Prediction of Meso-structure Parameters on Particle Assemblies

Victor Brun Ivan Flensburg Filip Westberg

February 1, 2023

Abstract

This report aims to investigate the possibility of generating homogeneous multi-component mixtures with desired porosity, particle shapes and volume distributions, from which to compute statistics that can be used to predict meso-structure properties using the advancing front algorithm.

The method successfully generates multi-component mixtures given desired volume distributions, particle shapes and scales, but cannot create the desired porosity. Furthermore statistics such as content uniformity and coordination number could not be produced due to unforeseen programming problems and time constraints.

The program developed to generate the mixtures was written in C++ and is available on GitHub[1].

[1] https://github.com/victorbrun/partycle--

Contents

1 Introduction

In general, particle mixing algorithms can be divided into two classes; static and dynamic mixtures. For dynamic mixtures, discrete element modelling (DEM) of particles under gravity (or isostatic compression) is a common method. The method is however computationally demanding and struggles to generate particle mixtures with a big span of particle sizes, also known as multi-scale mixtures. The goal of this report is to investigate if a static particle mixing algorithm, more specifically a variant of the advancing front algorithm presented in [2], can be a solution to the above mentioned problems for particle mixtures with several different components.

The report begins by motivating the importance of understanding the properties of particle mixtures, e.g. via characteristics such as coordination number and content uniformity. It then moves on to examine a portion of the current literature regarding static mixtures and motivate the authors choice of mixing algorithm. Some terminology and definitions used in the report are then specified, after which theory concerning superellipsoids, such as contact detection, is discussed since they play a central role as the mathematical model of the physical particles in the report.

This report does moreover enforces the restriction on mixture components that the only within component variability is the volume of a particle, which is defined to follow some distribution. This means that two particles belonging to the same mixture component has the same shape but most likely different volumes. Thus, after the superellipsoids the report describes the theory behind choosing the, on average, correct number of particles to achieve the specified volume fraction for each mixture component in a given mixture. Before moving on to explaining the inner workings of the advancing front algorithm, the complexity of the full program is analysed. The results, which include run time data for a complex mixture and porosity values for a mixture of identical spheres, are presented. Lastly, the results, are discussed in the context of the chosen methods and possible improvements are presented.

2 Background

Particle products have a key role in sectors of the manufacturing industry, with application in the pharmaceutical, chemical and food industries among others. While their use and production is ubiquitous and varied, some common performance goals are

- *Appearance:* Properties such as bulk density, size particle volume distribution and colour.
- *Dosing:* the amount of the active compound being deployed.
- *Mass transfer:* the change of the product upon use, from processes such as dissolution and swelling among others.

Achieving performance goals such as these may require design of the particle product at several different scales; generally being divided into four different categories:

- *Supramolecular scale:* the organisation of a collection of molecules in a spatial domain.
- *Particle scale:* the organisation of domains within a particle.
- *Meso-scale:* the representative spatial arrangement of particles within a volume.
- *Macro-scale:* the scale at which different meso-scale structures across volumes are arranged within a unit-operation.

The connection of the macro-scale with unit-operations is such that inhomogeneity of the unit-operation is the typical cause of macro-scale structure. As the properties of particulate systems are generally manipulated and designed on the supramolecular, particle and meso-scales, available degrees of freedom are higher than for other approaches.

One of, if not the most critical physical propertiy of a particle product is the particle volume. Properties of the product influenced by the particle volume is, among others

- melting and glass transition temperatures [1],
- dissolution velocity [10], and
- flowability [4].

The study of the effects of the particle volume distribution is then essential for performance characterisation.

2.1 Previous studies

During the initial stage of the project, a literature study into existing algorithms was done. Two different categories of static solutions to the problem of filling some volume with particles were identified. Either one can divide the domain into subdomains in which particles are placed, or the particles can be placed in the domain until it is divided.

The first category consists of using some sort of tessellation, e.g. in [6] Voronoi tessellation is utilised to partition the domain. Each cell generated by the tessellation is then filled by one particle with the same shape as the cell, which eliminates the possibility of overlapping particles. In order to fulfil certain volume distribution requirements on the different particle types, a stochastic method like Inverse Monte Carlo (IMC) can be used. This IMC step does however, in the case of [6], change the shape of the cells, which in turn changes the shape of the particle placed in the cell. Thus, for mixtures where the particle shapes are fixed on beforehand, this method is not suitable without large modifications making

the problem complex. Constraining the aspect ratio of the cells isn't included under this approach.

The second category consists of sequentially placing particles according to some algorithm and then move them to avoid overlap. In [3] the particles are randomly placed and replaced until they fit, or until some threshold is reached at which the whole mixture is discarded and the processes started over. This procedure is known as a random adsorption process. Since the number of discarded mixtures will increase with lower target porosities, in [3] results even suggests that the number of discarded mixtures grows faster than exponential, this method faces run time problems.

However, in [2] a method called advancing front is proposed, which packs fixed shape particles close to the minimum porosity while still being moderately complex. Therefore it was decided that the particle mixture algorithm this report will be based on is the advancing front method.

3 Theory

3.1 Terminology and definitions

To make the report more self-contained and minimise the risk of confusion, this section is dedicated to creating common ground regarding terminology and definitions.

Definition 3.1 (Hard particle)**.** *A hard particle is defined as a particle not allowed to deform, i.e. when put in contact with another particle, no overlap can occur.*

Definition 3.2 (Soft particle)**.** *A soft particle is defined as a particle allowed to deform, i.e. when put in contact with another particle a certain amount of overlap can occur. This approach is common in DEM.*

Definition 3.3 (Particle mixture)**.** *A particle mixture is defined as a set of hard or soft particles placed inside a domain such that contacts between them may occur. In a mixture, particles cannot be placed inside each other, and overlap between them are only allowed if the particles are soft.*

Definition 3.4 (Mixture component)**.** *A mixture component is defined as a grouping of particles by their substance.*

The above definition entails that the shape of two particles of the same mixture component may differ. In this report a restriction is however made on this definition, making particles belonging to the same component have the same shape but may have differing volume.

