Journal of Computational Physics 253 (2013) 471–490

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Random generation of periodic hard ellipsoids based on molecular dynamics: A computationally-efficient algorithm

Elias Ghossein, Martin Lévesque [*]

Laboratory for Multiscale Mechanics (LM^2), CREPEC, Départment of Mechanical Engineering, École Polytechnique de Montréal, C.P. 6079, succ. Centre-ville, Montréal, QC, H3C3A7, Canada

A R T I C L E   I N F O

Article history:
Received 30 November 2012
Received in revised form 22 June 2013
Accepted 1 July 2013
Available online 23 July 2013

Keywords:
Random periodic packings
Hard ellipsoids
Molecular dynamics
Binary collisions
Orientations distribution
High aspect ratios

A B S T R A C T

This paper presents a computationally-efficient algorithm for generating random periodic packings of hard ellipsoids. The algorithm is based on molecular dynamics where the ellipsoids are set in translational and rotational motion and their volumes gradually increase. Binary collision times are computed by simply finding the roots of a non-linear function. In addition, an original and efficient method to compute the collision time between an ellipsoid and a cube face is proposed. The algorithm can generate all types of ellipsoids (prolate, oblate and scalene) with very high aspect ratios (i.e., > 10). It is the first time that such packings are reported in the literature. Orientations tensors were computed for the generated packings and it has been shown that ellipsoids had a uniform distribution of orientations. Moreover, it seems that for low aspect ratios (i.e., $\leqslant 10$), the volume fraction is the most influential parameter on the algorithm CPU time. For higher aspect ratios, the influence of the latter becomes as important as the volume fraction. All necessary pseudo-codes are given so that the reader can easily implement the algorithm.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Random packings are found in several physics and engineering fields. Several studies relied on random packings to simulate the behavior of molecular fluids [1,2]. They are also widely used in numerical homogenization of random media [3,4] where periodic boundary conditions are usually imposed since they lead to smaller Representative Volume Elements (RVEs) [5]. When the effective properties are computed from the finite element technique, the imposition of periodic boundary conditions requires the unit cell to be periodic. In this case, the random packings must also be periodic.

Several types of random packings were studied in the literature: hard spherical particles [4,6,7], cylinders [8,9], ellipsoids [10–12], spherocylinders [13–15], etc. Following the work of Lubachevsky and Stillinger [16], algorithms based on molecular dynamics (MD) have become increasingly popular due to their computational efficiency with respect to algorithms where the particles are generated sequentially [17]. Algorithms that generate random packings of spherical particles are now well established [18,19]. They rely on simple equations and, in most cases, analytical solutions exist. This is not the case for non-spherical models where numerical methods are usually needed. Donev et al. [20] proposed an efficient algorithm that relies on the near-neighbor list (NNL) method to improve the binary collisions computation time. This algorithm was tested for generating random packings that contain ellipses and ellipsoids [21]. However, the technique used to determine the collision time between two ellipsoids may not be computationally efficient. Indeed, the collision time between two moving ellipsoids is the first non-zero time value for which the maximum of a certain function is equal to zero. Since the maximum of this

[*] Corresponding author.
E-mail addresses: elias.ghossein@polymtl.ca (E. Ghossein), martin.levesque@polymtl.ca (M. Lévesque).
URLs: http://www.polymtl.ca/lm2 (E. Ghossein), http://www.polymtl.ca/lm2 (M. Lévesque).

0021-9991/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.07.004

function is computed numerically, the problem could be computationally expensive. In addition, the authors considered periodic packings but without detailing the method for computing the collision between an ellipsoid and a cell face.

The aim of this paper is to present a fully detailed computationally-efficient algorithm for generating random periodic packings of hard ellipsoids. Special emphasis is put on obtaining a computationally-efficient algorithm that can deal with general ellipsoids.

The paper is structured as follows: Section 2 reviews the different techniques for generating random packings. Section 3 presents some preliminaries on the mathematical representation of an ellipsoid oriented in space. The random generation algorithm is detailed in Section 4. Section 5 describes the method for evaluating the orientations dispersion of random ellipsoids. The performance of the algorithm is discussed in Section 6. Several examples of random packings are also presented.

The following convention has been adopted, unless specified otherwise: scalars are denoted by lower case letters (i.e. a, α); column vectors and matrices are respectively denoted by boldfaced lower case letters (i.e. $\boldsymbol{a}$, $\boldsymbol{\alpha}$) and boldfaced upper case Latin letters (i.e. $\mathbf{A}$). " $\times$ " denotes a vector product.

2. Background

2.1. Existing algorithms for generating random packings

The Random Sequential Adsorption (RSA) algorithm [17] is certainly the most widely used algorithm for generating random packings. At the beginning of the computation, the position of a particle is randomly selected. Then, the position of another particle is drawn and the contact is checked between both particles. If there is interference, the position of the second particle is drawn again until there is no contact with the first. The process is repeated until the desired number of particles and volume fraction are reached. Several authors [6,7,22] have used this algorithm and have struggled to reach high volume fractions. In its simplest form, this algorithm can generate volume fractions of approximately 30% for identical spherical particles.

An improved version of the RSA algorithm was proposed by Segurado and Llorca [6]. A unit cell having a volume fraction lower than the desired value is first generated with the RSA algorithm. The cell is then compressed in several steps and the particles positions and volumes are updated at each compression stage. However, this compression leads to particles interpenetration. It is therefore necessary to check for particles overlapping at each compression stage. If two particles touch, one of them is displaced along a random vector. If the particles are still in interference, the particle is placed in its original position and the process is repeated until the two inclusions no longer intersect. The simulation continues until the target volume fraction is reached. This modification in the RSA algorithm allows for denser packings than with the original version (around 50% for identical spherical particles).

Lubachevsky and Stillinger proposed an algorithm based on molecular dynamics [16]. This algorithm was originally applied to discs (2D) and spheres (3D) [18]. The main idea of the algorithm is as follows. All inclusions are randomly created in the unit cell but have a null volume. Each inclusion has also a random velocity vector. The particles are then set in motion and their volumes gradually increase. Two types of events are checked at each iteration: binary collisions and collisions between particles and the cell faces. When a binary collision occurs, the velocities of the two concerned particles are updated according to the kinetic energy conservation principle. However, if a particle leaves the cell through a face, it must appear from the opposite side to meet the periodicity conditions. The simulation stops when the desired volume fraction is reached. This algorithm is more efficient than the RSA and the modified RSA algorithm since it can generate very dense packings in a low computation time. For example, a packing of 30 identical spheres with a volume fraction of 62% can be generated in less than 10 s. It was also possible to reach a volume fraction of 74% [4], which is close to the theoretical maximum dense packing for spheres of identical size ($\approx 74.05\%$). This type of algorithm is called event-driven molecular dynamics (EDMD) where a sequence of discrete events are predicted and processed. Other authors [13] have used a time-driven molecular dynamics (TDMD) approach where time is divided into small increments and, at each step time, differential equations based on Newton's law are integrated. TDMD algorithm is much easier to implement than EDMD but is far less efficient, especially for high densities.

The works of Lubachevsky and Stillinger have been extended by Donev et al. [20] to the case of non-spherical particles within an EDMD framework. In most cases, the collision time between inclusions is computed numerically. Since this step requires numerous computations, Donev et al. introduced the near-neighbor list (NNL) concept to avoid computing unnecessary collisions. Each particle has a bounding neighborhood and collision between a pair of particles is checked if their bounding neighborhoods overlap. This method is very useful for very aspherical inclusions and speeds up considerably the algorithm. The latter was applied to ellipses (2D) and ellipsoids (3D) [21]. To calculate the collision time between two ellipsoids, the authors made use of the overlap potentials [23,24]. The collision time between two moving ellipsoids is the first root of the overlap potential $F(t)$ that represents the maximum of a certain parametric function $f(t,\lambda)$, i.e., $F(t) = \max_{0 \leqslant \lambda \leqslant 1} f(t,\lambda)$. Since the maximum of $f(t,\lambda)$ cannot be computed analytically, the problem takes the form of two optimization subproblems, which can make the algorithm less computationally efficient.

Wang et al. [25] have developed an algebraic condition for defining the relative configuration of two static ellipsoids (separate, tangent or overlapping). Choi et al. [26,27] have used this condition to develop a continuous collision detection algorithm based on the Bézier clipping technique. However, their algorithm is only applicable to ellipsoids moving under rational motions (motions defined by rational function of time). Jia et al. [28] extended their work for ellipsoids moving under arbitrary motions by using a symbolic approach. In this method, the collision time between two ellipsoids is computed by finding the roots of a function. This technique is discussed in Section 4.2.

2.2. Orientation tensor

Random packings of ellipsoids are characterized by an Orientation Distribution Function that provides the probability of an inclusion to be oriented along a given vector. The nature (e.g. isotropic, transversely isotropic) of an ODF can be characterized with orientation tensors.

In a global coordinate system $O\boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$, the orientation of each ellipsoid is defined by its two Euler angles, θ and ϕ (it is assumed that the ellipsoids have an axis of revolution). θ is the angle between $\boldsymbol{e}_3$ and the ellipsoid's major axis, while ϕ is the angle between $\boldsymbol{e}_1$ and the projection of the ellipsoid's major axis in the $O\boldsymbol{e}_1\boldsymbol{e}_2$ plane. All possible orientations can be described when $0 \leqslant \theta \leqslant \pi$ and $0 \leqslant \phi \leqslant 2\pi$.

The second-order orientation tensor of a packing containing N ellipsoids is computed as follows [29]:

$$\mathrm{T}_{mn} = \langle w_m^i w_n^i \rangle \tag{1a}$$

where

$$\begin{cases} w_1^i = \sin\theta^i \cos\phi^i \\ w_2^i = \sin\theta^i \sin\phi^i \\ w_3^i = \cos\theta^i \end{cases} \tag{1b}$$

and $\langle\cdot\rangle$ denotes the unweighted average over the N ellipsoids. It should be noted that $\mathrm{tr}(\mathbf{T}) = 1$. In the case of isotropic packings, the orientation tensor is equal to $\frac{1}{3}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

3. Representation of an ellipsoid

In this section, the mathematical representation of an ellipsoid is defined. The methodology used to determine the configuration of an ellipsoid at time $(t + \Delta t)$ from its configuration at time t is also presented.

3.1. Static ellipsoid

Let i be an ellipsoid oriented in space and defined in a global coordinate system $O\boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$. Let vector $\boldsymbol{r}^i_{(t)}$ denote the position of the ellipsoid's center O' at time t. Let the local coordinate system $O\boldsymbol{e}'_1\boldsymbol{e}'_2\boldsymbol{e}'_3$ be aligned along the principal axes of i. The equation of the ellipsoid in its local coordinate system can be written as:

$$\boldsymbol{x}'^{\top}\mathbf{A}'^{i}_{(t)}\boldsymbol{x}' = 0 \tag{2a}$$

where

$$\mathbf{A}'^{i}_{(t)} = \begin{bmatrix} (a^i_{(t)})^{-2} & 0 & 0 & 0 \\ 0 & (b^i_{(t)})^{-2} & 0 & 0 \\ 0 & 0 & (c^i_{(t)})^{-2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2b}$$

and $a^i_{(t)}$, $b^i_{(t)}$ and $c^i_{(t)}$ represent the length of the semi-principal axes. It should be noted that $\boldsymbol{x}' = \{x_1, x_2, x_3, w\}^{\top}$ are the homogeneous coordinates of a point in space. It is the equivalent of the point $\boldsymbol{x}' = \{\frac{x_1}{w}, \frac{x_2}{w}, \frac{x_3}{w}\}^{\top}$ in Cartesian coordinates.

The mathematical representation of i in the global system is obtained with rotation matrices. Normalized quaternions are used for this purpose instead of Euler angles because they are more stable numerically [30]. The orientation of i at time t can be defined as a rotation of an angle $\Omega^i_{(t)}$ around an axis oriented along a unit vector $\boldsymbol{u}^i_{(t)}$. The quaternion of i, consisting of a scalar $\alpha^i_{(t)}$ and a vector $\boldsymbol{\psi}^i_{(t)}$, is written as follows:

$$\boldsymbol{q}^i_{(t)} = [\alpha^i_{(t)}, \boldsymbol{\psi}^i_{(t)}] = \left[\cos\frac{\Omega^i_{(t)}}{2}, \left(\sin\frac{\Omega^i_{(t)}}{2}\right)\boldsymbol{u}^i_{(t)}\right] \tag{3}$$

Since $\|\boldsymbol{u}^i_{(t)}\| = 1$, $\|\boldsymbol{q}^i_{(t)}\| = (\alpha^i_{(t)})^2 + \|\boldsymbol{\psi}^i_{(t)}\|^2 = 1$ (i.e. the quaternion $\boldsymbol{q}^i_{(t)}$ is normalized).

