Efficient algorithms for general periodic Lorentz gases in two and three dimensions

IOP Publishing Journal of Physics A: Mathematical and Theoretical

J. Phys. A: Math. Theor. **49** (2016) 025001 (20pp) doi:10.1088/1751-8113/49/2/025001

Efficient algorithms for general periodic Lorentz gases in two and three dimensions

Atahualpa S Kraemer[1,3], **Nikolay Kryukov**[2] and **David P Sanders**[2]

[1] Institut für Theoretische Physik II-Soft Matter Heinrich-Heine-Universität Düsseldorf Building 25.32 Room O2.56 Universitätsstrasse 1 D-40225 Düsseldorf, Germany
[2] Departamento de Física, Facultad de Ciencias, Universidad Nacional Autónoma de México, Ciudad Universitaria, México D.F. 04510, Mexico

E-mail: ata.kraemer@gmail.com, kryukov@ciencias.unam.mx and dpsanders@ciencias.unam.mx

Received 2 July 2015, revised 1 October 2015
Accepted for publication 20 October 2015
Published 8 December 2015

Abstract

We present efficient algorithms to calculate trajectories for periodic Lorentz gases consisting of square lattices of circular obstacles in two dimensions, and simple cubic lattices of spheres in three dimensions; these become increasingly efficient as the radius of the obstacles tends to 0, the so-called Boltzmann–Grad limit. The 2D algorithm applies continued fractions to obtain the exact disc with which a particle will collide at each step, instead of using periodic boundary conditions as in the classical algorithm. The 3D version incorporates the 2D algorithm by projecting to the three coordinate planes. As an application, we calculate distributions of free path lengths close to the Boltzmann–Grad limit for certain Lorentz gases. We also show how the algorithms may be applied to deal with general crystal lattices.

Keywords: Lorentz gas, numerical algorithms, billard models, free paths

(Some figures may appear in colour only in the online journal)

1. Introduction

Lorentz gases are simple physical systems that present deterministic chaos [1], and are a popular model in statistical mechanics and nonlinear dynamics. This model consists of point particles that move freely until they encounter obstacles, often spheres, where they undergo elastic collisions.

[3] Author to whom any correspondence should be addressed.

These systems can have different configurations of obstacles, e.g., random arrangements [2–4] or quasiperiodic structure [5, 6]. However, due to its simplicity, the periodic case has been most widely studied; see, e.g., [7–10]. In this case, the model is equivalent to a Sinai billiard [7]. Many of the results obtained theoretically for these gases are in the limit where obstacles are very small, i.e., the so-called Boltzmann–Grad limit [11–19]. There are still many interesting open questions in this area [20–23].

The standard simulation method for periodic Lorentz gases is to reduce to a single cell with periodic boundary conditions, and, in the simplest case, an obstacle in the centre of the cell [24, 25]. However, this requires that the program check in each cell whether the particle collides with the obstacle in the cell, or if it will move to the next cell. If the obstacle is large, it is quite likely that the particle will collide each time it crosses into a new cell. However, for very small obstacles, this method becomes very inefficient.

Instead, we would like to just find the coordinates of the next obstacle with which the particle will collide, given its initial position and velocity. This turns out to be closely related to the best rational approximant to an irrational number, and can be solved using the continued-fraction algorithm. Continued fractions have often been used to provide information about the free path distribution of the periodic Lorentz gas in the Boltzmann–Grad limit [8, 9, 11–17]. An algorithm along these lines was previously developed: see comments in [26]; however, it was never published [T Geisel, private communication]. Caglioti and Golse developed a method to encode the trajectories of particles using the continued fraction algorithm and the so-called 3-length theorem [11, 12].

However, Golse's algorithm works only if the particle leaves the surface of a disk. This restriction prevents the algorithm from being used in other geometries, such as two incommensurate overlapping arrays of square lattices, or with different shapes of obstacles; such systems may produce a number of surprising effects [27].

On the other hand, due to the construction of Golse's algorithm, it is not possible to use it in higher dimensions, which is 'a notoriously more difficult problem' [12]. Recent advances on multidimensional continued-fraction algorithms may provide a possible future direction [28, 29], although here we have opted for a different approach for higher-dimensional systems.

In this paper, we develop an efficient algorithm to find a collision with a 2D square lattice of discs starting from an arbitrary initial condition. We then use that 2D algorithm as part of an efficient algorithm for a 3D simple cubic lattice by projecting onto coordinate planes. Finally, we show how obstacles arranged on arbitrary (periodic) crystal lattices may be treated.

2. Classical algorithm for the periodic Lorentz gas

We begin by recalling the classical algorithm for a Lorentz gas on a d-dimensional (hyper)-cubic lattice, where each cell contains a single spherical obstacle of radius r. The simplest method is to locate the centre of the obstacle at the centre of a cubic cell $\left[-\frac{1}{2}, \frac{1}{2}\right)^d$, and to track which cell $\mathbf{n} \in \mathbb{Z}^d$ the particle is in using periodic boundary conditions: when a particle hits a cell boundary, its position is reset to the opposite boundary and the cell counter $\mathbf{n}$ is updated accordingly; see figure 1.

In each cell, the classical algorithm is as follows. For a particle with initial position $\mathbf{x}$ and velocity $\mathbf{v}$, a collision occurs with the disc with centre at $\mathbf{c}$ and radius r at a time t^* if

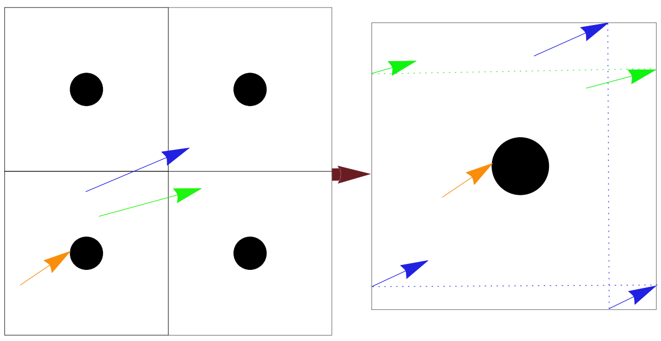

Figure 1. Reducing the dynamics in a periodic lattice to a single cell with periodic boundary conditions.

$$\left|\left|\mathbf{x} + \mathbf{v}t^* - \mathbf{c}\right|\right| = r. \tag{1}$$

This gives a quadratic equation for the collision time, and hence

$$t^* = -B - \sqrt{B^2 - C}, \tag{2}$$

where

$$B = \frac{(\mathbf{x} - \mathbf{c}) \cdot \mathbf{v}}{v^2}; \qquad C = \frac{(\mathbf{x} - \mathbf{c})^2 - r^2}{v^2}, \tag{3}$$

provided that the condition $B^2 - C \geqslant 0$ is satisfied. If this happens, then the collision position is $\mathbf{x} + \mathbf{v}t^*$. If the condition is not satisfied, then the trajectory misses the disc.

If no collision with the obstacle occurs, i.e. when $B^2 - C < 0$, the velocity is conserved and the particle will hit one of the cell boundaries. To determine which boundary will be hit, we find intersection times of the trajectory with each cell boundary (lines 2D or planes in 3D), given by

$$t_{i,\pm} = \frac{\pm\frac{1}{2} - x_i}{v_i},$$

where i runs from 1 to the number of dimensions (2 or 3) and the sign corresponds to the two opposite faces in direction i. The least positive time then gives the collision time with the boundary. Depending on which boundary was hit, we move to the new unit cell and repeat the process: if $t_{i,\pm}$ is the minimum time, then the positive (respectively negative) ith boundary is hit, and the ith component of the cell is updated to $n_i' = n_i \pm 1$.