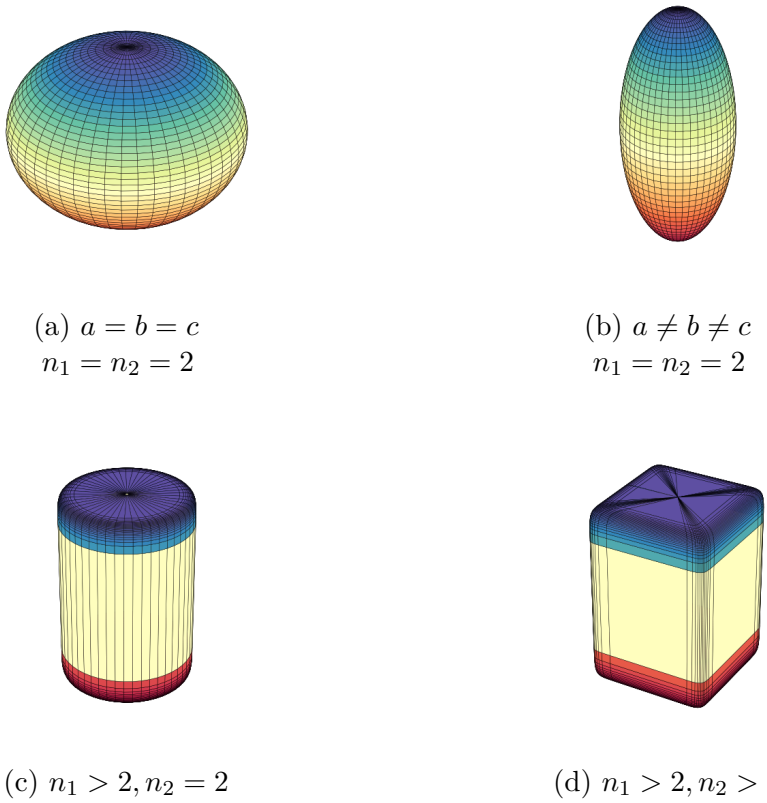
Definition 3.5 (Volume fraction)**.** *The volume fraction for a given mixture component is defined as the fraction of the total volume of all particles in a mixture occupied by particles from the given mixture component.*

Note that definition 3.5 does not conform with the definition usually found in the literature. The most common definition of volume fraction is as the fraction of the domain which is filled by particles, i.e. the complement to porosity. This report does not use the commonly used definition in order for it to conform with the source code of `partycle--`.

Definition 3.6 (Porosity)**.** *Porosity is defined as the fraction of the domain that is void of particles.*

Definition 3.7 (Coordination number)**.** *The coordination number for a given particle in a mixture is defined as the number of other particles it is in contact with.*

3.2 Superellipsoids



(a) $a = b = c$
$n_1 = n_2 = 2$

(b) $a \neq b \neq c$
$n_1 = n_2 = 2$

(c) $n_1 > 2, n_2 = 2$

(d) $n_1 > 2, n_2 > 2$

Figure 1: Renderings of superellipsoids for different shape parameters n_1, n_2 and scale parameters a, b, c as defined by (1).

To mathematically represent particles, this report utilises superellipsoids as they can represent a wide variety of shapes using only a few parameters. Superellipsoids are a special case of superquadrics which together with supertoroids form the whole set of superquadrics. Their horisontal sections are superellipses, also known as Lamé curves, which motivates the name. The surface of a superellipsoid is

defined by the implicit equation

$$\left(\left|\frac{x}{a}\right|^{n_2} + \left|\frac{y}{b}\right|^{n_2}\right)^{n_1/n_2} + \left|\frac{z}{c}\right|^{n_1} = 1, \tag{1}$$

where n_1 and n_2 are shape parameters and a, b and c are scale parameters in respective axis. In order for the Hessian, (10), to be defined for all x, y, z we restrict $n_1, n_2 \in [2, \infty)$. Figure 1 illustrates the different types of shapes that can be constructed with such restrictions; spheres, cubes, pointy and flat needles, cylinders etc. Shapes with sharp enough corners, i.e. less than two time differentiable corners, are the only convex shapes that cannot be constructed under these restrictions.

To render the particles in, e.g., figure 1 the below parametric representation of the radius vector is utilised

$$\boldsymbol{r}(\eta, \omega) = \begin{pmatrix} a\cos^{2/n_1}(\eta)\cos^{2/n_2}(\omega) \\ b\cos^{2/n_1}(\eta)\sin^{2/n_2}(\omega) \\ c\sin^{2/n_1}(\eta) \end{pmatrix}, \quad \eta \in [-\pi/2, \pi/2], \quad \omega \in [-\pi, \pi], \tag{2}$$

where $cos^{\alpha}(\theta) = \text{sign}(\cos(\theta))|\cos(\theta)|^{\alpha}$ and similarly for $\sin^{\alpha}(\theta)$.

3.2.1 Minimum distance between superellipsoids

The smallest distance between two superellipsoids does, in general, not have a closed analytic formula. This means that in order to calculate this distance, an optimisation problem needs to be solved. More specifically the problem of minimising the distance between two points in the global space under the condition that they each lie on the respective superellipsoid's surface.

Let p_x, p_y be two particles and $\boldsymbol{x}_l, \boldsymbol{y}_l \in \mathbb{R}^3$ be two points in the local space of the respective particle. Furthermore, let R_x and R_y be the corresponding rotational matrices and $\boldsymbol{c}_x$, $\boldsymbol{c}_y$ be the corresponding centres. Then, the minimum distance between the particles p_x and p_y is given by the solution to the problem

$$\begin{aligned} \min_{\boldsymbol{x}_l, \boldsymbol{y}_l} \quad & ||(R_x\boldsymbol{x}_l + \boldsymbol{c}_x) - (R_y\boldsymbol{y}_l + \boldsymbol{c}_y)||_2^2 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}_l) \leq 0, \\ & g_2(\boldsymbol{y}_l) \leq 0, \end{aligned} \tag{3}$$

where $g_1(\boldsymbol{x}_l)$ and $g_2(\boldsymbol{y}_l)$ is the inside-outside function for each particle given by

$$g(x, y, z) = \left(\left|\frac{x}{a}\right|^{n_2} + \left|\frac{y}{b}\right|^{n_2}\right)^{n_1/n_2} + \left|\frac{z}{c}\right|^{n_1} - 1. \tag{4}$$

The minimisation problem can be solved using the primal-dual interior point method, in which the Karush-Kuhn-Tucker (KKT) conditions are both sufficient and necessary. This due to the fact that Slater's constraint qualification is satisfied, i.e. the feasible region defined by our constraints is convex, and there exists an interior point such that it satisfies the strict inequalities of our constraints.

