



ELSEVIER

Available online at www.sciencedirect.com



Computer Physics Communications 166 (2005) 26–44

Computer Physics
Communications

www.elsevier.com/locate/cpc

An implementation of granular dynamics for simulating frictional elastic particles based on the DL_POLY code

Meenakshi Dutt ^{a,*}, Bruno Hancock ^c, Craig Bentham ^b, James Elliott ^a

^a *Pfizer Institute for Pharmaceutical Materials Science, Department of Materials Science and Metallurgy, University of Cambridge, Pembroke Street, Cambridge, CB2 3QZ, UK*

^b *Pfizer Global Research and Development, Sandwich, Kent, CT13 9NJ, UK*

^c *Pfizer Global Research and Development, Groton, CT 06340, USA*

Received 30 August 2004; accepted 21 October 2004

Available online 8 December 2004

Abstract

We have modified Daresbury Laboratory's replicated data strategy (RDS) parallel molecular dynamics (MD) package DL_POLY (version 2.13) to study the granular dynamics of frictional elastic particles. DL_POLY [Smith and Forester, The DL_POLY_2 User Manual v2.13, 2001; Forester and Smith, The DL_POLY_2 Reference Manual v2.13, 2001] is a MD package originally developed to study liquid state and macromolecular systems by accounting for various molecular interaction forces. The particles of interest in this study are macroscopic grains in pharmaceutical powders, with sizes ranging from tens to hundreds of microns. We have therefore substituted the molecular interaction forces with *contact* forces (including linear-dashpot, HKK interaction forces and Coulombic friction) while taking advantage of the RDS scheme. In effect, we have created a parallel Discrete Element Simulation (DES) code. In this paper, we describe the modifications made to the original DL_POLY code and the results from the validation tests of the granular dynamics simulations for systems of monodisperse spherical particles settling under gravity. The code can also be utilized to study particle packings generated via uniaxial compaction and, in some cases, simultaneous application of shear, at constant strain.

© 2004 Elsevier B.V. All rights reserved.

PACS: 81.05.R; 81.20.E

Keywords: Granular materials; Discrete element simulations; Granular dynamics; Particle packings; Pharmaceutical powders; DL_POLY

* Corresponding author. Tel.: +44 1223 767059, fax: +44 1223 767063.
E-mail address: md336@cus.cam.ac.uk (M. Dutt).

1. Introduction

Granular materials are found everywhere around us—common examples being salt, sugar, sand, rocks, agricultural grain among many others. As a result, they have applications in various industries: pharmaceutical, construction, agriculture, food and cosmetics. Despite their ubiquity, the properties of these materials are still not very well understood due to their ability to demonstrate behavior associated with solids, and fluids, depending upon the external forces applied [3]. An example of the multiphase behavior of granular materials is a vibrofluidized sand bed. Depending upon the driving acceleration, the sand will remain compacted (*solid phase*), or be a partially or completely fluidized sand bed (*fluid phase*); in the former case, the fluidized boundary layer sits on a compacted layer of sand particles [3,4]. Granular materials are loosely defined as agglomerates of particles whose sizes may range from microns to tens of metres [5]. Granular materials can be of two types: dry non-cohesive or cohesive agglomerates of particles. Interactions for the latter are determined by constituent particle sizes along with the presence of interstitial fluid of non-negligible viscosity or surface tension, which results in force contributions from van der Waals forces or liquid bridges (produced by surface tension due to presence of a thin liquid film on the surface of the grains). We are interested in studying dry non-cohesive granular materials where the interstitial fluid plays a negligible role in determining the behavior of the system. The constituent particles interact solely via inelastic mechanical contacts. The inelasticity of the contacts arises primarily due to the existence of surface friction, which couples the translational and rotational degrees of freedom.

The pharmaceutical industry daily handles large volumes of granular materials, in the form of pharmaceutical powder blends, for the production of drug tablets that are currently one of the most common drug delivery forms [6,7]. The tablets are generated by double-ended uniaxial compaction of the pharmaceutical blend [6,7]. The pharmaceutical blends are a complex mixture of the drug powder along with excipients, such as binders, disintegrants or diluents. During the uniaxial compaction process, the pharmaceutical powder blend in the tablet press is compressed by the upper and lower die punches with the binder particles gluing the other blend particles together, forming a tightly bound compact mass. Prediction of powder properties across length scales can be made via either a top-down or bottom-up approach. In the former, the macroscopic behavior of the blends can be described by various constitutive relations, depending upon the assumptions made about the powder particle properties, and the experiments conducted for the characterization of the bulk powder properties [8–10]. We are interested in the latter approach, i.e. to obtain the constitutive relations for these powder blends by making extrapolations of the behavior observed from simulating the microscopic properties and dynamics of a small but representative number of blend particles.

We have studied dry non-cohesive powders, whose constituent particle sizes range between 10 to 1000 microns, with emphasis on pharmaceutical powders. The particles possess both translational and rotational degrees of freedom, in other words the particles are able to rotate about their centres of mass. The particles interact with one another at their points of contact via elastic and viscoelastic interactions. We have also considered the surface friction acting at the interparticle contacts to reduce the relative motion of the contacting surfaces while simultaneously coupling the translational and rotational degrees of freedom of the contacting particles.

The nature of the physical interaction between the particles in a pharmaceutical powder would not allow us to utilize standard molecular simulations packages without considerable modifications. However, constructing an efficient granular dynamics code from first principles would be an unnecessary duplication of time and effort. We therefore modified Daresbury Laboratory's parallel MD package DL_POLY [1,2] so as to enable us to develop a DES [11] to study model pharmaceutical granular systems. The motivation behind doing so was to take advantage of DL_POLY's infrastructure to support parallel simulations of many-body systems in 3D. In this paper, we present changes made to DL_POLY to study a system under the effect of an external gravitational force acting along the z -direction, and optionally, constant strain shear and uniaxial compaction (z -direction). We describe modifications made to DL_POLY in Section 2, we present some results in Section 3 and we provide a discussion of our work in Section 4. In this paper we will provide results from some validation tests which have been carried out to ensure

that our simulations are physically consistent with existing theoretical and computational studies [12–14], and some benchmarks which demonstrate the scaling of the execution time with the system size.

2. Modifications made to DL_POLY

This section provides details of changes made to DL_POLY. We have divided this section into six subsections which address the following issues: the initialization phase or reading the input files; neighbor list calculations; force field calculations; external field calculations, integration of equations of motion and writing to output files.

