Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 197 (2004) 139–166

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A linear programming algorithm to test for jamming in hard-sphere packings

Aleksandar Donev [a,b], Salvatore Torquato [a,b,c,*], Frank H. Stillinger [c], Robert Connelly [d]

[a] *Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544, USA*
[b] *Princeton Materials Institute, Princeton University, Princeton, NJ 08544, USA*
[c] *Department of Chemistry, Princeton University, Princeton, NJ 08544, USA*
[d] *Department of Mathematics, Cornell University, Ithaca, NY 14853, USA*

Received 3 September 2003; received in revised form 25 November 2003; accepted 26 November 2003
Available online 17 January 2004

Abstract

Jamming in hard-particle packings has been the subject of considerable interest in recent years. In a paper by Torquato and Stillinger [J. Phys. Chem. B 105 (2001)], a classification scheme of jammed packings into hierarchical categories of *locally*, *collectively* and *strictly jammed* configurations has been proposed. They suggest that these jamming categories can be tested using numerical algorithms that analyze an equivalent contact network of the packing under applied displacements, but leave the design of such algorithms as a future task. In this work, we present a *rigorous* and *practical* algorithm to assess whether an ideal hard-sphere packing in two or three dimensions is jammed according to the aforementioned categories. The algorithm is based on linear programming and is applicable to regular as well as random packings of finite size with hard-wall and periodic boundary conditions. If the packing is not jammed, the algorithm yields representative multi-particle unjamming motions. Furthermore, we extend the jamming categories and the testing algorithm to packings with significant interparticle gaps. We describe in detail two variants of the proposed randomized linear programming approach to test for jamming in hard-sphere packings. The first algorithm treats ideal packings in which particles form perfect contacts. Another algorithm treats the case of jamming in packings with significant interparticle gaps. This extended algorithm allows one to explore more fully the nature of the feasible particle displacements. We have implemented the algorithms and applied them to ordered as well as random packings of circular disks and spheres with periodic boundary conditions. Some representative results for large disordered disk and sphere packings are given, but more robust and efficient implementations as well as further applications (e.g., non-spherical particles) are anticipated for the future.
© 2003 Elsevier Inc. All rights reserved.

Keywords: Hard-sphere system; Packings; Jamming; Rigidity theory; Linear programming

[*] Corresponding author. Tel.: +1-609-258-3341; fax: +1-609-258-6878.
E-mail address: torquato@princeton.edu (S. Torquato).

0021-9991/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2003.11.022

1. Introduction

Packings of hard particles interacting only with infinite repulsive pairwise forces on contact are applicable as models of complex many-body systems because repulsive interactions are the primary factor in determining their structure. Hard-particle packings are therefore widely used as simple models for granular media [1,2], glasses [3], liquids [4] and other random media [5], to mention a few examples. Furthermore, hard-particle packings, and especially hard-sphere packings, have inspired mathematicians and been the source of numerous challenging (many still open) theoretical problems [6].

We focus our attention in this paper on the venerable idealized hard-sphere model, i.e., the only interparticle interaction is an infinite repulsion for overlapping particles. This idealization is crucial because it enables us to be precise about the important concept of "jamming". This concept is closely related to that of *rigid* or *stable* packings in the mathematical literature. In the present work, hard-sphere jamming is presented from a rigorous perspective by focusing on the *geometry* of the *final* packed states. Rigidity of central-force (spring) networks, and in particular rigidity percolation, has been the subject of extensive work in the physics literature [7,8]. For disordered generic networks, the question of whether a network is rigid or not becomes combinatorial in nature and there are very efficient algorithms that give the answer for systems with millions of degrees of freedom on a personal workstation, at least in two dimensions [9]. However, for particle packings, geometry is crucial, and the problem of verifying jamming or rigidity is significantly more difficult. Nonetheless, the algorithm presented here is rigorous and polynomial (it is in the same class as linear programming (LP) problems) for ideal packings (as defined in Section 2.1), and thus appreciably enlarges the scope of rigidity problems which can be studied computationally.

There are still many important and challenging questions open even for the simplest type of hard-particle packings, i.e., packing of perfectly impenetrable equal (congruent) spheres (called monodisperse packings in the physics literature). One category of open problems pertains to the enumeration and classification of disordered disk and sphere packings, such as the precise identification and quantitative description of the maximally random jammed (MRJ) state [10], which has supplanted the ill-defined "random close packed" (RCP) state. Others pertain to the study of ordered systems and finding packing structures with extremal properties, such as the lowest or highest (for packings of unequal spheres, called polydisperse packings in the physics literature) density jammed disk or sphere packings, for the various jamming categories described below [11,12]. Numerical algorithms have long been the primary tool for studying random packings quantitatively. In this work, we take an important step toward future studies aimed at answering the challenging questions posed above by designing tools for algorithmic assessment of the jamming category of finite packings.

In Section 2, we present the conceptual theoretical framework underlying this work. Specifically, we review and expand on the hierarchical classification scheme for jammed packings into locally, collectively and strictly jammed packings proposed in [13]. In Section 3, we present a randomized linear programming algorithm for finding unjamming motions within the approximation of small displacements, focusing on periodic boundary conditions in Section 4. This algorithm is rigorous when applied to *ideal* packings, where interparticle gaps are very small. In Section 5, we extend the concepts of jamming and the randomized linear programming algorithm to packings that have significant interparticle gaps and do not fit well in the rigorous framework suitable for ideal packings. We also introduce a randomized sequential loading algorithm to study non-ideal packings. We discuss the two algorithms in detail and describe a preliminary, but efficient, implementation in Section 6. Results of physical relevance obtained using this implementation are presented in a separate publication [14]. Here, we only give some representative illustrations and timing statistics in order to illustrate the utility of the proposed algorithms. We focus here on sphere packings. However, extensions to packings of non-spherical smooth convex particles, and in particular, packings of ellipsoids, are possible and are the subject of current and future work. The ideas that we employ here are drawn heavily from the mathematics literature [7,15–17]. Some mathematical preliminaries are given here, and more technical points are deferred to Appendices A, B, C.

2. Jamming in hard-particle packings

The physical intuition behind the word jamming is strong: It connotes that a given configuration is "frozen" or "trapped". Two main approaches can be taken to define jamming, *kinematic* or *static*. In the kinematic approach, one considers the motion of particles away from their current positions, and this approach is for example relevant to the study of flow in granular media. [1] The term *jammed* seems most appropriate here. In the static approach, one considers the mechanical properties of the packing and its ability to resist external forces. [2] The term *rigid* is often used among physicists in relation to such considerations. However, due to the correspondence between kinematic and static properties, i.e., strains and stresses, these two different views are largely equivalent.

In this paper we largely adopt a kinematic approach, as we focus on the geometry of packings, but the reader should bear in mind the inherent ties to static approaches. We first give a general approach to jamming in hard-particle packings in Section 2.1, and then focus on the fundamental and rigorous case of packings with ideal interparticle contacts (i.e., no interparticle gaps) in Section 2.2, studied both in the physics and mathematics literature. Finally, we connect these definitions to the kinematic concept of unjamming motion in Section 2.3, and also to static concepts in Section 2.4. Since we are attempting to bring together several apparently different approaches and terminologies, as well as generalize to packings with interparticle gaps, the exposition will be gradual and more detailed discussion, illustrations and proofs are delayed to later parts of this paper.

2.1. Jamming as isolation in configuration space

Capitalized bold letters will be used to denote dN-dimensional vectors which correspond to the d-dimensional vectors of all N of the particles. Note also that we often use "sphere" and "ellipsoid" in any dimension d, but sometimes we will emphasize "disk" and "ellipse" in two dimensions for clarity.

A *hard-particle packing* $P(\mathbf{R})$ is characterized by the positions and orientations of N non-overlapping particles, which give the *configuration* $\mathbf{R}$. In particular, a *sphere packing* in a finite region in d-dimensional Euclidean space $\Re^d$ is characterized only by the positions of the sphere centers $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$,

$$P(\mathbf{R}) = \left\{ \mathbf{r}_i \in \Re^d,\ i = 1, \ldots, N \,\middle|\, \|\mathbf{r}_i - \mathbf{r}_j\| \geqslant \frac{D_i + D_j}{2}\ \forall j \neq i \right\},$$

where the diameter of the ith sphere is D_i. Two configurations are identical if all interparticle distances are the same, i.e., if the configurations are related via a rigid-body motion (and possibly a mirror inversion in addition). We focus here on monodisperse (i.e., $D_i = D = \text{const.}$) hard-sphere packings for simplicity, but some of the conclusions are in fact applicable to particles of any strictly convex shape, and in particular ellipsoids. In the case of ellipsoids though, there are $d(d-1)/2$ additional degrees of freedom per particle associated with the possible rotations of the particles, and these need to be considered as part of the configuration. We are currently working on generalizations and extensions of the theory and algorithms to packings of ellipsoids.

[1] In particular, the cessation of flow as jamming is approached.

[2] In particular, the infinite elastic moduli near jamming.

Our perspective on jamming focuses on the set $\mathcal{J}_{\mathbf{R}}$ of configurations around a particular initial configuration $\mathbf{R}$ reachable via continuous displacements of the spheres, subject to non-overlapping constraints and certain boundary conditions. An illustration of this is provided in Fig. 1 for a very simple case in which only one disk is free to move, i.e., there are only two degrees of freedom. If $\mathcal{J}_{\mathbf{R}}$ is isolated in configuration space, we call it a *basin of jamming*, and the configuration $\mathbf{R} \in \mathcal{J}_{\mathbf{R}}$ determines a *jammed packing* $P(\mathbf{R})$. To relate this to the physical intuition of jamming, we must further ask that the extent of $\mathcal{J}_{\mathbf{R}}$ be small, in the sense that only small *continuous* displacements of the particles from their initial configurations are possible for all $\mathbf{R} \in \mathcal{J}_{\mathbf{R}}$. The natural length scale defining the meaning of "small" is the typical size of the particles, and also the typical size of the interparticle gaps. A more strict mathematical definition of jamming considers packings that have perfect interparticle contacts, which we will call *ideal packings*. For a jammed ideal packing $\mathbf{R}$ is an isolated *point* in configuration space, i.e., $\mathcal{J}_{\mathbf{R}} = \{\mathbf{R}\}$, so that the particles cannot at all be displaced continuously from their current configuration (modulo trivial rigid-body motions). We focus first on this strict definition, and we will return to the issue of interparticle gaps later. By changing the boundary conditions, we get several different categories of jamming, namely local, collective and strict jamming.

2.2. Three jamming categories

First we repeat, with slight modifications as in [18], the definitions of several hierarchical jamming categories as taken from [13], and later we make them mathematically specific and rigorous for several different types of sphere packings. When defining jamming, one must be very specific about the type of boundary conditions imposed on the packing, for example, the packing may be contained inside a hard-

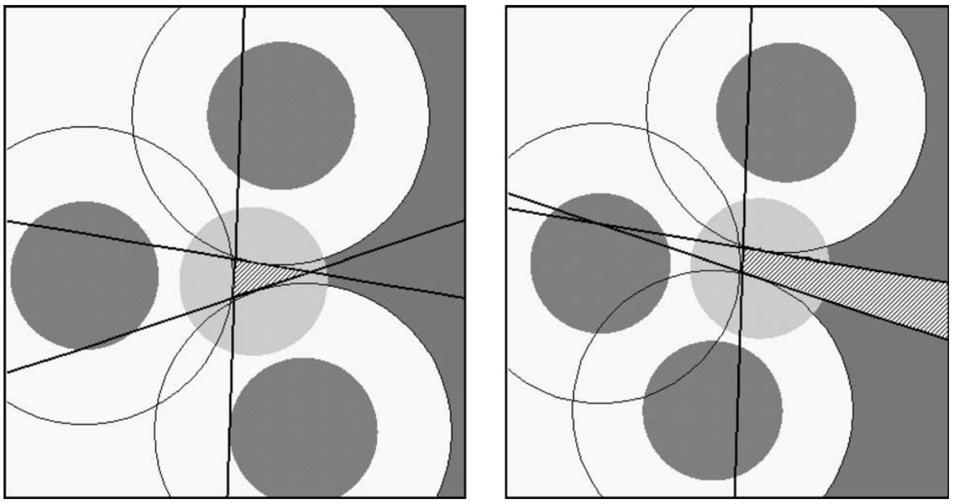
Fig. 1. Feasible displacements polyhedron. The figures show three stationary (dark gray) circular disks surrounding a mobile disk (light gray). For each of the three stationary disks, we have a nonlinear impenetrability constraint that excludes the mobile disk from a disk of radius D surrounding each stationary disk (dark circles), delimiting the (non-convex) region of reachable configurations $\mathcal{J}_{\mathbf{R}}$. Also shown are the linearized versions of these constraints (dark lines), which are simply tangents to the circles at the point of closest approach, as well as the region of feasible displacements bounded by these lines (shaded gray). This region is a polyhedral set, and in the left figure it is bounded, meaning that within the approximation of small displacements (ASD) the mobile disk is locally jammed (trapped) by its three neighbors, while on the right it is unbounded, showing the cone of locally unjamming motions (escape routes). Notice that with the true nonlinear constraints, the mobile disk can escape the cage of neighbors in both cases, showing that the ASD is not exact. However, it should also be clear that this is because we have relatively large interparticle gaps here.

wall container. For now we simply assume that some boundary conditions are imposed, and we specialize the meaning of the terms boundary and boundary deformation for specific types of packings in the following section.