For simplicity's sake, define our objective function as $f(\boldsymbol{x}) = ||(R_x\boldsymbol{x}_l + \boldsymbol{c}_x) - (R_y\boldsymbol{y}_l + \boldsymbol{c}_y)||_2^2$, our variables as $\boldsymbol{x} = (\boldsymbol{x}_l, \boldsymbol{y}_l)^\top$ and our constraints as

$\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}_l), g_2(\boldsymbol{y}_l))^\top$. By introducing the slack variables $\boldsymbol{s} = (s_1, s_2)^\top$, (3) is reformulated as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s} = \boldsymbol{0}, \\ & \boldsymbol{s} \geq \boldsymbol{0}. \end{aligned} \tag{5}$$

To eliminate the now introduced non-negativity constraint of our slack variables, the inequality is penalised through reformulating (5) using the barrier method, so that

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) - \mu \sum_{i=1}^{2} \ln s_i \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s} = \boldsymbol{0}, \end{aligned} \tag{6}$$

where $\mu > 0$ is the barrier parameter. It is shown in [2] that performing Lagrangian relaxation of the equality constraint of (6) and subsequently formulating the KKT-conditions results in the systems of linear equations

$$\begin{aligned} \nabla f(\boldsymbol{x}) + (\boldsymbol{J_g}(\boldsymbol{x}))^\top \boldsymbol{\lambda} &= \boldsymbol{0}, \\ \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s} &= \boldsymbol{0}, \\ \boldsymbol{LSe} - \mu \boldsymbol{e} &= \boldsymbol{0}, \end{aligned} \tag{7}$$

where $\lambda = (\lambda_1, \lambda_2)^\top$ is the vector of Lagrangian multipliers, $\boldsymbol{L} = \operatorname{diag}(\boldsymbol{\lambda})$, $\boldsymbol{S} = \operatorname{diag}(\boldsymbol{s})$ and $\boldsymbol{e} = (1, 1)^\top$. Furthermore, this is solved by Newton's method, in which the Newton steps $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{s})^\top$ are derived in [2] as

$$\begin{pmatrix} \boldsymbol{A_1} & \boldsymbol{A_2}^\top & \boldsymbol{0}_{6\times 2} \\ \boldsymbol{A_2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \\ \boldsymbol{0}_{2\times 6} & \boldsymbol{S} & \boldsymbol{L} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{s} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{F}_1 \\ -\boldsymbol{F}_2 \\ -\boldsymbol{F}_3 \end{pmatrix}, \tag{8}$$

where $\boldsymbol{A}_1 = \nabla^2 f(\boldsymbol{x}) + \sum_{i=1}^{2} \lambda_i \nabla^2 g_i(\boldsymbol{x})$, $\boldsymbol{A}_2 = \boldsymbol{J_g}(\boldsymbol{x})$, and $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, $\boldsymbol{F}_3$ represent each linear equation formulated in (7).

To efficiently solve the above system of linear equations, analytical expressions for the gradient vector and the Hessian matrix with respect to the two points in local coordinates, $\boldsymbol{x}_l$ and $\boldsymbol{y}_l$, for both the objective function and the constraint function are required. For the objective function, these are as follows:

$$\nabla f(\boldsymbol{x}) = 2 \begin{pmatrix} R_x^T (R_x \boldsymbol{x}_l - R_y \boldsymbol{y}_l + \boldsymbol{c}_x - \boldsymbol{c}_y) \\ R_y^T (R_x \boldsymbol{x}_l - R_y \boldsymbol{y}_l + \boldsymbol{c}_x - \boldsymbol{c}_y) \end{pmatrix},$$

$$\nabla^2 f(\boldsymbol{x}) = 2 \begin{pmatrix} R_x^T R_x & -R_x^T R_y \\ -R_y^T R_x & R_y^T R_y \end{pmatrix}.$$

For the inside-outside function, the first order derivatives with respect to each direction are defined as

$$\begin{aligned} g'_x(x, y, z) &= \frac{n_1}{a} \left| \frac{x}{a} \right|^{n_2 - 1} \nu^{n_1/n_2 - 1} \operatorname{sign}(x), \\ g'_y(x, y, z) &= \frac{n_1}{b} \left| \frac{y}{b} \right|^{n_2 - 1} \nu^{n_1/n_2 - 1} \operatorname{sign}(y), \\ g'_z(x, y, z) &= \frac{n_1}{c} \left| \frac{z}{c} \right|^{n_1 - 1} \operatorname{sign}(z). \end{aligned}$$

And the second order derivatives become

$$
\begin{aligned}
g'_{xx}(x,y,z) &= \frac{n_1(n_2-1)}{a^2}\left|\frac{x}{a}\right|^{n_2-2}\nu^{n_1/n_2-1} + \frac{n_2(n_1-n_2)}{a^2}\left|\frac{x}{a}\right|^{2n_2-2}\nu^{n_1/n_2-2},\\
g'_{yy}(x,y,z) &= \frac{n_1(n_2-1)}{b^2}\left|\frac{y}{b}\right|^{n_2-2}\nu^{n_1/n_2-1} + \frac{n_2(n_1-n_2)}{b^2}\left|\frac{y}{b}\right|^{2n_2-2}\nu^{n_1/n_2-2},\\
g'_{xy}(x,y,z) &= \frac{n_1(n_1-n_2)}{ab}\left|\frac{x}{a}\right|^{n_2-1}\left|\frac{y}{b}\right|^{n_2-1}\nu^{n_1/n_2-2}\mathrm{sign}(xy),\\
g'_{zz}(x,y,z) &= \frac{n_1(n_1-1)}{c^2}\left|\frac{z}{c}\right|^{n_1-2},\\
g'_{yz} &= g'_{zy} = g'_{xz} = g'_{zx} = 0,\\
g'_{xy} &= g'_{yx},
\end{aligned}
$$

where

$$\nu = \left|\frac{x}{a}\right|^{n_2} + \left|\frac{y}{b}\right|^{n_2}.$$

The gradient vector for the inside-outside function, (4), thus becomes

$$\nabla g(x,y,z) = (g_x, g_y, g_z)^\top, \tag{9}$$

and the Hessian matrix of the same function becomes

$$\nabla^2 g(x,y,z) = \begin{pmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz} \end{pmatrix}. \tag{10}$$