2.1. Description of system

Fig. 1 is a schematic diagram of the system of particles (labeled T2) bounded along the z -direction by two z -surfaces. The z -surfaces are generated by constraining the centers of mass of identical particles lying on it. Particles forming the lower and upper surfaces are labeled as types T1 and T3, respectively. The system is periodic along the x - and y -axes. The motivation behind our implementation of these boundaries is to simulate the effect of the simulation cell being a part of a large 3-dimensional system of particles, while simultaneously introducing granularity to the compacting surface. The type of each particle i is stored in an array `keytype(i)`. The particles on the upper and lower z -surface are constrained. The contacts between the type T1 and T3 particles, at a microscopic length scale (particle diameters), with the type T2 particles are modeled using the same force models as those for the contacts between the type T2 particles. However, at a macroscopic length scale, the surface particles should behave as a part of a reservoir of identical particles which form a solid object, in this case the compacting die. Therefore, we make the assumption that the surface particles (types T1 and T3) do not interact with other particles of the same type. During the gravity compaction phase, the type T2 particles settle under acceleration due to gravity onto the lower z -surface (or, type T1 particles). If the objective of the simulation is to study the effects of compaction and/or

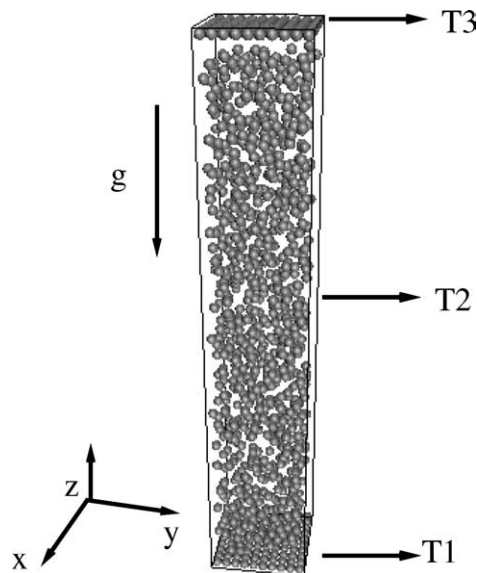


Fig. 1. Initial snapshot of 1000 monodisperse spheres accelerating due to gravity g , bounded by two z -surfaces which are modeled by monodisperse spheres arranged in a square lattice. The system is periodic along the x - and y -axes. The three classes of particles in this system have been labeled: T1 and T3 particles constituting the lower and upper z -surfaces, respectively, and T2 particles constituting the particles pouring onto the base.

shear on the T2 particles: either or both of the upper and lower z -surfaces will begin to uniaxially compact and/or shear the type T2 particles.

2.2. Input files

The simulation code requires three input files, which are very similar in name and functionality to those for DL_POLY: CONTROL, FIELD and CONFIG. The CONTROL file stores details relating to specifications pertinent to each phase of the simulation (gravity compaction, uniaxial compaction and/or shear) such as the number of iterations, the integrating time step δt and the various rates of output intervals (for the different output files) along with the total simulation run time. The FIELD file stores external force details such as the value for the gravitational acceleration, the shear and compaction strain rates and the interparticle interaction parameters. The interparticle interaction parameter values are read into a three-dimensional array $\text{cnt_int}(k_i, k_j, k)$, where $1 \leq k \leq 6$, and $k_i = \text{key_int}(i)$, $k_j = \text{key_int}(j)$ are particle interaction type indices (for particles i and j) such that $1 \leq k_i, k_j \leq N_{\text{inttype}}$. N_{inttype} is the total number of particles with different material properties and $\text{key_int}(i)$ is the interaction type array which stores the interaction type of each particle depending upon its material parameters. The `parset.f` routine will read the input files to assess the size of the various run time arrays. Once the arrays have been declared, the modified DL_POLY subroutines `simdef.f`, `sysdef.f` and `sysgen.f` read the CONTROL, FIELD and CONFIG files, respectively, to assign values to the simulation arrays. For example, the `parset.f` subroutine is invoked to determine the dimensions of the $\text{cnt_int}(k_i, k_j, k)$ array before data from the FIELD file are stored in the array. Relevant sections of the subroutine `sysdef.f` which store the interaction parameters in the array $\text{cnt_int}(i, j, k)$ are provided in [Appendix A](#) and are analogous to the subroutine `sysdef.f` in DL_POLY. The variables `irowmax`, `icolmax` and `iparmax` are read from the FIELD file; `irowmax` and `icolmax` are equal to N_{inttype} and `iparmax` = 6, following the discussion above. Each processor is assigned an index number which can be accessed through the integer variable `idnode`. The processor with `idnode` = 0 is responsible for reading and writing to files, unless explicitly mentioned. The designation of the input/output operations to the various nodes will be discussed later in this paper. The segment of code shown in [Appendix A](#) is within an `If` statement which ensures that the FIELD file is read until the end of file is reached. The CONFIG files contain individual particle information such as the mass m , diameter d , moment of inertia about an axis through the centre of mass and dynamical variables, namely, the position, linear and angular velocity vectors, \mathbf{r} , \mathbf{v} and $\boldsymbol{\omega}$, respectively. As mentioned earlier, the subroutine `sysgen.f` reads the CONFIG file to assign values to the arrays storing individual particle information.

We would like to readdress the issue of designation of the input/output operations: one of the processors is assigned the task of reading each line of the input files and distributing it to the remaining processors (the subroutine `getrec.f` is invoked for this purpose). Each node will process each line to store the relevant information in the appropriate variables. This procedure is followed while simultaneously adhering to the RDS paradigm [15–18]. We have retained DL_POLY's philosophy of allowing a simulation to be restarted from the point it was terminated through the help of the REVCN file. The REVCN file allows a simulation to be restarted from the point it stopped running, and is used in the place of the CONFIG file in such situations. The REVCN file is periodically updated (via the `revive.f` subroutine) with the most current simulation details, such as the number of timesteps elapsed and dimension of the simulation cell, along with the current values of the particle dynamical variables. The format of the REVCN and CONFIG files are very similar except that the former has information on the number of iterations elapsed at the end of the last simulation run, the integration time step and, in the case of parallel simulations, information which is locally stored on each processor (for example, individual particle dynamical variables essential for calculating interparticle contact forces and neighbor list information).

We have provided the relevant segments of the subroutine `sysgen.f` in [Appendix B](#) file which reads a CONFIG file for a restarted simulation (generated from a REVCN file) and distributes it for local storage to each of the processors. In contrast to the original DL_POLY code, the variable `levcfg` tracks whether a simulation is new or restarted from a previous run; for a restarted simulation `levcfg` > 0. As mentioned earlier, the subroutine

`getrec.f` reads lines (stored as a strings) from the input files and distributes the string to the other nodes for processing. The variable `id_cur` ($0 \leq id_cur \leq nodemx-1$, `nodemx` is the number of processors used for the parallel simulation) stores the index of the processor which has to locally store the information in the following lines of the `CONFIG` file. The variable `ii` tracks the local processor index of a particle whose information is being read and locally stored on the processor `id_cur`. The integer variable `i` is the global index of the particle which has a local index `ii` on processor `id_cur`. The global indices of the particles were used by the subroutine `revive.f` to write particle details to the `REVCN` file. The integer arrays `len_br(ii)` and `lentry(ii)` store the number of particles in contact with and residing in the neighbor list of, particle `ii`, respectively. This procedure is repeated over all the `nodemx` processors and for all the particles `ii` whose information is stored on each of the processors. A discussion related to the subroutine `revive.f` which writes to the `REVCN` file is provided in the section dedicated to output files.