A finite system of spheres is:

Locally jammed. Each particle in the system is locally trapped by its neighbors, i.e., it cannot be translated while fixing the positions of all other particles. This definition is analogous to the definition of *1-stability* in [16]. Because of its simplicity, this definition has been overused to obtain theoretical estimates of the density of random packings [19,20].

Collectively jammed. Any locally jammed configuration in which no subset of particles can simultaneously be continuously displaced so that its members move out of contact with one another and with the remainder set. An equivalent definition is to require that all finite subsets of particles be trapped by their neighbors. Compare this to the definition of *finite stability* in [16].

Strictly jammed. Any collectively jammed configuration that disallows all globally uniform volume-non-increasing deformations of the system boundary. Note the similarity with collective jamming but with the additional provision of a deforming boundary. This difference and the physical motivations behind it should become clearer in Section 4.3. Compare this to the definition of *periodic stability* in [16] for packings with periodic boundary conditions.

Observe that these are ordered *hierarchically*, with local being implied by collective and similarly collective being implied by strict jamming. We point out that these do not exhaust all possibilities and various intricacies can arise, especially when considering infinite packings [16].

2.3. Unjamming motions

Note that the mathematics literature often uses the term *rigid* or *stable packing* for what we call a jammed packing in Section 2.2. It can be shown [17] that to assess jamming for a given sphere packing, one need only look for the existence of *analytic* continuous displacements of the particles from their current configuration. [3] An *unjamming motion* $\Delta\mathbf{R}(t) = (\Delta\mathbf{r}_1(t), \dots, \Delta\mathbf{r}_N(t))$, where t is a time-like parameter, $t \in [0, 1]$, is a continuous analytic displacement of the spheres from their current position along the path $\mathbf{R} + \Delta\mathbf{R}(t)$, starting from the current configuration, $\Delta\mathbf{R}(0) = 0$, and ending at the final configuration $\mathbf{R} + \Delta\mathbf{R}(1)$, while observing all relevant constraints along the way, such that some of the contacting spheres lose contact with each other for $t > 0$. This means that impenetrability and any other particular (boundary) conditions must be observed, i.e., $P(\mathbf{R} + \Delta\mathbf{R}(t))$ is a *valid* packing for all $t \in [0, 1]$. If such an unjamming motion does not exist, we say that the packing is *jammed*. By changing the (boundary) constraints we get different categories of jamming, such as local, collective and strict.

It can be shown (see [17]) that an equivalent definition [4] is to say that a packing is jammed if it is isolated in the allowed configuration space, i.e., there is no valid packing within some (possibly small) finite region around $\mathbf{R}$ that is not equivalent (congruent) to $P(\mathbf{R})$. In the language of Section 2.1, $\mathscr{J}_{\mathbf{R}} = \{\mathbf{R}\}$.

Furthermore, it is a simple yet fundamental fact that we only need to consider first derivatives $\mathbf{V} = (\mathrm{d}/\mathrm{d}t)\Delta\mathbf{R}(t)$, which can be thought of as velocities, and then simply move the spheres in the directions $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_2)$ to obtain an unjamming motion $\Delta\mathbf{R}(t) = \mathbf{V}t$. Therefore, henceforth special consideration will be given to the final displacement $\Delta\mathbf{R}(1)$, so that we will most often just write $\Delta\mathbf{R} = \Delta\mathbf{R}(1)$. The formal statement is that *a packing is rigid if and only if it is infinitesimally rigid*, [5] see [15]. Although the proofs of this statement published in the mathematics literature consider packings of equal spheres in a hard-wall container, the proof carries directly to the case of collective jamming with periodic boundary conditions

[3] This is the third definition (definition *c*) in Section 2.1 of [17].

[4] This is the first definition (definition *a*) in Section 2.1 of [17].

[5] This is not true for packings of ellipsoids, which may be rigid but not infinitesimally rigid.

(i.e., packings on a flat torus), as well as packings of unequal spheres. As discussed in Section 4.3, the statement is also true for strict jamming with periodic boundary conditions. A sphere packing is not jammed if and only if one can give the spheres velocities $\mathbf{V}$ such that no two contacting spheres i and j, $\|\mathbf{r}_i - \mathbf{r}_j\| = D$, have a relative speed $v_{i,j}$ toward each other: [6]

$$v_{i,j} = (\mathbf{v}_i - \mathbf{v}_j)^{\mathrm{T}} \mathbf{u}_{i,j} \leqslant 0, \tag{1}$$

where

$$\mathbf{u}_{i,j} = \frac{\mathbf{r}_j - \mathbf{r}_i}{\|\mathbf{r}_i - \mathbf{r}_j\|}$$

is the unit vector connecting the two spheres. [7] Of course, some special and trivial cases like rigid body translations ($\mathbf{V} = \text{constant}$) or rigid body rotations need to be excluded since they do not really change the configuration of the system. We will elaborate on this "linearized" perspective in the context of packings with interparticle gaps in Section 3.1.

In this paper we will plot unjamming motions as "velocity" fields, and occasionally supplement such illustrations with a sequence of frames from $t = 0$ to $t = 1$ showing the unjamming process. Note that the lengths of the vectors in the velocity fields have been scaled to aid in better visualization. For the sake of clear visualization, only two-dimensional examples will be used, however, all of the techniques described here are fully applicable to three-dimensional packings as well. Interactive virtual reality modeling language (VRML) animations which are very useful in getting an intuitive feeling for unjamming mechanisms in sphere packings can be viewed on our webpage [22].

2.4. Jamming and forces

We have defined jamming above using kinematic concepts and focused on the positions of the particles, i.e., on the *geometry* of the packings. It is very instructive to discuss briefly the relations between contact forces and applied loads in the context of jamming. This is crucial because of the physical importance of statical considerations in the study of granular materials and the preponderance of force-based discussions in the physics literature. Furthermore, forces play a very important role in the analysis of the configuration-based definitions given above as dual variables associated with impenetrability constraints, and have appeared prominently in the mathematics literature as well [17]. Ref. [23] contains a wide-ranging discussion of the relation between geometry and forces.

Consider a configuration belonging to a basin of jamming, $\mathbf{R} \in \mathscr{J}_{\mathbf{R}}$, and an applied load $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_N)$ on the particles. In the case of spheres, $\mathbf{b}_i$ is just the total force acting on sphere i (for example, due to thermal or mechanical vibrations or externally applied fields). In the case of ellipsoids, it would also contain the total torque acting on each particle. Assume for simplicity that this load is independent of the configuration. Under this load, the particles will displace to a new configuration of minimal energy:

$$\begin{aligned} &\max_{\Delta\mathbf{R}} \mathbf{B}^{\mathrm{T}} \Delta\mathbf{R} \quad \text{for virtual work} \\ &\text{such that } \mathbf{R} + \Delta\mathbf{R} \in \mathscr{J}_{\mathbf{R}} \quad \text{for impenetrability.} \end{aligned} \tag{2}$$

Since the packing is jammed, the program (2) will have a bounded solution which lies on the boundary of $\mathscr{J}_{\mathbf{R}}$, i.e., some particles will be in contact in the new configuration. The Lagrange multipliers associated

[6] See Section 2.2 of [17].

[7] The sign notation may be a bit unorthodox but is taken from [21].

with the impenetrability constraints are in fact the reaction contact forces which resist the applied equilibrium load $\mathbf{B}$.

We thus see the meaning of the three jamming categories in the static context: In a locally jammed packing each particle i can support *any* load $\mathbf{b}_i$ if its neighbors are fixed. A collectively jammed packing can resist (support) *any* loading without rearrangements of the particles as long as the boundary is held fixed externally. Strictly jammed packings on the other hand can support *any* load with a compressive global (boundary) component (i.e., positive macroscopic pressure). Note, however, that a packing may be able to support all compressive global loads even though it is not strictly jammed, as it may be unstable due to the existence of collective unjamming mechanisms. [8]

2.5. Jammed subpackings

It should be mentioned that jammed random particle packings produced experimentally or in simulations typically contain a small population of "rattlers", i.e., particles trapped in a cage of jammed neighbors but free to move within the cage. For present purposes we shall assume that these have been removed before considering the (possibly) jammed remainder. This idea of excluding rattlers can be further extended to "rattling clusters" of particles, i.e., groups of particles that can be displaced collectively even though the remainder of the packing is jammed. In fact, one can consider any packing which has a jammed subpacking (collectively or strictly as defined above, with *identical* boundary conditions) to be jammed.

The physical meaning and mathematical basis for such a modified approach is more evident from the static perspective. Specifically, as long as there is a jammed subpacking, this subpacking will resist (support) global loads (stresses), and furthermore, this jammed subpacking is also able to resist local loads, such as, for example, induced by vibrations (shaking) in granular materials, therefore making the whole packing stable and rigid.

2.6. Boundary conditions

Large or infinite packings are most easily created by periodically repeating a certain finite (and possibly small) known packing. A *repetitive packing* $\hat{P}(\mathbf{R})$ is generated by replicating a finite *generating packing* $P(\hat{\mathbf{R}})$ on a lattice $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_d\}$, where $\boldsymbol{\lambda}_i$ are linearly independent *lattice vectors* and d is the spatial dimensionality. The positions of the spheres are generated by

$$\mathbf{r}_{\hat{i}(\mathbf{n}_c)} = \hat{\mathbf{r}}_i + \mathbf{\Lambda}\mathbf{n}_c \text{ and } \mathbf{n}_c \text{ is integer}, \quad \mathbf{n}_c \in Z^d, \tag{3}$$

where we think of the *lattice* $\mathbf{\Lambda}$ as a matrix with d^2 elements having the lattice vectors as columns and $\mathbf{n}_c$ the number of replications of the unit cell along each basis direction. The sphere $\hat{i}(\mathbf{n}_c)$ is the familiar *image sphere* of the *original sphere* $i \equiv \hat{i}(0)$, and of course for the impenetrability condition only the nearest image matters. Notice that condition (3) only gives the positions of the spheres, and additional boundary conditions need to be specified before applying the jamming definitions from Section 2.2.

As previously mentioned, the boundary conditions imposed on a given packing are very important, especially in the case of strict jamming. Here, we consider two main types of boundary conditions, hard-wall and periodic boundary conditions.

Hard-wall boundaries. The packing $P(\mathbf{R})$ is placed in an impenetrable concave *hard-wall container* K (see [15]). Fig. 2 shows that the honeycomb lattice can be unjammed inside a certain hard-wall container. We can also make an effective container out of N_f *fixed spheres* whose positions cannot change.

[8] An example is the Kagomé lattice disk packing, which can support all compressive global loads (sometimes called "loads at infinity" in the engineering literature), but is not collectively jammed with periodic boundary conditions.

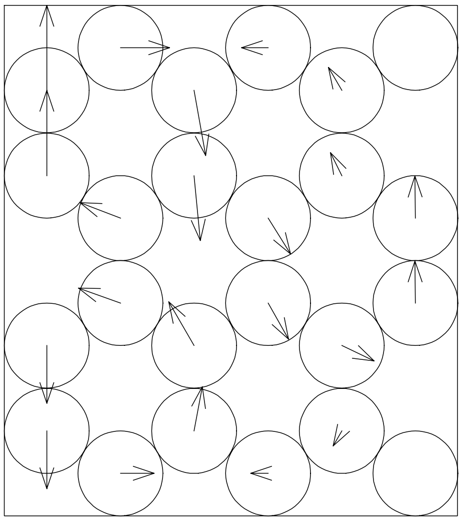
Fig. 2. Unjamming the honeycomb lattice. A subpacking of size $\mathbf{N}_c = (3, 2)$ unit cells of the infinite honeycomb lattice disk packing is placed inside a hard-wall rectangular container. The arrows in the figures given here show the direction of motion of the spheres $\mathbf{V}$ in the linear unjamming motion, scaled by some arbitrary constant to enhance the figure.

