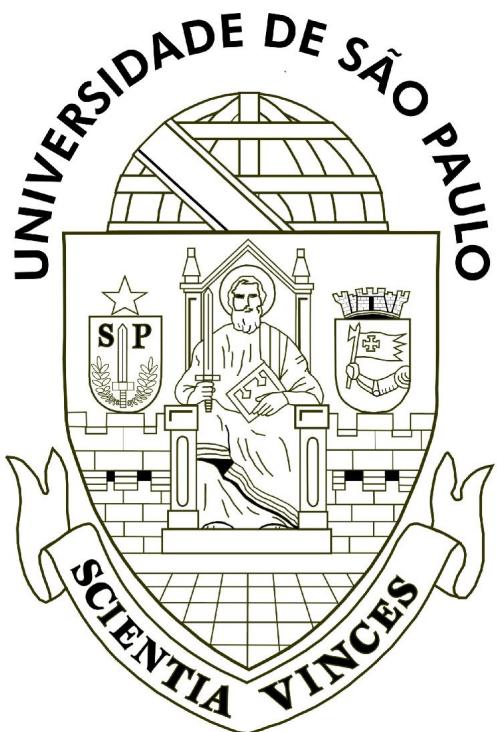


Física estatística computacional



DINÂMICA MOLECULAR

Aluno

Leonardo Vaz Ferreira n° 13862330

São Carlos
2025

1 Tarefa 1

Nesse projeto foi proposto a criação de um programa em fortran para analisar a dinâmica molecular de um sistema de N partículas interagentes de acordo com o potencial de Lennard-Jones.

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right] \quad (1)$$

Para a criação da simulação as calcularemos as forças de interação de partículas próximas a uma certa distância, dessa maneira conseguiremos criar uma rede de interação entre as partículas. Caso as partículas passem do comprimento L do sistema serão impostas condições periódicas de contorno, para que dessa maneira as partículas não escapem do sistema. E para a atualização das posições das partículas utilizaremos o método iterativo de Verlet.

$$t = n\Delta t, \quad (2)$$

$$x_i(n+1) = 2x_i(n) - x_i(n-1) + a_{i,x}(n)(\Delta t)^2, \quad n \geq 1, \quad (3)$$

$$y_i(n+1) = 2y_i(n) - y_i(n-1) + a_{i,y}(n)(\Delta t)^2, \quad n \geq 1, \quad (4)$$