2.3. Neighbor list calculations

As mentioned earlier, the particles in the system interact with one another via mechanical contacts, therefore a given particle indexed i will interact with those neighbors indexed j which lie within a sphere of diameter $\frac{1}{2} \times (\text{diameter}(i) + \text{diameter}(j))$. For a system of N particles, the process of having each particle i checking with the remaining $N - 1$ particles j to test whether the particles are in contact or not will result in approximately $N(N - 1)/2$ computations, assuming symmetry in the interaction criteria test. Hence, neighbor lists are essential for efficient computations of interaction forces, and are determined by the nature of the interaction between the particles whose trajectories are being simulated by the MD technique. Typically, neighbor lists [19] are utilized to reduce the computation time spent by each particle in determining its interaction partners depending upon the existence or non-existence of contacts. We use the link cell method [19,20] to calculate the neighbor list and appropriately select the dimensions of the link cells so that the neighbor list does not have to be recomputed at each iteration. For our application, the dimensions of the link cell are approximately $d_{\max} + r_{\text{skin}}$, where $d_{\max} = \max(\frac{1}{2} \times (\text{diameter}(i) + \text{diameter}(j)), (i, j = 1, \dots, N))$ is the largest interaction diameter and r_{skin} is the distance which we use to control the frequency with which the neighbor lists are recomputed. r_{skin} is about 1% of the smallest particle diameter. We take caution not to make r_{skin} too large, otherwise the frequency of recomputing the neighbor lists will decrease. Given the criteria used in determining interactions, frequent checks are needed to determine the existence of contacts. At each iteration, we use the subroutine `vertest1.f` from `DL_POLY` to check the necessity for recalculating the neighbor list. We use the `DL_POLY` subroutine `parlink.f` to calculate the neighbor list of particles residing on each processor. A new neighbor list requires updating the arrays which store interparticle contact information pertaining to the tangential force calculations and the state of the contacts. This point will be further addressed in following section.

2.4. Force field calculations

The molecular force field computation routines in `DL_POLY` have been replaced in our code by routines which address force field calculations pertinent to granular materials. The physical interactions at the contacts between the particles are modeled by normal and tangential elasticity and viscoelasticity, and macroscopic surface friction. This section summarizes the physical interactions modeled followed by details of the subroutines which carry out the relevant computations.

Fig. 2 is a schematic diagram of two particles i and j which are in contact with one another. During the contact between particles i and j , there are two relevant directions which will determine the forces acting on each particle: the normal and tangential directions, represented by unit vectors $\hat{\mathbf{n}}_{ij}$ and $\hat{\mathbf{t}}_{ij}$, respectively. The overlap along the normal direction is given by δ_n as shown in **Fig. 2**. There will be an analogous extension along the tangential direction denoted by δ_t . The relative velocity \mathbf{v}_c at the point of contact on the surface of particles i and j is given

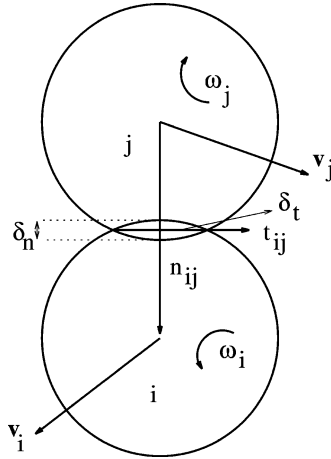


Fig. 2. Schematic diagram of a contact between two deformable particles i and j , indicating the normal and tangential deformations, and unit vectors δ_n and δ_t , and $\mathbf{v}_{i,j}$ and $\omega_{i,j}$, respectively.

by

$$\mathbf{v}_c = \mathbf{v}_i + \mathbf{v}_j + \left(\frac{d_i}{2} \boldsymbol{\omega}_i + \frac{d_j}{2} \boldsymbol{\omega}_j \right). \quad (1)$$

The forces acting on particles i and j will depend upon the degree of deformation and the rate of change of the deformation with time.

2.4.1. Normal interactions

The normal forces F_n acting on particles i and j are due to the elastic restoring force and a dissipative force due to the viscoelasticity of the materials. The elastic restoring force may have a linear dependence on the normal deformation δ_n [11] which is represented by the linear spring-dashpot model [21,22]

$$F_n = k_n \delta_n + \gamma_n \dot{\delta}_n. \quad (2)$$

A more accurate representation of the normal elastic restoring forces resulting from the contact between two spherical particles is given by the $\frac{3}{2}$ power dependence of the normal force on the normal deformation δ_n and these contacts are called Hertzian contacts [23,24]. The Hertz–Mindlin model captures this elastic behavior along with a linear viscoelastic term by the following normal force model

$$F_n = \kappa_n \delta_n^{3/2} + \gamma_n \dot{\delta}_n. \quad (3)$$

However, a study [25] has shown that the Hertz–Mindlin force results in an anomalous behavior of the particles: namely, the normal coefficient of restitution e_n approaches 1 as the impact speed approaches zero $(1 - e_n) \propto v_{c,n}^{-1/5}$, contrary to experimental evidence [26,27]. Further work [28,29] has developed a more accurate representation of the normal elastic restoring force which is known as the Hertz–Kuwabara–Kono (HKK) model. The HKK model results in the normal coefficient of restitution satisfying the functional relation $(1 - e_n) \propto v_{c,n}^{1/5}$ which has a higher degree of compatibility with experimental results [26,27] and has the following form

$$F_n = \kappa_n \delta_n^{3/2} + \Gamma_n \delta_n^{1/2} \dot{\delta}_n, \quad (4)$$

where k_n and κ_n are elastic force constants which depend upon material properties such as the Young's modulus E and the Poisson's ratio ν , and the particle radius R through the following relation: $\kappa_n = \frac{4}{3} E_{\text{eff}} \sqrt{R_{\text{eff}}} \gamma_n$ with

$1/E_{\text{eff}} = 1/E_i + 1/E_j$ and $1/R_{\text{eff}} = 1/R_i + 1/R_j$ [21–23]; γ_n and Γ_n are viscoelastic constants. k_n can be obtained from κ_n via dimensional analysis. δ_n can be substituted by the normal component of the relative velocity \mathbf{v}_c , Eq. (1). Our simulations have primarily used the HKK model to calculate the normal forces at the particles contacts due its better agreement with experimental results.

2.4.2. Tangential interactions

The tangential force F_t acting at the contact surface between particles i and j is modeled via a linear elastic restoring force and a velocity-dependent dissipative component (attributed to tangential viscoelasticity). This form of F_t is analogous to that for F_n in the linear spring-dashpot model, as shown below

$$F_t = k_t \delta_t + \gamma_t \dot{\delta}_t, \quad (5)$$

where k_t and γ_t are the tangential elastic and viscoelastic dissipative constants, respectively. δ_t and $\dot{\delta}_t$ are the tangential elongation and the time rate of change of the elongation, respectively, at the current time t . In order to calculate these two quantities, we have to keep a record of the tangential elongation for each contact at time $t - \delta t$ and update it with the change in the elongation during the time interval $t - \delta t$ and t . We have used the tangential component of \mathbf{v}_c Eq. (1) in the place of $\dot{\delta}_t$. As it is unphysical to allow the tangential elongation to increase unbounded, we have used Coulomb's law of friction [30] to determine the tangential interactions at an interparticle contact [31]. Coulomb's friction criteria acts to bound the values of the tangential contact force F_t and elongation δ_t , as shown below. When the normal and tangential forces satisfy the following relation

$$F_t < \mu_{\text{stat}} F_n, \quad (6)$$

where μ_{stat} is the coefficient of static friction, static friction is said to act at the contacts such that the relative velocity \mathbf{v}_c , Eq. (1), is zero, i.e. the contacts are non-sliding. During the non-sliding phase of the contacts, the tangential force takes the form given by Eq. (5). When the particle surfaces participating in a contact begin sliding over one another (i.e. when Eq. (6) is no longer satisfied), sliding friction acts at the contacts to reduce the sliding (non-zero \mathbf{v}_c) to zero [23]. The tangential force takes the following form

$$F_t = -\mu_k F_n \text{sign}(v_{ct}), \quad (7)$$

where μ_k is the coefficient of kinetic friction and v_{ct} is the tangential component of \mathbf{v}_c . The tangential elongation is adjusted appropriately so that the tangential contact force F_t satisfies Eq. (7).