Finally, the Jacobian of our reformulated constraints $\boldsymbol{g} = (g_1, g_2)^\top$ with respect to $\boldsymbol{x} = (\boldsymbol{x}_l, \boldsymbol{y}_l)^\top$, is

$$\boldsymbol{J_g}(\boldsymbol{x}) = \begin{pmatrix} \nabla g_1(\boldsymbol{x}_l) & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \nabla g_2(\boldsymbol{y}_l) \end{pmatrix}, \tag{11}$$

and the Hessian matrix

$$\boldsymbol{H_g}(\boldsymbol{x}) = \begin{pmatrix} \nabla^2 g_1(\boldsymbol{x}_l) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \nabla^2 g_2(\boldsymbol{y}_l) \end{pmatrix}. \tag{12}$$

3.2.2 Collision check between superellipsoids

The exact collision detection follows [8], where the collision detection problem is formulated as the minimization problem

$$\min F_1(\mathbf{X}) + F_2(\mathbf{X}) \tag{13}$$

$$: F_1(\mathbf{X}) = F_2(\mathbf{X}) \tag{14}$$

where $F_1(\cdot), F_2(\cdot)$ is the implicit function of each particle as defined in 1, and $\mathbf{X}$ are global coordinates. The problem solution $\mathbf{X}^*$ is a point between the particles but in a sense closest to both. The problem is reformulated as a system of non-linear equations

$$\nabla F_1(\mathbf{X}) + \mu^2 \nabla F_2(\mathbf{X}) = 0 \tag{15}$$

$$F_1(\mathbf{X}) - F_2(\mathbf{X}) = 0 \tag{16}$$

where the first three equations correspond to the stationary point of the Lagrangian of the initial problem. The solution is solved for using Newtons method

$$dZ \;:\; JdZ = -\Phi, \quad Z^{n+1} = Z^n + \alpha dZ \tag{17}$$

where J is the Jacobian of Φ, which in turn is the left-hand side of the four equations above. The step size α is chosen such that

$$||\Phi(Z^{n+1})|| < ||\Phi(Z^n)||.$$

For the sake of convergence the particle shapes are initially set as spheres and iteratively updated until the actual particle shape is achieved. If $F_1(\mathbf{X}) < 0$ and $F_2(\mathbf{X}) < 0$ for the point $\mathbf{X}$ at the end of these iterations, then the particles are colliding.

3.2.3 Achieving specified volume fractions

For mixtures with more than one component, the amount of each component need to be specified. This is done by specifying the volume fraction for each component, i.e. how big part of the total volume of all particles is made up of each component. However, since the volume of each particle follows some distribution, one can only ensure that the specified volume fractions for each component are on average achieved. To do this the expected number of particles of each component needed to fill the specified fraction of the domain volume is computed. It is important to note that the volume fractions refer to how large part of the domain volume is filled by each mixture component. In other words, the porosity is assumed to be zero. Due to this, the expected number of particles needed to fill the volume of the domain will be higher than what is actually needed, since it is impossible to pack the particles in such a manner that there is no gap between them, i.e. the mixture cannot have a porosity of zero.

Let $\mathcal{D}_i(\boldsymbol{\theta}_i)$ be the volume distribution with parameter vector $\boldsymbol{\theta}_i$ for component i. For a given component i, the volume of particle j is then described by the random variable $V_{ij} \mid i \overset{\text{i.i.d}}{\sim} \mathcal{D}_i(\boldsymbol{\theta}_i)$. Furthermore, let V_d be the volume of the domain into which the particles are to be placed, then the volume fraction of given component i is described by

$$F_i = \frac{\sum_{j=1}^{n_i} V_{ij}}{V_d}, \tag{18}$$

where n_i is the number of particles of component i.

Since V_{ij} is the only random variable in (18) the expected volume fraction becomes

$$\begin{aligned} \mathrm{E}F_i &= \frac{\mathrm{E}\left[\sum_{j=1}^{n_i} V_{ij}\right]}{V_d} \\ &= n_i \frac{\mathrm{E}V_{ij}}{V_d}, \end{aligned}$$

where the second equality follows from the fact that $V_{ij} \mid i$ are IID. By letting $\mathrm{E}F_i$ be the specified volume fraction f_i, the number of particles of component i needed to, on average, achieve the specified volume fraction is given by $n_i = f_i V_d / \mathrm{E}V_{ij}$.

3.3 Time complexity

Due to the construction of the advancing front algorithm the outermost layer of a mixture will always be in the advancing front. Furthermore, at least one of the particles that are chosen to define the approach point for the new particle are in the advancing front and the approach vector is orthogonal to the plane intersecting the three particles' center points. This results in large flat areas on the outside of a mixture tend to have more particles approach them. This means that large areas with the same curvature are minimised, which in turn yields that the mixture can be said to, approximately, grow spherically. Moreover, the volume of the intersection between a sphere and the domain between two planes, will grow proportional to the squared radius of the sphere. Using this insight it is possible to rigorously examine the time complexity of the algorithm.

Next the volume of the sphere of packed particles needs to be related to the number of particles in the mixture. Let n be the number of particles currently in the mixture, and let V_i be the volume of particle i, then, using the expected volume of each particle it follows that

$$n\mathrm{E}V_i \propto \frac{4}{3}\pi r^3 \implies r^2 \propto n^{2/3},$$

since $\mathrm{E}V_i$ is a constant. Note that since it is not specified of which component particle i is, this must be included in the computation of the expected volume. Now using this relation it follows that the number of particles in the intersection between the growing mixture and the domain between two planes grows as $n^{2/3}$, and therefore so does the result from the binary search in algorithm 3.

The time complexity of the algorithm can now be considered. Assume N particles is to be packed, then for each such particle the advancing front algorithm is performed, and thus also the binary approach. For each call to binary approach the intersection between the results from the binary search in each coordinate axis is computed. Since intersection of sets has linear time complexity the put together complexity of our algorithm becomes

$$O\left(N\sum_{n=4}^{N} n^{2/3}\right) \leq O\left(N\frac{N(N+1)}{2}\right) = O(N^3).$$

Lastly, since $O(N) < \sum_{n=4}^{N} n^{2/3} < O(N^2)$, the algorithm has a time complexity smaller than cubic but larger than squared.