3. Efficient 2D algorithm

The classical algorithm is efficient for large radii r, but very inefficient once r is small, since a trajectory will cross many cells before encountering a disc.

In this section, we develop an algorithm to simulate the periodic Lorentz gas on a 2D square lattice, based on the use of continued fractions, whose goal is to calculate *efficiently* the first disc hit by a particle, even for very small values of the radius r. Without loss of generality, we will use the lattice formed by the integer coordinates in the 2D plane.

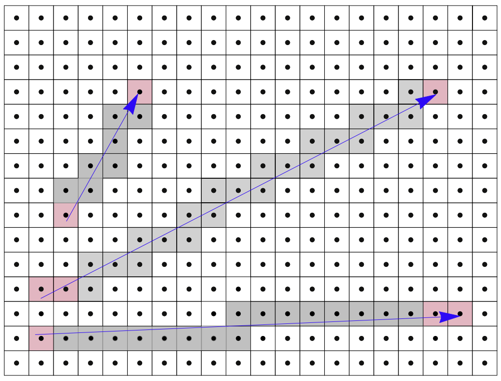

Figure 2. Cells covered by the classical algorithm, compared to the few steps required by the efficient algorithm.

We wish to calculate the minimal time $t^* > 0$ such that a collision occurs with some disc centred at $\mathbf{c} = (q, p)$, with q and p integers, by 'jumping' straight to the correct disc; see figure 2.

3.1. Continued fraction algorithm: approximation of an irrational number by a rational

In this section we recall the continued fraction algorithm and some properties of continued fractions; see, e.g., [30] for proofs. The geometrical interpretation has been suggested before by many other authors; see, for example, [31].

A continued fraction is obtained via an iterative process, representing a number α as the sum of its integer part, a_0, and the reciprocal of another number, $\alpha_1 \coloneqq \alpha - a_0$, then writing α_1 as the sum of its integer part, a_1, and the reciprocal of $\alpha_2 \coloneqq \alpha_1 - a_1$, and so on. This gives the continued fraction representation of α:

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}.$$

This iteration produces a sequence of integers $\lfloor \alpha \rfloor = a_0$, $\lfloor \alpha_1 \rfloor = a_1$, $\lfloor \alpha_2 \rfloor = a_2$, etc. We define inductively two sequences of integers $\{p_n\}$ and $\{q_n\}$ as follows:

$$p_{-2} = 0; \quad p_{-1} = 1; \quad p_i = a_i p_{i-1} + p_{i-2}; \tag{4}$$

$$q_{-2} = 1, \quad q_{-1} = 0, \quad q_i = a_i q_{i-1} + q_{i-2}. \tag{5}$$

With this sequence we can approximate any irrational number α using the Hurwitz theorem: for any irrational number, α, all the relative prime integers p_n, q_n of the sequences defined in equations (4) and (5) satisfy

$$\left| \alpha - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n^{\,2}}. \tag{6}$$

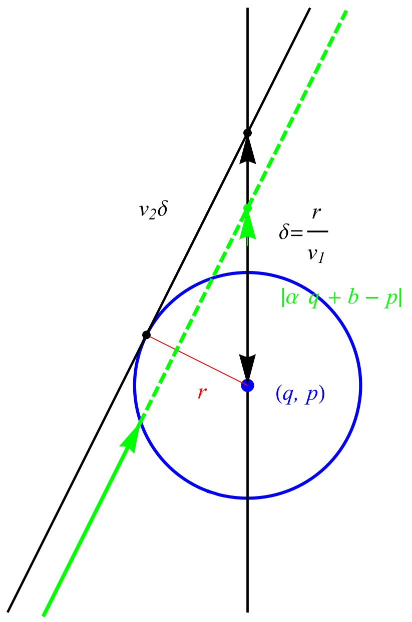

Figure 3. Relation between the intersection of a line and a circle with integer coordinates and the intersection of the line $x = q$.

3.2. Collision with a disc

The classical algorithm finds the intersection between a line, corresponding to the trajectory of the particle, and a circle, corresponding to the circumference of the disc, by solving the quadratic equation (2) for t^*. A first improvement follows from observing that we may instead look for the intersection of the trajectory with another line, as follows. In the following, we take $v_1 > 0$ and $v_2 > 0$; by symmetry of the system, we can always rotate or reflect it such that these conditions are satisfied.

We write the equation of the particle's trajectory as $y = \alpha x + b$, with slope $\alpha = v_2/v_1$, and look for its intersection with the vertical line $x = q$ passing through the disc at (q, p). As shown in figure 3, if $|\alpha q + b - p| < \delta \coloneqq r/v_1$, then a collision with the disc (q, p) will occur. Due to the periodic boundary conditions, we can redefine $b \coloneqq \{|\alpha q + b - p|\}$ where $\{\cdot\}$ denotes the fractional part. Thus, $0 < b < 1$, and we need only solve $b < \delta$.

We do not need to apply periodic boundary conditions at every step; rather, we only need to check

$$|\{\alpha q_n\} + b - 1| < \delta, \tag{7}$$

where $\{\cdot\}$ denotes the fractional part, and $q_n = q_{n-1} + 1$, where q_1 is the x-coordinate of the closest obstacle to the particle at $t = 0$. Then, the first q_n that satisfies this inequality will be q. To calculate p, we use that either $p = \lfloor \alpha q + b \rfloor$ or $p = \lfloor \alpha q + b \rfloor + 1$.

Now, to simplify the algorithm further, consider the integer coordinates (q_n, p_n) such that

$$|\alpha q_n - p_n + b| < \delta, \tag{8}$$

and for any pair of numbers (i, j) such that $i < q_n$, then $|\alpha i - j + b| > \delta$, $q = q_n$, and $p = p_n$.

But $\left|\alpha q_i - p_i + b\right|$ are the distances between the integer coordinates (q_i, p_i) and the point $(q_i, \alpha q_i + b)$. Thus, we would like a sequence such that

$$|\alpha q_i - p_i + b| < |\alpha q_{i-1} - p_{i-1} + b| \tag{9}$$

for every integer $i > 1$. Also, the first pair of integer coordinates q_0 and p_0 should be $(0, 0)$ or $(0, 1)$, minimizing $\left|\alpha q_0 - p_0 + b\right|$, that is

$$|\alpha q_1 - p_1 + b| < f(b) = \begin{cases} b, & \text{if } b < \dfrac{1}{2} \\ 1 - b, & \text{if } b > \dfrac{1}{2}. \end{cases} \tag{10}$$

Note that if $b < 1/2$, we have $p_n = \lfloor \alpha q_n + b \rfloor = \lfloor \alpha q_n \rfloor$, if $b + \alpha q_n - \lfloor \alpha q_n \rfloor < 1$, and $p_n = \lfloor \alpha q_n \rfloor + 1$, if $b + \alpha q_n - \lfloor \alpha q_n \rfloor > 1$. Whereas if $b > 1/2$, we have $p_n = \lfloor \alpha q_n + b \rfloor + 1 = \lfloor \alpha q_n \rfloor + 1$, if $b + \alpha q_n - \lfloor \alpha q_n \rfloor < 1$ and $p_n = \lfloor \alpha q_n \rfloor + 2$, if $b + \alpha q_n - \lfloor \alpha q_n \rfloor > 1$. Substituting these four cases in the two cases of equation (10), we obtain that indeed $p_1 = \lfloor \alpha q_1 \rfloor + 1$. Iterating the inequality (9) we obtain

$$p_n = \lfloor \alpha q_n \rfloor + 1. \tag{11}$$

Combining the inequality (8) with equation (11), we obtain again equation (7).

