74	74

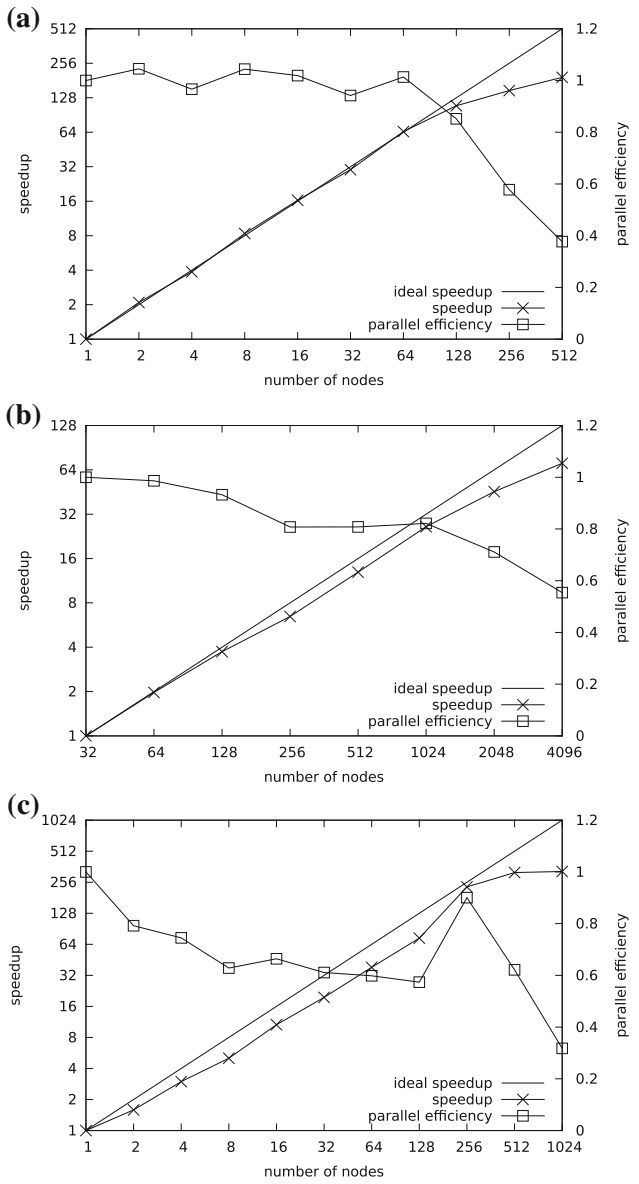

Fig. 12 Strong-scaling graphs for a granular gas test problem on all test machines **a** Strong-scaling graph on the Emmy cluster **b** Strong-scaling graph on the Juqueen supercomputer **c** Strong-scaling graph on the SuperMUC supercomputer

cluster, owed to the torus network which shows excellent performance for the nearest-neighbor communication.

The results of the strong-scaling experiments on the SuperMUC supercomputer are shown in Fig. 12c. In total 128^3 non-spherical particles are simulated. Hence, in the single-node run each process owns $32 \times 64 \times 64 = 131{,}072$ particles, and in the run on 1024 nodes, each process owns $8 \times 4 \times 4 = 128$ particles. The parallel efficiency is at 90.0 % on 256 nodes. Beyond that point it decreases dramatically, indicating that the scaling is good as long as at least about 500 particles are present per process.