This is because it is often hard to fit a packing into a simple container such as a square box, while it is easy to surround it with other fixed spheres, particularly if a periodic lattice is used to generate the packing. Specifically, one can take a finite subpacking of an infinite repetitive packing and freeze the rest of the spheres, thus effectively making a container for the subpacking. An example is depicted in Fig. 3. Note that hard-wall containers do not allow any trivial unjamming motions.

Periodic boundaries. Periodic boundary conditions are often used to emulate infinite systems, and they fit the algorithmic framework of this work very nicely. To obtain a *periodic packing* we wrap a repetitive packing $\hat{P}(\mathbf{R})$ around a flat torus, i.e., we ask that whatever happens to a sphere i also happens to all of the image spheres $\hat{\imath}(\mathbf{n}_c)$, with the additional provision that the lattice may also change by $\Delta\mathbf{\Lambda}$:

$$\Delta\mathbf{r}_{\hat{\imath}(\mathbf{n}_c)} = \Delta\hat{\mathbf{r}}_i + (\Delta\mathbf{\Lambda})\mathbf{n}_c. \tag{4}$$

When the lattice is fixed ($\Delta\mathbf{\Lambda} = 0$), periodic boundary conditions allow for trivial rigid body translations of the packing, but trivial rotations only exist if the lattice is allowed to change. Furthermore, by imposing a suitable condition on the deformation of the lattice $\Delta\mathbf{\Lambda}$, as described in Section 4.3, one can eliminate the trivial rigid-body rotations of the packing.

2.7. Generating hard-particle packings

Algorithms to generate large-scale hard-particle packings are very important, especially because experimental hard-particle configurations are very hard to obtain and are limited in applicability. Of particular interest are stochastic algorithms aimed at producing *random* (*disordered*) *packings*. Many such algorithms have been proposed and used in previous work, as explained in more detail in [14]. We produced most packings using the Lubachevsky–Stillinger compression algorithm [24] with periodic boundary conditions. This algorithm is essentially a hard-sphere molecular dynamics in which the spheres grow in size

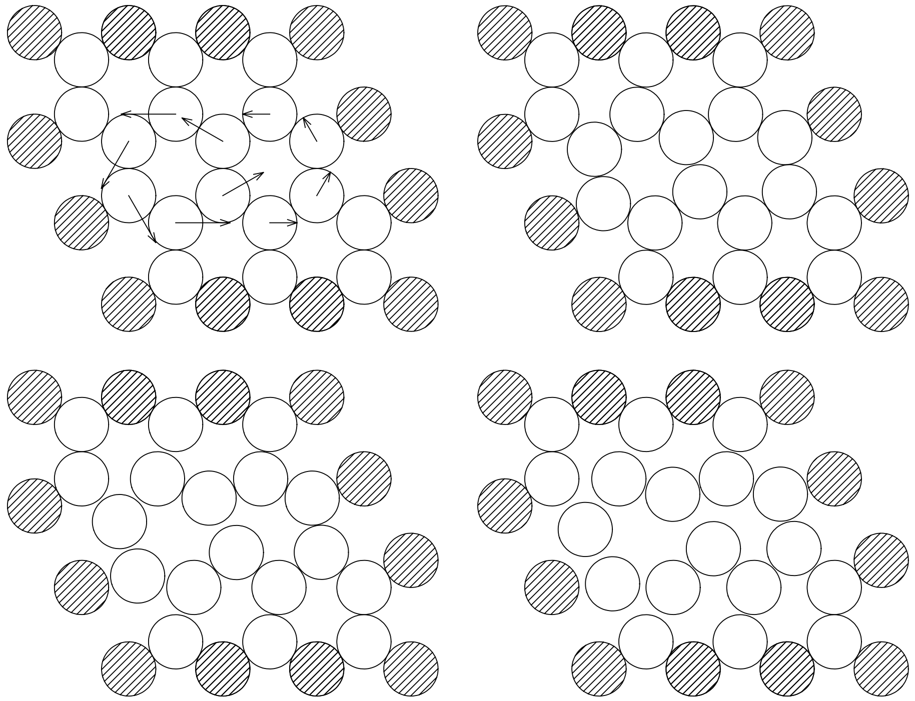

Fig. 3. Unjamming the honeycomb lattice. A subpacking of size $\mathbf{N}_c = (3, 3)$ of an infinite honeycomb packing is pinned by freezing all neighboring image disks. A representative unjamming motion is shown as a sequence of several frames between times $t = 0$ and $t = 1$ (in the order top left, top right, bottom left and bottom right). The unshaded disks represent the particles in the generating packing $P(\hat{\mathbf{R}})$, while the shaded ones are image disks that touch one of the original disks.

during the course of the simulation at a certain expansion rate and collide with each other elastically. In the limit of an infinite number of collisions, a final state is reached in which the collision rate diverges and the particles cannot grow any further. We have extended this algorithm to generate packings of ellipses and ellipsoids and developed a methodology to access jamming *during* the compression algorithm, however, this work will be presented in future publications. A number of packings produced by other methods, such as the Zinchenko algorithm [25], have also been tested, with similar results.

2.7.1. Using simple lattices to generate packings

Familiar lattices with a simple basis (unit cell), such as the triangular, honeycomb, Kagomé and square in two dimensions, or the simple cubic (SC), body-centered cubic (BCC), face-centered cubic (FCC) and hexagonal-close packed (HCP) in three dimensions, can be used to create a (possibly large) packing taking a subsystem of size $\mathbf{N}_c$ unit cells along each dimension from the infinite lattice packing. The properties of the resulting system can be studied with the tools developed here, provided that we restrict ourselves to finite $\mathbf{N}_c$. Moreover, it is important to specify which lattice vectors are to be used. We will usually take them to be primitive vectors (for which there is one particle per unit cell), but sometimes it will be more convenient to use conventional ones, as used in the physics literature (usually representing a cubic unit cell having more then one particle per unit cell for variations on the cubic lattice).

For hard-wall boundary conditions, we can take an infinite packing generated by these simple lattices and then freeze all but the spheres inside the window of $\mathbf{N}_c$ unit cells, thus effectively obtaining a hard-wall container. Fig. 3 illustrates an unjamming motion for the honeycomb lattice under these conditions.

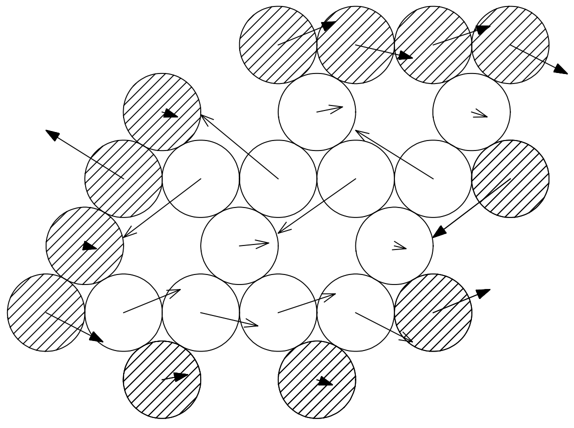

Fig. 4. Unjamming the Kagomé lattice. Periodic boundary conditions are used with $\mathbf{N}_c = (2, 2)$. Shaded disks represent periodic images and have the same velocity as their unshaded original disks.

For periodic boundary conditions, the generator $P(\hat{\mathbf{R}})$ can itself be generated using $\mathbf{N}_c$ unit cells of a simple lattice. [9] In this case the lattice $\mathbf{\Lambda}$ is a *sublattice* of the underlying (primitive) lattice $\tilde{\mathbf{\Lambda}}$, i.e., $\mathbf{\Lambda} = \tilde{\mathbf{\Lambda}}\mathrm{Diag}\{\mathbf{N}_c\}$, where $\mathrm{Diag}\{\mathbf{N}_c\}$ denotes a diagonal matrix whose diagonal is $\mathbf{N}_c$. This is not only a convenient way to generate simple finite periodic packings, but it is in general what we mean when we ask, for example, to analyze the jamming properties of the Kagomé lattice under periodic or hard-wall boundary conditions. Fig. 4 shows a periodic unjamming motion for the Kagomé lattice. Notice though that the jamming properties one finds depend on how many neighboring unit cells $\mathbf{N}_c$ are used as the "base" region (i.e., the generating packing), and therefore, we will usually specify this number explicitly. Some properties may be independent of $\mathbf{N}_c$ (for example, the triangular lattice packing is strictly jammed for all $\mathbf{N}_c$) and tailored mathematical analysis can be used to show this [16,26]. More systematic approaches based on Bloch wave (Fourier) decompositions of the set of feasible motions are being investigated for repetitive packings. We will not consider these issues in detail here, but rather focus on algorithmic approaches tailored for *finite* and *fixed* systems (i.e., $\mathbf{N}_c$ is fixed and finite), which is important when studying disordered particle packings, i.e., packings where the generator $P(\hat{\mathbf{R}})$ is itself a large disordered packing.

3. Linear programming algorithm to test for jamming

Given a sphere packing, we would often like to test whether it is jammed according to each of the categories given above, and if it is not, find one or several unjamming motions $\Delta\mathbf{R}(t)$. We now describe a simple algorithm to do this that is exact for *gapless* (*ideal*) *packings*, i.e., packings where neighboring spheres touch exactly, and for which the definitions given earlier apply directly. However, in practice, we would also like to be able to study *packings with small gaps*, such as produced by various heuristic compression schemes like the Lubachevsky–Stillinger algorithm [24], and we will consider these along with ideal packings. In this case the meaning of unjamming needs to be modified so as to fit physical intuition. We do this using what Roux [23] calls the *approximation of small displacements* (ASD), and propose an algorithm based on linear programming that can test whether a finite packing is jammed.

[9] This closely resembles the Born–von Karman boundary conditions used in solid-state physics models of lattice vibrations.

We believe that computer-generated packings which are almost ideal are often actually very close in configurational space to an ideal packing. This cannot be verified exactly in most cases, though some support for this claim can be obtained by setting the numerical precision of the generation algorithm to higher and higher values (for example, by increasing the number of collisions per particle in the Lubachevsky–Stillinger algorithm [24]) and verifying that the interparticle gaps monotonically decrease toward zero. It is possible though that gaps are natural and *essential* in certain applications, such as for example, the study of particle rearrangement in granular materials, and we therefore separately study packings which need not be (close to) ideal, using mathematical programming as the fundamental tool. When the configuration is known exactly, often the case for small ordered packings, jamming may be analyzed analytically.

3.1. Approximation of small displacements

As already explained, an unjamming motion for a sphere packing can be obtained by giving the spheres suitable velocities, such that neighboring spheres do not approach each other. Here, we focus on the case when $\Delta\mathbf{R}(t) = \mathbf{V}t + \mathcal{O}(t^2)$ are small finite displacements from the current configuration. We will drop the time designation and just use $\Delta\mathbf{R}$ for the displacements from the current configuration $\mathbf{R}$ to the new configuration $\tilde{\mathbf{R}} = \mathbf{R} + \Delta\mathbf{R}$. We defer discussion of packings with significant interparticle gaps to Section 5.

In this ASD approximation, we can linearize the impenetrability constraints

$$\left\|\tilde{\mathbf{r}}_i - \tilde{\mathbf{r}}_j\right\| = \left\|(\mathbf{r}_i - \mathbf{r}_j) + (\Delta\mathbf{r}_i - \Delta\mathbf{r}_j)\right\| \geqslant D \tag{5}$$

by expanding to first order in $\Delta\mathbf{R}$, to get the condition for the existence of a (first-order) *feasible displacement* $\Delta\mathbf{R}$,

$$(\Delta\mathbf{r}_i - \Delta\mathbf{r}_j)^{\mathrm{T}}\mathbf{u}_{i,j} \leqslant \Delta l_{i,j} \quad \text{for all } \{i,j\}, \tag{6}$$

where $\{i,j\}$ represents a *potential contact* between nearby spheres i and j, and

$$\Delta l_{i,j} = \left\|\mathbf{r}_i - \mathbf{r}_j\right\| - D$$

is the *interparticle gap* (or *interstice*). The set of contacts $\{i,j\}$ that we include in (6) form the *contact network* of the packing and they correspond to a subclass of the class of fascinating objects called *tensegrity frameworks*, namely *strut frameworks* (see [17] for details and also [2] for a treatment of more general packings). We only consider potential contacts $\{i,j\}$ between nearby, and not all pairs of spheres, that is we only consider a contact if

$$\left\|\mathbf{r}_i - \mathbf{r}_j\right\| \leqslant (1+\delta)D, \tag{7}$$

where $\delta \ll 1$ is a chosen *gap tolerance*.