2.4.3. Implementation of contact force interactions

We have developed a subroutine called `contact_forces_hkk.f` which calculates the normal and tangential contact forces between the particles adhering to the HKK contact model equation (4) and the linear dashpot model equation (5), respectively. At each iteration and for every possible contact, `contact_forces_hkk.f` computes the values of $\delta_{n,t}$ and $\dot{\delta}_{n,t}$ at the current time t . As a consequence of these calculations, the subroutine `contact_forces_hkk.f` requires the particle position, linear and angular velocity vectors at time t . DL_POLY uses the Verlet leap-frog integration scheme which computes the values of the position, linear and angular velocity vectors at time $t + \frac{\delta t}{2}$. We use a slightly modified integration scheme which computes the position, linear and angular velocity vectors of the particles at the current time t . This will be discussed in further detail in the section dedicated to the integration of the equations of motion. We have provided relevant segments of the subroutine `contact_forces_hkk.f`, interspersed with algorithms, in Appendix C. Following the RDS paradigm, each processor is responsible for calculating the interparticle forces and torques, and updating the total force and torque acting on each of the particles which fall into its domain of responsibility. For all the `mxnode` processors, each processor, indexed `idnode`, uses a local index `ii` to refer to the particles it is responsible for. Information germane to these particles can be accessed globally by the index `i`. The integer arrays `lentry(ii)` and `list(ii,k)` reside locally on processor `idnode`, and store the number of particles and the global indices

of particles, respectively, which are on particle ii 's neighbor list $list(ii, k)$. For example, particle j is the k th of the $lentry(ii)$ neighbors of particle ii . These two integer arrays are calculated in the `parlink.f` subroutine. The details associated with every contact between particle ii and its k th neighbor $j = list(ii, k)$ is stored locally on processor $idnode$ and is referred by the local index ik . As a first step to determining the interparticle forces and torques, the interparticle distances associated with each of the ik contacts are determined in a manner which is mindful of the 2D periodic boundary conditions used in the simulations. In the second step, normal and tangential elongations, $\delta_{n,t}$ and their rate of change with time, $\dot{\delta}_{n,t}$ are computed for further calculations of the normal and tangential contact forces, f_n and f_t , for the ik th contact. As mentioned earlier, the tangential force calculations require a record of the changes in the tangential elongation between two successive iterations for each contact as this is, in turn, used to determine the current total tangential elongation for the entire duration of a contact. We have introduced arrays `dtxold(ii, k)`, `dtyold(ii, k)` and `dtzold(ii, k)` which store the 3D Cartesian components of the tangential elongation for a contact ik between particles i and j . Details of contact ik is stored locally on the processor responsible for either or both of the particles i and j . The subroutine `tangential_elongation.f` computes the tangential elongation between successive iterations and is called by the subroutine `contact_forces_hkk.f` to assist in the calculations of the tangential contact force. The nature of the contact is determined by the Coulomb's friction criteria leading to possible changes in the values of the tangential contact force f_t and elongation δ_t . Contributions to the total particle force and torque are determined from the contact force computations. In the final step, the interparticle contact force details are written to the `FORCE_HISTORY` file, and the contact statistics are written to the `CONTACT_HISTORY` file.

The history of the tangential contact details is essential for determining the tangential forces due to the type of the tangential force interactions we have chosen to use while modeling microscopic contacts. Unless a contact between two particles ceases to exist, the arrays `dtxold(ii, k)`, `dtyold(ii, k)` and `dtzold(ii, k)` must always be updated to the current state of the contacts. There is a need to synchronize the updates of the tangential elongation arrays with those of the neighbor lists. We have introduced a subroutine called `tangential_adjustments.f` which updates the arrays storing the tangential elongation details of every contact while simultaneously ensuring that contact details are not lost with each neighbor list update.

2.5. External field calculations

The physical systems of our interest do not require the use any of the external force options present in `DL_POLY`, with the exception of the gravitational force. Typical external forces acting on a pharmaceutical powder blend, in a tablet press, are gravity and those arising from constant strain uniaxial compaction. We have introduced modifications into `DL_POLY` to account for the latter along with constant strain shear, to enable us to study the particle packing evolving from the simultaneous application of shear and uniaxial compaction on the powder compacts. We can use the code to study three different systems: (1) packing of particles under acceleration due to gravity; (2) uniaxial compaction of the particles, and (3) simultaneous uniaxial compaction and shear of the particles. Systems (2) and (3) can use the final spatial configuration of the particles generated via (1) as their initial spatial configurations. In the following subsections, we describe in some detail the implementation of the various external factors which act on the particles.

2.5.1. Gravity compaction

Fig. 1 shows the initial configuration of a typical system which is simulated by our code with the three types of particles described earlier on. For simulations studying the formation of particle compacts by gravitational compaction of a collection of particles the external force is mg where g is the acceleration due to gravity. Type T2 particles experience gravitational acceleration where as the type T1 and T3 particles do not. The gravitational compaction simulations will run until the average kinetic energy of the particles falls below 10^{-10} of the initial average kinetic energy or the prescribed number of iterations have been completed. The gravitational force is assigned to act on the type T2 particles in the subroutine `external_forces.f`.

2.5.2. Constant strain shear and/or uniaxial compaction

In the physical systems of our interest, constant strain (or speed) uniaxial compaction and shear are applied after the type T2 particles have undergone gravitational compaction (Fig. 1). We have implemented constant strain uniaxial compaction and shear by introducing predefined velocities for the particles of the top and bottom surfaces \mathbf{v}_{top} and \mathbf{v}_{bot} , respectively. The z -components of the particles forming the upper and lower surfaces ($\mathbf{v}_{\text{top}_z}$ and $\mathbf{v}_{\text{bot}_z}$) are the constant strain rates at which uniaxial compaction occurs. The x -, y -components of the particles forming the upper and lower surfaces ($\mathbf{v}_{\text{top}_{x,y}}$ and $\mathbf{v}_{\text{bot}_{x,y}}$) are the constant strain rates at which the type T2 particles in contact with the type T1 and T3 particles are sheared. The values of $\mathbf{v}_{\text{bot,top}_{x,y,z}}$ are provided in the FIELD file. The type T1 and T3 particle velocities are assigned the values of $\mathbf{v}_{\text{bot,top}_{x,y,z}}$ in the subroutine which handles the integration of the equations of motion.