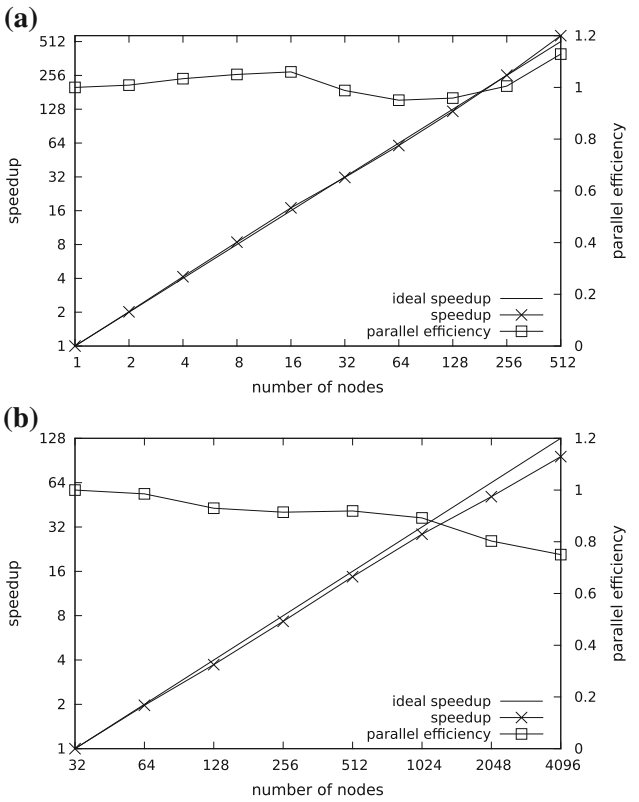

Fig. 13 Strong-scaling graphs for hexagonal close packings of spheres **a** Strong-scaling graph on the Emmy cluster **b** Strong-scaling graph on the Juqueen supercomputer

7.6.2 Hexagonal close packings of spheres

In the strong-scaling experiment on the Emmy cluster in total $1280 \times 640 \times 10 = 8{,}192{,}000$ particles were generated. The experiment was run for 1 to 512 nodes, such that the smallest setup with $5 \times 4 \times 1 = 20$ processes on a single node generated $256 \times 160 \times 10 = 409{,}600$ spherical particles per process, and the largest setup with $128 \times 80 \times 1 = 10{,}240$ processes on 512 nodes generated $10 \times 8 \times 10 = 800$ particles per process. Figure 13a presents the results. A super-linear speedup is observed for several benchmark runs, which is likely due to caching effects, since the working set size becomes very small on each core. In the strong-scaling experiment for 512 nodes of the granular gas scenario on Emmy also only 800 particles are generated per process. However, the computational intensity here is much higher in comparison to that of the granular gas, because far more contacts msut be resolved. This explains the high parallel efficiency of 113 % in comparison to the disappointing parallel efficiency of 37.7 % from Fig. 12a. In conclusion, the scaling experiments suggest that a few hundred particles per process are enough to achieve a very good parallel efficiency on the Emmy cluster if the granular material is dense.

For the strong-scaling experiment on the Juqueen supercomputer a hexagonal close packing with 41,943,040 particles in total is created. The smallest execution runs $64 \times 32 \times 1 = 2048$ processes on 32 nodes, where $32 \times 64 \times 10 = 20,480$ spherical particles are generated per process. The largest execution runs $512 \times 512 \times 1 = 262,144$ processes on 4096 nodes, where $4 \times 4 \times 10 = 160$ particles are generated per process. Figure 13b shows the speedup and the parallel efficiency on the second axis, both with respect to 32 nodes. A parallel efficiency of 75.0 % on 4096 nodes is achieved, where only 160 particles were owned per process. This suggests that a reasonable good efficiency can be achieved for a dense setup on the Juqueen supercomputer, as long as several hundred particles are handled per process.

8 Related work

Other authors have proposed approaches for parallelizing non-smooth contact dynamics on architectures with distributed memory. All of them are based on domain partitionings. A parallelization strategy termed non-smooth contact domain decomposition (NSCDD) implemented in the renowned LMGC90 code was lately presented in [40,41] by Visseq et al. The approach is inspired by the finite element tearing and interconnect (FETI) method for solving partial differential equations in computational mechanics. The authors suggest to decouple the multi-contact problem such that on each process a multi-contact problem is solved having the same structure as a multi-contact problem that is solved sequentially. Particles with multiple contacts that are associated with different subdomains are duplicated, similar to shadow copies used in this article. However, the mass and inertia are split among all instantiations. The coupling is recovered by adding linear equations gluing the duplicates back together through additional Lagrange multipliers. In contrast to the contact constraints, the interface equations are linear, and a block-diagonal system of linear equations must be solved after several sweeps over all contacts. In [41], the authors present simulations with up to $2 \cdot 10^5$ spherical particles and $2 \cdot 10^6$ contacts, time-integrated on up to 100 processes. The NSCDD allows non-nearest-neighbor communication in order to allow enlarged rigid bodies instead of introducing a concept analogous to global bodies.