For a gapless packing, we have $\Delta l = 0$ and the condition (6) reduces to (1), and the packing is jammed if and only if the only non-trivial solution to (6) is $\Delta\mathbf{R} = 0$. For packings with finite but small gaps though, condition (6) is only a first-order approximation. By transforming Eq. (5), we obtain the nonlinear analog of Eq. (6):

$$(\Delta\mathbf{r}_i - \Delta\mathbf{r}_j)^{\mathrm{T}}\mathbf{u}_{i,j} - \frac{\left\|\Delta\mathbf{r}_i - \Delta\mathbf{r}_j\right\|^2}{2\left\|\mathbf{r}_i - \mathbf{r}_j\right\|} \leqslant \Delta l_{i,j}\left(1 - \frac{\Delta l_{i,j}}{2\left\|\mathbf{r}_i - \mathbf{r}_j\right\|}\right) \quad \text{for all } \{i,j\}. \tag{8}$$

Notice that any displacement feasible under the linearized constraints (6) will also be feasible under the full nonlinear impenetrability constraints (8). In other words, within the ASD, $\mathscr{J}_{\mathbf{R}}$ is approximated with the *inscribed* polyhedral set $\mathscr{P}_{\Delta\mathbf{R}} \subseteq \mathscr{J}_{\mathbf{R}}$ of feasible (linearized) displacements, as determined by the system of linear inequalities (6), as illustrated in Fig. 1. This is a very special and most useful property of sphere packings which does not generalize to other convex particle shapes.

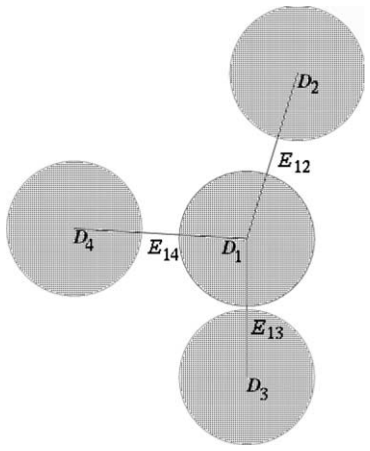

Fig. 5. The packing from Fig. 1 shown again with a numbering of the disks. D_i denotes particle i and E_{ij} denotes the contact between the ith and jth particles, i.e., the contact $\{i,j\}$.

Also note that for any non-trivial (i.e., $\Delta\mathbf{r}_i \neq \Delta\mathbf{r}_j$) solution to (6) the nonlinear inequality (8) is satisfied as a *strict* inequality, which means that particles i and j lose contact, even if the inequality is active to first order. This is an important property which shows that by scaling a first-order feasible displacement $\Delta\mathbf{R}$ appropriately one can always obtain a non-trivial feasible displacement which separates some of the contacting particles. This property does not directly generalize to other smooth strictly convex particle shapes, and in particular, it does not apply to packings of ellipsoids.

Comparison of (6) and (8) also suggests that the linearized constraints become too strict as the magnitude of the displacements becomes comparable to the size of the particles $\|\Delta\mathbf{r}_i - \Delta\mathbf{r}_j\| \approx D$. The complicated issue of how well the ASD approximation works when the gaps are not small enough is illustrated in Fig. 1.

By putting the $\mathbf{u}_{i,j}$'s as columns in a matrix of dimension $[Nd \times N_e]$, where N_e is the number of contacts in the contact network, we get the important *rigidity matrix* [10] of the packing $\mathbf{A}$. This matrix is sparse and has two blocks of d non-zero entries in the column corresponding to the particle contact $\{i,j\}$, namely, $\mathbf{u}_{i,j}$ in the block row corresponding to particle i and $-\mathbf{u}_{i,j}$ in the block row corresponding to particle j. Represented schematically:

$$\mathbf{A} = \begin{array}{c c} & \{i,j\} \\ & \downarrow \\ i \rightarrow & \\ j \rightarrow & \end{array} \hspace{-2em} \begin{bmatrix} \vdots \\ \mathbf{u}_{i,j} \\ \vdots \\ -\mathbf{u}_{i,j} \\ \vdots \end{bmatrix}.$$

For example, for the four-disk packing shown in Fig. 1, and with the numbering of the disks depicted in Fig. 5, we have the following rigidity matrix:

[10] This is in fact the normalized negative transpose of what is usually taken to be the rigidity matrix, and is chosen to fit the notation in [21], and also because it resembles the *node-arc incidence matrix* of the (directed) graph corresponding to the contact network.

$$\mathbf{A} = \begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \end{array} \begin{array}{c} \begin{array}{ccc} E_{12} & E_{13} & E_{14} \end{array} \\ \begin{bmatrix} \mathbf{u}_{12} & \mathbf{u}_{13} & \mathbf{u}_{14} \\ -\mathbf{u}_{12} & & \\ & -\mathbf{u}_{13} & \\ & & -\mathbf{u}_{14} \end{bmatrix} \end{array}.$$

Using this matrix, we can rewrite the linearized impenetrability constraints as a simple system of linear inequality constraints:

$$\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant \Delta\mathbf{l}. \tag{9}$$

3.1.1. Boundary conditions

Handling different boundary conditions within the above formulation is easy. For example, for usual periodic conditions, one adds a few columns to the rigidity matrix **A** with

$$\mathbf{u}_{i,\hat{j}(\mathbf{n}_c)} = \frac{\mathbf{r}_{\hat{j}(\mathbf{n}_c)} - \mathbf{r}_i}{\left\|\mathbf{r}_{\hat{j}(\mathbf{n}_c)} - \mathbf{r}_i\right\|}$$

for all images $\hat{j}(\mathbf{n}_c)$ which have contacts with one of the original spheres i. These columns correspond to the periodic contacts wrapping the packing around the torus.

For hard-wall boundaries, we add a potential contact to the contact network from each sphere close to a wall to the closest point on the wall and fix the endpoint on the wall. Such fixed points of contact and fixed spheres j, called *fixed nodes* in tensegrity terminology, are simply handled by transferring the corresponding term $\Delta\mathbf{r}_j^{\mathrm{T}}\mathbf{u}_{i,j}$ to the right-hand side of the constraints in (9).

3.2. Randomized linear programming algorithm

The question of whether an ideal packing is jammed, i.e., whether the system (9) is feasible for some $\Delta\mathbf{R} \neq 0$, can be answered rigorously by using standard LP techniques, as described in Appendix A. If a packing is jammed, then this LP test is enough. However, for packings that are not jammed, it is more useful to obtain a *representative* collection of unjamming motions, rather then use a binary classification into packings which are jammed and ones which are not jammed. A random collection of such unjamming motions is most interesting and can be obtained easily by solving several linear programs with a random cost vector.

We adapt such a *randomized LP algorithm* to testing for jamming, namely, we solve the following LP in the *displacement formulation*:

$$\begin{aligned} &\max_{\Delta\mathbf{R}} \mathbf{B}^{\mathrm{T}}\Delta\mathbf{R} \quad \text{for virtual work} \\ &\text{such that} \quad \mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant \Delta\mathbf{l} \quad \text{for impenetrability} \\ &|\Delta\mathbf{R}| \leqslant \Delta R_{\max} \quad \text{for boundedness,} \end{aligned} \tag{10}$$

for a *random load* **B**, where $\Delta R_{\max} \gg D$ is used to prevent unbounded solutions and thus improve numerical behavior. [11] The physical interpretation of **B** as an external load was elucidated in Section 2.4. Trivial solutions, such as uniform translations of the packing $\Delta\mathbf{R} = \text{const.}$ for periodic boundary conditions, can be eliminated a posteriori, for example by reducing $\Delta\mathbf{R}$ to zero mean displacement. Alternatively, trivial

[11] In our tests we usually set $\Delta R_{\max} \sim 100D$.

motions can be handled by introducing extra constraints in (10), for example, by fixing the position of one of the spheres, though we have found this less attractive, particularly for packings with gaps. Finally, trivial components of $\Delta\mathbf{R}$ can also be avoided by carefully choosing $\mathbf{B}$ to be in the null-space of $\mathbf{A}$, which usually means it needs to have zero total sum and total torque (see [21, Chapter 15]). We will discuss numerical techniques to solve (10) in Section 6.

The reason we have included possibly non-zero gaps $\Delta\mathbf{l}$ in (10) is that computer-generated packings, which we analyze, are never ideal and there are always small interparticle gaps between some particles, [12] typically much less than a percent of the typical particle size D. One can safely consider such packings purely within the ASD. However, we need to modify our definition of jamming to allow for very small particle rearrangements at the application of the load $\mathbf{B}$, i.e., we consider a solution to (10) an unjamming motion only if some particle is displaced a significant distance:

$$\exists i \text{ such that } \|\Delta\mathbf{r}_i\| \geqslant \Delta r_{\text{large}} \gg \overline{\Delta l},$$

where $\overline{\Delta l} \ll D$ is the typical size of the interparticle gap. Even though any solution to $\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant 0$ is also a solution to $\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant \Delta\mathbf{l}$, the latter may have other solutions with large components, corresponding to elongated corners of the polyhedron of feasible displacements $\mathscr{P}_{\Delta\mathbf{R}}$ (see Fig. 1), which should also be treated as unjamming motions. Therefore, the primary purpose of including the exact interparticle gaps in (10) is to ensure proper handling of degenerate cases, such as a near-180° angle between two contacts in 2D (see Fig. 5).

In summary, we treat any solution $\Delta\mathbf{R}$ to (10) with components significantly larger than $\overline{\Delta l}$ as an unjamming motion. For each $\mathbf{B}$, if we fail to find an unjamming motion, we apply $-\mathbf{B}$ as a loading also, for reasons detailed in Appendix B. We stress that despite its randomized character, this algorithm is almost rigorous when used as a test of jamming, in the sense that it is *strictly rigorous* for gapless packings, and also likely to work well if the interparticle gaps are sufficiently small, as explained in more detail in Appendix B. We will discuss more complicated adaptations of the randomized LP algorithm to non-ideal packings, i.e., packings with larger gaps, in Section 5.

3.3. Kinematic/static duality

The subject of kinematic/static duality and its physical meaning and implications have been discussed in numerous previous works [7,13,15,17,23]. The dual of the displacement formulation LP (10) (excluding the additional practical safeguard constraint $\Delta\mathbf{R} \leqslant \Delta\mathbf{R}_{\max}$), the *force formulation* LP,

$$\begin{aligned}
&\max_{\mathbf{f}} (\Delta\mathbf{l})^{\mathrm{T}}\mathbf{f} \quad \text{for virtual work}\\
&\text{such that } \mathbf{A}\mathbf{f} = \mathbf{B} \quad \text{for equilibrium,}\\
&\mathbf{f} \leqslant 0 \quad \text{for repulsion only,}
\end{aligned} \tag{11}$$

gives the *interparticle repulsive* [13] *force* $f_{i,j}$ between spheres i and j as the dual variable associated with the impenetrability constraint (6). The displacement- and force-based LPs are of great importance in studying the *stress–strain* behavior of granular materials, and since they are equivalent to each other, we can call them the *ASD stress–strain LP*. We have emphasized the displacement formulation (10) simply because we based our discussion of jamming on a kinematic perspective, but a parallel static interpretation can easily be given. For example, a random $\mathbf{B}$ used in the randomized LP algorithm that finds an unbounded unjamming motion physically corresponds to a load that the packing cannot

[12] These gaps may be an inherent and essential feature of disordered packings in general.
[13] We choose a negative sign for repulsive forces here in agreement with mathematical literature [17].

support, i.e., the force formulation (dual) LP is infeasible, implying that the displacement formulation (primal) LP is unbounded.