2.6. Integration of equations of motion

We have adhered to the same algorithm utilized by DL_POLY for integrating the equations of motions of the particles: the Verlet leap-frog algorithm [1,2,19]. The total force \mathbf{F}_i and acceleration \mathbf{a}_i acting on each particle i with mass m_i is calculated

$$\mathbf{F}_i(t) = \sum_j \mathbf{f}_{ij}(t) + \mathbf{F}_{\text{ext}}(t), \quad (8)$$

$$\mathbf{a}_i(t) = \frac{\mathbf{F}_i(t)}{m_i}, \quad (9)$$

where \mathbf{f}_{ij} is the force acting on particle i due to particle j , hence the summation is over all the particles in contact with particle i . \mathbf{F}_{ext} is the external force acting on particle i . From Newton's third law, the force experienced by particle j due to particle i is $\mathbf{f}_{ji} = -\mathbf{f}_{ij}$. According to the leap-frog algorithm, the equations for the position and velocity vectors, \mathbf{r}_i and \mathbf{v}_i , respectively, are

$$\mathbf{v}_i\left(t + \frac{\delta t}{2}\right) = \mathbf{v}_i\left(t - \frac{\delta t}{2}\right) + \delta t \mathbf{a}_i(t), \quad (10)$$

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \delta t \mathbf{v}_i\left(t + \frac{\delta t}{2}\right). \quad (11)$$

We have introduced angular spin or velocity $\boldsymbol{\omega}$ about the centre of mass of the individual particles. Hence, the equations for the rotational degrees of freedom have analogous forms to those for the translational degrees of freedom: the equations for the net torque $\boldsymbol{\tau}_i$ and the angular velocity $\boldsymbol{\omega}_i$ about the centre of mass of particle i are given by

$$\boldsymbol{\tau}_i(t) = \sum_j \mathbf{r}_{ij}(t) \times \mathbf{f}_{ij}(t), \quad (12)$$

$$\boldsymbol{\omega}_i\left(t + \frac{\delta t}{2}\right) = \boldsymbol{\omega}_i\left(t - \frac{\delta t}{2}\right) + \delta t \hat{\mathbf{I}}_i^{-1} \mathbf{a}_i(t), \quad (13)$$

where $\mathbf{r}_{ij}(t) = \mathbf{r}_i(t) - \mathbf{r}_j(t)$ is the vector joining the centre of masses of particles i and j , and $\hat{\mathbf{I}}_i$ is the diagonalized moment of inertia tensor about the centre of mass of particle i . The force calculation routines require current time (t) values of the linear and angular velocities of each particle, which are calculated using the following relations

$$\mathbf{v}_i(t) = \frac{1}{2}[\mathbf{v}_i(t + \delta t/2) + \mathbf{v}_i(t - \delta t/2)], \quad (14)$$

$$\boldsymbol{\omega}_i(t) = \frac{1}{2}[\boldsymbol{\omega}_i(t + \delta t/2) + \boldsymbol{\omega}_i(t - \delta t/2)]. \quad (15)$$

Constant strain uniaxial compaction and shear are implemented by assigning the appropriate components of $\mathbf{v}_{\text{bot,top}_{x,y,z}}$ to the current velocity $\mathbf{v}_i(t)$ of the type T1 and T3 particles.

2.7. Output files

Unlike DL_POLY, our code does not perform any runtime analysis such as radial distribution functions, z-density or any statistical variable calculations. All the output files generated belong to one of the three categories: raw data files which store individual particle trajectories and dynamical variables along with individual contact details, to be used for subsequent analysis (HISTORY, FORCE_HISTORY and CONTACT_HISTORY files); a record of all system details, taken at fixed intervals during the simulation, to restart simulations (REVCON file), and a record of the initial and final states of the simulation (OUTPUT file). The HISTORY, OUTPUT and REVCON files are analogous in philosophy to those generated by DL_POLY. The subroutine `traject.f` writes all the trajectory details of each particle in the HISTORY file. We have introduced the CONTACT_HISTORY and FORCE_HISTORY files to store the number of total, sliding and non-sliding contacts in the system, and the details of the contact vectors and forces, respectively. The HISTORY, CONTACT_HISTORY, FORCE_HISTORY and REVCON files are appended at regular intervals. The task of writing to the HISTORY and OUTPUT files is carried out by a pre-designated processor as the RDS algorithm ensures that all processors have copies of the arrays storing the relevant particle dynamical variables. The procedures used in writing to these data files is identical to those followed by DL_POLY. The procedures followed to write to the CONTACT_HISTORY, FORCE_HISTORY and REVCON files are slightly more involved as data written to these files reside locally on the individual processors. The calculations of the details relating to the interparticle contacts, such as the contact vectors and forces and the contact states, are carried out by the individual processors on the subset of particles it is responsible for (designated by the RDS scheme). As a consequence, the storage of these interparticle contact details into the data files can be either designated to the individual processors (in a sequential manner), for example, the subroutine `norm_tan_force_hist.f` writing to the FORCE_HISTORY file, or be carried out by a pre-designated processor (following global summation operations), for example, the subroutine `contact_output.f` writing to the CONTACT_HISTORY file. The `revive.f` subroutine which is responsible for updating the REVCON file adheres to the former approach as it contains individual particle neighbor list and contact details which are locally stored on each node.

Relevant segments of the subroutine `contact_output.f` are provided in [Appendix D](#). Following the calculations of the contact interactions among all the particles (on all processors), a pre-designated node (`idnode = 0`) carries out the global integer sum operation over all the processors, using DL_POLY's subroutine `gisum.f`, to compute the number of total, sliding and non-sliding contacts represented by the variables `ncnt`, `ncnt_roll` and `ncnt_slip`, respectively. After the global summation operations are complete, `idnode (= 0)` writes the values of the variables `ncnt`, `ncnt_roll` and `ncnt_slip` to the CONTACT_HISTORY file. We have designed the `norm_tan_force_hist.f` subroutine such that the force calculations for all the potential contacts must be completed before entries are made to the FORCE_HISTORY file. [Appendix E](#) provides the pertinent segments of the subroutine `norm_tan_force_hist.f` which write to the FORCE_HISTORY file. Following the variable notations used earlier, `norm_tan_force_hist.f` loops over all the `mxnode` processors, referred by the variable `nodeid`, and writes down the details of the `ikth` contact between particles `i` (local variable `ii`) and its `kth` neighbor, particle `j`, utilizing the neighbor list details of the former (`lentry(ii), list(ii,k)`). If `mxnode > 1`, it is essential that all nodes are at the same stage of operation for making entries sequentially to the FORCE_HISTORY file. To ensure sequential data entry to the FORCE_HISTORY file, we use a subroutine from DL_POLY called `gsync.f` which is the global synchronization call, and simultaneously, loop through the list of nodes `nodeid = 1, ..., mxnode`.

