Efficient algorithms for the periodic Lorentz gas in two and three dimensions

Atahualpa S. Kraemer[1], **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`

Abstract.

We present efficient algorithms to calculate collisions for a periodic Lorentz gas in two and three dimensions. The 2D algorithm uses continued fractions to obtain the exact disc with which a particle will collide at each step, instead of using periodic boundary conditions. The 3D version uses the 2D algorithm by projecting to coordinate planes. We study the efficiency of the algorithms, and, as an application, calculate distributions of free paths for three different geometries.

1. Introduction

Lorentz gases are simple physical systems that present deterministic chaos [**?**], 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.

These systems can have different configurations of obstacles, e.g., random arrangements [**?**, **?**, **?**] or quasiperiodic structure [**?**, **?**]. However, due to its simplicity, the periodic case has been most widely studied; see, e.g., [**?**, **?**, **?**, **?**]. In this case, the model is equivalent to a Sinai billiard [**?**]. 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 [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. There are still many interesting open questions in this area [**?**, **?**, **?**, **?**].

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[**?**, **?**]. 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 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 [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. An algorithm along these lines was previously developed: see comments in [**?**]; 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 [**?**, **?**].

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 quasiperiodic arrays, two incommensurate overlapping arrays of square lattices, or with different shapes of obstacles; such systems may produce a number of surprising effects [**?**]. On the other hand, due to the construction of the algorithm, it is not possible to use it in higher dimensions, which, to quote Golse, is "a notoriously more difficult problem".

In this paper, we develop a general, 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 to develop an algorithm for a 3D simple cubic lattice by projecting to coordinate planes.

2. Continued fractions

In this section, we derive the basis of the 2D algorithm using continued fractions. Its goal is to calculate efficiently where the first collision of a particle will occur in a square lattice. Without loss of generality, we will use the lattice formed by the integer coordinates in the 2D plane. We wish to calculate the minimal time $t^* > 0$, such that a collision occurs with some disc with centre $\vec{c} = (q, p)$, with q and p integers. The condition for a collision is then

$$\|\vec{x}_0 + \vec{v}_0 t^* - \vec{c}\| = r, \tag{1}$$

where r is the radius of the obstacle, and $\vec{x}_0$ and $\vec{v}_0$ are the initial position and velocity of the particle. The collision then occurs at the point $\vec{x}_0 + \vec{v}_0 t^*$.

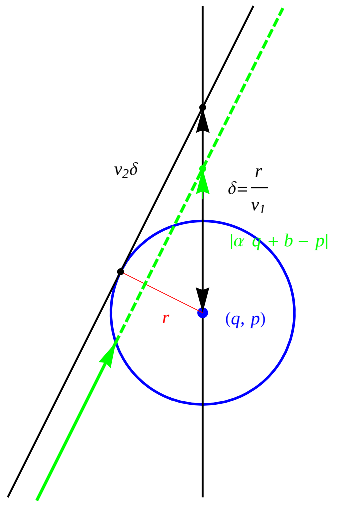

Figure 1. Relation between the intersection of a line and a circle with integer coordinates and the intersection of the line $x = q$.

2.1. 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 (**??**) for t^*. A first improvement follows from observing that we may instead look for the intersection with another line, as follows. We denote by (v_1, v_2) the components of the particle's velocity. Without loss of generality, in the following we take the speed $\sqrt{v_1^2 + v_2^2}$ to be 1 and $v_1 > 0$ and $v_2 > 0$; we can always rotate or reflect the system such that these conditions are satisfied, due to its symmetry (see below).

We write the equation of the particle's trajectory as $y = \alpha x + b$ and look for its intersection with the vertical line $x = q$. As shown in figure **??**, if $|\alpha q + b - p| = |b| < \delta := r/v_1$, then a collision will take place. Due to the periodic boundary conditions, we can always take $0 < b < 1$, so we need only solve $b < \delta$.

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

$$|\{\alpha q_n\} + b - 1| < \delta, \tag{2}$$

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

$$|\alpha q_n - p_n + b| < \delta, \tag{3}$$

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 $|\alpha q_i - p_i + b|$ 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

$$|\alpha q_i - p_i + b| < |\alpha q_{i-1} - p_{i-1} + b| \tag{4}$$

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 $|\alpha q_0 - p_0 + b|$, that is

$$|\alpha q_1 - p_1 + b| < f(b) = b, \quad \text{if } b < 1/2 \tag{5}$$
$$1 - b, \quad \text{if } b > 1/2. \tag{6}$$

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 (**??**), we obtain that indeed $p_1 = \lfloor \alpha q_1 \rfloor + 1$. Iterating the inequality (**??**) we obtain

$$p_n = \lfloor \alpha q_n \rfloor + 1. \tag{7}$$

Combining the inequality (**??**) with equation (**??**), we obtain again equation (**??**).