In general the stress–strain LP will be highly degenerate and its primal and/or dual solution not unique. However, as Roux points out [23], the existence of small gaps in random packings is very important in this context. Namely, if $\Delta\mathbf{l}$ is random and non-zero (even if small), and $\mathbf{B}$ is also random, both the primal and dual solutions will likely be non-degenerate (see [21]), and we have indeed observed this in practice for random packings. A non-degenerate (basic or vertex) solution to (11) corresponds to an isostatic force-carrying contact network [7,23]. We use interior-point linear programming algorithms, which are not sensitive to degeneracies, but also do not necessarily identify a vertex solution to the LP if the solution is not unique.

4. Testing for jamming with periodic boundary conditions

In this section we give more details on using the randomized linear programming approach to test for local, collective and strict jamming in ideal packings with periodic boundary conditions. An outline of the actual computational algorithm along with representative results is given in Section 6.1.

4.1. Local jamming

Recall that the condition for a packing to be locally jammed is that *each particle be fixed by its neighbors*. This is easy to check. Namely, each sphere has to have at least $d + 1$ contacts with neighboring spheres, not all in the same d-dimensional hemisphere. This can be tested in any dimension by solving a small linear program, and in two and three dimensions one can use more elementary geometric constructions.

We prefer the LP approach because it is in the spirit of this work and because of its dimensional independence, and so we present it here. Take a given sphere i and its set of contacts $\{\mathbf{u}_{i,*}\}$, and put these as rows in a matrix $\mathbf{A}_i^{\mathrm{T}}$. Then, solve the local portion of (10) (using the simplex algorithm):

$$\begin{aligned} &\min_{\Delta\mathbf{r}_i} (\mathbf{A}_i\mathbf{e})^{\mathrm{T}}\Delta\mathbf{r}_i \\ &\text{such that } \mathbf{A}_i^{\mathrm{T}}\Delta\mathbf{r}_i \leqslant \Delta\mathbf{l}_{i,*}, \end{aligned} \tag{12}$$

which will have an unbounded solution if the sphere i is not locally jammed, i.e., if it is a rattler, as illustrated in Fig. 1. Here, e is a vector of ones (see Appendix A.1). The local load $\mathbf{b}_i = \mathbf{A}_i\mathbf{e}$ can be replaced with two random loads of opposite direction, which is more suitable when larger gaps are present. When testing for jamming in ideal packings, we remove the rattlers from the packing before proceeding with tests for collective or strict jamming. Notice that checking each sphere for local jamming using (12) only once is not enough under this removal scheme. Namely, once a rattling sphere is removed, this removes some contacts from the packing and can make other spheres not locally jammed. We have observed that sometimes, particularly in two-dimensional systems, *all* disks can be removed on the basis of just the local jamming testing.

Of course we can define higher orders of local jamming by asking that *each set of n spheres be fixed by its neighbors*, called *n-stability* in [16]. However, for $n > 1$ it becomes combinatorially too difficult to check for this because the number of subsets to be tested grows exponentially. Computationally, we have found testing for local jamming using (12) to be quite efficient and simple.

4.2. Collective jamming

The randomized LP algorithm was designed to test for collective jamming in large packings, and in this case the linear program (10) that needs to be solved is very large and sparse. Notice that boundary conditions are only involved when making the list of contacts in the contact network and deciding if certain spheres or contact points are fixed. In the case of periodic boundary conditions, we simply add the usual contacts between original spheres near the boundary of the unit cell and any nearby periodic image spheres and also fix $\Delta\mathbf{\Lambda} = 0$ in Eq. (4).

4.3. Strict jamming

To extend the notion of collective jamming to strict jamming we introduced deformations of the boundary. In the case of periodic packings, *the lattice* $\mathbf{\Lambda}$ *plays the role of the boundary*. Therefore, the only difference with collective jamming is that we will now allow the lattice to change while the spheres move, i.e., $\Delta\mathbf{\Lambda} \neq 0$ in (4). The *lattice deformation* $\Delta\mathbf{\Lambda}$ will become part of the unknowns in (10), but since it too enters linearly in (4), we still get a linear program, only with coefficient matrix $\mathbf{A}$ augmented with new (denser) rows. These rows have non-zero entries in the columns corresponding to contacts across the periodic boundary, and for brevity we do not give details here but refer the interested reader to [27]. The actual implementation of the algorithm for strict jamming requires more care and bookkeeping, but the conceptual changes should be clear, and the randomized LP algorithm remains applicable.

Obviously, we cannot allow the volume of the unit cell to enlarge, since the unit cell is in a sense the container holding the packing together. Therefore, we only consider *volume-non-increasing* continuous lattice deformations $\Delta\mathbf{\Lambda}(t)$:

$$\det\left[\tilde{\mathbf{\Lambda}} = \mathbf{\Lambda} + \Delta\mathbf{\Lambda}(t)\right] \leqslant \det\mathbf{\Lambda} \quad \text{for } t > 0. \tag{13}$$

We now think of $[\Delta\mathbf{R}(t), \Delta\mathbf{\Lambda}(t)]$ as an unjamming motion and focus on linear motions $\Delta\mathbf{\Lambda}(t) = \mathbf{W}t$, $\mathbf{W} = \text{const.}$ and the final small deformations $\Delta\mathbf{\Lambda} = \Delta\mathbf{\Lambda}(1)$, and consider first-order linearizations of the non-expansion nonlinear constraint (13).

The linearized version of (13) is

$$\mathrm{Tr}[(\Delta\mathbf{\Lambda})\mathbf{\Lambda}^{-1}] \leqslant 0, \tag{14}$$

and this is just one extra linear constraint to be added to the linear program (10). An extra condition which needs to be added is that $(\Delta\mathbf{\Lambda})\mathbf{\Lambda}^{-1}$ be symmetric, which is also an added linear constraint,

$$(\Delta\mathbf{\Lambda})\mathbf{\Lambda}^{-1} = \boldsymbol{\varepsilon} \quad \text{and} \quad \boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^{\mathrm{T}}, \tag{15}$$

where we add the *strain tensor* $\boldsymbol{\varepsilon}$ as an unknown in the randomized LP algorithm. Even better, one can eliminate $\Delta\mathbf{\Lambda} = \boldsymbol{\varepsilon}\mathbf{\Lambda}$ and use the strain as the only added variable. Torquato et al. [18] and Donev and Torquato [27] discuss the interpretation of $(\Delta\mathbf{\Lambda})\mathbf{\Lambda}^{-1}$ as a macroscopic (global) strain tensor. Note that condition (15) does nothing more than eliminate trivial rotations of the lattice, which correspond to skew-symmetric strain tensors, so that uniform translations remain the only trivial unjamming motion. Appendix C proves that adding lattice deformations does not change the mathematical theory presented in Section 2.3.

The motivation for the category of strict jamming and its above interpretation in the periodic case should be clear: Changing the lattice in a volume non-increasing way models *macroscopic non-expansive strain* (i.e., a compressive macroscopic stress) and is therefore of great relevance to studying the macroscopic mechanical properties of random packings (see [18]). We also again point out that strict jamming is (significantly) stronger than collective jamming for periodic boundary conditions, particularly in two-dimensional packings. This point is illustrated in Fig. 6, which shows an unjamming motion involving a deformation of the lattice, even though this lattice packing is collectively jammed. Periodic boundary conditions are often used to model infinite systems, in the hope that a jammed periodic packing will produce a “jammed” infinite packing (for example, in the sense of uniform stability [16]) when periodically replicated in all directions. A simple counting argument demonstrates that isostatic collectively jammed

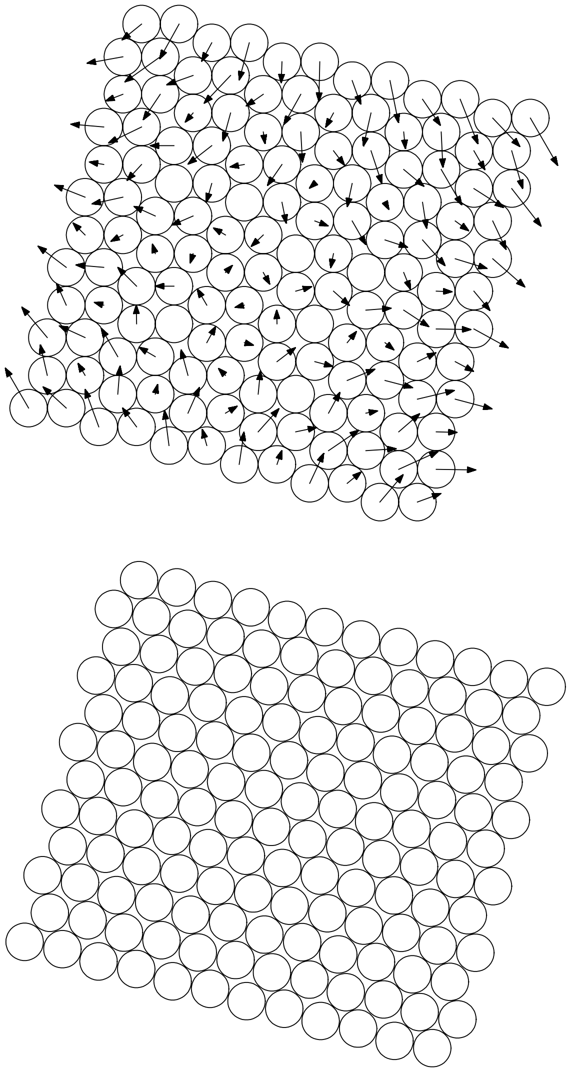

Fig. 6. Example of a lattice deformation. The above periodic packing (packing 3 in [26]) is collectively jammed, but not strictly jammed. It can be continuously sheared toward the triangular lattice by deforming the lattice in a volume non-decreasing manner, as shown here.

periodic packings *cannot* generate "jammed" infinite packings because they have too few contacts, but isostatic strictly jammed periodic packings *can* since they have enough contacts (due to the inclusion of additional degrees of freedom for the deforming lattice). We omit details of this counting argument for the sake of brevity.

5. Dealing with interparticle gaps

We originally motivated our perspective on jamming in Section 2.1 by looking at the set of available (reachable) configurations $\mathcal{J}_{\mathbf{R}}$ around a particular initial configuration $\mathbf{R}$, and have since focused mostly on ideal packings, though allowing for sufficiently small interparticle gaps. For these packings, $\mathscr{J}_{\mathbf{R}}$ is very localized around $\mathbf{R}$, and this makes it possible to define the three jamming categories meaningfully and rigorously, and also allows for a simple randomized linear programming testing algorithm. However, the either–or character of such a jamming criterion is often too restrictive or specialized when analyzing large disordered packings with possibly larger interparticle gaps, where particle displacements may be comparable to the typical particle size. Therefore, we investigate ways to study jamming in this practical sense.

One can make a rigorous definition of jamming even in the case when particle displacements are large. However, it is not clear that this has a physical significance. A somewhat ambitious but desirable goal is to efficiently obtain a grasp on the character and extent of $\mathscr{J}_{\mathbf{R}}$ and use this to judge whether the packing should be considered jammed or not. However, since $\mathscr{J}_{\mathbf{R}}$ is a very high-dimensional and non-convex set, it is a very complex object to describe or understand. We focus here on trying to judge the *extent* of $\mathscr{J}_{\mathbf{R}}$ by trying to displace the spheres away from their current position by as much as possible. This can be done with a *sequential random loading* algorithm: Repeatedly solve the LP (10), displace the spheres in the direction of $\Delta\mathbf{R}$ by as much as possible while still avoiding overlap, until the particles rearrange and form contacts that actually support the applied load $\mathbf{B}$. This should be repeated for several random loads, in the hope of exploring $\mathscr{J}_{\mathbf{R}}$ along several directions. We give an outline of an algorithm to do this along with representative results in Section 6.2. The important point here is that for packings which are almost jammed, mathematical programming is needed in order to efficiently find a direction in which the particles can be displaced by significant amounts. Traditional heuristics such as Monte Carlo schemes in which particles are displaced one-by-one simply get trapped easily, and algorithms which search for collective particle rearrangements are needed.

5.1. Shrink-and-bump heuristic

The following heuristic test for collective jamming has been suggested in [24]: Shrink the particles by a small amount γ and then start the Lubachevsky–Stillinger molecular dynamics algorithm with random velocities, and see if the system gets unjammed. One would also slowly enlarge the particles back to their original size while they bump around, so as to allow finite termination of this test (within numerical accuracies). We call this the *shrink-and-bump heuristic*. The idea is that the vector of velocities takes on random values in velocity space and if there is a direction of unjamming, it will be found with a high probability and the system will unjam. Animations of this process can be found in [22].