Prior to Visseq et al., Koziara et al. presented the parallelization implemented in the solfec code [22]. This approach dispenses with the separation into interface problems and local multi-contact problems. A classic NBGS is parallelized with a non-negligible but inevitable amount of serialization. Bodies are instantiated redundantly on all processes, prohibiting scaling beyond the memory limit. Instead of using accumulator and correction variables, as proposed in this paper, the authors synchronize dummy particles (particles that are in contact with shadow copies or original instances) in addition to shadow copies in order to implement contact shadow copies. As in the NSCDD, the system matrix (Delassus operator) is set up explicitly instead of using matrix-free computations as proposed here. Simulations are presented with up to $1 \cdot 10^4$ polyhedral particles or $6 \cdot 10^5$ contacts time-integrated on up to 64 processes.

At the same time, Shojaaee et al. presented another domain partitioning method in [36]. The presentation is restricted to two-dimensional problems. The solver in the paper corresponds to a subdomain NBGS with relaxation parameter $\omega = 1$, where the authors argue that divergence does typically not occur. At least for three-dimensional simulations this is in our experience not sufficient. Shadow copies are created not only if the hulls overlap the neighboring subdomain but also if the particles approach the subdomain boundaries, simplifying the intersection testing but introducing excessive shadow copies. Shojaaee et al. also introduce contact shadow copies instead of using accumulator and correction variables as proposed here. Simulations are presented with up to $1 \cdot 10^6$ circular particles in a dense packing on up to 256 processes.

The approach presented in this paper improves in general the robustness and scalability of previously published parallel algorithms. The matrix-free approach facilitates the evaluation of the particle wrenches in parallel as suggested in Sect. 6.3 and thus reduces the amount of communicated data. The separation of bodies into global and local bodies allows to restrict message-exchange communications to nearest neighbors as detailed in Sect. 6.4 and thus maps well to various interconnect networks. Furthermore, the synchronization protocol defined in Sect. 6.2 and Sect. 6.5 is not susceptible to numerical errors in contrast to the conventional rules which are based on contact locations. Last but not least the scaling experiments from Sect. 7 with up to $2.8 \cdot 10^{10}$ non-spherical particles or $1.1 \cdot 10^{10}$ contacts on up to $1.8 \cdot 10^6$ processes exceed all previously published numbers by a factor of 10^3–10^4.

9 Summary

This article presents models and algorithms for performing scalable direct numerical simulations of granular matter in hard contact as we implemented them in the *pe* open-source software framework for massively parallel simulations of rigid bodies. The *pe* framework already has been successfully used to simulate granular systems with and without surrounding fluid in the past [6,12]. Excellent scaling has also been achieved in a fluid-structure interaction context [15,16].

The discretization of the equations of motion underlying the time-stepping scheme uses an integrator of order

one. Contacts are modelled as inelastic and hard contacts with Coulomb friction. The hard contact model avoids the necessity to resolve the collision micro-dynamics and the time-stepping scheme avoids the necessity to resolve impulsive events in time. The one-step integration can be split into the integration of the velocities and the subsequent integration of the positions and orientations.

The velocity integration requires the solution of a non-linear system of equations per time step. In order to reduce the size of the system in the first place conventional broad-phase contact detection algorithms are applied to exclude contacts between intersection hulls. To solve the non-linear system of equations the subdomain non-linear block Gauss-Seidel is used. The numerical solution algorithm is a mixture between a non-linear block Gauss-Seidel (NBGS) and a non-linear block Jacobi with underrelaxation. In contrast to a pure non-linear block Jacobi it only requires a mild underrelaxation and in contrast to a non-linear block Gauss-Seidel it accommodates the subdomain structure of the domain partitioning and thus allows an efficient parallelization avoiding irregular data dependencies across subdomains. The implementation of the subdomain NBGS in the *pe* is matrix-free and thus avoids the expensive assembly of the Delassus operator. Furthermore, the use of accumulators and correction variables enables the evaluation of the particle wrenches in parallel, reuses partial results and reduces the number of particles that need to be synchronized.

The integration of the positions and orientations is entailed by the execution of a robust synchronization protocol that guarantees correctness while being highly efficient when proceeding to ultra large scale. The key to obtain this robustness is to add the rank of the parent process and the ranks of the shadow copy holders to the state of each particle and to explicitly communicate the state changes. Only then processes can reliably agree upon responsibilities such as contact treatment and particle integration without being susceptible to numerical errors.