4 Method

4.1 Mixture method

The bulk of this report consist of geometric mixture algorithm implementation details. These details are discussed in this section, and below is an outline of the overall approach to the generation of mixtures.

1. Specify the mixture by, for each component, providing the shape of the reference particle, the volume distribution, weight fraction and the shape of the domain in which this mixture is to be placed.

2. The expected number of particles of each component needed to completely fill the volume of this domain whilst maintaining the specified weight fraction is computed as discussed in section 3.2.3. Note that to completely fill the domain is equivalent to creating a 0% porosity mixture.

3. Generate the above number of particles of each component's reference particle and scale them such that their volume follows the respective volume distributions. Save these particles in a list and shuffle it.

4. Initialise the advancing front by placing the first four particles in the corners of a tetrahedron centred in the domain mid point. Then move the particles toward the domain mid point until at least one pair of them are touching.

5. Iterate over the remaining generated particles, adding them to the domain, one at a time, by using the advancing front algorithm discussed in section 4.1.1.

4.1.1 Generic advancing front algorithm

The advancing front algorithm is a static method used to achieve particle mixtures with low porosities. The algorithm builds on the notion of growing the mixture of particles incrementally along some front, this front is then updated to always act as a boundary between the part of the domain which has particles and the part of the domain which has yet no particles. This updating makes the front advance in the direction of no particles, which motivates the name – advancing front.

When new particles are introduced, i.e. the front is incremented, they are done so with the constraint of them being in contact with three particles already present in the mixture, of which at least one must be in the advancing front. It is this constraint which allows the method to achieve mixtures with low porosities.

Algorithm 1 Pseudocode for generic advancing front algorithm in 3D.

$\{p_1, \ldots, p_k\}$:= set of initial particles
G := containing geometry
$E := \{p_1, \ldots, p_k\}$
$C_{\text{front}} := \{p_1, \ldots, p_k\}$
$\tilde{r}$:= set of parameters that define the next particle except for its center
while $C_{\text{front}} \neq \varnothing$ **do**
  p_0 := select particle from C_{front}
  V := all neighbouring particles to p_0
  for all $p_1', p_2' \in V$ **do**
    if $W(p_0, p_1', p_2', G, V, \tilde{r}) \neq \varnothing$ **then**
      p_{new} := any element of $W(p_0, p_1', p_2', G, V, \tilde{r})$
      $C_{\text{front}} := C_{\text{front}} \cup p_{\text{new}}$
      $E := E \cup p_{\text{new}}$
      $\tilde{r}$:= set of parameters that define the next particle except for its center
      break
    end if
  end for
  $C_{\text{front}} := C_{\text{front}} \setminus \{p_0\}$
end while
return E

Algorithm 1 presents the same generic advancing front algorithm as in [8]. Albeit missing much detail, this pseudocode outline is the core of the algorithm implemented in this project.

The idea is to define two sets, E and C_{front}, where the former constitutes all the particles present in the constraining geometry, G, and the latter is the actual advancing front. The initial particles $p_1, \ldots, p_k$ are generated and placed in some initial configuration. The next step is to generate all parameters, except the center, of the next particle. This includes; two shape parameters, three scale parameters, orientation, and class. This completes the initialisation of the advancing front and next comes the actual advancing of the front defined by C_{front}.

To advance the front the auxiliary function $W(p_1, p_2, p_3, G, V, \tilde{r})$ is used. It is defined as the set of particles with shape, volume, orientation and class defined by $\tilde{r}$, which are in outer contact with p_1, p_2, p_3 simultaneously, are fully contained in G, and do not overlap with any particle in V. The advancing is done by selecting one particle, p_0, from the active front C_{front} and the set of all its neighbours, V. Then, for each pair of neighbours $p_1', p_2' \in V$ all possible new particles, which are in contact with p_0, p_1', p_2', are defined by $W(p_0, p_1', p_2', G, V, \tilde{r})$. If this set of possible new particles is empty a new pair is selected from the same set of neighbours. If the set defined by the auxiliary function, on contrary, is non-empty, one particle from the set is selected, added to both E and C_{front} and the procedure then continues to define the parameters for the next particle to add. Lastly, whether or not a particle has been added to the domain, the p_0 is removed from the active front since it has served it purpose. If the active front C_{front} does not contain any

particles after p_0 is removed it means that G cannot be filled with more particles and the algorithm terminates with the mixture available in E. If the active front contains particles then a new $p_0 \in C_{\text{front}}$ is selected and above procedure repeated.

The pseduocode in algorithm 1 does not specify how the initial particles are placed, nor does it specify the definition of $W(p_0, p_1', p_2', G, V, \tilde{r})$. The placement of the initial particles will determine how the front advances. As seen in figure 2; if the initial particles are grouped together in the domain the front will advance outward toward the boundary of G.

In our case, four particles are generated placed in the corners of a tetrahedron with sides the same length as the radius of the larges enclosing sphere of any of the initial particles. The four particles are then moved toward the center of the tetrahedron until they cannot move further without overlapping. As for the specification of $W(p_0, p_1', p_2', G, V, \tilde{r})$, it is discussed in section 4.1.2.