This kind of heuristic has the advantage of being very simple and thus easy to implement and use (and also incorporates nonlinear effects), and it is also very efficient, though still significantly slower than the linear programming algorithm since typically many collisions per particle are needed to significantly displace the particles due to the high density. By incorporating deformations of the lattice in the Lubachevsky–Stillinger algorithm, one can also use this to test for strict jamming, as discussed further in [14]. Its disadvantages are its non-rigorous character and indeterminacy, artificial introduction of dynamics into a geometrical problem, and most of all, its strong dependence on the exact value of γ. For example, animations showing how the Kagomé lattice inside a container made of fixed spheres (as in Fig. 3) can be unjammed with a large-enough γ, even though it is actually collectively jammed under these boundary conditions, can be found at our webpage [22]. In fact, many jammed *large* packings will appear unstable under this kind of test, as motivated with the notion of *uniform stability*, defined in [16].

6. Algorithmic details

In this section we outline in detail two algorithms to test for jamming in hard-sphere packings. The first one is applicable to ideal packings, while the second one deals with non-ideal packings. Although the core concept used in both is the randomized linear programming algorithms presented in Sections 3 and 4, the two differ in their goals and the way they process the results of the linear programming step: The first one attempts to give a binary classification of packings into jammed and not jammed, while the second tries to explore the extent of $\mathcal{J}_{\mathbf{R}}$ by trying to continuously displace the particles as much as possible, as discussed in Section 5.

6.1. Algorithm: ideal packings

We summarize the proposed algorithm to test for collective or strict jamming in ideal packings, applicable also to packings with very small interparticle gaps (Algorithm 1). This algorithm removes spheres which are not locally jammed. Once a rattling sphere is removed, this removes some contacts from the packing, which can make other spheres not locally jammed. Therefore, an implementation in which neighbors of rattlers are recycled on a stack of spheres to be checked is needed. This algorithm also classifies packings which have an ideal jammed subpacking as jammed, even if they have some rattling particles or rattling clusters of particles, as illustrated in Fig. 7 for a disordered binary disk packing.

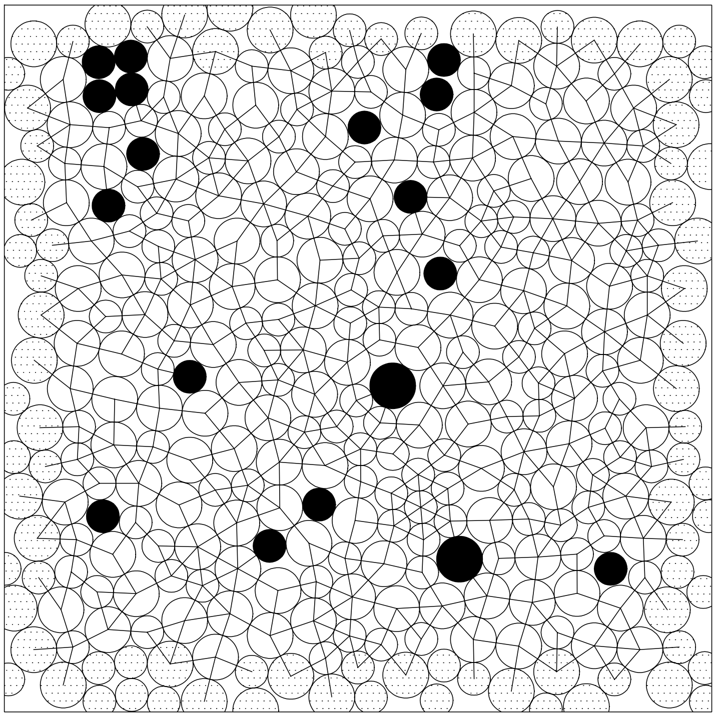

Fig. 7. Results from the algorithm of Section 6.1 (Algorithm 1). The algorithm to test for *collective* jamming in ideal packings was applied to this equimolar bidisperse disk packing of 250 disks ($\varphi = 0.846$) in order to identify a jammed subpacking (if any). A gap tolerance of $\delta = 0.01$ was used, and all disks that displaced by more than $\Delta r_{\text{large}} = 10^{-3}$ were removed (colored black) to leave a jammed subpacking of 232 disks, for which the average displacement during the test was $\overline{\|\Delta\mathbf{r}_i\|}/D \approx 7 \times 10^{-7}$ and the maximal was $\max_i \|\Delta\mathbf{r}_i\| \approx 2 \times 10^{-5}$, indicating a high numerical accuracy in the packing algorithm (about 20,000 collisions per particle were used). If the rattling particles or rattling clusters of particles were not removed, the displacements observed would have been higher, as, for example, in Table 1. On the other hand, if overly strict tolerances were chosen in the algorithm of Section 6.1 (for example, $\Delta r_{\text{large}} = 10^{-4}$), then no jammed subpacking would have been found. With reasonably tight tolerances, *there is no strictly jammed subpacking* of this packing. Note that it may be possible to remove some of the disks from the collectively jammed subpacking and still maintain the jamming property. The dotted disks represent periodic images.

Algorithm 1 (*Randomized linear programming algorithm*).

```
1. If testing for collective jamming, fix the strain $\boldsymbol{\varepsilon} = 0$.
2. Choose a suitable gap tolerance $\delta$, $\delta D \sim \Delta r_{\text{large}}$, in Eq. (7) and add all potential contacts $\{i, j\}$ between
   neighboring particles to the contact network.
3. If there are no spheres in the packing, declare the packing as not jammed and terminate.
4. Test for local jamming (rattlers):
   (a) Make a stack of all the spheres.
   (b) Remove the top (pop) sphere $i$ from the stack and solve the LP (12) with $\mathbf{b}_i = \mathbf{A}_i\mathbf{e}$ using the simplex
       algorithm. If $\|\Delta\mathbf{r}_i\| \geqslant \Delta r_{\text{large}}$, remove the sphere from the packing, push all its neighbors not on the
       stack back on the stack and remove all its contacts from the contact network.
   (c) Go back to step 4a if the stack is not empty.
   (d) Repeat step 3.
5. Choose a random load $\mathbf{B}$.
6. Solve the LP (10) along with constraint (14) (if testing for strict jamming).
7. Remove all spheres $i$ displaced by the load from the packing, $\|\Delta\mathbf{r}_i\| \geqslant \Delta r_{\text{large}}$.
8. Repeat steps 3–4, reverse the direction of $\mathbf{B}$, $\mathbf{B} \leftarrow -\mathbf{B}$ and repeat steps 6–7.
9. If no spheres were displaced by either load, declare the (sub)packing jammed and terminate. Otherwise
   go back to step 3.
```

6.2. Algorithm: non-ideal packings

When dealing with non-ideal packings, one has to abandon the strict "jammed" versus "not jammed" binary classification. Instead we focus on trying to judge the *extent* of $\mathscr{J}_{\mathbf{R}}$ by trying to displace the spheres away from their current position by as much as possible. We first give the algorithm to do this in Algorithm 2 and then we discuss specific steps and the choices one can make in each step. Some illustrative results are given in Section 7.2.

Algorithm 2 (*Sequential random loading algorithm*).

```
 1. If testing for collective jamming, fix the strain $\boldsymbol{\varepsilon} = 0$.
 2. Choose a suitable gap tolerance $\delta$, $\delta D \sim \Delta r_{\text{large}}$, in Eq. (7) and add all potential contacts $\{i, j\}$ between
    neighboring particles to the contact network.
 3. Test for rattlers:
    For all spheres $i$, solve the LP (12) using the simplex algorithm and two randomly chosen loads $\mathbf{b}_i$ of
    opposite direction. If $\|\Delta\mathbf{r}_i\| \geqslant \Delta r_{\text{large}}$, mark the sphere as a rattler.
 4. Choose a random load $\mathbf{B}$ and set $\mathbf{b}_i = 0$ for rattling particles.
 5. Solve the LP (10) along with constraint (14) to obtain a linearized unjamming motion $\Delta\mathbf{R}$.
 6. Find the largest scaling $\tau > 0$ for the displacements so that no spheres overlap for displacements from 0
    to $\tau\Delta\mathbf{R}$ and also require that the volume of the unit cell does not increase, $\det[\mathbf{I} + \tau\boldsymbol{\varepsilon}] \leqslant 1$. Displace the
    spheres to a new configuration, $\mathbf{R} \leftarrow \mathbf{R} + \tau\Delta\mathbf{R}$, $\boldsymbol{\Lambda} \leftarrow (\mathbf{I} + \tau\boldsymbol{\varepsilon})\boldsymbol{\Lambda}$. Note that this changes the rigidity matrix
    $\mathbf{A}$ of the packing and requires updating the contact network.
 7. If any particle was displaced by a significant amount, $\tau\|\Delta\mathbf{r}_i\| > \beta D$, go back to step 5. Also keep statis-
    tics of $\tau\|\Delta\mathbf{r}_i\|$ over all spheres, such as average $\overline{\|\Delta\mathbf{r}_i\|}$ and maximum value $\max\|\Delta\mathbf{r}_i\|$.
 8. Optionally repeat step 3 and set $\mathbf{b}_i = 0$ for (new) rattling particles.
 9. Reverse the direction of $\mathbf{B}$ and repeat steps 5–7.
10. If the average or maximal particle displacement exceed thresholds, declare packing as not "jammed"
    and terminate. Otherwise go back to step 4 until convinced packing is "jammed".
```

We discuss the different steps of this algorithm separately in the following sections. Note that the proposed algorithm is not as efficient as possible, mostly because not all linear programs need to be solved to full accuracy. Linear optimizers, and in particular interior-point algorithms, spend most of their effort in the final stages of the optimization, looking for the exact optimal vertex (or face) of the feasible polytope. Therefore, early termination is most desirable, and in future work we will develop specialized implementations that will replace step 5 with several Newton steps of a feasible interior-point algorithm. In a sense, the above algorithm resembles a sequential linear programming (SLP) algorithm for finding equilibrium configurations of packings under applied loads. It remains to be explored whether including information about the curvature of the nonlinear impenetrability constraints, as is done in most modern nonlinear optimization algorithms, will be useful in light of the increased complexity of the linear algebra involved. For packings of non-spherical particles, such as packings of ellipsoids, including second-order information is necessary in order to find feasible directions of displacements. Numerous optimizations related to reuse of information in the iterative process and linear solvers as well as parallelization will also be investigated.

We stress that one cannot directly use off-the-shelf nonlinear optimization software to explore $\mathscr{J}_{\mathbf{R}}$, since feasibility must be strictly maintained throughout the process. Furthermore, efficiency also demands a specialized implementation. This is why we present in this work algorithms based on linear programming, which allows one to use any of the numerous linear programming libraries available today without the complexity of dealing with nonlinear programming algorithms.

6.2.1. Choosing the gap tolerance

First, we discuss the choice of the gap tolerance δ. The larger this tolerance, the more possible particle contacts we will add to the set of constraints, and thus the more computational effort we need. Furthermore, we are including more redundant and/or stricter-than-necessary linearized impenetrability constraints. Choosing a very small tolerance makes it hard to treat systems with moderately large interparticle gaps (say of the order of $\delta = 0.1D$), since crucial constraints which become relevant as soon as the magnitude of the displacements becomes comparable to δD are omitted. We have found values of $\delta \approx 0.1D$–$1.0D$ reasonable, depending on the dimensionality and type of packing. The general rule is that the contacts of each sphere with *all* spheres in (only) its first coordination shell should be included, and of course physical intuition and close examination of the results are very helpful.

6.2.2. Testing for rattlers

Unlike the case of ideal packings, where we permanently remove rattlers from the packing, here we simply avoid placing a load on the rattlers, but still consider them as part of the packing, as they may provide important constraints as the spheres displace. It is desirable not to place a load on rattlers because for some smaller gap tolerances δ, the contact network may not provide sufficient constraints to locally trap all particles. The particles that are not locally trapped will displace by very large distances under any non-zero load, leading to a very small scaling τ and very slow progress of the algorithm. Unfortunately, some linear programming solvers may return a large displacement for a rattler even if no load is applied on it, as most solvers initialize the variables independently of the user. In practice, step 3 of the algorithm only helps in cases when there is a small number of clear rattlers, as it enables one to use a smaller gap tolerance δ and thus reduce the size of the linear programs, and in such cases the first test for rattlers will already identify the troublesome particles. In other cases, one simply must use a sufficiently large δ.