Beyond that, all messages are aggressively aggregated in order to reduce the communication overhead of small messages and all messages are restricted to nearest neighbors. The latter is achieved by splitting bodies into local and global bodies and identifying appropriate requirements. Both measures improve the scalability of the implementation.

Finally, the scalability was demonstrated for dilute and dense setups on three clusters, two of them having been in the top 10 of the world's largest publicly available supercomputers. The parallel efficiency on Juqueen is excellent. The inter-island scaling results on SuperMUC are satisfactory, however, the intra-island scaling results show room for possible improvements. This is not inherently caused by the parallelization approach, as can be shown by inspecting the results of the Emmy cluster, whose architecture is similar to a single island of SuperMUC.

The largest scaling experiments demonstrate that simulations of unprecedented scale with up to $2.8 \cdot 10^{10}$ non-spherical particles and up to $1.1 \cdot 10^{10}$ contacts are possible using up to $1.8 \cdot 10^{6}$ processes. The systematic evaluation also confirms that good parallel efficiency can be expected on millions of processes even if only a few hundred particles are allocated to each process provided that the computation exhibits a sufficiently high computational intensity and the architecture has a good interconnect network.

The favourable scalability results do not account for the fact that the NBGS solver may not scale (algorithmically) in terms of the number of iterations needed to achieve a given error bound when large ensembles of particles are in mutual contact. Possible future developments arise out of this: In such situations, the convergence rate of multigrid methods can still be independent of the number of unknowns and is in that sense optimal. The successful construction of a multigrid method for hard contact problems would be invaluable for simulating every-increasing system sizes.

Acknowledgments The second author gratefully acknowledges the support of the Institute of Mathematical Sciences of the National University of Singapore.

References

1. Anitescu M, Potra F (1997) Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dynam 14(3):231–247
2. Anitescu M, Potra F (2002) A time-stepping method for stiff multibody dynamics with contact and friction. In J Numer Methods Eng 55(7):753–784
3. Anitescu M, Tasora A (2010) An iterative approach for cone complementarity problems for nonsmooth dynamics. Comput Optim Appl 47(2):207–235
4. van den Bergen G (1999) A fast and robust GJK implementation for collision detection of convex objects. J Gr Tools 4(2):7–25
5. van den Bergen G (2001) Proximity queries and penetration depth computation on 3D game objects. In: Game developers conference, vol 170
6. Bogner S, Mohanty S, Rüde U (2015) Drag correlation for dilute and moderately dense fluid-particle systems using the lattice boltzmann method. Int J Multiph Flow 68:71–79
7. Bonnefon O, Daviet G (2011) Quartic formulation of Coulomb 3D frictional contact. Technical Report RT-0400, INRIA
8. Chen D, Eisley N, Heidelberger P, Senger R, Sugawara Y, Kumar S, Salapura V, Satterfield D, Steinmacher-Burow B, Parker J (2012) The IBM Blue Gene/Q interconnection fabric. IEEE Micro 32(1):32–43
9. Cohen J, Lin M, Manocha D, Ponamgi M (1995) I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In: Proceedings of the 1995 symposium on interactive 3D graphics, ACM, p 189
10. Diebel J (2006) Representing attitude: Euler angles, unit quaternions, and rotation vectors. Matrix 58:15–16
11. Esefeld B (2014) Numerische Integration von Mehrkörpersystemen mit mengenwertigen Kraftgesetzen. Herbert Utz Verlag, München