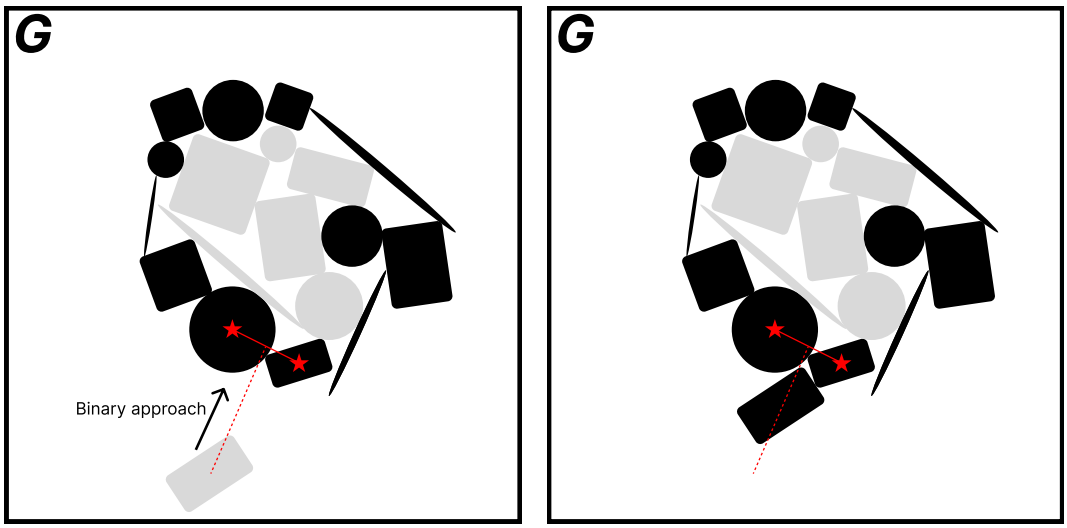


Figure 2: 2D illustration of the advancing front algorithm utilising binary approach (algorithm 2). The particles with red stars are p_0 and p_1' in algorithm 1, while the red line is r in algorithm 2.

4.1.2 Binary approach

In order to pack the particles in proximity to each other, an approximate approach method was used due to the implicit nature of the particles. Binary approach is used to find a particle in $W(p_0, p_1, p_2, G, V, \bar{r})$, defined previously.

Algorithm 2 Pseudocode for binary approach.

$p :=$ particle to be packed
$T = \{p_1, p_2, p_3\} :=$ set of target particles
$E = \{p_1, \ldots, p_k\} :=$ set of all packed particles
$r_m :=$ maximum circumscribing radius of packed particles
$r :=$ circumscribing radius of particle p.
$ct :=$ contact tolerance
$m :=$ target coordinate
$v :=$ start point of particle p
$p = m - v :=$ displacement vector
$\lambda = 0 :=$ displacement parameter
$\lambda_{low} = 0$
$\lambda_{high} = 1$
$cc \in \{TRUE, FALSE\} :=$ indicates collision
while TRUE **do**
  $x := v + \lambda r$
  $G :=$ set of particles within the box $x \pm (r_m + r + ct)$
  if $G \neq \emptyset$ **then**
    $cc := C(G)$ testing for collision
    if $CC = TRUE$ **then**
      $\lambda_{high} := \lambda$
    else if $CC = FALSE$ **then**
      $\lambda_{low} := \lambda$
    end if
  else
    $\lambda_{low} := \lambda$
  end if
  if EXIT **then**
  end while
end if
return x

Algorithm 2 was adapted from [8]. The collision checking $C(G)$ is hierarchical, done by first checking the euclidean distance between the particle centres and comparing with the sum of the circumscribing sphere radii. This then indicates a potential collision. A similar step is then run comparing against the sum of inscribed sphere radii. A definite collision can then be identified without needing the expensive exact collision check. Many unnecessary minimisation steps can then be avoided. The exact distance calculation is done only when there is a potential collision, and the second geometric check shows no definite collision.

What the λ:s does is to iteratively approach a point where the moving particle is in close contact to another. The endpoint is specified in $EXIT$ which incorporates several exit codes:

- 0.) Collision with tolerable overlap.
- 1.) No collision within contact tolerance of another particle.

- 2.) Maximum number of iterations reached (20).
- 3.) Non-viable start point.
- 4.) Maximum number of sequential collisions reached.
- 5.) Viable endpoint $x := m$.

The algorithm can then allow for deformable, soft particles.

4.1.3 Proximity search algorithm

When performing binary search the distance between every particle and the new particle need not be computed, the distance need only to be computed between the new particle and those particles which lie on the path along which the new particle travels. In order to find the possibly colliding particles quickly an algorithm has been developed which relies on binary search.

Algorithm 3 Pseudocode for proximity search algorithm.

$p_x, p_y, p_z :=$ x, y, z-coordinates of the particle to be packed
$E := \{p_1, \ldots, p_k\}$ set of all packed particles
$B_x :=$ E but ordered by increasing x-center coordinate
$B_y :=$ E but ordered by increasing y-center coordinate
$B_z :=$ E but ordered by increasing z-center coordinate
$R :=$ largest circumscribing sphere radius of all particles in $E \cup \{p\}$
$P_x :=$ `binary_search(`B_x`,` $[p_x - R, p_x + R]$`)`
$P_y :=$ `binary_search(`B_y`,` $[p_y - R, p_y + R]$`)`
$P_z :=$ `binary_search(`B_z`,` $[p_z - R, p_z + R]$`)`
return $P_x \cap P_y \cap P_z$

Algorithm 3 keeps three copies of the set of all packed particle, E, each of them ordered by the particles x, y, and z-center coordinates respectively. Then to find the particles which the new particle can collide with a box is created around the current center of the new particle. To find the particles present in this box, $[p_x - R, p_x + R] \times [p_y - R, p_y + R] \times [p_z - R, p_z + R]$, one firstly finds the particles has x, y, z-center coordinates in the respective intervals via `binary_search`, which preforms binary search on the given ordered set and returns the set of particles that lie in the specified interval in logarithmic time.

When adding a new particle to E the ordered sets B_x, B_y, and B_z are kept up to date by inserting the new particle in each set such that they stay ordered. The place at which to insert the new particle is again found in logarithmic time using binary search.

4.1.4 Minimum distance

In order to determine the minimum distance between the introduced particle and the particles of interest given by proximity search described in 4.1.3, a solution is found to the minimisation problem formalised in 3.2.1.

4.2 Contact statistics

4.2.1 Coordination number

The general formulation of the advancing front algorithm discussed in section 4.1.1 enforces the constraint on the new particles to be in contact with three particles. The implementation of the algorithm does however not place the new particles exactly, instead it uses the approximation of binary approach. This means that two particles will very seldom be in actual contact. Thus a tolerance distance is specified by the user, and if two particles are closer than the specified tolerance they are considered to be in contact.

4.2.2 Content uniformity

Content uniformity (CU) is a degree of consistency for different volumes of the mixture. The consistency can be expressed as the ratio of mean frequency of a certain particle type within a set of subsamples, to the expected frequency. In cases of high CU it is expected the above mentioned ratio is close to 1 and that the standard deviation among subsamples is small. Bootstrap is a possible method that can be used for drawing random samples and subsequently statistics of interest.

