Appendix A. Appendice: Explicit code

In this section we present the pseudo code to program efficient algorithm of the 2D periodic Lorentz gas. The Full packet it is made by 8 functions: $Frac(\alpha,\varepsilon)$ (see algorithm 1), $Eff(m,b,r)$ (see algorithm 2), $Next(\vec{x},\vec{v},r)$ (see algorithm 3), $Collision(\vec{x_0},\vec{x_c},\vec{v},r)$ (see algorithm 4), $Vel_{col}(\vec{x_0},\vec{x_c},\vec{v})$ (see algorithm 5), $Lor(\vec{x},\vec{v},\vec{r})$ (see algorithm 6), $Lorentz2D(\vec{x},\vec{v},r)$ (see algorithm 7), and $LorentzGas(\vec{x},\vec{v},r,steps)$ (see algorithm **??**).

The algorithm 1 is the continued fraction algorithm that will be used to calculate efficiently the first collision. This function calculates the smallest integers k_n and h_n such that $|\alpha - \frac{k_n}{h_n}| < \varepsilon$, for a given ε and α using the continued fraction algorithm.

Algorithm 1 Continued fraction algorithm

```
function FRAC(α, ε)
    h1, h2 = 1, 0
    k1, k2 = 0, 1
    b = α
    while |k1 α − h1| > ε do
        a = ⌊b⌋
        h1, h2 = a h1 + h2, h1
        k1, k2 = a k1 + k2, k1
        b = 1/(b − a)
    end while
     return k1, h1
end function
```

Using this algorithm it is possible to calculate efficiently the center of the first obstacle with radius r which a particle that has a initial position $(0,b)$ with $0<b<1$, and its trajectory is on a line with slope $0<m<1$ with both components of the velocity of the particle are positive, collides. We call the function to calculate this $Eff(m,b,r)$, and it is shown in the algorithm 2. This is the main function in this paper to optimize the efficiency of the simulations in the periodic Lorentz gases.

Using rotations, it is possible to use the algorithm 2 for any velocity, however, it is still needed to move the particle up to the next position of the form (n,b), where n is a integer number, and b is a real number. Because of the periodic boundary conditions, this is equivalent to a position of the form $(0,b)$ with $0<b<1$. The function $Next(\vec{x},\vec{v},r)$ (algorithm 3) calculates the first of the next possibilities: The intersection of a particle at a initial position $\vec{x}$ and velocity $\vec{v}$, with a line of the form (n,b), where n is an integer and b is a variable, or the center of the obstacle with which the particle collides. This center can only be one of the followings: $\left\lfloor \vec{x}+\vec{0.5} \right\rfloor$, or $\left\lfloor \vec{x}+\vec{0.5} \right\rfloor + \hat{e}_1$, or $\left\lfloor \vec{x}+\vec{0.5} \right\rfloor + \hat{e}_2$, or $\left\lfloor \vec{x}+\vec{0.5} \right\rfloor + \hat{e}_1 + \hat{e}_2$. Here $\vec{0.5} = (0.5, 0.5)$. The returned variable $test$ is 1 if there is not collision and is equal to 0 if there is a collision.

In the Lorentz gas, the simulations are on hard disk, so, the we need calculate the exact position of the collision, which is the intersection between a straight line and a circle, as well as the final velocity after the collision, which is simply a reflection. The function $Collision(\vec{x_0},\vec{x_c},\vec{v},r)$ calculates the intersection between a disc of radius r, at a position x_c, with a line with parametric equation $\vec{x} = \vec{x_0} + \vec{v}t$.

The function vel_{col}, calculates the velocity after a collision, if the collision takes place at position $\vec{x_0}$, with an obstacle with centre $\vec{x_c}$, and initial velocity $\vec{v}$.