$$x_i(1) = x_i(0) + v_{i,x}(0)\Delta t, \quad (5)$$

$$y_i(1) = y_i(0) + v_{i,y}(0)\Delta t, \quad (6)$$

$$v_{i,x}(n) = \frac{x_i(n+1) - x_i(n-1)}{2\Delta t}, \quad (7)$$

$$v_{i,y}(n) = \frac{y_i(n+1) - y_i(n-1)}{2\Delta t}. \quad (8)$$

Dessa maneira foi criado o seguinte código:

```

1 program tarefaA
2 integer, parameter:: n = 28, L = 10
3 real, parameter:: v_0 = 1.0d0, dt = 0.01d0, t_max = 50.0d0
4 real,dimension(n,2):: cord, prev_cord, v, a, temp, vel
5 real,dimension(n,2):: epsilon = 1.0d-20
6 integer:: i, j, k
7 integer:: t
8
9 open(unit=18, file='positions.out')
10 open(unit=28, file='velocities.out')
11
12 t = 0.0d0
13
14 call init_positions(cord, v, n, L, dt, prev_cord)
15
16 do while (t<=t_max)
17   t = t + dt
18
19   temp = cord
20
21   call calculate_forces(cord, a, n)
22   call update_positions(cord, prev_cord, a, n, dt)
23   call border_conditions(cord, n, prev_cord, L)
24
25   prev_cord = temp
26
27   do i = 1, n
28     write(18,*), cord(i,1), cord(i,2)
29     vel(i,1) = delta_periodic(cord(i,1) - prev_cord(i,1), real(L,8)) / (2.0d0*dt)
30     vel(i,2) = delta_periodic(cord(i,2) - prev_cord(i,2), real(L,8)) / (2.0d0*dt)
31   end do
32
33   if (mod(t, 28.0d0*dt) < epsilon) then
34     do i = 1, n
35       write(28,*), vel(i,1), vel(i,2), sqrt(vel(i,1)**2 + vel(i,2)**2)
36     end do
37   end if
38
39   end do
40
41 contains
42
43 function delta_periodic(d, l) result(dp)
44   integer, intent(in) :: d, l
45   real,dimension(2):: dp
46   dp = d - dnint(d / l) * l
47   end function
48
49 subroutine init_positions(cord, v, n, L, dt, prev_cord)
50   integer, intent(in) :: n, L
51   real,dimension(2):: v
52   real,dimension(2):: prev_cord
53   integer :: i, ix, iy
54   integer, parameter :: nx = 4, ny = 5
55   real,dimension(2):: ox, oy, theta, rand1, rand2
56   real,dimension(2):: dx, dy
57
58   dx = real(L) / real(nx)
59   dy = real(L) / real(ny)
60
61   i = 0
62   do iy = 0, ny-1
63     do ix = 0, nx-1
64       i = i + 1
65       if (i > n) exit
66
67       cord(i,1) = (ix + 0.5d0) * dx
68       cord(i,2) = (iy + 0.5d0) * dy
69
70       call random_number(rand1)
71       call random_number(rand2)
72       cord(i,1) = cord(i,1) + (rand1 - 0.5d0) * dx + 0.2d0
73       cord(i,2) = cord(i,2) + (rand2 - 0.5d0) * dy + 0.2d0
74
75       prev_cord = cord
76
77       call random_number(theta)
78       theta = theta + 2.0d0 * 3.141592653589793d0
79       v(i,1) = v_0 * cos(theta)
80       v(i,2) = v_0 * sin(theta)
81
82       write(18,*), i, cord(i,1), cord(i,2)
83     end do
84   end do
85
86   do i = 1, n
87     cord(i,1) = mod(cord(i,1) + v(i,1) * dt, real(L))
88     cord(i,2) = mod(cord(i,2) + v(i,2) * dt, real(L))
89     write(18,*), i, cord(i,1), cord(i,2)
90   end do
91
92 end subroutine init_positions
93
94 subroutine border_conditions(cord, n, prev_cord, l)
95   integer, intent(in) :: n, L
96   real,dimension(2):: prev_cord
97   integer:: i
98
99   do i = 1, n
100     cord(i,1) = modulo(cord(i,1), real(L))
101     cord(i,2) = modulo(cord(i,2), real(L))
102
103     prev_cord(i,:) = cord(i,:)
104   end do
105
106 end subroutine border_conditions
107
107 subroutine update_positions(cord, prev_cord, a, n, dt)
108   integer:: n, i
109   real,dimension(n,2):: cord, prev_cord, a
110   real,dimension(n,2):: dt
111
112   do i = 1, n
113     cord(i,1) = 2.0d0*cord(i,1) - prev_cord(i,1) + a(i,1)*dt*t**2
114     cord(i,2) = 2.0d0*cord(i,2) - prev_cord(i,2) + a(i,2)*dt*t**2
115   end do
116
117 end subroutine update_positions
118
118 subroutine calculate_forces(cord, a, n)
119   integer:: n
120   real,dimension(n,2):: cord, a
121   real,dimension(n,2):: r, f, dx, dy
122   integer:: i, j
123
124   a = 0.0d0
125
126   do i = 1, n
127     do j = i+1, n
128       dx = delta_periodic(cord(i,1) - cord(j,1), real(L,8))
129       dy = delta_periodic(cord(i,2) - cord(j,2), real(L,8))
130
131       r = sqrt(dx**2 + dy**2)
132       if (r <= 3.0d0) then
133         f = 24.0d0 * (2.0d0 * r**13 - 1.0d0 / r**7)
134         a(i,1) = a(i,1) + f * dx / r
135         a(i,2) = a(i,2) + f * dy / r
136         a(j,1) = a(j,1) - f * dx / r
137         a(j,2) = a(j,2) - f * dy / r
138       end if
139     end do
140   end do
141
142 end subroutine
143
143 end program

```

Figura 1: Código da tarefa 1

Dessa maneira, com as partículas distribuídas uniformemente ao longo do sistema e velocidades iniciais $v_0 = 1$ obtivemos os seguinte resultados da evolução do sistema.