Thus, we have reduced the solution from two linear equations and one quadratic to one linear equation. Furthermore, now we do not check in every periodic cell, because if $\alpha > 1$, for every q_n we advance $\left(\lfloor q_n \alpha \rfloor - \lfloor q_{n-1} \alpha \rfloor\right)$ cells. And we do not need to apply periodic boundary conditions until we reach the obstacle.

3.3. The Diophantine inequality: $|\alpha p - q| \leqslant \delta$

Now, a better algorithm should find a way to find the set of q_i, such that inequality (9) holds for every i, and there is no integer q such that $q_i < q < q_{i-1}$ for some i and $\left|\{\alpha q_i\} + b - 1\right| < |\{\alpha q\} + b - 1| < \left|\{\alpha q_{i-1}\} + b - 1\right|$.

In order to do this, we can use the continued fraction algorithm to obtain solutions to the inequality $|\alpha q - p| \leqslant \delta$. This algorithm already gives a sequence of (q_n, p_n) such that $\left|\alpha q_i - p_i\right| < \left|\alpha q_{i-1} - p_{i-1}\right|$ if $q_{i-1} < q_i$. In addition, the convergents of the continued fractions provide best approximants and hence the smallest solution of the inequality (10). So, if we turn our inequality (10) into this other inequality, we will find our algorithm just by using the continued fraction algorithm. Indeed, using equation (11) and the inequality (10), we obtain $\left|\{\alpha q_1\} - 1\right| < 2b$ if $b < 1/2$ or $<2(1 - b)$ if $b > 1/2$, which is almost the continued fraction inequality, except that p_1 is always equal to $\lfloor \alpha q_1 \rfloor + 1$.

$$|\alpha q - p| < \begin{cases} 2b, & \text{if } b < \dfrac{1}{2}, \\ 2(1 - b), & \text{if } b > \dfrac{1}{2}. \end{cases} \tag{12}$$

Now we can apply the continued fraction algorithm to obtain p_1 and q_1 of inequality (12). If $\alpha q_1 + b < \delta$ or $1 - \alpha q_1 + b < \delta$, then we have found the center of the obstacle at (p_1, q_1), with $p_1 = \lfloor \alpha q_1 \rfloor + 1$; otherwise, we have not found it, but we know that if the center of collision is at (p, q) then $p \geqslant p_1$, and $q \geqslant q_1$. Hence, we can just use (q_1, p_1) even if they do not satisfy inequality (9). Redefining b_i as $b_0 = b$, $b_i = \{\alpha q_i + b\}$, we can calculate a succession of (p_i, q_i). If $b_n < \delta$ the algorithm stops, and the collision will take place with the obstacle centred at the coordinates (q_n, p_n). Otherwise, if $b_n = b$, then the particle has a rational slope equal to that of a channel, and so is travelling along and parallel to that channel, and hence will never undergo another collision with an obstacle. In this case, the algorithm throws an exception.

3.4. Complete 2D algorithm

We now have the necessary tools to implement the algorithm. Pseudo-code for the complete efficient 2D algorithm is given in the appendix; source code for our implementation, written in the Julia programming language [32, 33], may be found in the supplementary information.

The functions described above work only for velocities in the first octant, i.e. such that $0 < v_2 < v_1$. If the velocity does not satisfy this condition, we use the symmetry of the system, applying rotations and reflections and then, after obtaining the coordinates of the collision, use the inverse transformations to return to the original system; see the appendix for details.

Finally, to calculate the exact collision point, we use the classical algorithm to obtain the intersection between a line and a circle, and from there the resulting post-collision velocity.

4. Efficient 3D algorithm

We now develop an efficient algorithm for calculating the next collision with a sphere in 3D on a simple cubic lattice, which again is designed to be efficient for a small radius r. The algorithm works by projecting the geometry onto the 2D coordinate planes and then using the above efficient 2D algorithm in each plane, as follows.

Suppose we project a particle trajectory in a 3D lattice onto one of the $x{-}y$, $x{-}z$ or $y{-}z$ planes. We will obtain a periodic square lattice with a 2D trajectory. This trajectory is *not* a trajectory of the 2D Lorentz gas, however—it may pass through certain discs as if they were not there, and may have non-elastic reflections with other discs. Furthermore, the speed varies.

However, we will use this to apply the 2D algorithm for the projections in each plane, in order to obtain coordinates of a disc in each of the three planes at which the first collision in that plane is predicted to occur. We now check whether the obstacle coordinates in these projections correspond to the *same* 3D obstacle, i.e. if the coordinates coincide pairwise. If not, then we have not found a true collision in 3D. We move the particle to the cell containing the obstacle that is furthest away, i.e., has the maximum collision time in its respective plane, and continue.

If the obstacle coordinates do coincide pairwise, then this algorithm predicts that there is a collision. However, this may not be true, due to the geometry, as follows. Calculating a collision with a disc in one of the planes $x{-}y$, $x{-}z$ or $y{-}z$ is equivalent to calculating a collision in space with a cylinder orthogonal to that plane. Joining these coordinates together means calculating a collision with the intersection of three orthogonal cylinders with the same radius. Figure 4 shows such an intersection of three cylinders, called a tricylinder or Steinmetz solid [34], together with a sphere of the same radius. The sphere is contained inside the intersection of the cylinders, and has a smaller volume.

Thus the algorithm may predict a false collision—with the tricylinder—even though the particle does not collide with the sphere. To control this, we check if the particle really does collide with this obstacle by using the classical algorithm; if so, then we have found a true collision, and if not, we move the particle to the next cell and continue applying the algorithm. Numerically, we find the probability of a false collision to be around 0.17.

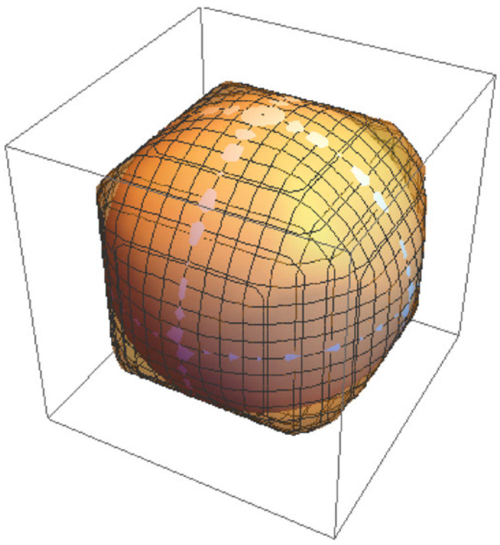

Figure 4. A sphere of radius r embedded into the intersection of three orthogonal cylinders of the same radius. The volume inside the intersection but outside the sphere is the region where the 3D algorithm predicts false collisions.

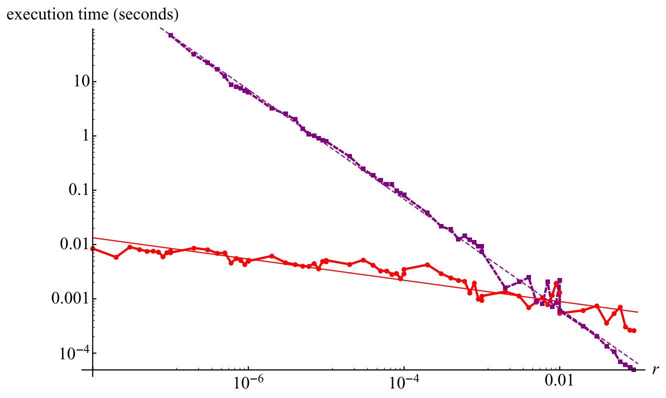

Figure 5. Mean execution time to find the first collision in the 2D square Lorentz gas, for the classical (dotted curve) and efficient (solid curve) algorithms. The straight lines show power-law fits.