Algorithm 2 Function Eff

```
function EFF(m, b, r)
    k_n = 0
    b_1 = b
    ε = r√(m² + 1)
    if b < ε or (1 − b) < ε then
        if b < 0.5 then
            q, p = Frac(m, 2b)
        else
            q, p = Frac(m, 2(1 − b))
        end if
        b = mod(mq + b, 1)
        kn = q + 1
    end if
    while b > ε and 1 − b > ε do
        if b < 0.5 then
            (q, p) = Frac(m, 2b)
        else
            (q, p) = Frac(m, 2(1 − b))
        end if
        b = mod(mq + b, 1)
        k_n = q + 1
        if |b − b_1| < 0 then
            return false
        end if
    end while
    q = k_n
    p = ⌊mq + b_1 + 0.5⌋
    return (q, p)
end function
```

Algorithm 3 Function Next

function NEXT($\vec{x}$,$\vec{v}$,r)
  $\hat{e}_1 = (1,0)$
  $\hat{e}_2 = (0,1)$
  $\vec{n} = \lfloor x \rfloor$
  $\vec{x}' = \vec{x} - \vec{n}$
  $t = (1 - x'_1)/v_1$
  $t_2 = -x'_/ v_1$
  $\vec{x}'' = \vec{x}' + \vec{v}t$
  $\vec{x}''' = \vec{x}' + \vec{v}t_2$
  $b_1 = x''_2$
  $b_2 = x'''_2$
  $\varepsilon = r/v_1$
  $test = 0$
  if $(\vec{x}' - \hat{e}_1) \cdot \vec{v} < 0$ **then**
    if $|b_1| < \varepsilon$ **then**
      return $\hat{e}_1 + \vec{n}, 0$
    end if
  end if
  if $\vec{x}' - \hat{e}_2) \cdot \vec{v} < 0$ **then**
    if $|1 - b_2| < \varepsilon$ **then**
      return $\hat{e}_2 + \vec{n}, 0$
    end if
  end if
  if $\vec{x}' - \hat{e}_2 - \hat{e}_1) \cdot \vec{v} < 0$ **then**
    if $|1 - b_1| < \varepsilon$ **then**
      return $\hat{e}_1 + \hat{e}_2 + \vec{n}, 0$
    end if
  end if
  if $\vec{x}' - \hat{e}_2 - \hat{e}_2 - \hat{e}_1) \cdot \vec{v} < 0$ **then**
    if $|2 - b_1| < \varepsilon$ **then**
      return $\hat{e}_1 + \hat{e}_2 + \hat{e}_2 + \vec{n}, 0$
    end if
  end if
  $test = 1$
  return $\vec{x}'' + \vec{n}, test$
end function

Algorithm 4 Intersection between a line with parametric equation $\vec{x} = \vec{x_0} + \vec{v}t$ an a circle of radius r, center x_c

function COLLISION($\vec{x_0}$, $\vec{x_c}$, $\vec{v}$,r)
    $b = \frac{(\vec{x_0}-\vec{x_c})\cdot\vec{v}}{\vec{v}^2}$
    $c = \frac{(\vec{x_0}-\vec{x_c})^2-r^2}{\vec{v}^2}$
    if $b^2 - c < 0$ **then**
        return "false"
    end if
    $t = -b - \sqrt{b^2 - c}$
    $\vec{x} = \vec{v}t + \vec{x_0}$
    return $\vec{x}$
end function

Algorithm 5 Resulting velocity after a collision at the point x_0 of particle with initial velocity $\vec{v}$ and a disk with center x_c

function $vel_{col}(\vec{x_0},\vec{x_c},\vec{v})$
    $\hat{n} = \frac{\vec{x_0}-\vec{x_c}}{||\vec{x_0}-\vec{x_c}||}$
    $\vec{v_n} = (\hat{n}\cdot\vec{v})\hat{n}$
    $\vec{v} = \vec{v} - 2\vec{v_n}$
    $\vec{v} = \frac{\vec{v}}{||\vec{v}||}$
    return $\vec{v}$
end function