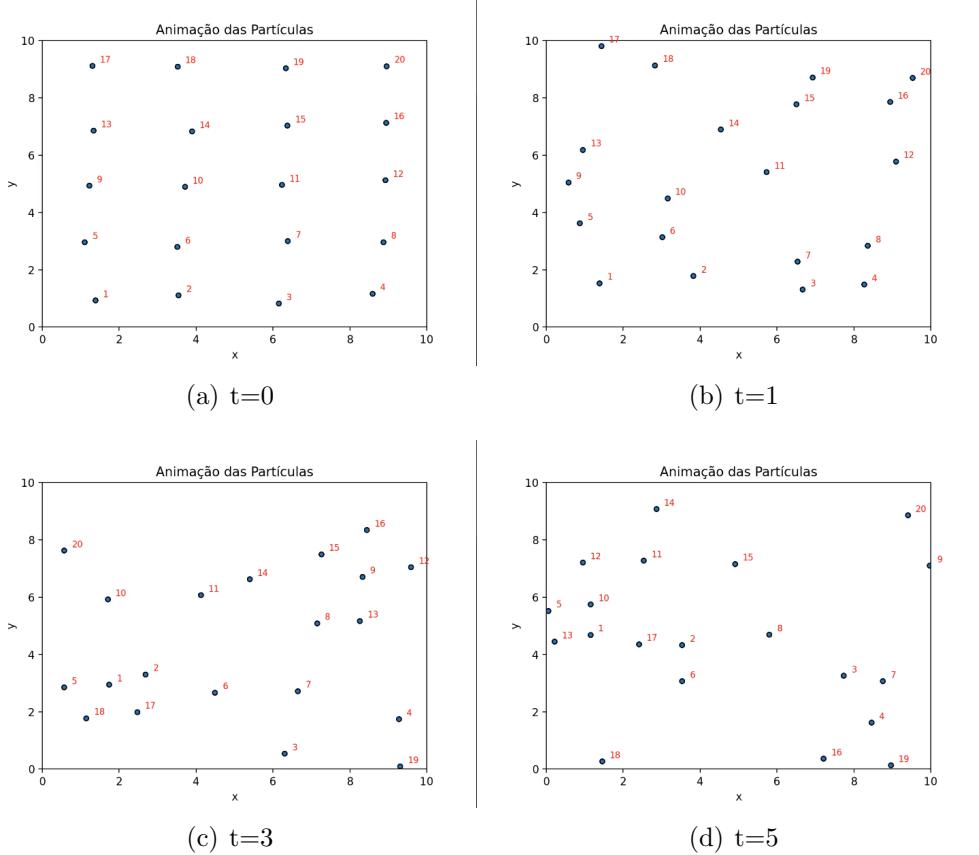


Figura 2: Evolução temporal do sistema com 20 partículas em um sítio de 10x10

De acordo com as imagens, podemos ver que as pertículas se mantiveram corretamente dentro do sistema, sem haver nenhuma explosão de velocidade e foi possível implementar corretamente as condições de contorno para fazer as partículas aparecerem do outro lado do grid sem haver aumento da velocidade das partículas.

2 Tarefa 2

Na segunda tarefa foi proposto analisar as velocidades nos eixos x e y juntamente com seu módulo do sistema da tarefa anterior e comparar com as distribuições de velocidades de Boltzman.

$$P(v) \sim \frac{v}{k_B T} \exp\left(-\frac{mv^2}{2k_B T}\right), \quad (9)$$

$$P(v_x) \sim \frac{1}{\sqrt{k_B T}} \exp\left(-\frac{mv_x^2}{2k_B T}\right), \quad (10)$$

$$P(v_y) \sim \frac{1}{\sqrt{k_B T}} \exp\left(-\frac{mv_y^2}{2k_B T}\right). \quad (11)$$

Dessa maneira, foram salvas as velocidades da simulação anterior e foi possível fazer a seguinte comparação:

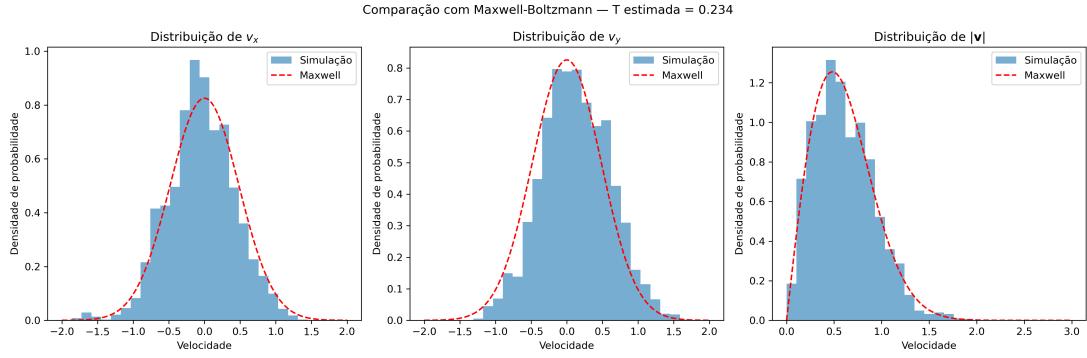


Figura 3: Histograma de comparação das velocidades da simulação juntamente com o esperado pela distribuição de Boltzman