The REVCON file stores all the necessary macroscopic and microscopic system information necessary to restart a simulation from the point it stopped running, and therefore, will need to store information regarding the individual particle neighbor lists and the interparticle contact details pertaining to the tangential elongations, as discussed in earlier. To expand on the latter point, the REVCON files must store details of the particle dynamical variables and individual contacts such as tangential elongations and the state of the contacts (sliding or non-sliding) which reside locally on the individual processors. Information relating to the neighbor list of particle `i` and its contact details with particle `j` are stored on the processor which is responsible for either particle `i` (or `j` but not both under the RDS

paradigm). Therefore, the REVCN files are written in two phases: (1) the predesignated node writes the values of all the particle dynamical variables (available to all the processors); (2) each processor proceeds to write other information which resides locally in a sequential format. Appendix F lists the relevant lines for the latter phase of the subroutine `revive.f`. Following the same variable notations used for earlier descriptions of code segments, each node `idnode` writes the details of the contacts formed by particle `ii` (local index for node `idnode`, global index is `i`) with all its `len_nbr(ii)` neighbors. This operation is followed by writing the neighbor list details of particle `ii` which has `lentry(ii)` neighbors. For simulations run on more than one node, `mxnode > 1`, we use the global synchronization call, as described earlier, to maintain a sequential output to the REVCN file, and we loop through the list of nodes.

3. Results and discussion

In contrast to the steady state MD simulations, our granular simulations model dissipative systems where the dynamics of systems may get quenched quickly if sufficient external energy, for example, in the form of shear or vibration, is not fed into the systems to compensate for the dissipation due to surface frictional or viscoelastic losses. As a consequence, we set a low average kinetic energy threshold in the CONTROL file; a simulation is stopped, if at any time during its execution, the average kinetic energy of the system falls below the preset threshold. Therefore, it is not uncommon to find the gravity compaction simulations terminating before completion of the preset total number of iterations on account of the aforementioned energy criterion. The two most computationally demanding operations in our modified code are: determination of the neighbor lists for each particle which is a procedure carried out locally on each node involving the calculation of the inter-particle distance, and the calculation of the inter-particle interaction forces, for all the particle contacts. The latter operation is assumed to be symmetric and the associated computations are carried out once for every contact. Hence, simulating a large dense granular system comprised of particles from a large size distribution will be computationally very expensive on both accounts.

We have benchmarked our code on the following criteria: (1) the physical validity of the simulation results, and (2) the computational performance of the code. Our studies have focused, primarily, on the statics and dynamics of dense granular media. We have used previous theoretical and computational studies of dense granular media to benchmark the implementation of the contact model we have chosen to utilize in our code, in terms of the physical validity of our results. Some of the earlier studies [12–14] have looked at details of particle packings of frictionless (no rotational degrees of freedom) and frictional (both translational and rotational degrees of freedom) monodisperse spheres; we have compared our numerical results with those obtained from these studies. For testing the computational performance of our code, we have looked at the run times of the code as a function of system size. We have provided detailed discussions of the two benchmark criteria below.

3.1. Validation tests

Earlier computational and theoretical work [12–14] on dense granular media have extensively studied particle packing characteristics such as the average coordination number Z (the number of nearest neighbors which are touching, for granular materials), the global packing fraction ρ and, to a lesser extent, the radial distribution function. In this paper, we shall focus on the former two quantities to benchmark the physical accuracy of our results. Theoretical predictions [12,13] have shown that for a dense granular material with every particle in static equilibrium in a d -dimensional space, frictionless spheres need a minimum of $2d$ contacts, and frictional spheres need a minimum of $d + 1$ spheres. These predictions assume a monodisperse collection of hard spheres. Hence, for 2- and 3-dimensional space ($d = 2, 3$, respectively), individual monodisperse frictionless and frictional spheres will need a minimum of 4 and 3, and 6 and 4 contacts, respectively. A detailed computational study by Grest et al. [14] has confirmed the theoretical predictions for both frictional and frictionless monodisperse spheres.

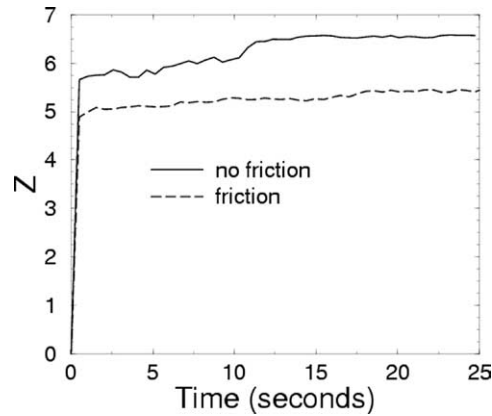


Fig. 3. Time evolution of the coordination number Z for an agglomeration of frictionless and frictional particles ($N = 1000$), with $\mu_{\text{stat}} = 0.7$ and $\mu_k = 0.2$, settling under acceleration due to gravity.

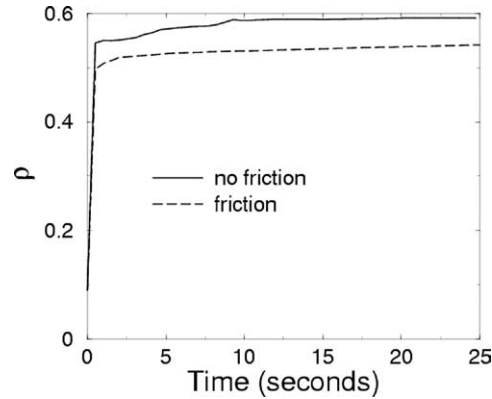


Fig. 4. Time evolution of the packing fraction ρ for an agglomeration of frictionless and frictional particles ($N = 1000$), with $\mu_{\text{stat}} = 0.7$ and $\mu_k = 0.2$, settling under acceleration due to gravity.

We generate particle packings of monodisperse spheres (frictionless and frictional) by allowing particles (type T2) to settle under acceleration due to gravity (Fig. 1). We have found the values of average coordination number Z and packing fraction ρ computed from the simulations of frictionless and frictional monodisperse spheres to agree well with previous numerical studies [14]. Figs. 3 and 4 represent the time evolution of the average coordination number and packing fraction, respectively, for simulations using 1000 monodisperse spherical particles (with and without surface friction). We find that as the particles settle under gravitational acceleration and gradually manoeuvre around one another so as to achieve a configuration which allows them to attain maximum static equilibrium, final values of Z and ρ are found to be greater than 6 and 4, and approximately 60 and 55%, for frictionless and frictional spheres, respectively. We have repeated calculations of the aforementioned particle packing characteristics for gravity packing simulations using varying number of frictionless and frictional spheres, and have tabulated the results for final average coordination number in Table 1. We have the final average values of the coordination number approach 5.5^+ and 6.4^+ for frictional and frictionless spheres, respectively. Our calculations of the packing fraction have values approaching 55^+ % and 63% for frictional and frictionless spheres, respectively, which are in good agreement with [14].