6.2.3. Scaling the displacements

We emphasized in Section 3.1 that any solution of the linearized impenetrability constraints is also a solution to the full nonlinear impenetrability constraints. However, there are several reasons why it is important to choose an appropriate scaling for the displacements τ. First, we do not include all pairs of particles in the constraints, and therefore any $\Delta\mathbf{R}$ for which some particle displaces by more than δD is not necessarily a feasible displacement, and may need to be scaled down appropriately. Furthermore, the linearized constraints are significantly stricter than the nonlinear ones for larger displacements, and therefore it is often possible to scale up the displacements by a significant factor without violating feasibility.

Since our aim is to displace the particles as much as possible from their initial configuration, we choose the largest scaling factor possible. To find this scaling, one thinks of $\Delta\mathbf{R}$ as a vector of particle velocities and finds the time of the first interparticle collision τ. This can be found with exactly the same procedure as used to build collision schedules in the Lubachevsky–Stillinger packing algorithm [24]. For highest efficiency, the computational domain is partitioned into cells and only collisions between particles in neighboring cells are considered, along with transfers of the particles between the cells. The same partitioning is used when building the contact network of the packing after displacing the particles, though depending on the value of the gap tolerance δ more than just the neighboring cells might need to be searched. One should also ensure that the volume of the unit cell does not increase during the deformation of the lattice when testing for strict jamming.

6.2.4. *Termination criteria*

We do not give detailed criteria on when to terminate the iterated linear programming in step 7, since these should really be adapted to a nonlinear feasible interior-point algorithm to be used in place of the linear optimizer. Typically $\beta \approx 0.01 - 0.1$. When none of the particles can be displaced further despite repeating step 5, the dual variables obtained by the LP solver will become (close to) the true interparticle forces that resist the load. A primal–dual nonlinear solver would also terminate at such a point and return the appropriate Lagrange multipliers. However, outside the ASD these forces are no longer unique [23], nor is it guaranteed that a packing that can support a random load $\mathbf{B}$ and $-\mathbf{B}$ can support all loads. Therefore, we need to use several random loads. We do not have estimates or bounds on how many loads need to be used, however, experience has shown that only a few (3–5) loads are sufficient to find large displacements if such displacements exist.

6.2.5. *Interpreting the results*

Processing the results of the above algorithm is somewhat of an art. However, by observing the statistics of the multiple particle displacements, and especially by visualizing the path traversed by the particles during the loading, one can get a sense of the character of $\mathscr{J}_{\mathbf{R}}$. Particularly useful is observing the average magnitude of the particle displacements, and we use this metric in reporting some results for disordered computer-generated packings in Section 7.2. It may also be useful to observe the distribution of displacement magnitudes among the particles.

In general, it is best to first try the algorithm of this section, and then use a visualization tool (like our VRML animations) or a histogram of the magnitudes of the particle displacements to judge whether there appears to be a jammed subpacking (to within a tight tolerance), or whether all particles seem to be able to displace significantly. If the former is the case, then using the algorithm of Section 6.1 one can identify such a jammed subpacking if it exists within the tolerances used.

7. Implementation and results

We have developed a practical numerical implementation of the randomized LP algorithms using a primal–dual interior-point linear optimizer. We have applied the algorithms to test for the different jamming categories in practice and verified their utility and efficiency. Illustrations of results obtained using our implementations are given throughout this paper, and results from the application of the algorithm of Section 6.2 to large computer-generated mondisperse and bidisperse disk and sphere packings are given in

[14]. Here, we briefly discuss numerical aspects of the implementation in Section 7.1 and present some representative results and timing statistics in Section 7.2.

7.1. Numerical aspects: linear optimization

We have implemented an efficient numerical solution of (10) using the primal–dual interior-point algorithm in the LOQO optimization library [28]. Both Fortran 95 codes, which directly invoke the LOQO library, and algebraic modeling programming language (AMPL) models have been developed, along with VRML visualization tools. The AMPL models are particularly simple to use and modify, and are available on our website [22]. We wish to emphasize that by using primal–dual interior point algorithms we automatically get both forces and displacements using the same implementation. For example, both LOQO and PCx (see [28]) return both primal and dual solutions to the user. Illustrations of results obtained using these implementations are given throughout this paper. Primal–dual interior-point algorithms are very well suited for problems of this type. Nonetheless, for three-dimensional problems the available high-quality implementations of interior-point algorithms (such as [28]) are based on direct linear solvers are too memory-demanding and inefficient. Tuned implementations based on conjugate-gradient iterative solvers are needed.

7.2. Results

In this section we apply the algorithm of Section 6.2 to random disk and sphere packings generated via the Lubachevsky–Stillinger algorithm [24,29], although we have also used the ideal packing algorithm of Section 6.1, which typically gives a reasonable classification of the packings into jammed and not jammed. However, Algorithm 1 does not give a feeling for the character of $\mathscr{J}_{\mathbf{R}}$ for packings which are not jammed (within the framework of ideal packings), and therefore Algorithm 2 (with loose tolerances) is the preferred first choice when analyzing a certain type of packing for the first time. In particular, we have learned through experience that is easy to misclassify disordered monodisperse disk packings as collectively jammed by choosing inappropriate parameters in the simpler algorithm of Section 6.1.

As a quantitative measure of jamming in these packings, we report the average particle displacement $\overline{\|\Delta\mathbf{r}_i\|}$ achieved during random loading. Another statistic we report is the time (in seconds) spent by the AMPL implementation (with some Fortran) of the algorithm of Section 6.2 on a typical personal computer. [14] Since most of the computational time is spent in LOQO, similar running times are typical of the Fortran codes as well. For each packing, we applied three different random loads (with opposite orientations), and for each load we successively solved three linear programs (so a total of 18 linear programs for each packing). We are currently developing more efficient and robust implementations of these algorithms, for both packings of disks/spheres and ellipses/ellipsoids.

Testing for strict jamming typically takes more time, by as much as 25%, since additional denser rows/columns are included in the rigidity matrix, and this is more pronounced in three dimensions where more of the contacts are on the boundary. The exact way the strain and the associated constraints are handled makes a difference in this case. We emphasize that for three-dimensional packings the sparse factorization linear solver in LOQO is not the best choice, so much smaller running times are possible with specialized implementations. The running time of the linear solver depends non-trivially on both the number of spheres *and* the number of contacts in the contact network. The number of contacts is very sensitive to the choice of the gap tolerance δ, which we usually decreased as the packing size increased (and thus the average displacements decreased). Therefore, the running times below should not be taken as a measure of the scaling

[14] More precisely, calculations were performed on an 1666 MHz AMD Athlon PC running Linux with 1 GB of memory.

of the LP solver computational effort with the number of spheres, but rather as typical runtimes for some representative packing sizes.

In general, the random packings we tested *were* collectively jammed, in the sense that only small (average) displacements of the particles are possible. This is illustrated in Table 1. The small feasible displacements are mostly due to rattlers and/or early termination of the packing algorithm and we believe that any (final) Lubachevsky–Stillinger packing with infinite collision rate will in fact have an *ideal* collectively jammed subpacking. It is in fact very important to verify that any packing algorithm claimed to produce jammed packings can indeed produce jammed ideal packings given enough (possibly) infinite time. One way to test this is to verify that all tolerances in the test for jamming can be tightened progressively as the numerical accuracy is increased and the convergence criteria in the packing algorithm are tightened. We demonstrate this for collective jamming in monodisperse sphere packings for the Lubachevsky–Stillinger algorithm in Table 2.

These quantitative tabular results illustrate the feasibility and utility of the proposed algorithms. However, qualitative observations, which are best obtained from the numerous animations of the "unjamming" process, are indeed invaluable to getting a better physical intuition and understanding of hard-

Table 1
Results of Algorithm 2 for equimolar binary disk packings of diameter ratio 1.4 (from [14])

N	ϕ	t (s), collective	$\overline{\Vert\Delta\mathbf{r}_i\Vert}/D_i$, collective	$\overline{\Vert\Delta\mathbf{r}_i\Vert}/D_i$, strict
50	0.845	2.1	0.010	0.060
100	0.842	6.4	0.0034	0.011
250	0.846	21	0.0037	0.0053
500	0.847	72	0.0016	0.0067
750	0.849	88	0.0022	0.012
1000	0.849	130	0.0016	0.018
1500	0.848	247	0.0016	0.020
2500	0.849	248	0.0039	0.010

The first column shows the total number of particles N, the second the packing fraction, the third the running time for the AMPL model that tests for collective jamming and the last two columns show the average particle displacement during collective (i.e., with a fixed lattice) and strict jamming (i.e., with a deforming lattice) testing. Notice that the displacements are significantly larger for the strict jamming test, especially for small packings. The analogous table for three dimensions, given in [14], shows similar behavior but significantly larger computational times due to the inefficiency of the direct linear solver in LOQO for three-dimensional contact networks.

Table 2
The average particle displacement $\overline{\Vert\Delta\mathbf{r}_i\Vert}/D$ during the test for collective jamming is shown for a series of sphere packings produced by the (original) Lubachevsky–Stillinger algorithm (from [14])

$N/N_{\text{coll}}(10^3)$	1	5	10	25
50	0.041	0.015	0.0018	4.9×10^{-10}
100	0.036	0.016	0.0011	0.00014
250	0.050	0.023	0.0015	0.00036
500	0.047	0.024	0.0028	0.0014
750	0.046	0.019	0.0030	0.0011
1000	0.052	0.020	0.0025	0.00067

From top to bottom the packing size N increases, and from left to right the number of collisions per particle N_{coll} (in thousands) increases (and thus the density also slowly increases). No special handling of rattlers was employed. It is easily observed that the packings uniformly become "more jammed" as the packing algorithm is run longer (though rattlers may continue to give a finite contribution to the observed displacements). Similar behavior is expected of any algorithm which in the limit of infinite numerical precision produces packings with a collectively jammed subpacking. Note that the analogous table for strict jamming, given in [14], demonstrates that the packings do not become strictly jammed in the ideal sense even in the limit of infinite number of collisions.

particle packings. Such animations can be found on our webpage [22], and we would be happy to share many more with interested readers.

8. Conclusions

In this work and [14] we have proposed, implemented and tested a practical algorithm for verifying jamming categories in finite sphere packings based on linear programming. We demonstrated its simplicity and utility and presented some representative results for ordered lattices and random packings. Future extensions and applications of the proposed algorithms are awaiting exploration.

The jamming concepts and algorithms presented here can be extended to packings of non-spherical particles with certain non-trivial modifications, however, mathematical developments in this area are lacking. We are investigating such extensions and will report our findings in future work. Other important tasks include extending various packing generation algorithms to generate strictly jammed packings, as well as designing algorithms with guarantees of producing jammed packings. The algorithms to test for jamming, and more generally to explore the set of reachable configurations $\mathcal{J}_{\mathbf{R}}$ for hard-particle packings can

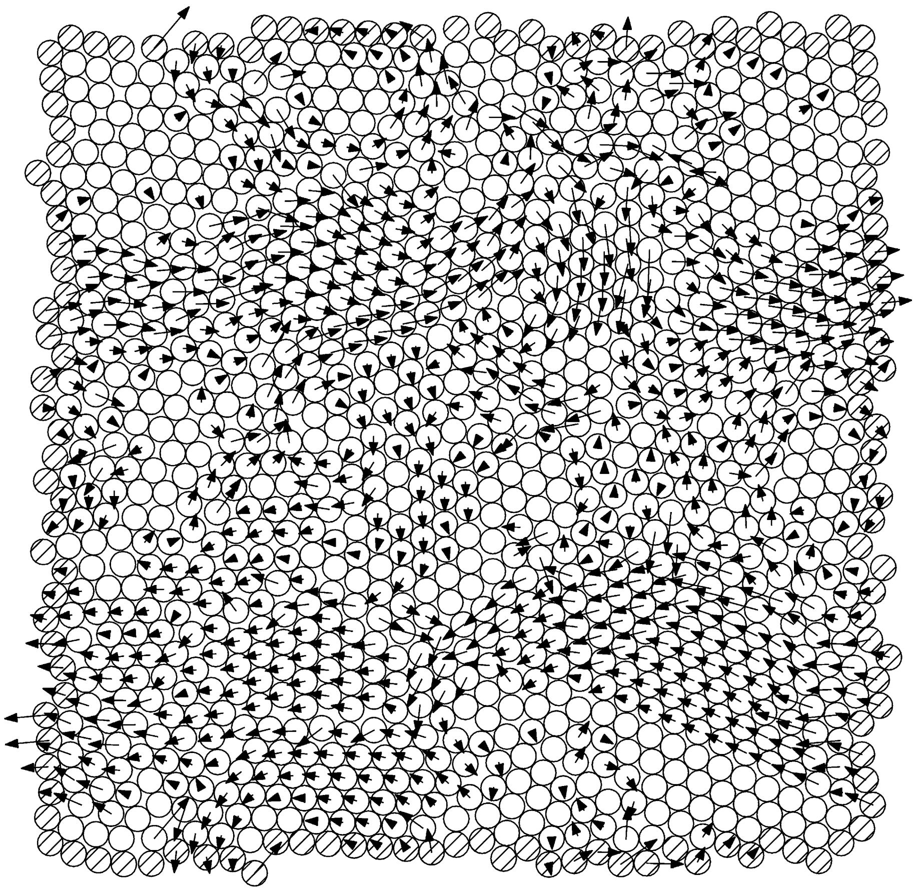

Fig. 8. Locally jammed disk packing. A random packing ($\varphi = 0.82$) of 1000 disks that is *not collectively jammed* and a representative periodic unjamming motion. More insightful animations can be found at the webpage [22].

be further improved. In particular, a carefully tuned implementation of linear solvers for three-dimensional packings is needed as a building block in implementations of various nonlinear programming algorithms related to packings. Work is already under way to provide highly efficient implementations of various optimization algorithms for linear and nonlinear programming on large-scale (contact) networks (Fig. 8).