Quaternions have the property that if the ellipsoid undergoes a motion defined by the quaternion $\boldsymbol{q_1} = [\alpha_1, \boldsymbol{\psi}_1]$, followed by a movement defined by the quaternion $\boldsymbol{q_2} = [\alpha_2, \boldsymbol{\psi}_2]$, the total motion of the ellipsoid can be summarized in a single quaternion $\boldsymbol{q_{12}}$, such that:

$$\boldsymbol{q_{12}} = [\alpha_{12}, \boldsymbol{\psi_{12}}] = [\alpha_1\alpha_2 - \boldsymbol{\psi_1}^{\top}\boldsymbol{\psi_2}, \alpha_1\boldsymbol{\psi_2} + \alpha_2\boldsymbol{\psi_1} - \boldsymbol{\psi_1} \times \boldsymbol{\psi_2}] \tag{4}$$

The rotation matrix can be obtained from the quaternion as follows:

$$\mathbf{R}^i_{(t)} = \left[2\left(\alpha^i_{(t)}\right)^2 - 1\right]\mathbf{I} + 2\boldsymbol{\psi}^i_{(t)}\boldsymbol{\psi}^{i\,\top}_{(t)} + 2\alpha^i_{(t)}\mathbf{S}^i_{(t)} \tag{5a}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{S}^i_{(t)}$ is given by:

$$\mathbf{S}^i_{(t)} = \begin{bmatrix} 0 & -\psi^i_{(t)}[3] & \psi^i_{(t)}[2] \\ \psi^i_{(t)}[3] & 0 & -\psi^i_{(t)}[1] \\ -\psi^i_{(t)}[2] & \psi^i_{(t)}[1] & 0 \end{bmatrix} \tag{5b}$$

where $\psi^i_{(t)}[k]$ refers to the kth term of vector $\boldsymbol{\psi}^i_{(t)}$.

In homogeneous coordinates, the transition from $O\mathbf{e}'_1\mathbf{e}'_2\mathbf{e}'_3$ to $O\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ is performed directly by combining rotation and translation:

$$\boldsymbol{x} = \mathbf{M}^i_{(t)}\boldsymbol{x}' \tag{6a}$$

where

$$\mathbf{M}^i_{(t)} = \begin{bmatrix} \mathbf{R}^i_{(t)} & \boldsymbol{r}^i_{(t)} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \Leftrightarrow \quad \left(\mathbf{M}^i_{(t)}\right)^{-1} = \begin{bmatrix} (\mathbf{R}^i_{(t)})^\top & -(\mathbf{R}^i_{(t)})^\top \boldsymbol{r}^i_{(t)} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{6b}$$

Using Eqs. (2a) and (6a), the ellipsoid equation in the global coordinate system becomes:

$$\boldsymbol{x}^\top \mathbf{A}^i_{(t)}\boldsymbol{x} = 0 \tag{7a}$$

where

$$\mathbf{A}^i_{(t)} = \left(\mathbf{M}^i_{(t)}\right)^{-\top}\mathbf{A}'^i_{(t)}\left(\mathbf{M}^i_{(t)}\right)^{-1} \tag{7b}$$

where $(\mathbf{M}^i_{(t)})^{-\top}$ denotes the inverse of $(\mathbf{M}^i_{(t)})^\top$.

3.2. Moving and growing ellipsoid

Let i be an ellipsoid whose configuration at a given time t is determined by its position vector $\boldsymbol{r}^i_{(t)}$, its quaternion $\boldsymbol{q}^i_{(t)}$ and the lengths of its semi-principle axes $\{a^i_{(t)}, b^i_{(t)}, c^i_{(t)}\}$. Moreover, i has linear and angular velocity vectors, denoted respectively by $\boldsymbol{v}^i$ and $\boldsymbol{\omega}^i$. The objective is to find the configuration of ellipsoid i at time $(t+\Delta t)$, i.e. $\boldsymbol{r}^i_{(t+\Delta t)}$, $\boldsymbol{q}^i_{(t+\Delta t)}$ and $\{a^i_{(t+\Delta t)}, b^i_{(t+\Delta t)}, c^i_{(t+\Delta t)}\}$, while considering that i grows gradually over the time. The new position of the ellipsoid's center at that instant is given by:

$$\boldsymbol{r}^i_{(t+\Delta t)} = \boldsymbol{r}^i_{(t)} + \boldsymbol{v}^i\Delta t \tag{8}$$

The new lengths of the semi-principle axes are:

$$\begin{cases} a^i_{(t+\Delta t)} = a^i_{(t)} + a^i_0\Delta t \\ b^i_{(t+\Delta t)} = b^i_{(t)} + b^i_0\Delta t \\ c^i_{(t+\Delta t)} = c^i_{(t)} + c^i_0\Delta t \end{cases} \tag{9}$$

where a^i_0, b^i_0 and c^i_0 represent the semi-principle axes growth rates.

During the time interval Δt, the ellipsoid has rotated by an angle $\Omega^i = \|\boldsymbol{\omega}\|\Delta t$ about an axis oriented along the unit vector $\boldsymbol{u}^i = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|$. The quaternion associated with this motion, denoted by $\boldsymbol{q}_{\Delta t}$, can be written as:

$$\boldsymbol{q}^i_{\Delta t} = \left[\alpha^i_{\Delta t}, \boldsymbol{\psi}^i_{\Delta t}\right] = \left[\cos\left(\frac{\|\boldsymbol{\omega}^i\|\Delta t}{2}\right), \left(\sin\left(\frac{\|\boldsymbol{\omega}^i\|\Delta t}{2}\right)\right)\frac{\boldsymbol{\omega}^i}{\|\boldsymbol{\omega}^i\|}\right] \tag{10}$$

Using Eq. (4), the quaternion at time $(t+\Delta t)$ becomes:

$$\begin{aligned} \boldsymbol{q}^i_{(t+\Delta t)} &= \left[\alpha^i_{(t+\Delta t)}, \boldsymbol{\psi}^i_{(t+\Delta t)}\right] \\ &= \left[\alpha^i_{(t)}\alpha^i_{\Delta t} - \boldsymbol{\psi}^{i\,\top}_{(t)}\boldsymbol{\psi}^i_{\Delta t}, \alpha^i_{(t)}\boldsymbol{\psi}^i_{\Delta t} + \alpha^i_{\Delta t}\boldsymbol{\psi}^i_{(t)} - \boldsymbol{\psi}^i_{(t)} \times \boldsymbol{\psi}^i_{\Delta t}\right] \end{aligned} \tag{11}$$

By knowing the position and the quaternion at time $(t+\Delta t)$, it is possible to deduce the rotation matrix $\mathbf{R}^i_{(t+\Delta t)}$ and the transition matrix $\mathbf{M}^i_{(t+\Delta t)}$ from Eqs. (5) and (6b). Finally, the equation of the ellipsoid in the coordinate system $O\boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$ at time $(t+\Delta t)$ is given by:

$$\boldsymbol{x}^\top \mathbf{A}^i_{(t+\Delta t)}\,\boldsymbol{x} = 0 \tag{12a}$$

where

$$\mathbf{A}^i_{(t+\Delta t)} = \left(\mathbf{M}^i_{(t+\Delta t)}\right)^{-\top} \mathbf{A}'^i_{(t+\Delta t)} \left(\mathbf{M}^i_{(t+\Delta t)}\right)^{-1} \tag{12b}$$

The matrix $\mathbf{A}'^i_{(t+\Delta t)}$ is calculated with Eq. (2b) using the lengths of the semi-principal axes at time $(t+\Delta t)$ obtained with Eq. (9).

The algorithm we implemented to define the state of an ellipsoid at time $(t+\Delta t)$ from its state at time t is presented in Appendix A (see Algorithm 2).

4. Proposed new algorithm

4.1. Algorithm outline

At the beginning of the simulation, N ellipsoids are randomly created into a cube of side L. The ellipsoids volumes are initially null. Each ellipsoid has a random linear and angular velocity, as well as a random quaternion (i.e. orientation). The semi-principle axes growth rates $\{a_0, b_0, c_0\}$ are chosen such that $b_0 = a_0/R_1$ and $c_0 = a_0/R_2$, where R_1 and R_2 denotes respectively the two aspect ratios that are input in the algorithm. It was found, after trials and errors, that $a_0 = 0.1$ led to good results. The elliptical particles are then put in translational and rotational motion and their volumes gradually increase. At each step, two types of collisions are checked and computed: binary collision between two ellipsoids and collision between a particle and a cube face. If the first type of collision occurs, the linear and angular velocities of the involved particles are updated. However, if an ellipsoid intersects a cube face, its periodic image is created on the opposite side. The algorithm stops when the volume fraction $\mathcal{V}_f$ is reached.

The principal steps of the algorithm are summarized in Algorithm 1.

4.2. Collision times between ellipsoids

4.2.1. Overlap detection between two static ellipsoids

Let i and j be two ellipsoids at time t with the respective equations $\boldsymbol{x}^\top \mathbf{A}^i_{(t)}\,\boldsymbol{x} = 0$ and $\boldsymbol{x}^\top \mathbf{A}^j_{(t)}\,\boldsymbol{x} = 0$. Introduce the characteristic equation of i and j as:

$$\det\left(\lambda \mathbf{A}^i_{(t)} + \mathbf{A}^j_{(t)}\right) = p_{1(t)}\lambda^4 + p_{2(t)}\lambda^3 + p_{3(t)}\lambda^2 + p_{4(t)}\lambda + p_{5(t)} = 0 \tag{13}$$

Algorithm 3 details the computation of coefficients $p_{k(t)}$ from matrices $\mathbf{A}^i_{(t)}$ and $\mathbf{A}^j_{(t)}$ [27].

Eq. (13) is a 4th-order polynomial and has therefore four roots. Wang et al. [25] showed that two of the roots are always real and negative. In addition, the authors have established a relationship between the nature of the other two roots and the configuration of the two ellipsoids. They have shown that:

(i) i and j are separate if Eq. (13) admits two negative and two positive roots.
(ii) i and j are externally tangent if Eq. (13) admits two negative roots and a double positive root.
(iii) i and j overlap in all other cases.

Therefore, the next collision time between two moving ellipsoids is the first moment where condition (ii) is met. The problem takes the form of an eigenvalue optimization and can be relatively challenging to solve. It would be more efficient to determine the relationship between the coefficients p_k which ensures the presence of a positive double root.

Jia et al. [28] have identified a relationship between the coefficients $p_{k(t)}$ and the ellipsoids configuration (separate, externally tangent or overlap). The technique is based on the Sylvester–Habicht matrix of the characteristic equation and its first derivative [31]. Five coefficients $\{\eta_{1(t)}, \eta_{2(t)}, \eta_{3(t)}, \eta_{4(t)}, \eta_{5(t)}\}$ are computed from the coefficients $p_{k(t)}$ (see Algorithm 4). The configuration of ellipsoids i and j at time t is then determined according to the value of $\eta_{k(t)}$. All possible combinations of $\eta_{k(t)}$ are summarized in Table 1 (see [28] for more details).

4.2.2. Computing the first collision time between two moving ellipsoids

Table 1 shows that $\eta_1 = 0$ when two ellipsoids are externally tangent. Algorithm 5 details the implementation of function $\eta_{1(t+\Delta t)}$ that provides the value of coefficient η_1 for any value of Δt. Once the function $\eta_{1(t+\Delta t)}$ is implemented, the procedure to compute the first collision time is as follows. First, all the roots of $\eta_{1(t+\Delta t)}$ are computed numerically. This can be done with the algorithm of Brent [32] which combines the bisection method, the secant method and the inverse

Table 1
Relation between the coefficients $\eta_{k(t)}$ and the ellipsoids configuration at time t.

Cases	$\eta_{1(t)}$	$\eta_{2(t)}$	$\eta_{3(t)}$	$\eta_{4(t)}$	$\eta_{5(t)}$	Ellipsoids configuration
1	$=0$	>0	>0		>0	separate
2	>0	>0			>0	separate
3	$=0$	>0	<0		>0	externally tangent
4	$=0$	$=0$		<0	>0	externally tangent
5			For all other cases			overlap

quadratic interpolation. Then, all computed values of Δt are sorted in ascending order and the collision time t_c is the smallest Δt value that satisfies case 3 or 4 in Table 1.

This technique is very efficient since computing the roots of function $\eta_{1(t+\Delta t)}$ is usually very fast, and in most cases, there are only two.