5 Results

Table 1 presents the parameters used to define the different mixture components displayed in figure 3. The mixture displayed in 3 does, furthermore, not have the constraint that each particle particle must be placed inside the defined domain. This explains the somewhat irregular form of the mixture and that some of the particles clearly lie outside the domain.

Component	Volume distribution	Volume fraction	Shape parameters (a, b, c)	Scale parameters (n_1, n_2)
1	$\mathcal{U}(1, 10)$	0.25	$(1, 1, 1)$	$(2, 2)$
2	$\mathcal{LN}(1, 0.25)$	0.25	$(1, 1, 1)$	$(8, 8)$
3	$\mathcal{N}(10, 2)$	0.25	$(5, 1, 1)$	$(5, 2)$
4	$\mathcal{WB}(1, 5)$	0.25	$(1, 3, 1)$	$(2, 2)$

Table 1: Four component particle mixture input parameters with volume distributions: Uniform ($\mathcal{U}$), Log-normal ($\mathcal{LN}$), Normal, ($\mathcal{N}$) and Weibull ($\mathcal{WB}$).

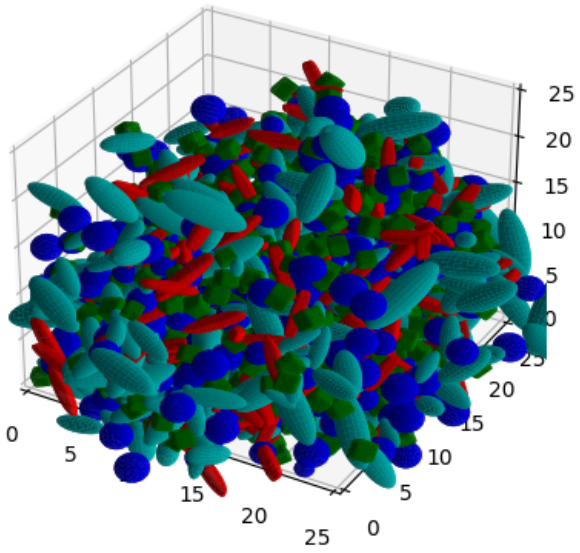


Figure 3: Particle mixture with 3273 particles of four different components placed in the domain $[0, 25]^3$ depicted with colours blue (1), green (2), red (3), cyan (4). The different components are defined in table 1.

5.1 Mixture algorithm run time

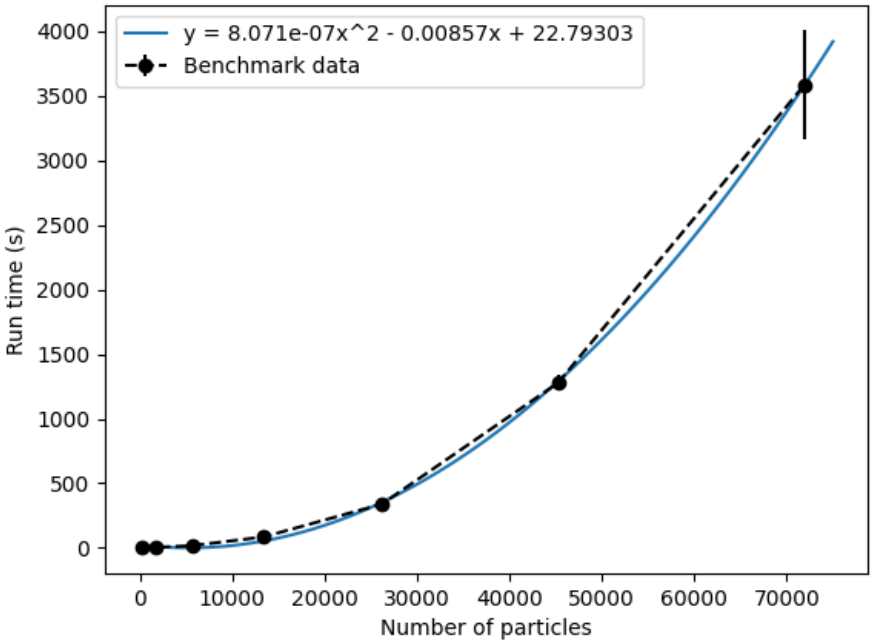


Figure 4: Run time of the advancing front algorithm plotted against the number of particles to be included in the final mixture. The mixtures' components are defined in table 1. The plot also contains a second degree polynomial fitted via polynomial regression to the benchmark data.

In figure 4 the total run time for the mixture algorithm is plotted against the number of particles that it present in the mixture. The different components in the mixture is defined in table 1 and there is no constraint added to inhibit particles to be placed outside the domain. This means that the run time data corresponds to mixtures alike the one presented in figure 3, but with different number of particles and in different sized domains. In the same figure a second degree polynomial fitted to the benchmark data is also included, which can be seen align well with the benchmark data. This can be used to argue that the time complexity of the mixture algorithm is polynomial, which aligns with the theoretical results presented in section 3.3.

5.2 Porosity

For a domain of size $20 \times 20 \times 20$, filling it with spheres with radius 1, a mean porosity of 0.658 with a standard deviation of 0.00420 was achieved. A realisation of one such mixture can be seen in figure 5. Note that in these mixtures, contrary to the ones discussed in section 5.1, particles are allowed to have parts outside the domain, but not centres. This entails that the stated porosity is an over-estimate.

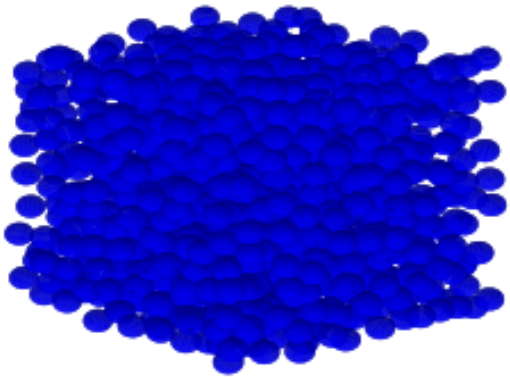


Figure 5: Mixture of single-size spheres of radius 1 filling a domain of size 20^3. The porosity is 0.657.

As the placed particles are fixed once placed, the method used is similar to random sequential adsorption. The achieved porosity may then be compared to the packing density achieved by [11] using RSA at 0.6159. Something to note here is that this is far from a maximally dense packing achievable using spheres at with a porosity $\approx 26\%$ or the maximum density of random jammed packings at $\approx 36\%$ [9].