Table 1

Summary of final average coordination number Z for simulations carried out for system sizes or number of particles N , using frictional and frictionless monodisperse spheres

N	Z	No friction	Friction
784	6.48	×	
784	5.68		×
820	6.84	×	
820	5.49		×
856	6.81	×	
856	5.50		×
928	6.19	×	
928	5.73		×
964	6.54	×	
964	5.58		×
1000	6.57	×	
1000	5.62		×

3.2. Performance comparisons

We have carried out some preliminary performance benchmarks to study how the run time of the code scales with the system size. We have chosen to study a system similar to the one shown in Fig. 1 with N type T2 particles and N_z type T1 and T3 particles each. In the simulation, the T2 particles will be allowed to settle under gravitational acceleration onto the z -surface formed by the T1 particles and then experience constant strain uniaxial compaction on account of the T3 particles. As mentioned earlier, three types of inter-particle interactions have to be considered: those among the type T2 particles, and the type T2 particles with type the T1 and T3 particles. The CPU time benchmarks have to be performed mindful of the values of N and N_z , and the initial packing fraction of the type T2 particles. The values of N and N_z will determine the number of the different types of interactions, and thereby affect the total run time of a given simulation. In this paper, we present benchmark results for simulations with variations in the number of type T2 particles N while maintaining a fixed number of type T1 and T3 particles N_z . In each simulation, the initial packing fraction of the type T2 particles is the same, and the type T1 and T3 particles have identical arrangements in the bottom and top z -surfaces, respectively. The packing fraction is maintained at a constant value with increasing N by increasing the dimension of the system along the z -axes while leaving unchanged the dimensions along the x - and y -axes. The particles in the simulations have been endowed with surface friction which allows them both translational and rotational degrees of freedom. The diameter ratio of the T1 particles to that for the T2 (or T3) particles is 4:1.

Fig. 5 plots the run time of each simulation as a function of the number of type T2 particles. For each simulation, the number of type T1 and T3 particles N_z are identical with $N_z = 75$. The gravitational and the uniaxial compaction phases are set to run for 1500001 and 100001 iterations, respectively. The uniaxial compaction strain rate was set at 1 mm/s. The simulations have been performed with identical FIELD and CONTROL files. All simulations were run a single node (Itanium 2 IA-64 1.3 GHz). A fit of the data points in Fig. 5 shows a weakly quadratic dependence of the execution time as a function of the number of particles in the system. For each of the systems used in our CPU benchmark, on the average the number of contacts between the T2 particles with the T1 and T3 particles remains the same. However, the number of contacts between the T2 particles increases with N .

4. Conclusions

In this paper, we have presented details of changes which we have made to the macromolecular systems MD simulation package DL_POLY to allows us to study the statics, dynamics and the resulting packings of the con-

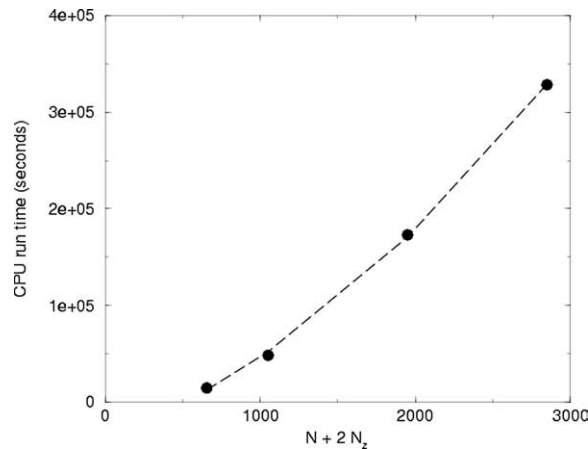


Fig. 5. Simulation execution time, in seconds, as a function of the total number of particles ($N + N_2$). Each of the simulations are identical in all respects except for the number of type T2 particles N . The dotted line is a quadratic fit.

stituent particles of a granular material under different external conditions. With hindsight, we could have made the surface particle velocity assignments in the subroutine `external_forces.f`, however, we decided to use the subroutine for the assignment of external forces only. Our validation tests for the physical accuracy of our simulation results have demonstrated good agreement with existing theoretical and numerical work on 3-dimensional monodisperse sphere packings. We have also carried out CPU time performance benchmarks which have shown the run time to have a weak quadratic dependence on the number of particles in the system.

The code can be used to simulate agglomerates of particles of different sizes and material properties with relevance to a wide range of applications; our primary focus is to utilize this code to study pharmaceutical powder blends. For simulations of polydisperse systems, care must be taken in generating the appropriate `CONTROL`, `FIELD` and `CONFIG` files. A possible future direction would be to utilize the `DES` code to extrapolate constitutive relations and properties of bulk powders by simulating the dynamics of a small but representative set of powder particles. Readers with queries or interests in the specific details of the code should contact us.

Acknowledgements

We would like to thank Daresbury Laboratory for allowing us to utilize the `DL_POLY` package. We would also like to acknowledge Pfizer for grant support. We would also like to thank Dr. Serena Best for reading through the manuscript and providing insightful comments.

Appendix A. Changes made to `sysdef.f`

```

if(idnode.eq.0)write(nwrite,'(a80)') record(1:80)
do i=1,irowmax
  do j=1,icolmax
    do k=1,iparmax
      call getrec(safe,idnode,mxnode,nfield,record)
      if(.not.safe)go to 2000
      irow=intstr(record(1:1),5,idum)
      icol=intstr(record(6:6),5,idum)
      cnt_int(i,j,k)=dblstr(record(11:11),16,idum)
    
```

```

        f(idnode.eq.0) write(nwrite,'(a80)') record(1:80)
    enddo
enddo
enddo

```

Appendix B. Changes made to `sysgen.f` for restarted simulations

```

do nd=1,nnodemx
  call getrec(safe,idnode,mxnode,nconf,record)
  if(.not.safe)go to 100
  if(record(1:10).eq.'begin node') id_cur = intstr(record,80,idum)
  ii=0
  do i=id_cur+1,natms,nnodemx
    ii=ii+1
    call getrec(safe,idnode,mxnode,nconf,record)
    if(.not.safe)go to 100
    i2=intstr(record(1:1),10,idum)
    lnnbr=intstr(record(11:11),10,idum)
    if(idnode.eq.id_cur) len_nbr(ii)=lnnbr
    if(lnnbr.gt.0)then
      do k=1,lnnbr
        Details of contacts between particle ii and
          its contacting neighbors are read and stored
          locally on processor id_cur
      enddo
    endif
    call getrec(safe,idnode,mxnode,nconf,record)
    if(.not.safe)go to 100
    i4=intstr(record(1:1),10,idum)
    lent=intstr(record(11:11),10,idum)
    if(idnode.eq.id_cur) lentry(ii)=lent
    if(lent.gt.0)then
      do k=1,lent
        Details the neighbor lists for particle ii
          are read and stored locally on processor id_cur
      enddo
    endif
  enddo
  :
enddo
call getrec(safe,idnode,mxnode,nconf,record)
if(.not.safe)go to 100
if(record(1:8).eq.'end node') id_cur =intstr(record,80,idum)
enddo

```

Appendix C. Contact force calculations

```

ii=0
ik=0
do i=idnode+1,natms,mxnode

```

```

ii=ii+1
do k=1,lentry(ii)
  j=list(ii,k)
  if(j.gt.0) then
    ik = ik + 1
    Calculate the interparticle distance associated with contact ik
    between particle i and its neighboring particle j, taking into
    account the 2D periodic boundary conditions.
  endif
enddo
enddo
:
ii=0
ik=0
do i=idnode+1,natms,mxnode
  ii=ii+1
  do k=1,lentry(ii)
    j=list(ii,k)
    if(j.gt.0) then
      ik = ik + 1
      For overlapping contacts between particles i and j
      1. Calculate  $\delta_n(t)$ ,  $\dot{\delta}_n(t)$  and  $\dot{\delta}_t(t)$ 
      2. Utilize tangential_elongation.f to calculate  $\dot{\delta}_t(t - \delta t)$ 
      3. Utilize  $\dot{\delta}_t(t - \delta t)$  and  $\delta_t(t - \delta t)$  to calculate  $\delta_t(t)$ 
      4. Calculate  $f_n$  and  $f_t$ 
      5. Check to see whether Coulomb's friction criteria
      equation7 is satisfied; if not, adjust  $f_t$  and  $\delta_t(t)$ ; update  $\delta_t(t)$ .
      6. Track number of total, sliding and non-sliding contacts
      7. Update the total force and torque acting on particles
      i and j using equation8 (with the absence of external forces)
      and equation12, respectively
    endif
  enddo
enddo
Write to FORCE_HISTORY and CONTACT_HISTORY