4.2.3. Contact point between the two ellipsoids

Once t_c is computed, it is possible to find the four roots of the characteristic equation $\det(\lambda \mathbf{A}^i_{(t+t_c)} + \mathbf{A}^j_{(t+t_c)}) = 0$, corresponding to the eigenvalues of the matrix $(\mathbf{A}^i_{(t+t_c)})^{-1}\mathbf{A}^j_{(t+t_c)}$. Among these four roots, there is a positive double root denoted by λ_0. The homogeneous coordinates of the point of contact $\boldsymbol{x}_c$ are the solution of the following equation [27]:

$$\left(\lambda_0 \mathbf{A}^i_{(t+t_c)} - \mathbf{A}^j_{(t+t_c)}\right)\boldsymbol{x}_c = 0 \tag{14}$$

where $\boldsymbol{x}_c$ corresponds to the eigenvector of the matrix $(\mathbf{A}^i_{(t+t_c)})^{-1}\mathbf{A}^j_{(t+t_c)}$ associated with the eigenvalue λ_0.

4.2.4. Improving the computation time of binary collisions

At each iteration, all binary collision times are saved. During an iteration, the collision time between two ellipsoids is calculated only if the velocities of at least one ellipsoid have changed since the last iteration or if one ellipsoid is a new periodic image that was created in the previous iteration. Indeed, if the velocities of a pair of ellipsoids has not changed, the collision time between this pair can be derived from the collision time obtained at the previous iteration. It suffices to subtract from the latter the time spent between the last two iterations.

Furthermore, the algorithm for finding the next binary collisions can be improved if it is possible to avoid computing collisions that are unlikely to occur. First, the concept of bounding spheres is introduced. Two ellipsoids cannot collide if their respective bounding spheres will never overlap. At time t, the bounding sphere of ellipsoid i, denoted by BS_i, has the same position $\boldsymbol{r}^i_{(t)}$ and the same linear velocity $\boldsymbol{v}^i$ as the latter. The radius and the radius growth rate of BS_i are respectively given by $R^i_{(t)} = \max(a^i_{(t)}, b^i_{(t)}, c^i_{(t)})$ and $R^i_0 = \max(a^i_0, b^i_0, c^i_0)$. Two bounding spheres BS_i and BS_j are in contact if:

$$\left\|\boldsymbol{r}^i_{(t)} - \boldsymbol{r}^j_{(t)}\right\| \leqslant R^i_{(t)} + R^j_{(t)} \tag{15}$$

If Eq. (15) is satisfied, the collision time between the two bounding spheres, denoted by t_c^{BS}, is set to 0. Otherwise, the collision time between BS_i and BS_j is given by [4]:

$$t_c^{BS} = \begin{cases} \gamma_1^{-1}[-\gamma_2 - \sqrt{\gamma_2^2 - \gamma_1\gamma_3}] & \text{if } (\gamma_2 \leqslant 0 \text{ or } \gamma_1 < 0) \text{ and } \gamma_2^2 - \gamma_1\gamma_3 \geqslant 0 \\ \text{No collision} & \text{if } (\gamma_2 > 0 \text{ and } \gamma_1 \geqslant 0) \text{ or } \gamma_2^2 - \gamma_1\gamma_3 < 0 \end{cases} \tag{16a}$$

where

$$\begin{cases} \gamma_1 = \left\|\boldsymbol{v}^i - \boldsymbol{v}^j\right\|^2 - \left(R^i_0 + R^j_0\right)^2 \\ \gamma_2 = \left(\boldsymbol{r}^i_{(t)} - \boldsymbol{r}^j_{(t)}\right)^\top\left(\boldsymbol{v}^i - \boldsymbol{v}^j\right) - \left(R^i_{(t)} + R^j_{(t)}\right)\left(R^i_0 + R^j_0\right) \\ \gamma_3 = \left\|\boldsymbol{r}^i_{(t)} - \boldsymbol{r}^j_{(t)}\right\|^2 - \left(R^i_{(t)} + R^j_{(t)}\right)^2 \end{cases} \tag{16b}$$

Collision between ellipsoids i and j is computed only if BS_i and BS_j overlap or will collide after a certain time. The main advantage of using the bounding spheres method is that the collision between two spheres is computed analytically unlike ellipsoids where the collision time is obtained numerically. In addition, the algorithm stores the minimum collision time ($t_{\min}$) obtained whenever a pair of ellipsoids has been checked. Therefore, the computation of collisions between ellipsoids i and j is performed if the collision time between BS_i and BS_j is lower than the minimum time stored, i.e. $t_c^{BS} < t_{\min}$.

The algorithm uses also the near-neighbor list (NNL) concept introduced by Donev et al. [20]. The neighborhood of an ellipsoid i, denoted by N_i, is also an ellipsoid which is concentric with i and has the same parameters as the latter. The only difference lies in the length of the semi-principle axes of N_i which are equal to $\{\mu a^i_{(t)}, \mu b^i_{(t)}, \mu c^i_{(t)}\}$, where μ is a scale factor. The value of μ was set to 1.2 in the algorithm. It is possible to reduce the value of μ during the course of the simulation such that $\mu \to 1$ near the jamming point. The collision between a pair of ellipsoids i and j is checked only if their respective neighborhood N_i and N_j overlap or are externally tangent. This reduces considerably the number of collisions to compute. Overlap between N_i and N_j at time t is checked analytically by computing their five coefficients $\eta_k(t)$ using Algorithm 5 and by deducing their configuration with Table 1.

Algorithm 6 details the next binary collision time computation while considering the bounding spheres and the NNL concepts. The algorithm provides also the coordinates of the contact point $\boldsymbol{x}_c$.

4.3. Collision time between ellipsoids and the cube faces

4.3.1. Intersection between a stationary ellipsoid and a cube face

Let i be an ellipsoid at time t defined by the equation $\boldsymbol{x}^\top \mathbf{A}^i_{(t)}\, \boldsymbol{x} = 0$. $\mathbf{A}^i_{(t)}$ can be written as:

$$\mathbf{A}^i_{(t)} = \begin{bmatrix} \mathbf{B}^i_{(t)} & \frac{1}{2}\boldsymbol{d}^i_{(t)} \\ \frac{1}{2}\boldsymbol{d}^{i\,\top}_{(t)} & F^i_{(t)} \end{bmatrix} \tag{17}$$

where $\mathbf{B}^i_{(t)}$ is a 3×3 matrix and $\boldsymbol{d}^i_{(t)}$ is a 3×1 vector. The ellipsoid equation becomes:

$$\boldsymbol{x}^\top \mathbf{B}^i_{(t)} \boldsymbol{x} + \boldsymbol{d}^{i\,\top}_{(t)} \boldsymbol{x} + F^i_{(t)} = 0 \tag{18}$$

where $\boldsymbol{x}$ denotes Cartesian coordinates. It should be noted that $\mathbf{B}^i_{(t)}$ is symmetric and positive definite since Eq. (18) is the equation of an ellipsoid.

Computation of the intersection equation between i and a cube face $x_k = \beta$ ($k \in \{1, 2, 3\}$ and $\beta \in \{0, L\}$) can be accomplished through a change of variables from 3D coordinates ($\boldsymbol{x}$) to 2D coordinates ($\boldsymbol{x}^*$). The transformation can be written as follows:

$$\boldsymbol{x} = \mathbf{P}\boldsymbol{x}^* + \boldsymbol{p} \tag{19a}$$

where

$$\begin{cases} \boldsymbol{x} = \{x_1, x_2, x_3\}^\top \\ \boldsymbol{x}^* = \{x_l, x_m\}^\top \\ \mathbf{P} = [\boldsymbol{e}_l, \boldsymbol{e}_m] \\ \boldsymbol{p} = \beta \boldsymbol{e}_k \end{cases} \quad l, m \in \{1, 2, 3\};\ l < m;\ l \neq k \neq m \tag{19b}$$

For example, if the cube face equation is $x_1 = L$, Eq. (19a) becomes:

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x_2 \\ x_3 \end{Bmatrix} + \begin{Bmatrix} L \\ 0 \\ 0 \end{Bmatrix} \tag{20}$$

Combination of Eqs. (18) and (19a) yields the portion of i that intersect the cube's face and leads to:

$$g_{(t)}(\boldsymbol{x}^*) = \boldsymbol{x}^{*\top} \mathbf{B}^{i\,*}_{(t)} \boldsymbol{x}^* + \boldsymbol{d}^{i\,*\,\top}_{(t)} \boldsymbol{x}^* + F^{i\,*}_{(t)} = 0 \tag{21a}$$

where

$$\begin{cases} \mathbf{B}^{i\,*}_{(t)} = \mathbf{P}^\top \mathbf{B}^i_{(t)} \mathbf{P} \\ \boldsymbol{d}^{i\,*}_{(t)} = 2\mathbf{P}^\top \mathbf{B}^i_{(t)} \boldsymbol{p} + \mathbf{P}^\top \boldsymbol{d}^i_{(t)} \\ F^{i\,*}_{(t)} = \boldsymbol{p}^\top \mathbf{B}^i_{(t)} \boldsymbol{p} + \boldsymbol{d}^{i\,\top}_{(t)} \boldsymbol{p} + F^i_{(t)} \end{cases} \tag{21b}$$

The intersection of an ellipsoid with a plane is an ellipse, a point or the empty set. To determine the nature of the intersection, it is necessary to identify the number of solutions that satisfy Eq. (21a). Since $\mathbf{B}^{i\,*}_{(t)}$ is positive definite, $g_{(t)}(\boldsymbol{x}^*)$ is a quadratic convex function which has a global minimum. The minimum of the function, denoted by $\boldsymbol{x}^*_{\min}$, is derived by setting $\nabla_{\boldsymbol{x}^*}[g_{(t)}(\boldsymbol{x}^*)] = 0$ and leads to:

$$\boldsymbol{x}^*_{\min} = -\frac{1}{2}\left(\mathbf{B}^{i\,*}_{(t)}\right)^{-1} \boldsymbol{d}^{i\,*}_{(t)} \tag{22}$$

The minimum value of $g_{(t)}(\boldsymbol{x}^*)$, denoted by $g^\dagger_{(t)}$, is therefore given by:

$$g^\dagger_{(t)} = g_{(t)}(\boldsymbol{x}^*_{\min}) = -\frac{1}{4}\boldsymbol{d}^{i\,*\,\top}_{(t)} \left(\mathbf{B}^{i\,*}_{(t)}\right)^{-1} \boldsymbol{d}^{i\,*}_{(t)} + F^{i\,*}_{(t)} \tag{23}$$

The intersection type depends on the value of $g^{\dagger}_{(t)}$:

(i) If $g^{\dagger}_{(t)} < 0$, $g_{(t)}(\boldsymbol{x}^*)$ crosses zero at various points. Therefore Eq. (21a) has several solutions and the intersection is an ellipse.
(ii) If $g^{\dagger}_{(t)} = 0$, $g_{(t)}(\boldsymbol{x}^*)$ crosses zero at a single point. Therefore Eq. (21a) has a unique solution and the intersection is a point.
(iii) If $g^{\dagger}_{(t)} > 0$, $g_{(t)}(\boldsymbol{x}^*)$ does not cross zero. Therefore Eq. (21a) has no solution and the intersection is the empty set.

4.3.2. *First collision time between an ellipsoid and a cube face*

The collision time between an ellipsoid i and a cube face can be computed by determining the first value t_s such that $g^{\dagger}_{(t+t_s)} = 0$. First, the function $g^{\dagger}_{(t+\Delta t)}$ is implemented (see Algorithm 7). The roots of $g^{\dagger}_{(t+\Delta t)}$ are then computed using the Brent algorithm and the smallest Δt value corresponds to the next collision time.

4.3.3. *Improvement of the next collision time computation efficiency*

As in the case of binary collisions, the collision time between an ellipsoid and a cube face is computed only if the velocities of the ellipsoid have changed since the previous iteration. It is also possible to improve the performance of the algorithm by using the concept of bounding spheres. An ellipsoid i cannot collide with a cube face if its bounding sphere BS_i will never touch this face. At time t, BS_i is in contact with the cube face $x_k = \beta$ if:

$$\begin{cases} r^i_{(t)}[k] - R^i_{(t)} \leqslant \beta & \text{if } \beta = 0 \\ r^i_{(t)}[k] + R^i_{(t)} \geqslant \beta & \text{if } \beta = L \end{cases} \tag{24}$$

If Eq. (24) is satisfied, the collision time between BS_i and the cube face $x_k = \beta$, denoted by t^{BS}_s, is set to 0. Otherwise, t^{BS}_s is given by [4]:

$$t^{BS}_s = \begin{cases} [R^i_{(t)} - r^i_{(t)}[k]][v^i_{(t)}[k] - R^i_0]^{-1} & \text{if } \beta = 0 \\ [\beta - R^i_{(t)} - r^i_{(t)}[k]][v^i_{(t)}[k] + R^i_0]^{-1} & \text{if } \beta = L \end{cases} \tag{25}$$

Given that the algorithm stores the minimum time ($t_{\min}$) obtained whenever a collision between an ellipsoid and a cube face has been processed, the collision between an ellipsoid i and a cube face $x_k = \beta$ is computed only if $t^{BS}_s < t_{\min}$.