Pseudo-code for the efficient 3D algorithm is given in the appendix.

5. Numerical measurements

5.1. Execution time

In order to test the efficiency of our algorithms, we measure the average execution time of the function that finds the first collision, starting from an initial point near the origin, as a function of obstacle radius, for both the classical and efficient algorithms, in 2D and 3D.

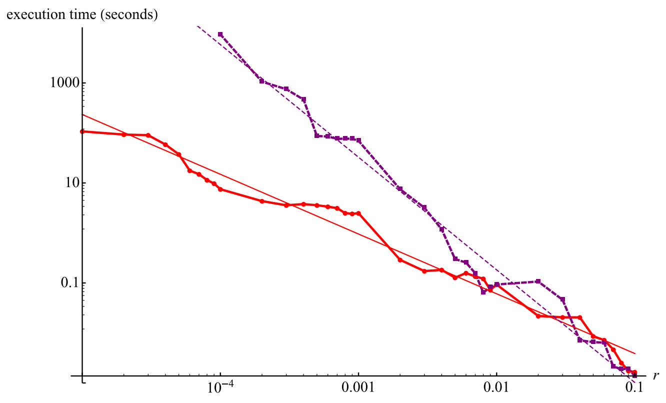

Figure 6. Mean execution time to find the first collision in the 3D simple cubic Lorentz gas, for the classical (dotted curve) and efficient (solid curve) algorithms. The straight lines show power-law fits.

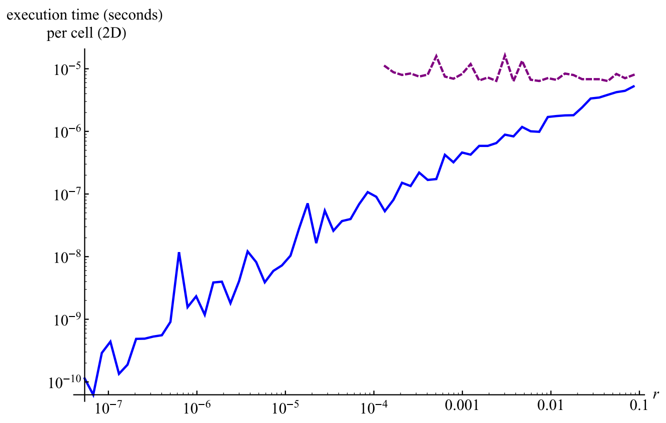

Figure 7. Mean execution time per cell to find the first collision in a 2D square Lorentz gas, for the classical (dashed curve) and efficient (solid curve) algorithms, as a function of disc radius, r.

Figures 5 and 6 show the results for the 2D and 3D algorithms, respectively. We performed power-law fits for the execution time as a function of obstacle radius. For the 2D case, we find an exponent of -1.01 for the classical algorithm and -0.20 for the efficient algorithm. For the 3D case, the exponents are -2.25 and -1.20 for classical and efficient, respectively. As we can see, our algorithms are increasingly more efficient for $r < 0.01$.

Similarly, we calculated the execution time per cell as a function of the obstacle radius, for both the 2D and 3D efficient algorithms, with comparison to the corresponding classical algorithms; see figures 7 and 8. Since the classical algorithms use periodic boundary conditions, the time per cell is basically constant, independent of the obstacle radius. For small radii, we again observe the efficiency of the new algorithms.

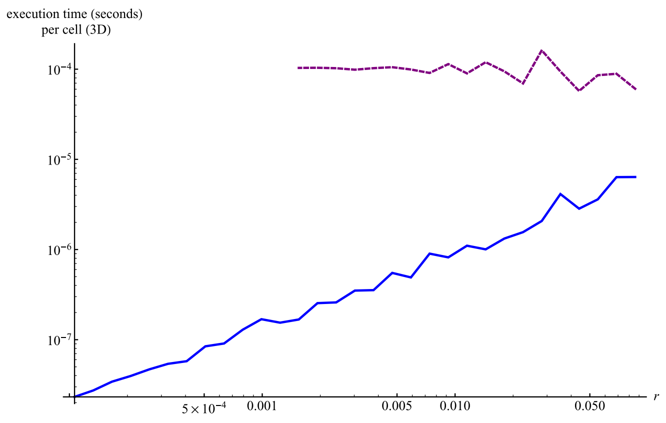

Figure 8. Mean execution time per cell to find the first collision in the 3D cubic Lorentz gas, for the classical (dashed curve) and efficient (solid curve) algorithms, as a function of sphere radius, r.

5.2. Asymptotic complexity of the classical and efficient algorithms

The scaling of the complexity of the classical algorithm as $1/r$ may be explained as follows. The distance that a particle travels before it collides with an obstacle, i.e. the free path length, is a function of obstacle size: the smaller the obstacles, the longer the free paths.

In periodic Lorentz gases, there is a simple formula for the mean free path between consecutive collisions, τ, that arises from geometrical considerations [35]: it is, up to a dimension-dependent constant, the ratio of the volume of the available space outside the obstacles to the surface area of the obstacles. For the square 2D Lorentz gas with discs of radius r, we have

$$\tau(r) = \frac{1 - \pi r^2}{2r}, \tag{13}$$

with asymptotics r^{-1} for small r. Since the classical algorithm must cross this distance at speed 1, it takes time proportional to $1/r$, as we find numerically. In applying this algorithm, approximately $1/r$ quadratic equations and four times as many linear equations must be solved.

For the simple cubic Lorentz gas in 3D with spheres of radius r, we have

$$\tau(r) = \frac{1 - \frac{4}{3}\pi r^3}{\pi r^2}, \tag{14}$$

with asymptotics r^{-2}, which is not far from our numerical results.

On the other hand, the efficient algorithm checks only one quadratic equation, and around $2r^{-1/2}$ linear equations, giving an upper bound of $r^{-1/2}$ for the complexity. (This calculation uses Hurwitz's theorem.) Numerically, it turns out to be significantly more efficient than that.

5.3. Free flight distribution

As an example application of our algorithm, we measure the distribution of free flight lengths for the first collision for certain systems studied by Marklof and Strömbergsson [27]. They studied N incommensurable, overlapping periodic Lorentz gases in the Boltzmann–Grad

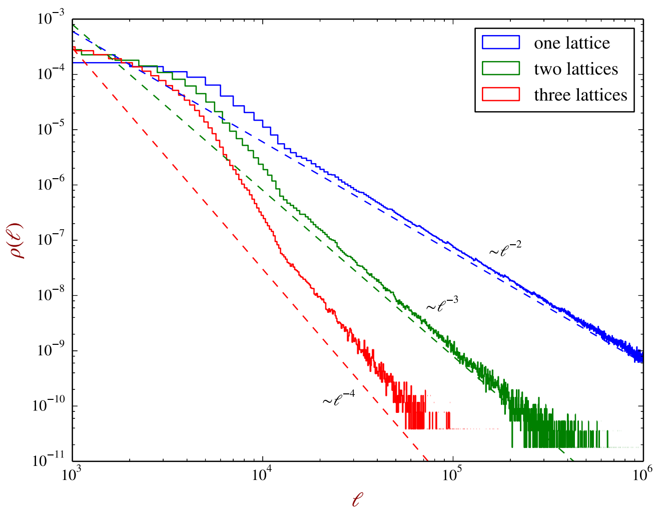

Figure 9. Probability density of the first free flight for two and three incommensurable, overlapping periodic Lorentz gases with angles $\pi/5$ and $\pi/7$; a total of 10^8 initial conditions was used. The results for a single lattice are shown for comparison. The dashed lines and labels show the theoretical asymptotics.

limit, $r \rightarrow 0$, and proved that the asymptotic decay of the probability density for free flights in that system is $\sim\ell^{-N-2}$. It follows that the asymptotic density of the *first* free flight should be $\rho(\ell) \sim \ell^{-N-1}$.