12. Fischermeier E, Bartuschat D, Preclik T, Marechal M, Mecke K (2014) Simulation of a hard-spherocylinder liquid crystal with the pe. Comput Phys Commun 185(12):3156–3161
13. Gilbert E, Johnson D, Keerthi S (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Autom 4(2):193–203
14. Gilge M et al (2013) IBM system Blue Gene solution Blue Gene/Q application development. IBM Redbooks, Durham
15. Götz J, Iglberger K, Feichtinger C, Donath S, Rüde U (2010) Coupling multibody dynamics and computational fluid dynamics on 8192 processor cores. Parallel Comput 36(2–3):142–151
16. Götz J, Iglberger K, Stürmer M, Rüde U (2010) Direct numerical simulation of particulate flows on 294912 processor cores. In: Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis, pp 1–11
17. Iglberger K, Rüde U (2009) Massively parallel rigid body dynamics simulations. Comput Sci Res Dev 23(3–4):159–167
18. Iglberger K, Rüde U (2010) Massively parallel granular flow simulations with non-spherical particles. Comput Sci Res Dev 25(1–2):105–113
19. Iglberger K, Rüde U (2011) Large-scale rigid body simulations. Multibody Syst Dynam 25(1):81–95
20. Jean M (1999) The non-smooth contact dynamics method. Comput Methods Appl Mech Eng 177(3–4):235–257
21. Kollmer J, Sack A, Heckel M, Pöschel T (2013) Relaxation of a spring with an attached granular damper. New J Phys 15(9):093,023
22. Koziara T, Bićanić N (2011) A distributed memory parallel multibody contact dynamics code. Int J Numer Methods Eng 87(1–5):437–456
23. Leyffer S (2006) Complementarity constraints as nonlinear equations: theory and numerical experience. In: Optimization with multivalued mappings, Springer, pp 169–208
24. Liu C, Jain S (2012) A quick tutorial on multibody dynamics. Tech. rep., Georgia institute of technology
25. McCalpin J (1995) Memory bandwidth and machine balance in current high performance computers. In: IEEE computer society technical committee on computer architecture (TCCA) newsletter pp 19–25
26. McNamara S, Young W (1994) Inelastic collapse in two dimensions. Phys Rev E 50(1):R28–R31
27. Miller S, Luding S (2004) Event-driven molecular dynamics in parallel. J Comput Phys 193(1):306–316
28. Mitarai N, Nakanishi H (2012) Granular flow: dry and wet. Eur Phys J Spec Top 204(1):5–17
29. Moreau J, Panagiotopoulos P (1988) Nonsmooth mechanics and applications, vol 302. Springer, Wien-New York
30. Negrut D, Tasora A, Mazhar H, Heyn T, Hahn P (2012) Leveraging parallel computing in multibody dynamics. Multibody Syst Dynam 27(1):95–117
31. Popa C, Preclik T, Rüde U (2014) Regularized solution of LCP problems with application to rigid body dynamics. Numer Algorithm 67:1–12
32. Sauer J, Schömer E (1998) A constraint-based approach to rigid body dynamics for virtual reality applications. In: Proceedings of the ACM symposium on virtual reality software and technology, pp 153–162
33. Schindler T, Acary V (2014) Timestepping schemes for nonsmooth dynamics based on discontinuous Galerkin methods: definition and outlook. Math Comput Simul 95:180–199
34. Schütte K, van der Waerden B (1952) Das Problem der dreizehn Kugeln. Mathematische Annalen 125(1):325–334
35. Shen Y, Stronge W (2011) Painlevé paradox during oblique impact with friction. Eur J Mech A/Solids 30(4):457–467
36. Shojaaee Z, Shaebani M, Brendel L, Török J, Wolf D (2012) An adaptive hierarchical domain decomposition method for parallel contact dynamics simulations of granular materials. J Comput Phys 231(2):612–628
37. Spahn F, Petzschmann O, Schmidt J, Sremčević M, Hertzsch JM (2001) Granular viscosity, planetary rings and inelastic particle collisions. In: Granular gases, Springer, Berlin, pp 363–385
38. Studer C (2009) Numerics of unilateral contacts and friction: modeling and numerical time integration in non-smooth dynamics, Lecture Notes in Applied and Computational Mechanics, vol 47. Springer, Berlin
39. Tasora A, Anitescu M (2011) A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. Comput Methods Appl Mech Eng 200(5):439–453
40. Visseq V, Martin A, Dureisseix D, Dubois F, Alart P (2012) Distributed nonsmooth contact domain decomposition (NSCDD): algorithmic structure and scalability. In: Proceedings of the international conference on domain decomposition methods
41. Visseq V, Alart P, Dureisseix D (2013) High performance computing of discrete nonsmooth contact dynamics with domain decomposition. Int J Numer Methods Eng 96(9):584–598
42. Wautelet P, Boiarciuc M, Dupays J, Giuliani S, Guarrasi M, Muscianisi G, Cytowski M (2014) Best practice guide—Blue Gene/Q. v1.1.1 edn
43. van der Weele K, van der Meer D, Versluis M, Lohse D (2001) Hysteretic clustering in granular gas. Europhys Lett 53(3):328