Algorithm 8 details the computation of the next collision time between an ellipsoid and a cube face while considering the bounding sphere concept.

4.4. *Updating linear and angular velocities after impact*

Let i and j be two ellipsoids that collide at time $t = t_c$. The linear ($\boldsymbol{v}^{i-}$ and $\boldsymbol{v}^{j-}$) and angular ($\boldsymbol{\omega}^{i-}$ and $\boldsymbol{\omega}^{j-}$) velocities before collision are known. The objective is to determine the linear and angular velocities after impact, i.e. $\boldsymbol{v}^{i+}$, $\boldsymbol{v}^{j+}$, $\boldsymbol{\omega}^{i+}$ and $\boldsymbol{\omega}^{j+}$. A system of 12 equations should be defined in order to find these 12 unknowns. Friction is assumed negligible at the contact surface.

First, the unit normal vector $\boldsymbol{n}$ at the contact point $\boldsymbol{x}_c$ is computed. $\boldsymbol{n}$ is defined as going from i to j (see Fig. 1) and is given by:

$$\boldsymbol{n} = \frac{\nabla(\boldsymbol{x}^\top \mathbf{A}^i_{(t)} \boldsymbol{x})}{\|\nabla(\boldsymbol{x}^\top \mathbf{A}^i_{(t)} \boldsymbol{x})\|} = \frac{2\mathbf{B}^i_{(t)}\boldsymbol{x}_c + \boldsymbol{d}^i_{(t)}}{\|2\mathbf{B}^i_{(t)}\boldsymbol{x}_c + \boldsymbol{d}^i_{(t)}\|} \tag{26}$$

where $\mathbf{B}^i_{(t)}$ and $\boldsymbol{d}^i_{(t)}$ are computed from Eq. (17)

Two other unit vectors $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ are defined such that $\boldsymbol{t_1}$, $\boldsymbol{t_2}$ and $\boldsymbol{n}$ form an orthonormal basis in $\mathbb{R}^3$. The kinematics theory used to model the impact is based on the linear and angular momentum conservation and is detailed below.

4.4.1. *Ellipsoid mass and moment of inertia*

Assuming a unit density, the mass and the moment of inertia of an ellipsoid i are given by:

$$m^i_{(t)} = \frac{4}{3}\pi a^i_{(t)} b^i_{(t)} c^i_{(t)} \tag{27a}$$

$$\mathbf{H}'^i_{(t)} = \frac{m^i_{(t)}}{5}\begin{bmatrix} (b^i_{(t)})^2 + (c^i_{(t)})^2 & 0 & 0 \\ 0 & (a^i_{(t)})^2 + (c^i_{(t)})^2 & 0 \\ 0 & 0 & (a^i_{(t)})^2 + (b^i_{(t)})^2 \end{bmatrix} \tag{27b}$$

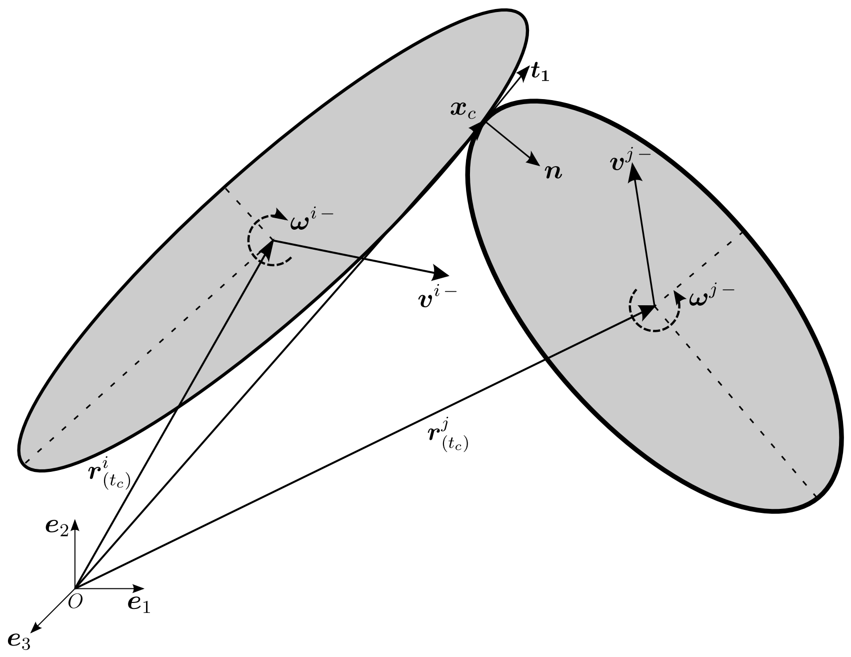

Fig. 1. Two colliding ellipsoids at time $t_c - \delta t$. $\boldsymbol{n}$ is defined as going from i to j. Vector $\boldsymbol{t_2}$ (not shown) is perpendicular to $\boldsymbol{n}$ and $\boldsymbol{t_1}$ such that $\boldsymbol{t_1} \times \boldsymbol{t_2} = \boldsymbol{n}$.

The moment of inertia given by Eq. (27b) is expressed in the ellipsoids local coordinate system. The rotation matrix $\mathbf{R}^i_{(t)}$ can be used to calculate the moment of inertia expression in the global coordinate system as:

$$\mathbf{H}^i_{(t)} = \mathbf{R}^i_{(t)} \mathbf{H}'^i_{(t)} \left(\mathbf{R}^i_{(t)}\right)^\top \tag{28}$$

4.4.2. Governing equations

For each ellipsoid, the linear momentum along $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ is conserved during the impact. This allows to write the following four equations:

$$m^r_{(t)} \boldsymbol{v}^{r+\top} \boldsymbol{t_1} = m^r_{(t)} \boldsymbol{v}^{r-\top} \boldsymbol{t_1} \quad r = \{i, j\} \tag{29a}$$

$$m^r_{(t)} \boldsymbol{v}^{r+\top} \boldsymbol{t_2} = m^r_{(t)} \boldsymbol{v}^{r-\top} \boldsymbol{t_2} \quad r = \{i, j\} \tag{29b}$$

In addition, the total linear momentum along $\boldsymbol{n}$ is conserved:

$$m^i_{(t)} \boldsymbol{v}^{i+\top} \boldsymbol{n} + m^j_{(t)} \boldsymbol{v}^{j+\top} \boldsymbol{n} = m^i_{(t)} \boldsymbol{v}^{i-\top} \boldsymbol{n} + m^j_{(t)} \boldsymbol{v}^{j-\top} \boldsymbol{n} \tag{30}$$

Furthermore, the angular momentum about the contact point $\boldsymbol{x}_c$ is conserved during the impact for each ellipsoid. It is then possible to define six new equations:

$$\mathbf{H}^r_{(t)} \boldsymbol{\omega}^{r+} + m^r_{(t)} \left[\left(\boldsymbol{r}^r_{(t)} - \boldsymbol{x}_c\right) \times \boldsymbol{v}^{r+}\right] = \mathbf{H}^r_{(t)} \boldsymbol{\omega}^{r-} + m^r_{(t)} \left[\left(\boldsymbol{r}^r_{(t)} - \boldsymbol{x}_c\right) \times \boldsymbol{v}^{r-}\right] \quad r = \{i, j\} \tag{31}$$

The last equation is obtained by applying the coefficient of restitution along $\boldsymbol{n}$ while considering the effect of the ellipsoids growth rate. This equation can be written as:

$$\boldsymbol{v}^{c+\top} \boldsymbol{n} = -e\left(\boldsymbol{v}^{c-\top} \boldsymbol{n}\right) - 2\left[\max\left(a^i_0, b^i_0, c^i_0\right) + \max\left(a^j_0, b^j_0, c^j_0\right)\right] \tag{32a}$$

where $e = 1$ for a perfect elastic collision. $\boldsymbol{v}^{c-}$ and $\boldsymbol{v}^{c+}$ denote the closing velocities of the contact point of i with respect to that of j, before and after collision respectively:

$$\begin{cases} \boldsymbol{v}^{c-} = \left[\boldsymbol{v}^{i-} + \boldsymbol{\omega}^{i-} \times \left(\boldsymbol{x}_c - \boldsymbol{r}^i_{(t)}\right)\right] - \left[\boldsymbol{v}^{j-} + \boldsymbol{\omega}^{j-} \times \left(\boldsymbol{x}_c - \boldsymbol{r}^j_{(t)}\right)\right] \\ \boldsymbol{v}^{c+} = \left[\boldsymbol{v}^{i+} + \boldsymbol{\omega}^{i+} \times \left(\boldsymbol{x}_c - \boldsymbol{r}^i_{(t)}\right)\right] - \left[\boldsymbol{v}^{j+} + \boldsymbol{\omega}^{j+} \times \left(\boldsymbol{x}_c - \boldsymbol{r}^j_{(t)}\right)\right] \end{cases} \tag{32b}$$

Eqs. (29), (30), (31) and (32) form a system of 12 linear equations that can be easily solved to obtain $\boldsymbol{v}^{i+}$, $\boldsymbol{v}^{j+}$, $\boldsymbol{\omega}^{i+}$ and $\boldsymbol{\omega}^{j+}$.

Algorithm 9 summarizes the steps needed to compute the velocities after collision.

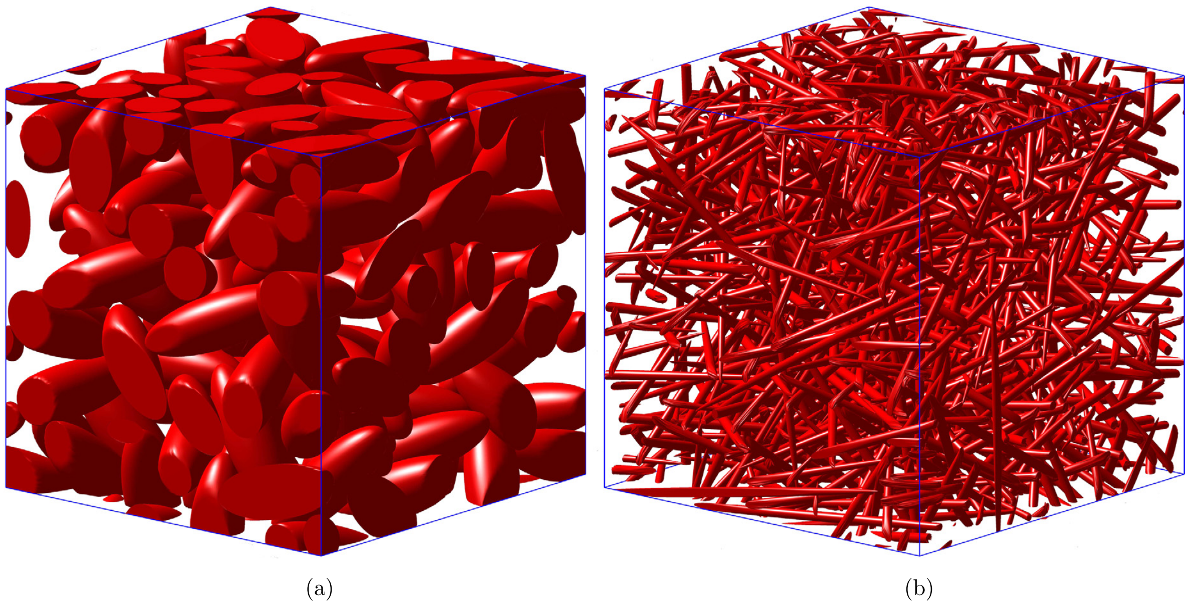

Fig. 2. Random packings of prolate ellipsoids. (a) $N = 100$, $R_1 = 3$, $R_2 = 3$, $\mathcal{V}_f = 40\%$. (b) $N = 300$, $R_1 = 40$, $R_2 = 40$, $\mathcal{V}_f = 10\%$.

4.5. *Post-collision with the cube faces: creation of periodic ellipsoids*

When an ellipsoid intersects with a cube face, its periodic image must appear on the opposite side. The number of periodic images varies depending on the number of faces that intersect ellipsoid i. Each periodic image of i has the same quaternion and the same linear and angular velocities as i. The offset between i and one of its periodic particles is given by the vector $\beta \boldsymbol{e}_k$, where $\beta \in \{L, -L\}$ and $k \in \{1, 2, 3\}$, depending on which face the periodic particle appears.

Algorithm 10 describes the creation of periodic particles when an ellipsoid is in contact with one or more cube faces.