Figure 9 shows our numerical results for this distribution in the case of two and three overlapping lattices, compared to the asymptotic decay given by the rigorous result of [27]. To obtain this plot, we fixed the radius as $r = 10^{-4}$ and calculated free flights for a given initial condition for a 2D lattice, and for the same lattice rotated by $\pi/5$ and $\pi/7$, respectively. The first free flight for each lattice is calculated separately, and the minimum of those results is then taken to give the first free flight for the superposition of either two or three incommensurable lattices. The distributions obtained numerically do indeed follow the power laws predicted. Naturally, it becomes increasingly difficult to obtain the asymptotic behaviour of the densities as the number of lattices increases.

6. Extension to general periodic lattices

So far, we have restricted attention to spherical obstacles on simple cubic lattices. In this section, we will show how to deal with *arbitrary* periodic crystal lattices. Such lattices consist of a *basis* (finite collection) of different spheres (atoms), in unit cells of a Bravais lattice; see, e.g., [36].

This may be considered as the superposition of distinct Bravais lattices, one for each of the distinct atoms in the basis. Thus the efficient algorithm may be used separately for each such lattice, and then we take the minimum time to determine the next collision. In this way, we can now restrict attention to simulating a Bravais lattice with a single atom per unit cell. For simplicity we will describe the method in 2D; the 3D case is similar.

A Bravais lattice in 2D is the set of points given by linear combinations of the form $a_1\mathbf{u}_1 + a_2\mathbf{u}_2$ of vectors $\mathbf{u}_i$ defining the directions of the lattice, where the a_i are integers. We

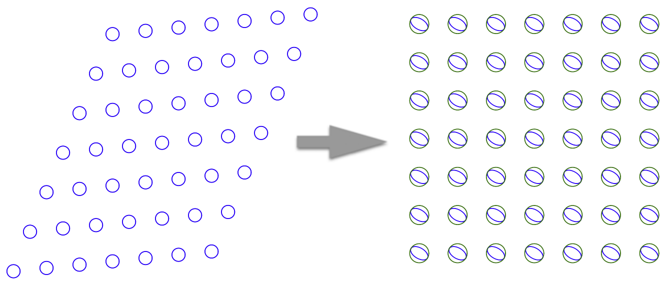

Figure 10. Effect of applying the transformation M_{os} to an oblique lattice of discs (left). The result is a square lattice of ellipses; circumscribed circles are also shown (right).

pass from the square lattice to the oblique lattice by applying the transformation matrix M_{so}, defined such that its columns are the vectors $\mathbf{u}_i$:

$$\mathsf{M}_{so} := (\mathbf{u}_1|\mathbf{u}_2). \tag{15}$$

To transform back from the Bravais lattice to the square lattice, we apply the inverse transformation $\mathsf{M}_{os} := \mathsf{M}_{so}^{-1}$. Starting from circular obstacles of radius r in the Bravais lattice and applying M_{os} gives one obstacle per unit cell at integer coordinates in the square lattice. However, this stretches the shape of the resulting obstacles into ellipses, as follows from the singular-value decomposition of M_{os}; see, e.g., [37]. The semi-major axis of the resulting ellipses is $r' = r\sigma_1$, where σ_1 is the first singular value of M_{os}. We circumscribe the resulting ellipse by a circular obstacle of radius r', giving a standard square periodic Lorentz gas, suitable for analysis using the corresponding efficient 2D algorithm; see figure 10.

Starting from a given initial condition $\mathbf{x}_0,\mathbf{v}_0$ in the Bravais lattice we wish to simulate, we transform these to $\mathbf{x}_0' := \mathsf{M}_{os} \cdot \mathbf{x}_0$ and $\mathbf{v}_0' := \mathsf{M}_{os} \cdot \mathbf{v}_0$ in the square lattice. We then apply the efficient algorithm in the square lattice to obtain a proposed disc or sphere with integer coordinates $\mathbf{n}$. These coordinates are mapped to the oblique lattice, giving a proposed disc or sphere with coordinates $\mathbf{n}' := \mathsf{M}_{so} \cdot \mathbf{n}$. We must check, however, if this is a true collision with the obstacle at $\mathbf{n}'$ using the classical algorithm, since the proposed collision with a disc in the square lattice may not actually hit the true elliptical obstacle there. If it is not a true collision, then we move to the next cell and continue; if it is a true collision, we calculate the new post-collision velocity.

Provided the transformation M_{so} does not stretch the obstacles too much, and the radius is small, this algorithm will still be very efficient.

Finally, non-spherical obstacles may be dealt with in a similar way, using a circumscribed circular or spherical obstacle. In this way, we may simulate completely general crystal lattice structures.

7. Conclusions

We have introduced efficient algorithms to simulate periodic Lorentz gases in two and three dimensions, that work particularly well when the obstacles are small. We have compared the efficiency of these algorithms with the standard ones, showing that the relative efficiency indeed increases very fast in 2D and fast in 3D, and we have shown a sample application to calculate free flight distributions near the Boltzmann–Grad limit.

We have also shown how to extend our methods to arbitrary crystal lattices. The extension of the 3D algorithm to higher dimensions and applications are in progress.

Acknowledgments

We thank Michael Schmiedeberg for useful comments and discussions about the algorithms, and the anonymous referees for their insightful remarks. ASK received support from the DFG within the Emmy Noether program (grant Schm 2657/2). NK is the recipient of a DGAPA-UNAM postdoctoral fellowship. DPS acknowledges financial support from CONACYT grant CB-101246 and DGAPA-UNAM PAPIIT grants IN116212 and IN117214.

Appendix A. Pseudo-code for the 2D efficient algorithm

Here we give pseudo-code for the efficient algorithm for simulating the 2D periodic Lorentz gas with small circular obstacles on a square lattice.

A.1. Continued fractions

The continued fraction algorithm (algorithm 1) calculates the smallest integers k_n and h_n such that $\left|\alpha - \frac{k_n}{h_n}\right| < \delta$.

Algorithm 1. Continued fraction algorithm

```
function RATIONAL_APPROXIMATION($\alpha$, $\delta$)
    $h_1, h_2 = 1, 0$
    $k_1, k_2 = 0, 1$
    $b = \alpha$
    while $|k_1\alpha - h_1| > \delta$ do
        $a = \lfloor b \rfloor$
        $h_1, h_2 = ah_1 + h_2, h_1$
        $k_1, k_2 = ak_1 + k_2, k_1$
        $b = 1/(b - a)$
    end while
    return $k_1, h_1$
end function
```

A.2. First collision

Using the continued fraction algorithm, it is possible to efficiently calculate the center of the first obstacle with which a particle collides. First suppose that the particle has initial position $(0, b)$ with $0 < b < 1$, and its trajectory has velocity vector (v_1, v_2) with $0 < v_2 < v_1$. Let $m = v_2/v_1$ be the slope of the trajectory, which thus satisfies $0 < m < 1$. The function EFFICIENT_DISC_COLLISION(m, b, r) calculates the first disc of radius r, located at integer coordinates, with which such a trajectory collides; see algorithm 2. This is the main function required to optimize the efficiency of simulations in periodic Lorentz gases.