Podemos ver que com a evolução do sistema até o equilíbrio o sistema se mantém de acordo com o esperado pelas distribuições de velocidades de Boltzman, de maneira que as velocidades em v_x e v_y se encontram centradas em 0, já que elas tem as mesmas probabilidades de se mover para todos os lados, assim dessa maneira, por simetria as velocidades médias se encontram em 0. Já o módulo da velocidade de encontra maior que 0, semelhante a distribuição esperada, já que o sistema está com temperatura maior que 0, dessa maneira existe movimento das partículas.

3 Tarefa 3

Nessa tarefa foi proposto realizar a mesma análise do anterior, porém dessa vez ao invés das velocidades iniciais serem aleatórias de módulo 1 para qualquer lado agora teremos as velocidades iniciais como:

$$\vec{v}_0(v_x, v_y) = (1, 0) \quad (12)$$

ou

$$\vec{v}_0(v_x, v_y) = (0, 1) \quad (13)$$

com metade na primeira configuração e outra metade na segunda da seguinte maneira no código.

```

1  call random_number(theta)
2
3  if (theta >= 0.5d0) then
4      v(i,1) = -v_0
5      v(i,2) = 0.0d0
6  else
7      v(i,1) = 0.0d0
8      v(i,2) = -v_0
9  end if

```

Figura 4: Alteração do código da tarefa 1

Dessa maneira, conseguimos os seguintes resultados:

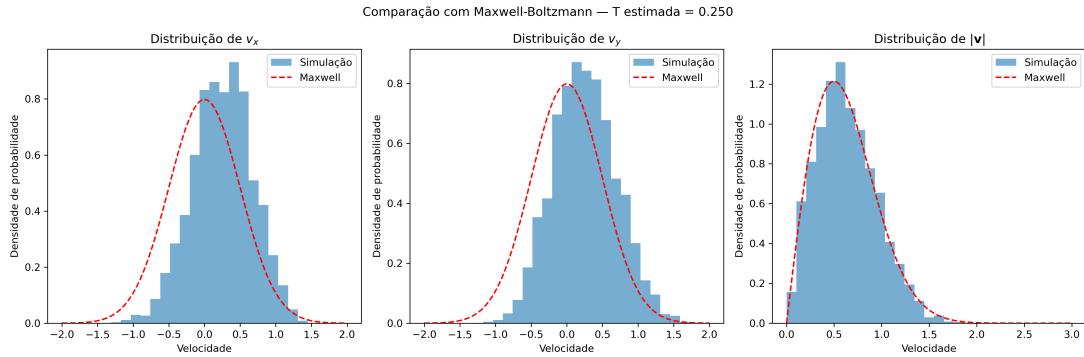


Figura 5: Histograma de comparação das velocidades da simulação juntamente com o esperado pela distribuição de Boltzman

Assim podemos ver que as médias das velocidades em v_x e v_y aparecem levemente deslocadas em relação a origem, aparecendo mais centradas próximas a 0.5, devido a alteração nas velocidades iniciais, mas quando olhamos o módulo da velocidade ele permanece igual ao anterior, devido ao fato de não mudarmos o módulo das velocidades.

4 Tarefa 4

Nessa tarefa foi proposto analisar as temperaturas das duas situações anteriores da tarefa 2 e tarefa 3. Segundo o teorema da equipartição da energia podemos analisar a temperatura do sistema da seguinte maneira.

$$k_B T = \left\langle \frac{m}{2} (v_x^2 + v_y^2) \right\rangle \quad (14)$$

Aplicando esse calculo no código:

```

● ● ●
1 temperature = temperture + vel(i,1)**2.0d0 + vel(i,2)**2.0d0

```

Figura 6: Adição no código da tarefa 1

Dessa maneira foi possível obter as seguintes curvas de temperaturas para os sistemas:

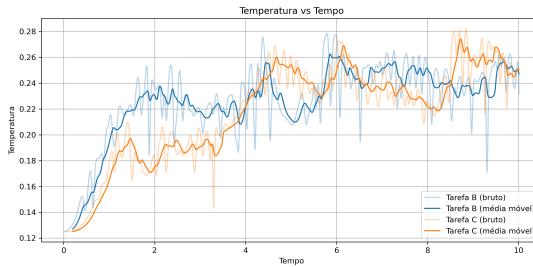


Figura 7: Evolução da temperatura do sistema

Dessa maneira, podemos ver que o sistema B (azul) que apresenta as velocidades distribuídas aleatoriamente nos eixos x e y consegue alcançar o equilíbrio térmico mais rápido do que o sistema C (laranja) onde as velocidades são $(v_x, v_y) = (1, 0)$ ou $(v_x, v_y) = (0, 1)$. Isso ocorre pois o sistema C leva mais tempo para as velocidades se equilibrarem, já que elas estão centradas apenas em uma direção, já o sistema B podemos ver uma curva mais acentuada no começo, nos mostrando que a temperatura aumenta rapidamente alcançando o equilíbrio rapidamente.

5 Tarefa 5

Nessa tarefa, utilizaremos o mesmo código da tarefa 1, com apenas uma pequena diferença, incrementaremos o tempo com um fator de $dt = 0.005$, e aumentando a densidade de partículas para $N = 1600$ e $L = 40$, dessa maneira temos a densidade de 1 em nosso sistema, o que significa que existe uma partícula por unidade de área do sistema, sendo assim um sistema densamente povoado.

Dessa maneira podemos analisar o sistema evoluindo com passos menores e assim podemos notar alguns fenômenos do sistema, como a cristalização do sistema, que é quando o sistema está em equilíbrio e as partículas vibram em sítios fixos, de maneira que a velocidade delas não fique muito alta.

Dessa maneira foram obtidos os seguintes resultados:

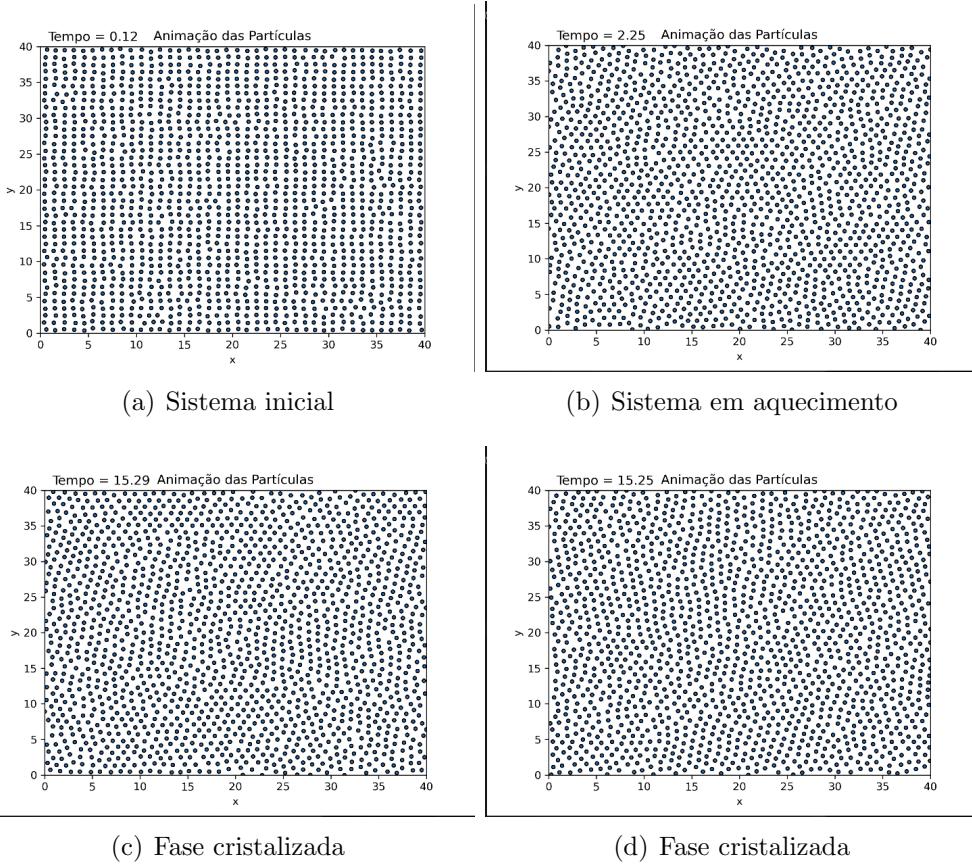


Figura 8: Evolução temporal do sistema com alta densidade

Com a evolução do sistema, podemos observar que, após um período inicial de reorganização das partículas, o sistema atinge uma configuração estável, caracterizada por uma estrutura ordenada. Nesse regime, as partículas passam a vibrar em torno de posições fixas, formando padrões regulares semelhantes a uma rede cristalina — esse fenômeno é conhecido como cristalização. Essa fase cristalizada indica que o sistema alcançou um equilíbrio termodinâmico em que a energia cinética média das partículas é suficientemente baixa para que elas permaneçam organizadas, oscilando levemente ao redor de seus sítios, sem rompimento da estrutura geral.