```

Appendix D. Writing population of contact states details to CONTACT_HISTORY file

```

if(idnode.eq.0)then
:
if(mxnode.gt.1) call gisum(ncnt,1,ibuff)
if(mxnode.gt.1) call gisum(ncnt_slip,1,ibuff)
if(mxnode.gt.1) call gisum(ncnt_roll,1,ibuff)
Write ncnt, ncnt_slip, ncnt_roll
:
endif

```

Appendix E. Writing contact force and vector information to FORCE_HISTORY file

```

do nodeid=idnode+1,mxnode
  if(mxnode.gt.1) call gsync()
  if(idnode.eq.(nodeid-1))then
    ii=0
    ik=0
    do i=idnode+1,natms,mxnode
      ii=ii+1
      do k=1,lentry(ii)
        j=list(ii,k)
        if(j.gt.0) then
          ik = ik + 1
          Write details of the contact ik between particles i and j
        endif
      enddo
    enddo
  endif
enddo

```

Appendix F. Writing system information necessary for restarting simulation to the REVCON file

```

:
if(idnode.eq.0)then
  write(nconf, "('begin node ',5x,i10)") idnode
  ii=0
  do i=idnode+1,natms,mxnode
    ii=ii+1
    write(nconf, '(2i10)')ii,len_nbr(ii)
    if(len_nbr(ii).gt.0)then
      do k=1,len_nbr(ii)
        Details of tangential elongation at contacts formed with particle ii or i
      enddo
    endif
    write(nconf, '(2i10)')ii,lentry(ii)
    if(lentry(ii).gt.0)then
      do k=1,lentry(ii)
        Details of neighbor lists for particle ii or i
      enddo
    endif
  enddo
  write(nconf, "('end node ',5x,i10)") idnode
  :
endif
if(mxnode.gt.1)then
  do j=1,mxnode-1
    call gsync()
    if(idnode.eq.j)then

```

```

open(nconf, file='REVCN', position='append')
write(nconf, "( 'begin node ', 5x, i10) ") idnode
ii=0
do i=idnode+1, natms, mxnode
  ii=ii+1
  write(nconf, '( 2i10' )ii, len_nbr(ii)
  if(len_nbr(ii).gt.0)then
    do k=1, len_nbr(ii)
      Details of tangential elongation at contacts formed with particle ii or i
    enddo
  endif
  write(nconf, '( 2i10' )ii, lentry(ii)
  if(lentry(ii).gt.0)then
    do k=1, lentry(ii)
      Details of neighbor lists for particle ii or i
    enddo
  endif
  :
enddo
write(nconf, "( 'end node ', 5x, i10) ") idnode
:
endif
enddo
endif

```

References

- [1] W. Smith, T.R. Forester, The DL_POLY_2 User Manual v2.13, 2001.
- [2] T.R. Forester, W. Smith, The DL_POLY_2 Reference Manual v2.13, 2001.
- [3] H.M. Jaeger, S.R. Nagel, R.P. Behringer, Rev. Mod. Phys. 68 (1996) 1259–1273.
- [4] P.G. de Gennes, Rev. Mod. Phys. 71 (1999) S374.
- [5] I. Goldhirsch, M.-L. Tan, G. Zanetti, J. Sci. Comput. 8 (1993) 1.
- [6] M.E. Aulton (Ed.), Pharmaceuticals the Science of Dosage Form Design, second ed., Churchill Livingstone, Spain, 2002.
- [7] G. Alderborn, C. Nyström (Eds.), Pharmaceutical Powder Compaction Technology, Drugs and Pharmaceutical Sciences, vol. 71, Marcel Dekker Inc., New York, 1996.
- [8] I. Aydin, B.J. Briscoe, K.Y. Sanliturk, Powder Tech. 89 (1996) 239.
- [9] D.H. Zeuch, J.M. Grazier, J.G. Argüello, K.G. Ewsuk, J. Mat. Sci. 36 (2001) 2911–2924.
- [10] L.C.R. Schneider, A.C.F. Cocks, Powder Met. 45 (2002) 237.
- [11] P.A. Cundall, O.D.L. Strack, Géotechnique 29 (1979) 47–65.
- [12] S. Alexander, Phys. Rep. 296 (1998) 65.
- [13] S.F. Edwards, Physica D 249 (1998) 226.
- [14] L.E. Silbert, D. Ertas, G.S. Grest, T.C. Halsey, D. Levine, Phys. Rev. E 65 (2002) 031304.
- [15] S. Brode, R. Ahlrichs, Comput. Phys. Comm. 42 (1986) 41.
- [16] W. Smith, D. Fincham, Molecular Simulation 10 (1993) 67.
- [17] W. Smith, Comput. Phys. Comm. 62 (1991) 229.
- [18] W. Smith, Theoretica Chim. Acta 84 (1993) 385.
- [19] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford University Press, UK, 2001.
- [20] G.S. Grest, B. Dünweg, K. Kremer, Comput. Phys. Comm. 55 (1989) 269.
- [21] J. Schäfer, S. Dippel, D.E. Wolf, J. Phys. I France 6 (1996) 5–20.
- [22] S. Luding, in: Collisions and contacts between two particles, in: H.J. Herrmann, J.-P. Hovi, S. Luding (Eds.), Physics of Dry Granular Media, Kluwer Academic Publishers, Dordrecht, 1998.
- [23] K.L. Johnson, Contact Mechanics, Cambridge University Press, UK, 2001.
- [24] E.M. Lifshitz, L.D. Landau, Theory of Elasticity, vol. 7, third ed., Butterworth-Heinemann, 1986.

- [25] Y.-H. Taguchi, *J. Phys. II France* 2 (1992) 2013.
- [26] F.G. Bridges, A. Hazes, D.N.C. Lin, *Nature* 309 (1984) 333.
- [27] R. Sondergaard, K. Chaney, C.E. Brennen, *J. Appl. Mech.* 57 (1990) 694.
- [28] G. Kuwabara, K. Kono, *Jap. J. Appl. Phys.* 26 (1987) 1230.
- [29] N.V. Brilliantov, F. Spahn, J.M. Hertzsch, T. Pöschel, *Phys. Rev. E* 53 (1996) 5382–5392.
- [30] C. Coulomb, in: *Memoir de Mathematique et de Physique*, vol. 7, Academie des Sciences, L'Imprimerie Royale, Paris, 1773, p. 343.
- [31] J.A. Åström, H.J. Herrmann, J. Timonen, *Phys. Rev. Lett.* 84 (2000) 638–641.