5. Orientations distributions of the generated packings

Orientation tensors were computed using Eqs. (1) to verify the orientations distributions in the generated packings. θ and ϕ were obtained from the quaternions. If an ellipsoid i has the quaternion $\boldsymbol{q}^i = [\alpha^i, \boldsymbol{\psi}^i]$, its angles θ^i and ϕ^i are given by:

$$\theta^i = \frac{\pi}{2} - \sin^{-1}\left(2\left(\alpha^i \psi^i[2] - \psi^i[1]\psi^i[3]\right)\right) \tag{33a}$$

$$\phi^i = \tan 2^{-1}\left(2\left(\alpha^i \psi^i[3] + \psi^i[1]\psi^i[2]\right), 1 - 2\left(\psi^i[2]^2 + \psi^i[3]^2\right)\right) \tag{33b}$$

Define the ratio $\rho^n = \lambda^n_{\max}/\lambda^n_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ denotes respectively the maximum and minimum eigenvalue of the orientation tensor and n a given realization of a random packing. Let $\bar{\rho}$ be the average value of the ρ^n. In the sequel, an ensemble of packings was considered isotropic when $\bar{\rho} \approx 1$ since $\rho^n \geqslant 1$.

6. Results and discussion

6.1. *Examples of ellipsoids packings*

Figs. 2, 3, 4 and 5 show several examples of periodic packings generated with the algorithm presented in this paper. Fig. 2(a) shows a packing containing 100 prolate ellipsoids ($R_1 = R_2 = 3$) with a volume fraction of 40%. Prolate ellipsoids are also shown in Fig. 2(b), but where the aspect ratios are much higher ($R_1 = R_2 = 40$) and $\mathcal{V}_f = 10\%$. Fig. 3 presents two examples of oblate ellipsoids. Fig. 3(a) shows a packing containing 100 ellipsoids occupying 40% of the space and with aspect ratios $R_1 = 3$ and $R_2 = 1$ while Fig. 3(b) illustrates a packing containing 50 ellipsoids particles with aspect ratios $R_1 = 20$ and $R_2 = 1$ and a volume fraction of 15%. The algorithm can also generate ellipsoids that do not have an axis of revolution (asymmetric or scalene ellipsoids) as shown in Fig. 4. Fig. 4(a) shows 100 scalene ellipsoids with a volume fraction of 30% and aspect ratios of $R_1 = 2$ and $R_2 = 5$. Fig. 4(b) shows a packing containing 1000 asymmetric ellipsoids at a volume fraction of 20% where $R_1 = 4$ and $R_2 = 0.5$. Finally, it is possible to generate ellipsoids particles with random aspect ratios. Fig. 5(a) shows 50 polydisperse ellipsoids with a volume fraction of 30% and where R_1 and R_2 range from 1 to 5. Fig. 5(b) illustrates a packing of 125 ellipsoids occupying 20% of the space and where $0.1 \leqslant R_1, R_2 \leqslant 2$.

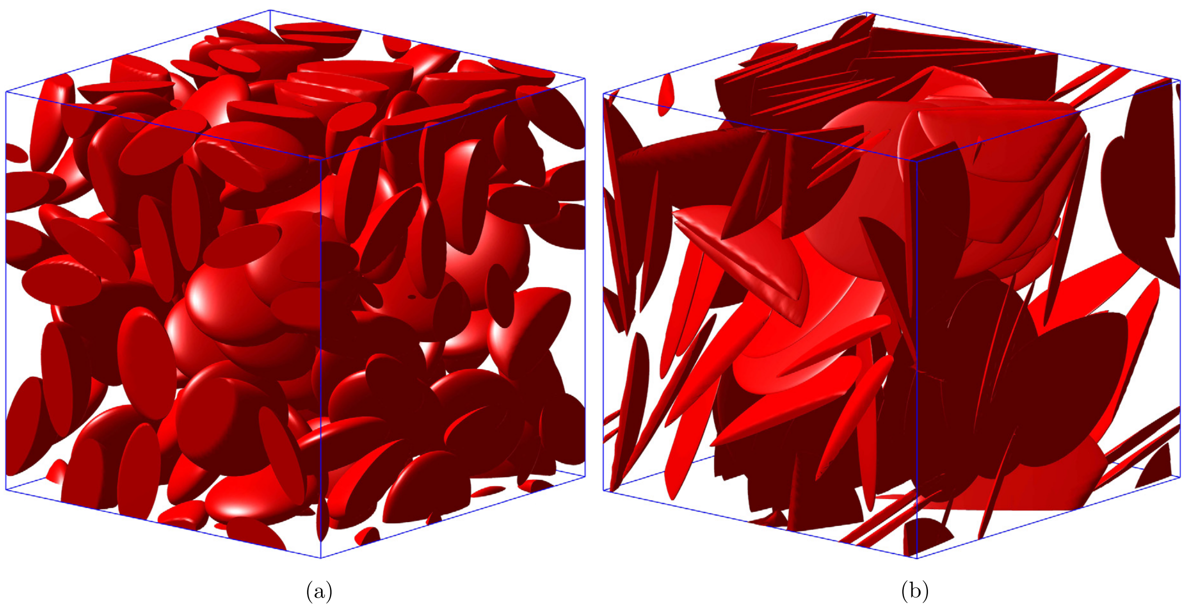

Fig. 3. Random packings of oblate ellipsoids. (a) $N = 100$, $R_1 = 3$, $R_2 = 1$, $\mathcal{V}_f = 40\%$. (b) $N = 50$, $R_1 = 20$, $R_2 = 1$, $\mathcal{V}_f = 15\%$.

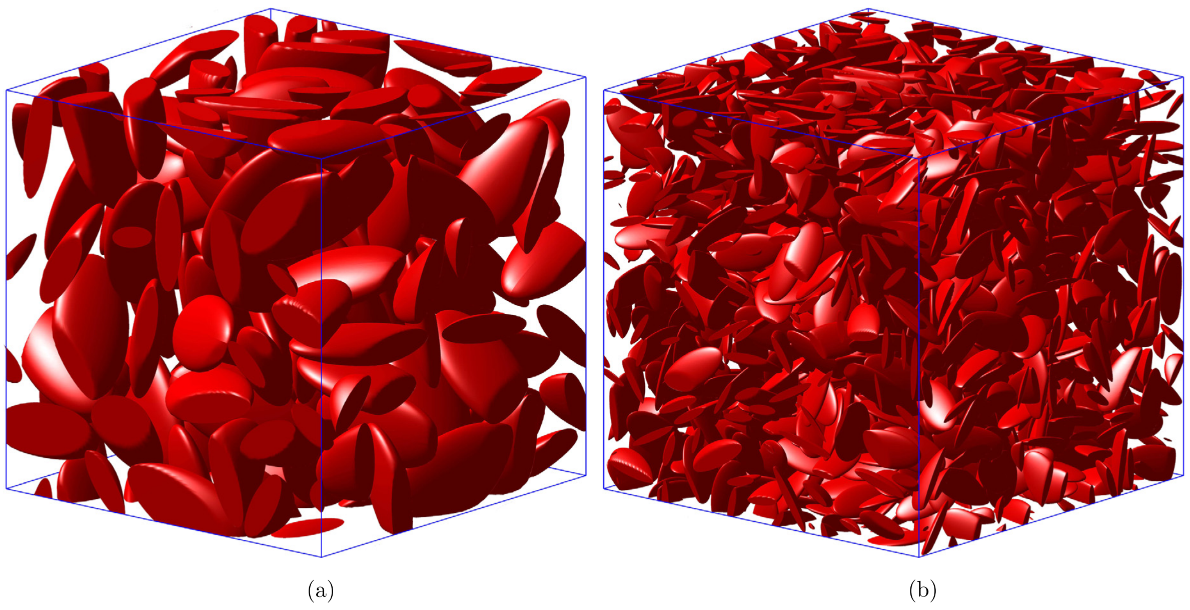

Fig. 4. Random packings of scalene ellipsoids. (a) $N = 100$, $R_1 = 2$, $R_2 = 5$, $\mathcal{V}_f = 30\%$. (b) $N = 1000$, $R_1 = 4$, $R_2 = 0.5$, $\mathcal{V}_f = 20\%$.

6.2. Orientations dispersion in the generated packings

The distribution of the packings were characterized with the orientation tensors. Two cases were considered. In the first, the ellipsoids had aspect ratios $R_1 = 2$ and $R_2 = 2$ and the volume fraction was set to 30%. In the second case, the ellipsoids occupied 10% of the space and had higher aspect ratios: $R_1 = 10$ and $R_2 = 10$. For each case, six ellipsoids numbers N were studied, namely $\{10, 25, 50, 100, 200, 500\}$.

Fig. 6 shows the evolution of the ratio $\bar{\rho}$ as a function of N and 10 realizations. For both cases, it can be seen that $\bar{\rho} \to 1$ when N is large enough. In addition, the 95% confidence intervals are tighter as N increases. This shows that the packings generated by the algorithm are totally random and the ellipsoids orientations dispersion is uniform, even for higher aspect ratios. This will remain true even if $R_1, R_2 > 10$. Indeed, when the aspect ratios are high, it is necessary to generate smaller

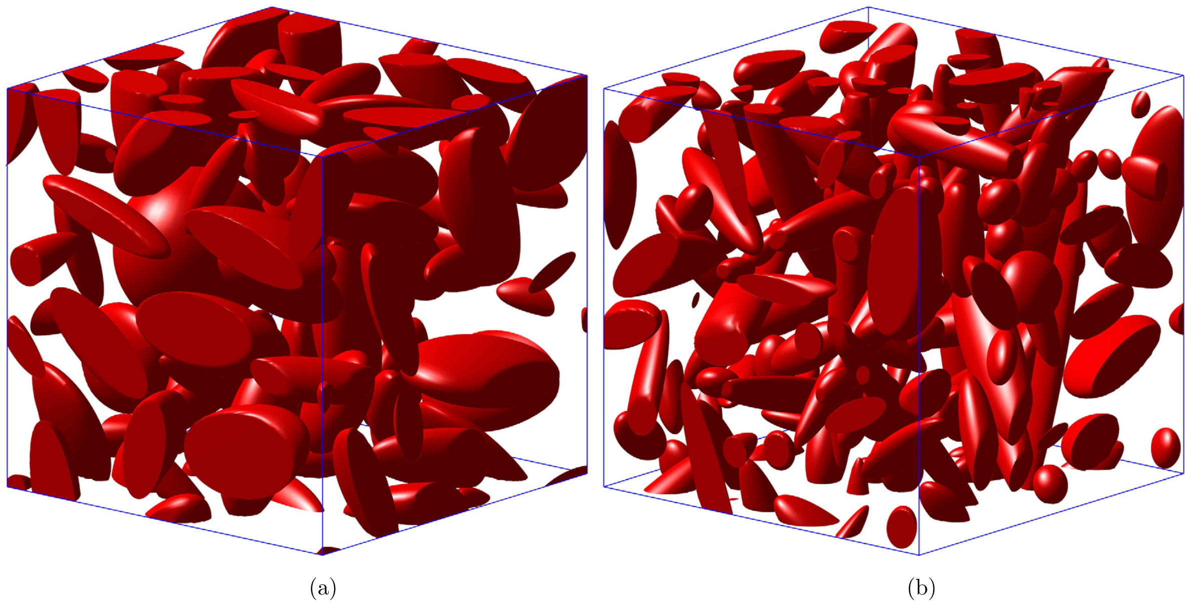

Fig. 5. Random packings of polydisperse ellipsoids. (a) $N = 50$, $1 \leqslant R_1, R_2 \leqslant 5$, $\mathcal{V}_f = 30\%$. (b) $N = 125$, $0.1 \leqslant R_1, R_2 \leqslant 2$, $\mathcal{V}_f = 20\%$.

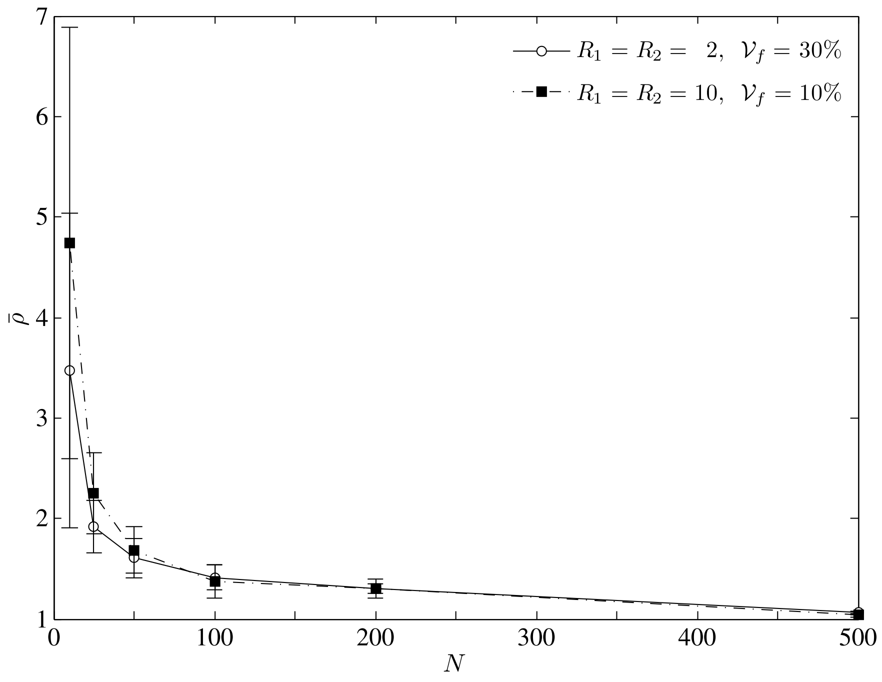

Fig. 6. Ratio $\bar{\rho}$ as a function of the ellipsoids number N. Each point represents the mean value obtained for 10 random realizations. 95% confidence intervals are shown.

particles to obtain a uniform orientations distribution, and therefore, a greater number of particles for a fixed volume fraction.