6 Tarefa 6

Nesse ultima reutilizaremos a simulação anterior mas depois que o sistema atingir o equilíbrio incrementaremos a velocidade com um fator de $R=1.5$ da seguinte maneira.

$$\vec{r}_{\text{ant}} \rightarrow \vec{r}_{\text{atual}} - (\vec{r}_{\text{atual}} - \vec{r}_{\text{ant}}) R \quad (15)$$

Dessa maneira conseguimos o seguinte resultado:

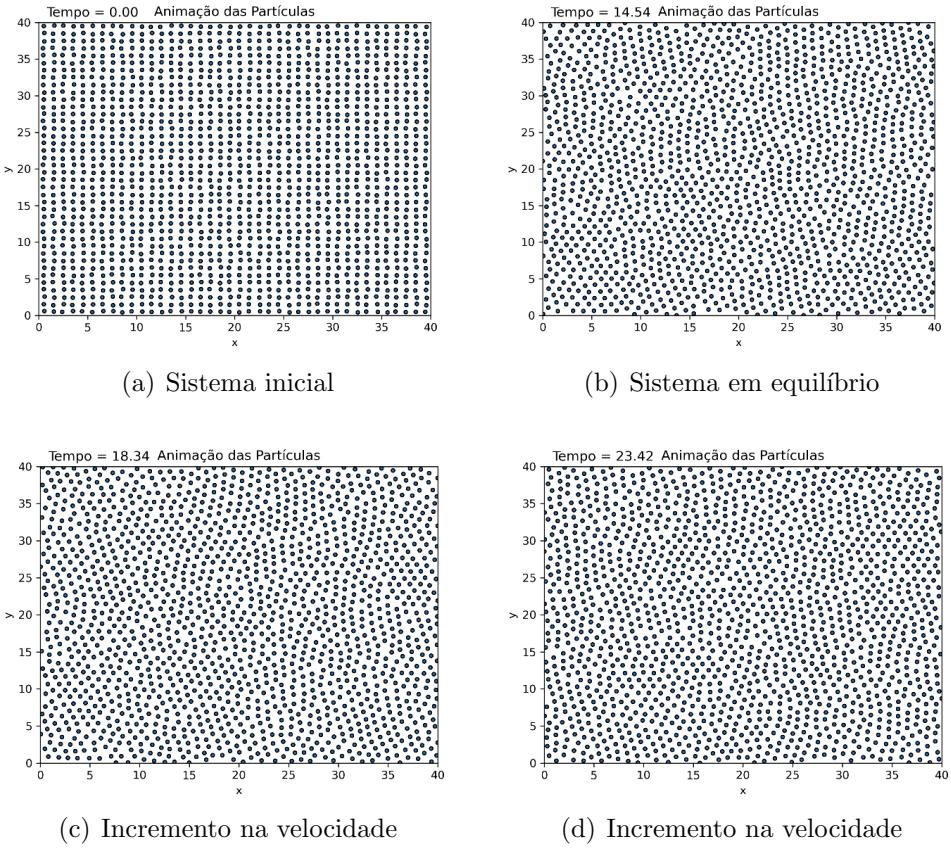


Figura 9: Evolução temporal do sistema com alta densidade

Após o sistema atingir o equilíbrio em uma configuração cristalina, aplicamos o aumento da velocidade das partículas com o fator $R=1,5$, conforme indicado na equação. Esse incremento simula um aquecimento súbito do sistema, elevando a energia cinética das partículas. Como resultado, observamos a ruptura da estrutura cristalina: as partículas passam a se mover de forma mais desordenada, saindo de suas posições fixas e rompendo a simetria da rede. Esse comportamento indica a transição de fase do estado sólido para o estado líquido, caracterizado pela mobilidade aumentada e pela ausência de ordem de longo alcance entre as partículas. A simulação, portanto, reproduz de forma clara o fenômeno de liquefação induzida por aquecimento, evidenciando a sensibilidade do sistema à variação da energia térmica.