Algorithm 2. Function efficient_disc_collision

```
function EFFICIENT_DISC_COLLISION($m$, $b$, $r$)
    $k_n = 0$
    $b_1 = b$
    $\delta = r\sqrt{m^2 + 1}$
```

(Continued.)

```
    **if** $b < \delta$ or $(1 - b) < \delta$ **then**
        **if** $b < \frac{1}{2}$ **then**
            $q, p$ = RATIONAL_APPROXIMATION$(m, 2b)$
        **else**
            $q, p$ = RATIONAL_APPROXIMATION$(m, 2(1 - b))$
        **end if**
        $b = \mathrm{mod}(mq + b, 1)$
        $k_n = k_n + q$
    **end if**
    **while** $b > \delta$ and $1 - b > \delta$ **do**
        **if** $b < \frac{1}{2}$ **then**
            $q, p$ = RATIONAL_APPROXIMATION$(m, 2b)$
        **else**
            $q, p$ = RATIONAL_APPROXIMATION$(m, 2(1 - b))$
        **end if**
        $b = \mathrm{mod}(mq + b, 1)$
        $k_n = k_n + q$
        **if** $|b - b_1| = 0$ **then**
            exception("Particle is parallel to a channel with slope $m$")
        **end if**
    **end while**
    $q = k_n$
    $p = \mathrm{ROUND}(mq + b_1)$        ▷ $y$-component of equation $y = mx + b_1$ with $x = q$
    return $(q, p)$
**end function**
```

A.3. Starting point for efficient algorithm via local step

At each step, we need to find the correct starting point for the efficient algorithm, by moving the particle up to the first collision with a local disc *or* the next position of the form (n, b), where $n \in \mathbb{Z}$ and $b \in \mathbb{R}$, i.e. that corresponds to a position $(0, b)$ with $0 < b < 1$ under periodic boundary conditions. The function LOCAL_STEP$(\mathbf{x}, \mathbf{v}, r)$ (algorithm 3) finds the collision with one of the 4 possible local discs shown in figure A1; if there is no collision with any of these discs, then the particle is moved to the position $(n + 1, b)$, where n is the integer part of the x component of the initial position. A boolean output is also returned: 1 if there is a collision with a disc, or 0 if not.

Algorithm 3. Function local_step

```
**function** LOCAL_STEP$(\mathbf{x}, \mathbf{v}, r)$
    $\hat{\mathbf{e}}_1 = (1, 0)$ and $\hat{\mathbf{e}}_2 = (0, 1)$
    $\mathbf{n} = \lfloor \mathbf{x} \rfloor$
    $\mathbf{x} = \mathbf{x} - \mathbf{n}$                ▷ translate to unit square centered at $\left(\frac{1}{2}, \frac{1}{2}\right)$
    $t_1 = (1 - x_1)/v_1$                  ▷ time to reach position $(1, b_1)$
    $b_1 = x_2 + t_1 v_2$
    $t_2 = -x_1/v_1$;                     ▷ time (negative) to reach position $(0, b_2)$
    $b_2 = x_2 + t_2 v_2$
    $\delta = r/v_1$                       ▷ impact parameter
```

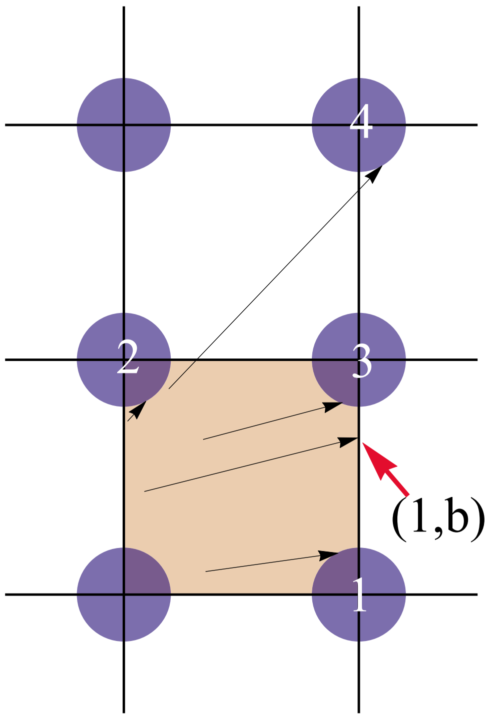

Figure A1. Possible outcomes of the function LOCAL_STEP. This function applies for velocities such that $0 < v_2 < v_1$, so that trajectories move to the right and have slope $m < 1$. Different types of trajectories are represented by arrows. The red square represents the possible positions after translation. There are 5 possible types of collision: on one of the 4 labelled discs, or at $(1, b_1)$.

(Continued.)

```
    if $(\mathbf{x} - \hat{e}_1) \cdot \mathbf{v} < 0$ then
        $|b_1| < \delta \implies$ return $\hat{\mathbf{e}}_1 + \mathbf{n}, 0$        ▷ collision with disc 1
    end if
    if $(\mathbf{x} - \hat{\mathbf{e}}_2) \cdot \mathbf{v} < 0$ then
        $|1 - b_2| < \delta \implies$ return $\hat{\mathbf{e}}_2 + \mathbf{n}, 0$        ▷ collision with disc 2
    end if
    if $(\mathbf{x} - \hat{e}_2 - \hat{e}_1) \cdot \mathbf{v} < 0$ then
        $|1 - b_1| < \delta \implies$ return $\hat{\mathbf{e}}_1 + \hat{\mathbf{e}}_2 + \mathbf{n}, 0$        ▷ collision with disc 3
    end if
    if $(\mathbf{x} - 2\hat{\mathbf{e}}_2 - \hat{\mathbf{e}}_1) \cdot \mathbf{v} < 0$ then
        $|2 - b_1| < \delta \implies$ return $\hat{\mathbf{e}}_1 + 2\hat{\mathbf{e}}_2 + \mathbf{n}, 0$        ▷ collision with disc 4
    end if
    return $(1, b_1) + \mathbf{n}, 1$        ▷ no collision with disc
end function
```

The function FIND_NEXT_DISC_FIRST_OCTANT($\mathbf{x}$, $\mathbf{v}$, $\mathbf{r}$) integrates the two previous functions to find the next disc with which a particle collides, either local or remote. This function assumes that the velocity $\mathbf{v}$ is in the first octant.

Algorithm 4. Calculate the first collision if the initial velocity is in the first octant

```
function FIND_NEXT_DISC_FIRST_OCTANT($\mathbf{x}$, $\mathbf{v}$, $r$)
    if $\|\mathrm{round}(\mathbf{x}) - \mathbf{x}\| < r$ then                         ▷ if inside obstacle
        return round($\mathbf{x}$)                         ▷ then first collision is with same obstacle
    end if
    $\mathbf{x}'$, collided $=$ LOCAL_STEP($\mathbf{x}$, $\mathbf{v}$, $r$)
    if collided $= 0$ then                                                        ▷ hit disc
        return $\mathbf{x}'$
    end if
    $m = v_2/v_1$
    $b = x_1'$
    $b = b - \lfloor b \rfloor$
    $\mathbf{c}$ = EFFICIENT_DISC_COLLISION($m$, $b$, $r$)                           ▷ find disc hit
    $\mathbf{d}$ = (0, ROUND($b$))      ▷ translation to check distance with closest obstacle, i.e. obstacle at
position (0, ROUND(b))
    $\mathbf{x}''$= ROUND($\mathbf{x}'$) $+ \mathbf{c} - \mathbf{d}$
    return $\mathbf{x}''$
end function
```

A.4. Transforming the velocity

For an arbitrary velocity, we transform the velocity to the first octant in order to use the above results, using the function TRANSFORMATION, which calculates the necessary combination of rotations and reflections to transform the velocity to the first octant. In an implementation, these would be computed once only and stored in an array.