6.3. *Algorithm performance*

Table 2 gives the maximum volume fraction reached as a function of aspect ratios R_1 and R_2 for prolate and oblate ellipsoids. During a simulation, it is considered that the maximum volume fraction is reached when the increase in the latter, after 50 successive iterations, is less than 0.1%. It is obvious that the maximum volume fraction reached decreases

Table 2
Maximum volume fraction reached as a function of aspect ratios R_1 and R_2. For each case, the minimum number of ellipsoids necessary to obtain a uniform distribution of orientations is given. Approximative CPU times (averaged over 10 runs) are also indicated.

R_1	Prolate ellipsoids ($R_2 = R_1$)			Oblate ellipsoids ($R_2 = 1$)		
	$\mathcal{V}_f$	N	t_{CPU} (s)	$\mathcal{V}_f$	N	t_{CPU} (s)
2	60%	30	54.3	60%	30	52.7
5	55%	45	69.9	55%	45	65.2
10	30%	90	48.6	30%	60	39.3
20	20%	175	112.5	20%	75	47.8
30	10%	175	104.9	10%	90	53.7
40	10%	310	181.3	10%	100	92.8

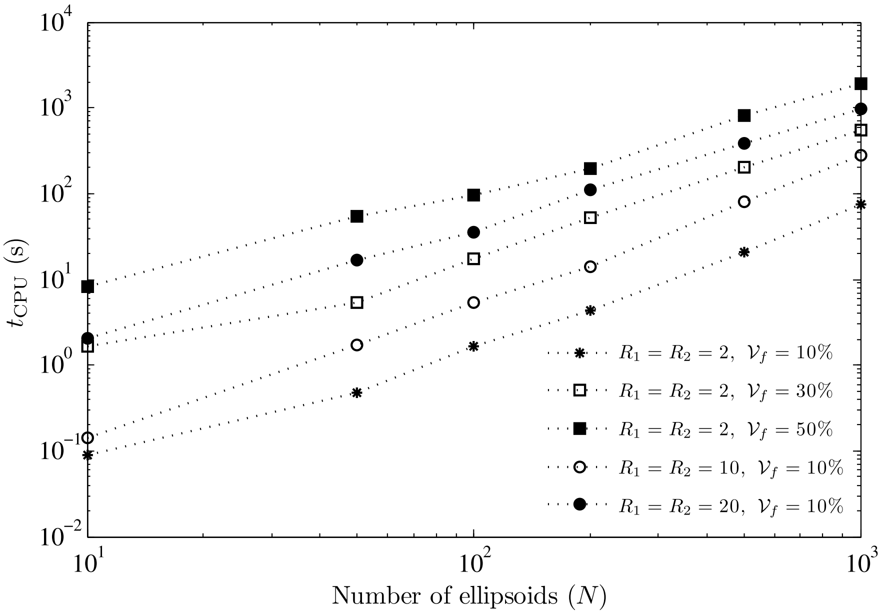

Fig. 7. CPU time (averaged over 10 realizations) as a function of the ellipsoids number (prolate).

when the aspect ratios increase. In addition, the volume fractions limit values are identical for both prolate and oblate ellipsoids. When $R_1, R_2 \leqslant 10$, results are similar to those reported by Donev et al. [21]. However, the authors did not present any results for which $R_1, R_2 > 10$. This is the first time that such results are reported in the literature.

Table 2 also provides, for each packing, the minimum number of particles required to obtain a uniform distribution of orientations. These values have been determined by varying the number of particles and by computing, for each number, the ratio ρ^n for each of the 10 random realizations. The number of particle N was considered sufficient when $\bar{\rho}$ was close to 1 (see Sections 5 and 6.2 for more details). An average CPU time (t_{CPU}) is also given for each combination of aspect ratios and volume fraction.

Donev et al. [21] reported that the binary collision computation took about 2 ms using Fortran 95 on a 1666 MHz Athlon running Linux. The proposed algorithm uses about 0.07 ms per ellipsoid binary collision using MATLAB 2011a on an Intel i7 Quad Core, 1.60 GHz, 8 GB RAM. The time saving between the two algorithm is much more important than the improvement in terms of processor. This is due to the fact that in the proposed algorithm, the collision time is obtained by finding the zero of a function while in that proposed by Donev et al., the collision time is obtained by solving two optimization subproblems. Finally, the proposed algorithm takes about 0.02 ms to compute the collision time between an ellipsoid and a cube face.

To better evaluate the influence of various parameters on the algorithm computation time, Figs. 7, 8, and 9 show the variation of CPU time (averaged over 10 runs) with the number of particles, the volume fraction and the aspect ratio. Results are only presented for prolate ellipsoids since similar results were obtained for oblate ellipsoids. Fig. 7 shows t_{CPU} as a function of the number of ellipsoids, for different combinations of volume fractions and aspect ratios. It seems that t_{CPU} varies linearly (on a log–log scale) with N, for $N > 100$. The average slope for the five different combinations is 1.5. Thus, $t_{\text{CPU}} \propto N^{1.5}$. Fig. 8 shows t_{CPU} as a function of the volume fraction, for a fixed aspect ratio ($R_1 = R_2 = 2$). A trend similar to that shown in Fig. 7 is observed, whatever the number of particles N. However, the average slope is approximately 2.6, which means that $t_{\text{CPU}} \propto {\mathcal{V}_f}^{2.6}$. Therefore, t_{CPU} seems to be more sensitive to the volume fraction than to the number

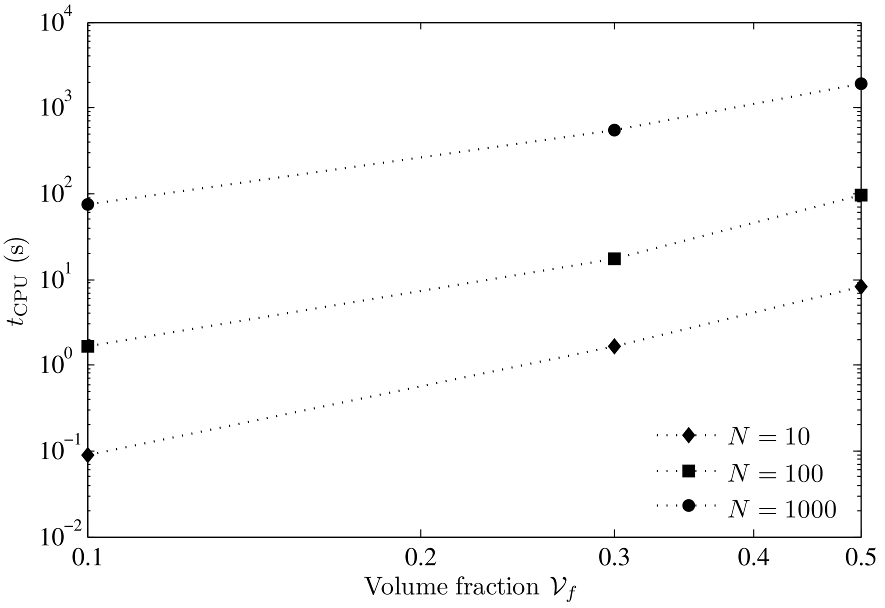

Fig. 8. CPU time (averaged over 10 realizations) as a function of the volume fraction for prolate ellipsoids. $R_1 = 2$, $R_2 = 2$.

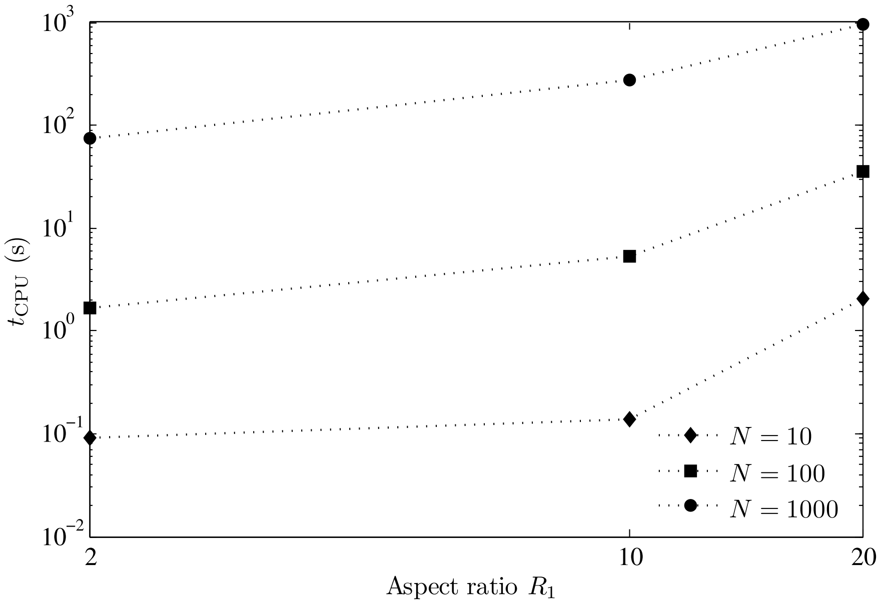

Fig. 9. CPU time (averaged over 10 realizations) as a function of the aspect ratio for prolate ellipsoids. $\mathcal{V}_f = 10\%$.

of particles (for the studied ranges). On the other hand, Fig. 9 shows that when $R_1 \leqslant 10$, $t_{\mathrm{CPU}} \propto {R_1}^{0.6}$. However, for high aspect ratios (i.e., $R_1 > 10$), $t_{\mathrm{CPU}} \propto {R_1}^{2.8}$. One could conclude that the aspect ratio has a negligible effect on the CPU time when its value is low (i.e. $\leqslant 10$). This parameter becomes as dominant as the volume fraction when its value is quite high (i.e., > 10).

7. Conclusion

The main contribution of this paper is a detailed computationally-efficient MD algorithm for generating random packings of periodic ellipsoids. The specific contributions of this paper are:

- The binary collision times are computed by simply finding the roots of a non-linear function, which is a more efficient and simple technique than that presented by Donev et al. [20,21].
- The paper puts more emphasis on periodic packings and presents a novel and efficient technique to compute the collision time between an ellipsoid and a cell face.
- The necessary steps for processing the impact between two ellipsoids are well established and can be used for any types of ellipsoids (prolate, oblate, scalene).
- It is the first time that ellipsoids packings with high aspect ratios (i.e., $R_1, R_2 > 10$) are presented in the literature.
- The algorithm is comprehensive and well documented. Detailed pseudo-codes are given so the algorithm can be easily implemented by other researchers.

Computation of the orientation tensors has shown a uniform dispersion of the ellipsoids orientations in the generated packings.

Finally, packings generated with this algorithm will be used in numerical homogenization where it would be possible to study the effect of fibers orientation and aspect ratios on the mechanical effective properties of ellipsoidal (i.e. 3D) particles reinforced composites.

Acknowledgements

The authors thank the National Science and Engineering Reseach Council of Canada (PhD Grant for E. Ghossein) and the Canada Research Chair (M. Lévesque) program for their financial support.

Appendix A. Algorithms pseudo-code

All algorithms pseudo-code are given in this appendix.