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 α cells. And we do not need to apply periodic boundary conditions until we reach the obstacle.

2.2. The Diophantine inequality: $|\alpha p - q| \leq \varepsilon$

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

In order to do this, we can use the continued fraction algorithm to obtain solutions to the inequality $|\alpha q - p| \leq \varepsilon$. This algorithm already gives a sequence of (q_n, p_n) such that $|\alpha q_i - p_i| < |\alpha q_{i-1} - p_{i-1}|$ if $q_{i-1} < q_i$. So, if we turn our inequality (**??**) into this other inequality, we will find our algorithm just by using the continued fraction algorithm. Indeed, using equation **??** and the inequality **??**, we obtain $|\{\alpha q_1\} - 1| < 2b$ 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 $\lfloor \alpha q_1 \rfloor + 1$.

$$|\alpha q - p| < 2b, \quad \text{if } b < 1/2 \tag{8}$$
$$2(1-b), \quad \text{if } b > 1/2 \tag{9}$$

Now we can apply the continued fraction algorithm to obtain p_1 and q_1 of inequality (**??**). 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 $p \geq p_1$, and $q \geq q_1$. Hence, we can just use (q_1, p_1) even if they do not satisfy inequality **??**. 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).

3. Complete 2D algorithm

We now have the necessary tools to implement the algorithm. The source code may be found in the supplementary information.

We call `obstacle(`α`, b)` the function that gives the centre of an obstacle with which a particle collides that has initial position $(0, b)$ and positive velocity with slope $0 < \alpha < 1$.

This function uses the algorithm mentioned above. However, in general the particle's velocity does not satisfy these conditions, and will need to be converted into this form.

Suppose first that we do have the good conditions on the velocity. Then we use the classical algorithm for one step, or at most two steps. In the classical algorithm, collisions with an obstacle inside a periodic cell (a unit square) are calculated. If there is no collision, then the particle moves to the next wall of the cell and periodic boundary conditions are applied. Due to the fact that the slope satisfies $0 < \alpha < 1$, after a maximum of 2 steps, the classical algorithm will return the position of a collision or a coordinate of the form (n, b), that may be rewritten as $(0, b)$ when periodic boundaries conditions are applied. So, if we use first the classical algorithm, and then the function `obstacle(`α`, b)`, we obtain the coordinates of the obstacle of the next collision.

Now we need only to transform the velocity into the correct form. To do so, 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. We need a transformation R that converts the velocity into a positive velocity with slope $0 < \alpha < 1$. Then, we redefine $v \to \mathsf{R} \cdot v$ and $x \to \mathsf{R} \cdot x$ and apply the efficient algorithm described above to obtain the position of the obstacle x_c. Finally, we apply the inverse of R to obtain the final values: $v \to \mathsf{R}^{-1} \cdot v$, $x \to \mathsf{R}^{-1} \cdot x$ and $x_c \to \mathsf{R}^{-1} \cdot x_c$. Here, R is a transformation that depends on the velocity.

If $|\alpha| > 1$, we reflect using the reflection matrix $\mathsf{M}_{\text{refl}} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. If $v_x < 0$ and $v_y > 0$, or $v_x < 0$ and $v_y < 0$, or $v_x > 0$ and $v_y < 0$, we rotate the through an angle $\pi/2$, π or $3\pi/2$, respectively, using the rotation matrix $\mathsf{M}_{\text{rot}}(\theta) := \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$. So, for example, if $v_y < v_x < 0$, then $\mathsf{R} = \mathsf{M}_{\text{refl}} \cdot M_{\text{rot}}(\pi) = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$.

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

4. 3D algorithm

We now turn to an efficient algorithm for calculating the next collision with a sphere in 3D on a simple cubic lattice. The algorithm works by projecting to 2D planes and using the 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, but with a speed in that plane that is, in general, not 1. Thus, we can apply the 2D algorithm for each of the 3 projections, and obtain the coordinates of 3 obstacles in the 3 projected planes where the first collision occurs.

We now check whether the obstacle coordinates in these projections correspond to the *same* 3D obstacle. If not, then we move the particle to the cell containing the obstacle that is furthest away (i.e., has the maximum collision time), and continue.

If the obstacle coordinates coincide correctly, i.e. the integer obstacle coordinates returned by the 2D algorithm for each plane coincide pairwise, then our 3D algorithm predicts that there is a collision. However, this may not be the case, due to the geometry, as follows.

Using the 2D continued fraction algorithm in one of the planes xy, xz and yz is equivalent to calculating the first collision with a cylinder orthogonal to that plane. Joining those coordinates together is equivalent to calculating a collision with the intersection of three orthogonal cylinders with the same radius. Figure **??** shows such an intersection of three