The function FIND_NEXT_DISC then applies the respective transformation to find the next disc with which a particle with arbitrary velocity collides.

A.5. Complete simulation

Given the disc with which a collision occurs, whose centre is returned by FIND_NEXT_DISC, we need to calculate the exact position of the collision, the intersection between the straight trajectory and the disc circumference, as well as the final velocity after the collision, given by a reflection with respect to the normal vector at the point of collision. The function COLLISION ($\mathbf{x}$, $\mathbf{x}_c$, $\mathbf{v}$, r) calculates the intersection between a disc of radius r, at a position c, with a line with parametric equation $\mathbf{x}(t) = \mathbf{x} + \mathbf{v}t$. The function COLLISION_VELOCITY calculates the velocity after a collision, if the collision takes place at position $\mathbf{x}$, with an obstacle with centre $\mathbf{c}$, and initial velocity $\mathbf{v}$. These two functions are identical to those in the classical algorithm.

Algorithm 5. Given initial position $\mathbf{x}$, initial velocity $\mathbf{v}$ and radius r, finds the first collision in a periodic Lorentz gas with circular obstacles of radius r

```
function OCTANT($\mathbf{v}$)                   ▷ find octant of the velocity between 1 and 8
    $v_1, v_2 = \mathrm{v}$
    $\theta$ = ATAN2($v_2$, $v_1$)                          ▷ arctan taking into account signs
    if $\theta < 0$ then
        $\theta = \theta + 2\pi$
    end if
    return $\lceil 4\theta/\pi \rceil$
end function
```

(Continued.)

```
function TRANSFORMATION($n$)                     ▷ transformation for octant $n$ between 1 and 8
    $R_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$                ▷ rotation matrix clockwise by angle $\pi/2$
    $R_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$                ▷ reflection matrix $(x, y) \mapsto (y, x)$
    if $n$ is odd then
        return $R_1^{(n-1)/2}$                       ▷ rotate according to quadrant
    else if $n$ is even then
        return $R_2 \cdot R_1^{(n-2)/2}$              ▷ also reflect if in other octant
    end if
end function
function FIND_NEXT_DISC($\mathbf{x}$, $\mathbf{v}$, $r$)
    $n$ = OCTANT($\mathbf{v}$)
    T = TRANSFORMATION($n$)                  ▷ transformation for the given octant
    $\mathbf{x} = \mathsf{T} \cdot \mathbf{x}$
    $\mathbf{v} = \mathsf{T} \cdot \mathbf{v}$
    $\mathbf{x}$ = FIND_NEXT_DISC_FIRST_OCTANT($\mathbf{x}$, $\mathbf{v}$, $r$)
    return $\mathsf{T}^{-1} \cdot \mathbf{x}$
end function
```

Algorithm 6. Lorentz gas: given initial conditions $\mathbf{x}$ and $\mathbf{v}$, the radius of the obstacles, and the number of collisions (steps), calculates the complete trajectory

```
function COLLISION($\mathbf{x}$, $\mathbf{c}$, $\mathbf{v}$, $r$)          ▷ find collision time with given disc
    $B = [(\mathbf{x} - \mathbf{c}) \cdot \mathbf{v}]/v^2$
    $C = [(\mathbf{x} - \mathbf{c})^2 - r^2]/v^2$
    if $B^2 - C < 0$ then
      return $+\infty$                                   ▷ no collision
    end if
    $t = -B - \sqrt{B^2 - C}$
    return $t$
end function
function POST_COLLISION_VELOCITY($\mathbf{x}$, $\mathbf{c}$, $\mathbf{v}$, $r$)      ▷ velocity after collision
    $\hat{\mathbf{n}} = (\mathbf{x} - \mathbf{c})/r$
    $\mathbf{v} = \mathbf{v} - 2(\mathbf{v} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$
    $\mathbf{v} = \mathbf{v}/\| \mathbf{v} \|$
    return $\mathbf{v}$
end function
function LORENTZGAS($\mathbf{x}$, $\mathbf{v}$, $r$, steps)
    for $i$ = 1: steps do
        $\mathbf{c}$ = FIND_NEXT_DISC($\mathbf{x}$, $\mathbf{v}$, $r$)
        $t$ = COLLISION($\mathbf{x}$, $\mathbf{c}$, $\mathbf{v}$, $r$)
        $\mathbf{x} = \mathbf{x} + \mathbf{v}t$
        $\mathbf{v}$ = POST_COLLISION_VELOCITY($\mathbf{x}$, $\mathbf{c}$, $\mathbf{v}$, $r$)
    end for
end function
```

Note that in the efficient algorithm, COLLISION is called only after it is known with which disc the collision will occur, and thus is guaranteed to find a collision. However, the case where no collision occurs often occurs in the classical algorithm and in the efficient 3D algorithm.

Appendix B. Pseudocode for the 3D efficient algorithm

The 3D version of the efficient algorithm calculates the 2D collisions in each of the 3 coordinate planes *x–y*, *y–z* and *x–z*. We project the velocity vector **v** onto each planes, and normalize. Using the 2D efficient algorithm, we calculate the first disc collision in each plane and the corresponding time required to reach the obstacle, and take the maximum of the three. We check if the three collisions correspond to the same obstacle in 3D space. If not, we move to the furthest obstacle and continue. Finally, if the three 2D collisions do correspond to the same obstacle, we use the function COLLISION to check if the collision is a true collision with the corresponding sphere.

Algorithm 7. Function Lorentz3D: Find the next collision with an obstacle in 3D

```
function LORENTZ3D($\mathbf{x}$, $\mathbf{v}$, $r$)
    $\Pi_i$ := plane with normal vector $\hat{e}_i$   ($i = 1, 2, 3$)
    for $i \in \{1, 2, 3\}$ do
        for $j \in \{1, 2\}$ do
            $(\mathsf{T}_i)_{jk} = \begin{cases} 0 & \text{if } k = i \\ \delta_{jk} & \text{if } k < i \\ \delta_{j,k-1} & \text{if } k > i \end{cases}$
        end for                                        ▷ $\mathsf{T}_i$ is the identity matrix without row $i$
    end for                                            ▷ $\mathsf{T}_i$ projects to plane $\Pi_i$
    while true do
        for $i \in \{1, 2, 3\}$ do
            $\mathbf{v}_i = \mathsf{T}_i \cdot \mathbf{v}$             ▷ 2D projected velocity in plane $\Pi_i$
            $\mathbf{u}_i = \mathbf{v}_i / \| \mathbf{v}_i \|$        ▷ normalized projected velocity
            $\mathbf{c}_i = \binom{i}{i}$         ▷ initialise coordinates of 3 distinct obstacles in plane
        end for
        while $(\mathbf{c}_1)_1 \neq (\mathbf{c}_2)_1$ or $(\mathbf{c}_1)_2 \neq (\mathbf{c}_3)_1$ or $(\mathbf{c}_2)_2 \neq (\mathbf{c}_3)_2$ do
            for $i \in \{1, 2, 3\}$ do
                $\mathbf{x}_i = \mathsf{T}_i \cdot \mathbf{x}$                 ▷ projected coordinates in plane $\Pi_i$
                $\mathbf{c}_i$ = LORENTZ2D($\mathbf{x}_i$, $\mathbf{u}_i$, $r$)          ▷ disc hit in plane $\Pi_i$
                $t_i = (\| \mathbf{c}_i - \mathbf{x}_i \| - r)/\| \mathbf{v}_i \|$   ▷ approximate collision time in plane $\Pi_i$
            end for
            $t_{\max} = \max(t_1, t_2, t_3)$
            if $t_{\max} < 0$ then     ▷ particle in 3 planes collides with obstacle nearest to particle
                $t_{\max} = (\sqrt{2} - 1)r/2$     ▷ maximal distance that particle on boundary of sphere
needs to move back such that at least in 2 planes it is outside the obstacles
            end if
            $\mathbf{x} = \mathbf{x} + \mathbf{v}(t_{\max} - (\sqrt{2} - 1)r/2)$       ▷ move close to the furthest of the 3 possible
cylinders
            end while
        $t'$ = COLLISION($\mathbf{x}$, [$\mathbf{x}$], $r$, $\mathbf{v}$)     ▷ if obstacles coincide, calculate if true collision; return +
$\infty$ if not
            if $t < \infty$ then
                break
            end if
```