Algorithm 1 Main Algorithm.

```
Input: $N$, $R_1$, $R_2$, $\mathcal{V}_f$
Output: $\{a, b, c\}$, $\boldsymbol{r}$ and $\boldsymbol{q}$ of all ellipsoids
1: Define a cube of side $L$. Set a corner as origin.
2: for $i = 1 \to N$ do
3:     Assign a random position vector $\boldsymbol{r}^i_{(0)}$.
4:     Assign a random quaternion vector $\boldsymbol{q}^i_{(0)} = [\alpha^i_{(0)}, \boldsymbol{\psi}^i_{(0)}]$.
5:     Assign a random linear ($\boldsymbol{v}^i_{(0)}$) and angular ($\boldsymbol{\omega}^i_{(0)}$) velocity vector.
6:     Assign the semi-principle axis growth rate $a^i_0 = 0.1$.
7:     Compute $b^i_0 = a^i_0/R_1$ and $c^i_0 = a^i_0/R_2$.
8: end for
9: Initialize the actual volume fraction $\mathcal{V}_{(0)} = 0$.
10: while $\mathcal{V}_{(t)} < \mathcal{V}_f$ do
11:     Compute $t_c$ using Algorithm 6.
12:     Compute $t_s$ using Algorithm 8.
13:     Compute $\Delta t = \min(t_c, t_s)$.
14:     for $i = 1 \to N$ do
15:         Move the particle $i$ from time $t$ to time $(t + \Delta t)$ using Algorithm 2.
16:     end for
17:     if $\Delta t = t_c$ then
18:         Update the velocities of the concerned particles using Algorithm 9.
19:     else if $\Delta t = t_s$ then
20:         Create periodic image(s) of the concerned particle using Algorithm 10.
21:     end if
22:     Compute the new volume fraction: $\mathcal{V}_{(t+\Delta t)} = \frac{1}{L^3}\sum_{i=1}^{N} \frac{4}{3}\pi a^i_{(t+\Delta t)} b^i_{(t+\Delta t)} c^i_{(t+\Delta t)}$.
23:     $t \leftarrow t + \Delta t$.
24: end while
25: Perform a scaling by reducing $\{a^i_{(t+\Delta t)}, b^i_{(t+\Delta t)}, c^i_{(t+\Delta t)}\}$ in order to reach $\mathcal{V}_{(t+\Delta t)} = \mathcal{V}_f$.
```

Algorithm 2 Moving an ellipsoid i between time t and time $(t+\Delta t)$.

```
Input: $\boldsymbol{r}^i_{(t)}$, $\boldsymbol{q}^i_{(t)} = [\alpha^i_{(t)}, \boldsymbol{\psi}^i_{(t)}]$, $\boldsymbol{v}^i$, $\boldsymbol{\omega}^i$, $\{a^i_{(t)}, b^i_{(t)}, c^i_{(t)}\}$, $\{a^i_0, b^i_0, c^i_0\}$, $\Delta t$
Output: $\{a^i_{(t+\Delta t)}, b^i_{(t+\Delta t)}, c^i_{(t+\Delta t)}\}$, $\boldsymbol{r}^i_{(t+\Delta t)}$, $\boldsymbol{q}^i_{(t+\Delta t)}$, $\mathbf{A}^i_{(t+\Delta t)}$, $\mathbf{R}^i_{(t+\Delta t)}$, $\mathbf{M}^i_{(t+\Delta t)}$
1: Compute the new position $\boldsymbol{r}^i_{(t+\Delta t)}$ using Eq. (8).
2: Compute $a^i_{(t+\Delta t)}$, $b^i_{(t+\Delta t)}$ and $c^i_{(t+\Delta t)}$ using Eq. (9).
3: Compute the quaternion $\boldsymbol{q}^i_{\Delta t}$ using Eq. (10).
4: Compute the quaternion $\boldsymbol{q}^i_{(t+\Delta t)}$ using Eq. (11).
5: Compute $\mathbf{R}^i_{(t+\Delta t)}$ and $\mathbf{M}^i_{(t+\Delta t)}$ using Eqs. (5) and (6b).
6: Compute $\mathbf{A}'^i_{(t+\Delta t)}$ using Eq. (2b).
7: Compute $\mathbf{A}^i_{(t+\Delta t)}$ using Eq. (12b).
```

Algorithm 3 Computing the coefficients $p_{k(t)}$ of Eq. (13) for ellipsoids i and j [27].

```
Input: $\mathbf{M}^i_{(t)}$, $\{a^i_{(t)}, b^i_{(t)}, c^i_{(t)}\}$, $\mathbf{A}^j_{(t)}$.
Output: $p_{k(t)}$
1: Compute $\mathbf{C} = (\mathbf{M}^i_{(t)})^\top \mathbf{A}^j_{(t)} (\mathbf{M}^i_{(t)})$.
2: Let $\delta_1 = (1/a^i_{(t)})^2$, $\delta_2 = (1/b^i_{(t)})^2$ and $\delta_3 = (1/c^i_{(t)})^2$.
3: Compute $p_{1(t)} = -\delta_1\delta_2\delta_3$.
4: Compute $p_{2(t)} = -(\delta_2\delta_3 C_{11} + \delta_1\delta_3 C_{22} + \delta_1\delta_2 C_{33} - \delta_1\delta_2\delta_3 C_{44})$.
5: Compute $p_{3(t)} = \delta_1\delta_2(C_{33}C_{44} - C_{34}C_{43}) + \delta_2\delta_3(C_{11}C_{44} - C_{14}C_{41}) + \delta_1\delta_3(C_{22}C_{44} - C_{24}C_{42}) + \delta_1(C_{23}C_{32} - C_{22}C_{33}) + \delta_2(C_{13}C_{31} - C_{11}C_{33}) + \delta_3(C_{12}C_{21} - C_{11}C_{22})$.
6: Compute $p_{4(t)} =$
     $\delta_1(C_{22}C_{33}C_{44} - C_{22}C_{34}C_{43} - C_{33}C_{42}C_{24} - C_{44}C_{32}C_{23} + C_{32}C_{24}C_{43} + C_{42}C_{23}C_{34}) +$
     $\delta_2(C_{11}C_{33}C_{44} - C_{11}C_{34}C_{43} - C_{33}C_{14}C_{41} - C_{44}C_{13}C_{31} + C_{31}C_{14}C_{43} + C_{41}C_{13}C_{34}) +$
     $\delta_3(C_{11}C_{22}C_{44} - C_{11}C_{24}C_{42} - C_{22}C_{14}C_{41} - C_{44}C_{12}C_{21} + C_{21}C_{14}C_{42} + C_{41}C_{12}C_{24}) +$
     $C_{11}C_{23}C_{32} + C_{22}C_{13}C_{31} + C_{33}C_{12}C_{21} - C_{11}C_{22}C_{33} - C_{21}C_{13}C_{32} - C_{31}C_{12}C_{23}$.
7: Compute $p_{5(t)} = \det(\mathbf{A}^j_{(t)})$.
```

Algorithm 4 Computing the coefficients $\eta_{k(t)}$ for ellipsoids i and j [28].

```
Input: $p_{k(t)}$
Output: $\eta_{k(t)}$
1: Compute $\bar{p}_1 = -p_{2(t)}/(4p_{1(t)})$, $\bar{p}_2 = p_{3(t)}/(6p_{1(t)})$, $\bar{p}_3 = -p_{4(t)}/(4p_{1(t)})$ and $\bar{p}_4 = p_{5(t)}/p_{1(t)}$.
2: Compute $\beta_1 = (\bar{p}_4 - \bar{p}_1\bar{p}_3) + 3(\bar{p}_2^2 - \bar{p}_1\bar{p}_3)$.
3: Compute $\beta_2 = -\bar{p}_3(\bar{p}_3 - \bar{p}_1\bar{p}_2) - \bar{p}_4(\bar{p}_1^2 - \bar{p}_2) - \bar{p}_2(\bar{p}_2^2 - \bar{p}_1\bar{p}_3)$.
4: Compute $\eta_{1(t)} = \beta_1^3 - 27\beta_2^2$.
5: Compute $\eta_{2(t)} = -9(\bar{p}_3 - \bar{p}_1\bar{p}_2)^2 + 27(\bar{p}_1^2 - \bar{p}_2)(\bar{p}_2^2 - \bar{p}_1\bar{p}_3) - 3(\bar{p}_4 - \bar{p}_1\bar{p}_3)(\bar{p}_1^2 - \bar{p}_2)$.
6: Compute $\eta_{3(t)} = \beta_1(\bar{p}_3 - \bar{p}_1\bar{p}_2) - 3\bar{p}_1\beta_2$.
7: Compute $\eta_{4(t)} = -(\bar{p}_4 - \bar{p}_1\bar{p}_3)$.
8: Compute $\eta_{5(t)} = (\bar{p}_1^2 - \bar{p}_2)$.
```

Algorithm 5 Function $\eta_{k(t+\Delta t)}$ for ellipsoids i and j.

```
Input: $\Delta t$
Output: $\eta_{k(t+\Delta t)}$
1: Compute $\{a^i_{(t+\Delta t)}, b^i_{(t+\Delta t)}, c^i_{(t+\Delta t)}\}$ and $\mathbf{M}^i_{(t+\Delta t)}$ for ellipsoid $i$ using Algorithm 2.
2: Compute $\mathbf{A}^j_{(t+\Delta t)}$ for ellipsoid $j$ using Algorithm 2.
3: Compute $p_{k(t+\Delta t)}$ using Algorithm 3.
4: Compute $\{\eta_{1(t+\Delta t)}, \eta_{2(t+\Delta t)}, \eta_{3(t+\Delta t)}, \eta_{4(t+\Delta t)}, \eta_{5(t+\Delta t)}\}$ using Algorithm 4.
```

Algorithm 6 Finding the next binary collision time.

Input: $\{a_{(t)}, b_{(t)}, c_{(t)}\}$, $\{a_0, b_0, c_0\}$, $\boldsymbol{r}_{(t)}$, $\boldsymbol{q}_{(t)}$, $\boldsymbol{v}$, $\boldsymbol{\omega}$ of all ellipsoids
Output: t_c, $\boldsymbol{x}_c$

```
1: Set $t_{\min} = \texttt{realmax}$ and $\mu = 1.2$.
2: for $i = 1 \to N - 1$ (including periodic ellipsoids) do
3:     for $j = i + 1 \to N$ (including periodic ellipsoids) do
4:         if $\boldsymbol{v}^i$ or $\boldsymbol{v}^j$ have changed since last iteration then
5:             Compute $R^r_{(t)} = \max(a^r_{(t)}, b^r_{(t)}, c^r_{(t)})$ and $R^r_0 = \max(a^r_0, b^r_0, c^r_0)$ where $r = \{i, j\}$.
6:             Compute $t^{BS}_c$ using Eqs. (15) and (16).
7:             if $0 \leqslant t^{BS}_c < t_{\min}$ then
8:                 Define ellipsoids $N_i$ and $N_j$ using $\mu$.
9:                 Compute $\eta_{k(t)}$ for $N_i$ and $N_j$ using Algorithm 5 (use $\Delta t = 0$).
10:                if $N_i$ and $N_j$ are externally tangent or overlap then
11:                    $\text{test}_1 = \texttt{true}$.
12:                else
13:                    $\text{test}_1 = \texttt{false}$.
14:                end if
15:            else
16:                $\text{test}_1 = \texttt{false}$.
17:            end if
18:            if $\text{test}_1 = \texttt{true}$ then
19:                Define the function $\eta_{1(t+\Delta t)}$ using Algorithm 5.
20:                Compute all roots of $\eta_{1(t+\Delta t)}$: $\Delta t_1 < \Delta t_2 < \cdots < \Delta t_n$.
21:                $\text{test}_2 = \texttt{false}$.
22:                for $p = 1 \to n$ do
23:                    Compute $\eta_{k(t+\Delta t_p)}$ for $i$ and $j$ using Algorithm 5.
24:                    if $\{\eta_{1(t+\Delta t_p)} = 0, \eta_{2(t+\Delta t_p)} > 0, \eta_{3(t+\Delta t_p)} < 0$ and $\eta_{5(t+\Delta t_p)} > 0\}$ or
                           $\{\eta_{1(t+\Delta t_p)} = 0, \eta_{2(t+\Delta t_p)} = 0, \eta_{4(t+\Delta t_p)} < 0$ and $\eta_{5(t+\Delta t_p)} > 0\}$ then
25:                        Compute $t_{\min} = \min(\Delta t_p, t_{\min})$.
26:                        $\text{test}_2 = \texttt{true}$.
27:                        break
28:                    end if
29:                end for
30:                if $\text{test}_2 = \texttt{false}$ then
31:                    Return No collision between $i$ and $j$.
32:                end if
33:            else
34:                Return No collision between $i$ and $j$.
35:            end if
36:        else
37:            Deduce the binary collision time $\Delta t$ from last iteration.
38:            Compute $t_{\min} = \min(\Delta t, t_{\min})$.
39:        end if
40:    end for
41: end for
42: Set $t_c = t_{\min}$.
43: Compute $\mathbf{A}^i_{(t+t_c)}$ and $\mathbf{A}^i_{(t+t_c)}$ using Algorithm 2.
44: Compute the eigenvalues of matrix $(\mathbf{A}^i_{(t+t_c)})^{-1}\mathbf{A}^j_{(t+t_c)}$.
45: Search for the double positive eigenvalue $\lambda_0$.
46: Compute $\boldsymbol{x}_c$ using Eq. (14).
```

Algorithm 7 Function $g^{\dagger}_{(t+\Delta t)}$ between an ellipsoid i and a cube face $x_k = \beta$.