Algorithm 6 Integrate the functions Eff and $Next$ in one function that calculates the first collision in a Lorentz gas, if the initial velocity $\vec{v}$ is positive and $v_1 > v_2$

function LOR($\vec{x}$,$\vec{v}$,$\vec{r}$)
    $\vec{x'}, test = Next(\vec{x}, \vec{v}, r)$
    if $test = 0$ **then**
        return $\vec{x'}$
    end if
    $m = v_2/v_1$
    $b = x'_1$
    $b = b - \lfloor b \rfloor$
    $\vec{d} = [0, int(b)]$
    $\vec{c} = Eff(m, b, r)$
    if c=false **then**
        return false
    end if
    $x'' = \lfloor x' + 0.5 \rfloor - \vec{d} + \vec{c}$
    return x''
end function

Algorithm 7 Given the initial position $\vec{x}$, the initial velocity $\vec{v}$ and the radius r of the obstacles, this algorithm finds the first collision in a periodic Lorentz gas.

function LORENTZ2D($\vec{x}$,$\vec{v}$,r)

$$ROT = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

#Rotational matrix $\pi/2$ radians

$$REF = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

#Reflection matrix, change $(x,y) \to (y,x)$

$v' = v$

$x' = x$

$m_1 = v_2/v_1$

if $|| \lfloor (\rfloor \vec{x} + \vec{0.5}) - \vec{x} || < r$ **then**

#if a particle begin inside an obstacle, then the first collision is considered with the same obstacle.

return $\lfloor \vec{x} \rfloor$

end if

if $m_1 > 0$ and $v_2 > 0$ **then**

if the velocity is in the quadrant I

if $m_1 < 1$ **then**

$x' = Lor(x', v', r)$

if $x' = false$ **then**

return false

end if

else if $m_1 > 1$ **then**

$x' = REFx'$

$v' = REFv'$

$x' = Lor(x', v', r)$

if $x' = false$ **then**

return false

end if

$x' = REFx'$

$v' = REFv'$

end if

return x'

else if $m_1 > 0$ and $v_2 < 0$ **then**

#if the velocity is in the quadrant III

$x' = ROT^2 x'$

$v' = ROT^2 v'$

if $m_1 < 1$ **then**

$x' = Lor(x', v', r)$

if $x' = false$ **then**

return false

end if

else if $m_1 > 1$ **then**

$x' = REFx'$

$v' = REFv'$

$x' = Lor(x', v', r)$

if $x' = false$ **then**

return false

end if

$x' = REFx'$

$v' = REFv'$

end if

$x' = ROT^2 x'$

$v' = ROT^2 v'$

return x'

```
    else if m_1 < 0 and v_2 > 0 then
        #if the velocity is in the quadrant II
        x' = ROT x'
        v' = ROT v'
        if m_1 < -1 then
            x' = Lor(x', v', r)
            if x' = false then
                return false
            end if
        else if m_1 > -1 then
            x' = REF x'
            v' = REF v'
            x' = Lor(x', v', r)
            if x' = false then
                return false
            end if
            x' = REF x'
            v' = REF v'
        end if
        x' = ROT^3 x'
        v' = ROT^3 v'
        return x'
    else if m_1 < 0 and v_2 < 0 then
        #if the velocity is in the quadrant IV
        x' = ROT^3 x'
        v' = ROT^3 v'
        if m_1 < -1 then
            x' = Lor(x', v', r)
            if x' = false then
                return false
            end if
        else if m_1 > -1 then
            x' = REF x'
            v' = REF v'
            x' = Lor(x', v', r)
            if x' = false then
                return false
            end if
            x' = REF x'
            v' = REF v'
        end if
        x' = ROT x'
        v' = ROT^3 v'
        return x'
    end if
end function
```

Algorithm 8 Lorentz gas model: given initial conditions $\vec{x}$ and $\vec{v}$, the radius of the obstacles, and the number of collisions $steps$, this function calculates the final position and velocity

```
function LORENTZGAS(x⃗, v⃗, r, steps)
    for i = 1 : steps do
        c⃗ = Lorentz(x⃗, v⃗, r)
        x⃗, t = Collision(x⃗, c⃗, r, v)
        v⃗ = vel_col(x⃗, c⃗, v⃗)
    end for
    return x⃗, v⃗
end function
```