(Continued.)

```
        $\mathbf{x} = \mathbf{x} + \mathbf{v}(1 - 2r)$   $\triangleright$ if no collision, move distance $(1 - 2r)$ forwards, such that the collision candidate is left behind, but no other obstacle is crossed
    **end while**
    $\mathbf{x} = \mathbf{x} + t'\mathbf{v}$
    return $\mathbf{x}$
**end function**
```

References

[1] Cvitanović P, Gaspard P and Schreiber T 1992 Investigation of the Lorentz gas in terms of periodic orbits *Chaos* **2** 85–90

[2] Latz A, Van Beijeren H and Dorfman J R 1997 Lyapunov spectrum and the conjugate pairing rule for a thermostatted random Lorentz gas: kinetic theory *Phys. Rev. Lett.* **78** 207

[3] Dellago C and Posch H A 1997 Lyapunov spectrum and the conjugate pairing rule for a thermostatted random Lorentz gas: numerical simulations *Phys. Rev. Lett.* **78** 211

[4] Beijeren H Van, Arnulf Latz and Dorfman J R 1998 Chaotic properties of dilute two-and three-dimensional random Lorentz gases: equilibrium systems *Phys. Rev.* E **57** 4077

[5] Kraemer Atahualpa S and David P Sanders 2013 Embedding quasicrystals in a periodic cell: dynamics in quasiperiodic structures *Phys. Rev. Lett.* **111** 125501

[6] Wennberg B 2012 Free path lengths in quasi crystals *J. Stat. Phys.* **147** 981–90

[7] Bunimovich Leonid A and Sinai Ya G 1981 Statistical properties of Lorentz gas with periodic configuration of scatterers *Commun. Math. Phys.* **78** 479–97

[8] Bleher P M 1992 Statistical properties of two-dimensional periodic Lorentz gas with infinite horizon *J. Stat. Phys.* **66** 315–73

[9] Chernov N I 1994 Statistical properties of the periodic Lorentz gas. Multidimensional case *J. Stat. Phys.* **74** 11–53

[10] Gilbert T, Nguyen H C and Sanders D P 2011 Diffusive properties of persistent walks on cubic lattices with application to periodic Lorentz gases *J. Phys. A: Math. Theor.* **44** 065001

[11] Caglioti E and Golse F 2003 On the distribution of free path lengths for the periodic Lorentz gas III *Commun. Math. Phys.* **236** 199–221

[12] Golse F 2012 Recent results on the periodic Lorentz gas *Nonlinear Partial Differential Equations* (Berlin: Springer) pp 39–99

[13] Boca F P and Zaharescu A 2007 The distribution of the free path lengths in the periodic two-dimensional Lorentz gas in the small-scatterer limit *Commun. Math. Phys.* **269** 425–71

[14] Golse F 2006 The periodic Lorentz gas in the Boltzmann–Grad limit *Proc. ICM* (*Madrid, 2006*) vol 3, pp 183–20

[15] Caglioti E and Golse F 2008 The Boltzmann–Grad limit of the periodic Lorentz gas in two space dimensions *Comp. Rend. Math.* **346** 477–82

[16] Caglioti E and Golse F 2010 On the Boltzmann–Grad limit for the two dimensional periodic Lorentz gas *J. Stat. Phys.* **141** 264–317

[17] Golse F and Wennberg B 2000 On the distribution of free path lengths for the periodic Lorentz gas: II *ESAIM: Math. Modelling Numer. Anal.* **34** 1151–63

[18] Marklof J and Strömbergsson A 2008 Kinetic transport in the two-dimensional periodic Lorentz gas *Nonlinearity* **21** 1413

[19] Bourgain J, Golse F and Wennberg B 1998 On the distribution of free path lengths for the periodic Lorentz gas *Commun. Math. Phys.* **190** 491–508

[20] Gilbert T and Sanders D P 2009 Persistence effects in deterministic diffusion *Phys. Rev.* E **80** 041121

[21] Marklof J and Strömbergsson A 2011 The periodic Lorentz gas in the Boltzmann-Grad limit: asymptotic estimates *Geom. Funct. Anal.* **21** 560–647

[22] Nándori P, Szász D and Varjú T 2014 Tail asymptotics of free path lengths for the periodic Lorentz process: on Dettmann's geometric conjectures *Commun. Math. Phys.* **331** 111–37

[23] Dettmann C P 2012 New horizons in multidimensional diffusion: the Lorentz gas and the Riemann hypothesis *J. Stat. Phys.* **146** 181–204

[24] Sanders D P 2005 Fine structure of distributions and central limit theorem in diffusive billiards *Phys. Rev.* E **71** 016220

[25] Sanders D P 2008 Normal diffusion in crystal structures and higher-dimensional billiard models with gaps *Phys. Rev.* E **78** 060101
[26] Zacherl A, Geisel T, Nierwetberg J and Radons G 1986 Power spectra for anomalous diffusion in the extended Sinai billiard *Phys. Lett.* A **114** 317–21
[27] Marklof J and Strömbergsson A 2014 Power-law distributions for the free path length in Lorentz gases *J. Stat. Phys.* **155** 1072–86
[28] Khanin K, Dias J and Marklof J 2007 Multidimensional continued fractions, dynamical renormalization and Kam theory *Commun. Math. Phys.* **270** 197–231
[29] Lagarias J C 1994 Geodesic multidimensional continued fractions *Proc. London Math. Soc.* (*3*) **69** 464–88
[30] Niven I, Zuckerman H S and Montgomery H L 2008 *An Introduction to the Theory of Numbers* (New York: Wiley)
[31] Nogueira A 1995 The three-dimensional poincaré continued fraction algorithm *Isr. J. Math.* **90** 373–401
[32] http://julialang.org/
[33] Bezanson J, Edelman A, Karpinski S and Shah V B 2014 Julia: A fresh approach to numerical computing arXiv:1411.1607
[34] Moore M 1974 Symmetrical intersections of right circular cylinders *Math. Gaz.* **58** 181–5
[35] Chernov N 1997 Entropy, Lyapunov exponents, and mean free path for billiards *J. Stat. Phys.* **88** 1–29
[36] Ashcroft N W and Mermin N D 1976 *Solid State Physics* (Philadelphia: Saunders College)
[37] Trefethen L N and Bau D 1997 *Numerical Linear Algebra* (Philadelphia, PA: Society for Industrial and Applied Mathematics)