Input: Δt
Output: $g^{\dagger}_{(t+\Delta t)}$

```
1: Compute $\mathbf{A}^i_{(t+\Delta t)}$ using Algorithm 2.
2: Deduce $\mathbf{B}^i_{(t+\Delta t)}$, $\boldsymbol{d}^i_{(t+\Delta t)}$ and $F^i_{(t+\Delta t)}$ using Eq. (17).
3: Compute $\mathbf{P}$ and $\boldsymbol{p}$ using Eq. (19b).
4: Compute $\mathbf{B}^{i*}_{(t+\Delta t)}$, $\boldsymbol{d}^{i*}_{(t+\Delta t)}$ and $F^{i*}_{(t+\Delta t)}$ using Eq. (21b).
5: Compute $g^{\dagger}_{(t+\Delta t)}$ using Eq. (23).
```

Algorithm 8 Finding the next collision time between an ellipsoid and a cube face.

Input: $\{a_{(t)}, b_{(t)}, c_{(t)}\}$, $\{a_0, b_0, c_0\}$, $\boldsymbol{r}_{(t)}$, $\boldsymbol{q}_{(t)}$, $\boldsymbol{v}$, $\boldsymbol{\omega}$ of all ellipsoids
Output: t_s

```
1: Set $t_{\min} = \texttt{realmax}$.
2: for $i = 1 \to N$ do
3:     Compute $R^i_{(t)} = \max(a^i_{(t)}, b^i_{(t)}, c^i_{(t)})$ and $R^i_0 = \max(a^i_0, b^i_0, c^i_0)$.
4:     for $k = 1 \to 3$ do
5:         for $\beta = \{0, L\}$ do
6:             if $\boldsymbol{v}^i$ has changed since last iteration then
7:                 if particle $i$ does not intersect the cube face $x_k = \beta$ then
8:                     Compute $t_s^{BS}$ using Eqs. (24) and (25).
9:                     if $0 \leqslant t_s^{BS} < t_{\min}$ then
10:                        Define the function $g^{\dagger}_{(t+\Delta t)}$ using Algorithm 7.
11:                        Compute all $n$ roots of $g^{\dagger}_{(t+\Delta t)}$: $\Delta t_1 < \Delta t_2 < \cdots < \Delta t_n$.
12:                        if $n \neq 0$ then
13:                            Compute $t_{\min} = \min(\Delta t_1, t_{\min})$.
14:                        else
15:                            Return No collision between $i$ and $x_k = \beta$.
16:                        end if
17:                    else
18:                        Return No collision between $i$ and $x_k = \beta$.
19:                    end if
20:                end if
21:            else
22:                Deduce the collision time $\Delta t$ from last iteration.
23:                Compute $t_{\min} = \min(\Delta t, t_{\min})$.
24:            end if
25:        end for
26:    end for
27: end for
28: Set $t_s = t_{\min}$.
```

Algorithm 9 Computing velocities after impact at time $t = t_c$.

Input: $\boldsymbol{r}^i_{(t)}$, $\boldsymbol{r}^j_{(t)}$, $\boldsymbol{q}^i_{(t)}$, $\boldsymbol{q}^j_{(t)}$, $\boldsymbol{v}^{i-}$, $\boldsymbol{v}^{j-}$, $\boldsymbol{\omega}^{i-}$, $\boldsymbol{\omega}^{j-}$, $\{a^i_{(t)}, b^i_{(t)}, c^i_{(t)}\}$, $\{a^j_{(t)}, b^j_{(t)}, c^j_{(t)}\}$, $\{a^i_0, b^i_0, c^i_0\}$, $\{a^j_0, b^j_0, c^j_0\}$, $\boldsymbol{x}_c$
Output: $\boldsymbol{v}^{i+}$, $\boldsymbol{v}^{j+}$, $\boldsymbol{\omega}^{i+}$, $\boldsymbol{\omega}^{j+}$

```
1: Compute $\mathbf{A}^r_{(t)}$, $\mathbf{R}^r_{(t)}$ using Algorithm 2, where $r = \{i, j\}$.
2: Compute $\boldsymbol{n}$ using Eq. (26).
3: Compute $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ such that $\boldsymbol{t_1} \times \boldsymbol{t_2} = \boldsymbol{n}$.
4: Compute $m^r_{(t)}$ and $\mathbf{H}'^r_{(t)}$ using Eqs. (27a) and (27b), where $r = \{i, j\}$.
5: Compute $\mathbf{H}^r_{(t)}$ using Eq. (28), where $r = \{i, j\}$.
6: Define the linear equations system using Eqs. (29), (30), (31) and (32).
7: Solve the equations system to find $\boldsymbol{v}^{i+}$, $\boldsymbol{v}^{j+}$, $\boldsymbol{\omega}^{i+}$ and $\boldsymbol{\omega}^{j+}$.
8: if ellipsoid $i$ has $P^i$ periodic image(s) ($P^i \neq 0$) then
9:     for $p = 1 \to P^i$ do
10:        $\boldsymbol{v}^p = \boldsymbol{v}^{i+}$.
11:        $\boldsymbol{\omega}^p = \boldsymbol{\omega}^{i+}$.
12:    end for
13: end if
14: if ellipsoid $j$ has $P^j$ periodic image(s) ($P^j \neq 0$) then
15:    for $p = 1 \to P^j$ do
16:        $\boldsymbol{v}^p = \boldsymbol{v}^{j+}$.
17:        $\boldsymbol{\omega}^p = \boldsymbol{\omega}^{j+}$.
18:    end for
19: end if
```

Algorithm 10 Creation of periodic images following the collision between a particle and the cube faces at time $t = t_s$.

Input: $\boldsymbol{r}^i_{(t)}$, $\boldsymbol{q}^i_{(t)}$, $\boldsymbol{v}^i$, $\boldsymbol{\omega}^i$, $\{a^i_{(t)}, b^i_{(t)}, c^i_{(t)}\}$, $\{a^i_0, b^i_0, c^i_0\}$
Output: Periodic particles of ellipsoid i

```
1: Suppose that particle $i$ collided with $m$ faces.
2: if $m = 1$ then
3:     $P^i = 1$.
4: else if $m = 2$ then
5:     $P^i = 3$.
6: else if $m = 3$ then
7:     $P^i = 7$.
8: end if
9: Create $P^i$ periodic image(s) of ellipsoid $i$.
10: for $p = 1 \rightarrow P^i$ do
11:     $\boldsymbol{r}^p_{(t)} = \boldsymbol{r}^i_{(t)} + \beta \boldsymbol{e}_k$ where $\beta \in \{L, -L\}$ and $k \in \{1, 2, 3\}$.
12:     $\boldsymbol{q}^p_{(t)} = \boldsymbol{q}^i_{(t)}$.
13:     $\boldsymbol{v}^p = \boldsymbol{v}^i$.
14:     $\boldsymbol{\omega}^p = \boldsymbol{\omega}^i$.
15: end for
```

Appendix B. Supplementary material

Supplementary material related to this article can be found online at 10.1016/j.jcp.2013.07.004.

References

[1] C. Gray, K. Gubbins, Theory of Molecular Fluids, vol. 1, Clarendon Press, Oxford, 1984.
[2] J. Kolafa, I. Nezbeda, Monte Carlo simulations on primitive models of water and methanol, Molecular Physics 61 (1987) 161–175.
[3] S. Torquato, Random Heterogeneous Materials: Microstructure and Macroscopic Properties, vol. 16, Springer-Verlag, 2002.
[4] E. Ghossein, M. Levesque, A fully automated numerical tool for a comprehensive validation of homogenization models and its application to spherical particles reinforced composites, International Journal of Solids and Structures 49 (2012) 1387–1398.
[5] T. Kanit, S. Forest, I. Galliet, V. Mounoury, D. Jeulin, Determination of the size of the representative volume element for random composites: statistical and numerical approach, International Journal of Solids and Structures 40 (2003) 3647–3679.
[6] J. Segurado, J. Llorca, A numerical approximation to the elastic properties of sphere-reinforced composites, Journal of the Mechanics and Physics of Solids 50 (2002) 2107–2121.
[7] R.B. Barello, M. Lévesque, Comparison between the relaxation spectra obtained from homogenization models and finite elements simulation for the same composite, International Journal of Solids and Structures 45 (2008) 850–867.
[8] L. Iorga, Y. Pan, A. Pelegri, Numerical characterization of material elastic properties for random fiber composites, Journal of Mechanics of Materials and Structures 3 (2008) 1279–1298.
[9] C. Redenbach, I. Vecchio, Statistical analysis and stochastic modelling of fibre composites, Composites Science and Technology 71 (2011) 107–112.
[10] W. Man, A. Donev, F. Stillinger, M. Sullivan, W. Russel, D. Heeger, S. Inati, S. Torquato, P. Chaikin, Experiments on random packings of ellipsoids, Physical Review Letters 94 (2005) 198001.
[11] A. Bezrukov, D. Stoyan, Simulation and statistical analysis of random packings of ellipsoids, Particle & Particle Systems Characterization 23 (2006) 388–398.
[12] B. Buchalter, R. Bradley, Orientational order in amorphous packings of ellipsoids, Europhysics Letters 26 (2007) 159.
[13] M. Allen, D. Frenkel, J. Talbot, Molecular dynamics simulation using hard particles, Computer Physics Reports 9 (1989) 301–353.
[14] S. Williams, A. Philipse, Random packings of spheres and spherocylinders simulated by mechanical contraction, Physical Review E 67 (2003) 051301.
[15] J. Zhao, S. Li, R. Zou, A. Yu, Dense random packings of spherocylinders, Soft Matter 8 (2012) 1003–1009.
[16] B. Lubachevsky, F. Stillinger, Geometric properties of random disk packings, Journal of Statistical Physics 60 (1990) 561–583.
[17] M.D. Rintoul, S. Torquato, Reconstruction of the structure of dispersions, Journal of Colloid and Interface Science 186 (1997) 467–476.
[18] B. Lubachevsky, F. Stillinger, E. Pinson, Disks vs. spheres: Contrasting properties of random packings, Journal of Statistical Physics 64 (1991) 501–523.
[19] B. Lubachevsky, How to simulate billiards and similar systems, Journal of Computational Physics 94 (1991) 255–283.
[20] A. Donev, S. Torquato, F.H. Stillinger, Neighbor list collision-driven molecular dynamics simulation for nospherical hard particles. I. Algorithmic details, Journal of Computational Physics 202 (2005) 737–764.
[21] A. Donev, S. Torquato, F.H. Stillinger, Neighbor list collision-driven molecular dynamics simulation for nospherical hard particles. II. Applications to ellipses and ellipsoids, Journal of Computational Physics 202 (2005) 765–793.
[22] S. Kari, H. Berger, R. Rodriguez-Ramos, U. Gabbert, Computational evaluation of effective material properties of composites reinforced by randomly distributed spherical particles, Composite Structures 77 (2007) 223–231.
[23] J. Perram, M. Wertheim, Statistical mechanics of hard ellipsoids. I. Overlap algorithm and the contact function, Journal of Computational Physics 58 (1985) 409–416.
[24] J. Perram, J. Rasmussen, E. Præstgaard, J. Lebowitz, Ellipsoid contact potential: Theory and relation to overlap potentials, Physical Review E 54 (1996) 6565–6572.
[25] W. Wang, J. Wang, M. Kim, An algebraic condition for the separation of two ellipsoids, Computer Aided Geometric Design 18 (2001) 531–539.
[26] Y. Choi, W. Wang, M. Kim, Exact collision detection of two moving ellipsoids under rational motions, IEEE International Conference on Robotics and Automation, vol. 1, IEEE, 2003, pp. 349–354.
[27] Y. Choi, J. Chang, W. Wang, M. Kim, G. Elber, Continuous collision detection for ellipsoids, IEEE Transactions on Visualization and Computer Graphics 15 (2009) 311–325.
[28] X. Jia, Y. Choi, B. Mourrain, W. Wang, An algebraic approach to continuous collision detection for ellipsoids, Computer Aided Geometric Design (2011).

[29] S. Advani, C. Tucker, The use of tensors to describe and predict fiber orientation in short fiber composites, Journal of Rheology 31 (1987) 751–784.
[30] D. Eberly, Rotation representations and performance issues, Magic Software 6006 (2002).
[31] S. Basu, R. Pollack, M. Roy, Algorithms in Real Algebraic Geometry, vol. 10, Springer-Verlag, New York, Inc, 2006.
[32] R. Brent, Algorithms for Minimization Without Derivatives, Dover Publications, 2002.