Acknowledgements

The authors thank Robert Vanderbei for numerous helpful discussions on mathematical programming issues and for providing us with the LOQO optimization library. A.D. and S.T. were supported by the Petroleum Research Fund as administered by the American Chemical Society and by the MRSEC Grant at Princeton University, NSF DMR-0213706. R.C. was partially supported by NSF Grant No. DMS-0209595.

Appendix A. Non-randomized LP testing for jamming

Determining whether the linear system of inequalities (9) with $\Delta\mathbf{l} = 0$ has non-trivial solutions is an interesting mathematical programming problem. One approach is the following: Solve the following linear program aimed at maximizing the sum of the (positive) gap dilations:

$$\begin{aligned}&\min_{\Delta\mathbf{R}} \sum_{\{i,j\}} (\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R})_{i,j} = \min\,(\mathbf{Ae})^{\mathrm{T}}\Delta\mathbf{R}\\ &\text{such that } \mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant 0,\end{aligned} \tag{A.1}$$

where $\mathbf{e}$ is the unit vector, and if this returns $\Delta\mathbf{R} = 0$ as one of the optimal solutions, test the rigidity matrix $\mathbf{A}$ for rank-defficiency, i.e., look for non-trivial solutions of $\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} = 0$. If this also fails to find an unjamming motion, the packing is jammed. Notice that this will usually produce a single unjamming motion, which we have found to be rather uninteresting for lattice packings in the sense that it is extremely dependent upon $\mathbf{N}_c$.

Appendix B. The geometry of the set of unjamming motions

The linearized impenetrability constraints $\mathbf{A}^{\mathrm{T}}\Delta\mathbf{R} \leqslant \Delta\mathbf{l}$ define a polyhedral set $\mathscr{P}_{\Delta\mathbf{R}}$ of *feasible* (*linearized*) *displacements*. Every such convex polyhedron consists of a finite piece $\mathscr{P}^{\mathrm{hull}}_{\Delta\mathbf{R}}$, a convex polytope given by the convex hull of its extreme points and possibly an unbounded piece $\mathscr{C}_{\Delta\mathbf{R}}$, a finitely generated *polyhedral cone*. In some cases this cone will be empty (i.e., $\mathscr{C}_{\Delta\mathbf{R}} = \{0\}$), but in others it will not, as can be seen in Fig. 1. The full nonlinear impenetrability constraints given by (8) define the true set of feasible displacements $\mathscr{P}^{\mathrm{NL}}_{\Delta\mathbf{R}} = \mathscr{J}_{\mathbf{R}} - \{\mathbf{R}\}$, which always relaxes the linearization: $\mathscr{P}_{\Delta\mathbf{R}} \subseteq \mathscr{P}^{\mathrm{NL}}_{\Delta\mathbf{R}}$. A mathematically well-defined definition of jamming is to take any ray in the cone $\mathscr{C}_{\Delta\mathbf{R}}$ as an unjamming motion, and exclude others, however, as Fig. 1 shows, the elongated corners of $\mathscr{P}_{\Delta\mathbf{R}}$ are in fact very likely to be unbounded in the true nonlinear feasible set of displacements $\mathscr{P}^{\mathrm{NL}}_{\Delta\mathbf{R}}$, so we prefer to take any "long" direction in $\mathscr{P}_{\Delta\mathbf{R}}$ as an unjamming motion.

We note that the randomized LP algorithm proposed here strictly answers the question of whether the polyhedral set of feasible displacements contains an unbounded ray (i.e., whether $\mathscr{C}_{\Delta\mathbf{R}} \neq \{0\}$) just by applying two (non-zero) loads $\mathbf{b}$ and $-\mathbf{b}$. This is because an attempt to find such a ray will be unsuccessful only if $-\mathbf{b} \in \mathscr{C}^*_{\Delta\mathbf{R}}$, where $\mathscr{C}^*_{\Delta\mathbf{R}}$ is the dual (conjugate) cone of $\mathscr{C}_{\Delta\mathbf{R}}$, and in this case $\mathbf{b} \notin \mathscr{C}^*_{\Delta\mathbf{R}}$, so that using the load

$-\mathbf{b}$ will find a ray if such a ray exists. Also, we note that one cannot hope to fully characterize the cone of first-order unjamming motions $\mathscr{C}_{\Delta\mathbf{R}}$ (i.e., find its convex hull of generating rays), as this is related to the hard problem of fully enumerating the vertices of a polyhedron. Our randomized approach essentially finds a few sample rays in $\mathscr{C}_{\Delta\mathbf{R}}$.

Appendix C. Strict jamming with periodic boundary conditions

We demonstrate that the mathematical statement that a packing is rigid if and only if it is infinitesimally rigid (see, for example, [15]) is true also even if we allow the periodic lattice to change. This argument is motivated by Swinnerton-Dyer [30]. In practice this means that any (first-order) solution obtained by the linear programming algorithm can be appropriately scaled to obtain a truly feasible displacement in which some particles strictly lose contact and the volume of the unit cell strictly decreases. We include this proof because this argument has not previously appeared. If we consider the nonlinear corrections to (14) by expanding (13) to second order, we get:

$$\begin{aligned}\det(\mathbf{\Lambda}+\Delta\mathbf{\Lambda}) &= \det(\mathbf{\Lambda})\det(\mathbf{I}+\boldsymbol{\varepsilon}) = \det(\mathbf{\Lambda})\prod_{i=1}^{d}(1+\lambda_i)\\ &= \det(\mathbf{\Lambda})\left(\sum_i \lambda_i + \sum_{i>j}\lambda_i\lambda_j + \text{higher-order terms}\right),\end{aligned}$$

where λ_i are the eigenvalues of the strain, which are all real because of (15) and also have a non-positive sum due to (14). If $\mathrm{Tr}(\boldsymbol{\varepsilon}) = \sum\lambda_i < 0$, then the first-order term will dominate for sufficiently small deformations and the nonlinear constraint (13) will be satisfied. Furthermore, if $\mathrm{Tr}(\boldsymbol{\varepsilon}) = 0$, then we have that

$$\sum_{i>j}\lambda_i\lambda_j = -\frac{1}{2}\sum_i \lambda_i^2 < 0$$

for any non-trivial deformation, which shows that the second-order term is of the correct sign and the volume of the unit cell strictly decreases for sufficiently small deformations.

References

[1] A. Mehta (Ed.), Granular Matter, Springer-Verlag, New York, 1994.

[2] S.F. Edwards, D.V. Grinev, The tensorial formulation of volume function for packings of particles, Chem. Engrg. Sci. 56 (2001) 5451–5455.

[3] R. Zallen, The Physics of Amorphous Solids, Wiley, New York, 1983.

[4] J.P. Hansen, I.R. McDonald, Theory of Simple Liquids, Academic Press, New York, 1986.

[5] S. Torquato, Random Heterogeneous Materials: Microstructure and Macroscopic Properties, Springer-Verlag, New York, 2002.

[6] T. Aste, D. Weaire, The Pursuit of Perfect Packing, IOP Publishing, 2000.

[7] M.F. Thorpe, P.M. Duxbury (Eds.), Rigidity Theory and Applications, Fundamental Materials Research, Kluwer/Plenum, 1999.

[8] C.M.M.J. Alava, P.M. Duxbury, H. Rieger (Eds.), Exact Combinatorial Algorithms: Ground States of Disordered Systems, Domb and Lebowitz series on Phase Transitions and Critical Phenomena, vol. 18, Academic Press, New York, 2001.

[9] D.J. Jacobs, B. Hendrickson, An algorithm for two dimensional rigidity percolation: the pebble game, J. Comput. Phys. 137 (2) (1997) 346–365.

[10] S. Torquato, T.M. Truskett, P.G. Debenedetti, Is random close packing of spheres well defined?, Phys. Rev. Lett. 84 (2000) 2064–2067.

[11] F.H. Stillinger, H. Sakai, S. Torquato, Lattice-based random jammed configurations for hard particles, Phys. Rev. E 67 (2003) 031107.

[12] L. Fejes Tóth, Regular Figures, Pergamon Press, New York, 1964.
[13] S. Torquato, F.H. Stillinger, Multiplicity of generation, selection, and classification procedures for jammed hard-particle packings, J. Phys. Chem. B 105 (2001) 11849–11853.
[14] A. Donev, S. Torquato, F.H. Stillinger, R. Connelly, Jamming in hard sphere and disk packings, J. App. Phys. (in press).
[15] R. Connelly, Rigid circle and sphere packings. Part I: Finite packings, Structural Topology 14 (1988) 43–60, see also [26].
[16] R. Connelly, K. Bezdek, A. Bezdek, Finite and uniform stability of sphere packings, Discrete Comput. Geometry 20 (1998) 111–130.
[17] R. Connelly, W. Whiteley, Second-order rigidity and prestress stability for tensegrity frameworks, SIAM J. Discrete Math. 9 (3) (1996) 453–491.
[18] S. Torquato, A. Donev, F.H. Stillinger, Breakdown of elasticity theory for jammed hard-particle packings: conical nonlinear constitutive theory, Int. J. Solids Struct. 40 (25) (2003) 7143–7153.
[19] E.L. Hinrichsen, J. Feder, T. Jossang, Random packing of disks in two dimensions, Phys. Rev. A 41 (8) (1990) 4199–4209.
[20] W. Uhler, R. Schilling, A local description of stable 2*d* random packings, J. Phys. C 18 (1985) L979–L983.
[21] R. Vanderbei, Linear Programming: Foundations and Extensions, Kluwer Academic Publishers, Dordrecht, 1997.
[22] A. Donev, Homepage for the sphere packing project, with useful supplementary materials. Available from http://atom.princeton.edu/donev/Packing.
[23] J.N. Roux, Geometric origin of mechanical properties of granular materials, Phys. Rev. E 61 (6) (2000) 6802–6836.
[24] B.D. Lubachevsky, F.H. Stillinger, Geometric properties of random disk packings, J. Statist. Phys. 60 (1990) 561–583, see also [29].
[25] A. Zinchenko, Algorithm for random close packing of spheres with periodic boundary conditions, J. Comput. Phys. 114 (1994) 298–307.
[26] R. Connelly, Rigid circle and sphere packings. Part II: Infinite packings, Structural Topology 16 (1991) 57–76, second part of [15].
[27] A. Donev, S. Torquato, Energy-efficient actuation in infinite lattice structures, Int. J. Solids Struct. 51 (8) (2003) 1459–1475.
[28] WWW, The general purpose interior-point LOQO optimization library is not public-domain, but can be tried at http://orfe.princeton.edu/loqo. The public-domain PCx library implements interior point linear programming algorithms and can be found at http://www-fp.mcs.anl.gov/otc/Tools/PCx/.
[29] B.D. Lubachevsky, F.H. Stillinger, E.N. Pinson, Disks vs. spheres: contrasting properties of random packings, J. Statist. Phys. 64 (1991) 501–525, second part of [24].
[30] H.P.F. Swinnerton-Dyer, Extremal lattices of convex bodies, Proc. Cambridge Philos. Soc. 49 (1953) 161–162.