6 Discussion

6.1 Binary approach

In hindsight, the method chosen to place particles, *binary approach* has the same flaw as DEM using multi-scale components; when a large particle is moved, many collision checks need to be run against small particles, meaning that multi-scale modelling runs into problems of high run times. Another problem is that without a step where the particles can be moved after their initial placement, the minimum porosity is roughly 62%. A fundamentally different approach is needed to solve the placement problem for low porosities, with multi-scale particles.

6.2 Proximity search algorithm

One way of improving the run time, and even the time complexity, is to revise the proximity search method. In [3] they employ a method where the domain is divided into spacial cells and each particle is then mapped into one of them depending on their center point. This method makes it easy to find possible colliding particles by querying the current and the neighbouring cells for their containing particles. In contrast with our proximity search algorithm, this method does not rely on the expensive intersection operation between sets that grows almost quadratic with the number of particles in the mixture. The cell method instead utilises a non-binary search method explained in [7], which subsequently, in the context of generating particle mixtures by, was adapted to parallel usage on GPUs by [5].

6.3 Interior point solver and coordination number

In order for binary approach to place particles next to each other, a way of determining the particles distance to each other was needed. Initially we intended to solve this problem by finding the minimum distance between two points by solving the minimisation problem formulated in 3.2.1. Using the python package `PYOMO`, which employed the interior point solver `IPOPT` made this possible, but it managed no more than 100 distance checks per second. This was too slow, since each binary approach iteration could use up to 100 distance checks, resulting in only one particle generated per second. We believed that the low speed was due to `IPOPT` having to approximate Jacobians and Hessians of both constraint and objective functions, due to all settings in `IPOPT` not being available in the `PYOMO` package. After moving on with another Python package `PYPOWER` which allowed for inputs of exact expressions of Jacobians and Hessians, the speed was still too slow. Even worse, it produced incorrect solutions sometimes. At this point, we took more interest in an alternative approach based on 3.2.2. This yielded a much better result and enabled us to generate larger particle mixtures at a much higher rate than previously, with the disadvantage of not knowing the exact minimum distance between the particles. We also moved our project from Python to C++ in hopes of increasing computational speed further, which was achieved.

Since the binary approach wasn't dependent on an interior point solver to work, its main use was to collect coordination number statistics of generated particle mixtures. We found and tried implementing a C++ based library called `IFOPT` into our project in order to get a working version of minimum distance checks, but were unable to make it work due to technical difficulties and unforeseen errors. After many attempts we instead decided to implement a solver ourselves, which just aimed to solve the KKT-conditions formulated in 3.2.1 by performing Newton steps and adaptively reducing the barrier parameter. Our own solver did however fail to converge for approximately 10% of the randomly generated particles during testing. This unwanted behaviour occurred increasingly for particles with higher shape parameters, i.e. "blocky" particles. At this point further investigation into why convergence was not always guaranteed was limited due to project time constraints. Since the solver could produce incorrect results, it would not be suitable for generating coordination number statistics, and is thus the reason that no such statistics could be provided.

References

[1] Jasim Ahmed, Muhammad Al-Foudari, Fatimah Al-Salman, and Abdulwahab S Almusallam. Effect of particle size and temperature on rheological, thermal, and structural properties of pumpkin flour dispersion. *Journal of Food Engineering*, 124:43–53, 2014.

[2] Nilanjan Chakraborty, Jufeng Peng, Srinivas Akella, and John E. Mitchell. Proximity queries between convex objects: an interior point approach for implicit surfaces. *IEEE Transactions on robotics*, pages 1910–1916, May 2006. MAG ID: 2162537142 S2ID: 0117f9b83c9eae07154c1a9944afe26046062c42.

[3] Alejandro C. Frery, Lorena Rivarola-Duarte, Lorena Rivarola-Duarte, Viviane Carrilho Leão Ramos, Viviane Carrilho Leão Ramos, Adeildo S. Ramos, and William Wagner Matos Lira. Stochastic particle packing with specified granulometry and porosity. *Granular Matter*, July 2012. ARXIV_ID: 1207.2961 MAG ID: 2029949458 S2ID: 7f38cb9ea68bb7e7765b0d333d56cfef4bcae5cd.

[4] Lian X Liu, Ivan Marziano, AC Bentham, Jim D Litster, ET White, and T Howes. Effect of particle properties on the flowability of ibuprofen powders. *International journal of pharmaceutics*, 362(1-2):109–117, 2008.

[5] Lucas G.O. Lopes, Diogo T. Cintra, and William W.M. Lira. A particle packing parallel geometric method using gpu. *Computational Particle Mechanics*, 8(4):931–942, 2021.

[6] Guilhem Mollon and Jidong Zhao. Fourier–voronoi-based generation of realistic samples for discrete modelling of granular materials. *Granular Matter*, 14(5):621–638, June 2012. MAG ID: 2004166774 S2ID: cc4d71f206dfcc76f18a9d647021b490499ddd51.

[7] Andrews Munjiza and KRF Andrews. Nbs contact detection algorithm for bodies of similar size. *International Journal for Numerical Methods in Engineering*, 43(1):131–149, 1998.

[8] Carlos Recarey, Irvin Pérez, Roberto Roselló, Márcio Muniz, Elizabeth Hernández, Robinson Giraldo, and Eugenio Oñate. Advances in particle packing algorithms for generating the medium in the discrete element method. *Computer Methods in Applied Mechanics and Engineering*, 345:336–362, March 2019. MAG ID: 2901630381 S2ID: f8676f566c69fe858802927f674d59e8ae88baeb.

[9] Chaoming Song, Ping Wang, and Hernán A Makse. A phase diagram for jammed matter. *Nature*, 453(7195):629–632, 2008.

[10] Jiao Sun, Fan Wang, Yue Sui, Zhennan She, Wenjun Zhai, Chunling Wang, and Yihui Deng. Effect of particle size on solubility, dissolution rate, and oral bioavailability: evaluation using coenzyme q10 as naked nanocrystals. *International journal of nanomedicine*, pages 5733–5744, 2012.

[11] Ge Zhang and Salvatore Torquato. Precise algorithm to generate random sequential addition of hard hyperspheres at saturation. *Physical Review E*, 88(5):053312, 2013.