Projeto 02: Sistemas aleatórios

7600017 - Introdução à Física Computacional - 2023/02
24/09/2023

Prof. Dr. José Abel Hoyos
Gabriel de Freitas de Azeredo (11810964)

Resumo

Sistemas aleatórios desempenham um papel fundamental na física, modelos estocásticos descrevem desde interações ecológicas até aceleração de raios cósmicos em ambientes astrofísicos. Este projeto explorou a importância desses sistemas, com foco no problema do caminhante aleatório e sua solução utilizando a linguagem de programação Fortran 77. Ao aplicar técnicas de programação e métodos numéricos, pudemos compreender melhor o comportamento de sistemas físicos sujeitos a flutuações e descritos por variáveis aleatórias.

---

1 Tarefa 1 - Momentos de uma distribuição

1.1 Introdução teórica

O objetivo da primeira tarefa é basicamente verificar o gerador de números aleatórios da linguagem *Fortran 77*. Para isso, com a função *RAND*, vamos gerar uma distribuição de N números e seus calcular momentos para $i = 1, 2, 3$ e 4, definidos por

$$\langle x^i \rangle = \frac{1}{N} \sum_{n=1}^{N} x_n^i. \tag{1}$$

Supondo uma distribuição uniforme da função *RAND*, uma variável aleatória contínua gerada por ela respeitaria $X \sim f(x) = 1$ em $[0, 1]$. Seus momentos são suas médias, então

$$\langle x^i \rangle = \int_0^1 x^i \frac{1}{1-0} dx = \frac{1}{i+1}. \tag{2}$$

Assim, para $i = 1, 2, 3$ e 4, respectivamente, temos $\langle x^i \rangle = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ como valores esperados.

1.2 Estratégias de resolução

1.2.1 Entradas e saídas de dados

O código pede como entrada a quantidade N de números aleatórios que será gerada. De saída, retorna o valor dos momentos da distribuição.

1.2.2 Desenvolvimento do código

Consiste de um *loop* que gera os N valores aleatórios, guardando em um vetor de 4 dimensões, em cada iteração, a soma do retorno da função *RAND*, o retorno ao quadrado, ao cubo e elevado a 4.

```
program tarefa1

real*8 :: dmoment(4) = 0.d0

call srand(1)

N = 0

write(*,*) "Digite o tamanho (N) da distribuição: "
read(*,*) N

do i = 1, N

    x = rand()

    dmoment(1) = dmoment(1) + x
    dmoment(2) = dmoment(2) + x**2
    dmoment(3) = dmoment(3) + x**3
    dmoment(4) = dmoment(4) + x**4

end do

write(*,*) "<x> = ", dmoment(1) / real(N, 8)
write(*,*) "<x^2> = ", dmoment(2) / real(N, 8)
write(*,*) "<x^3> = ", dmoment(3) / real(N, 8)
write(*,*) "<x^4> = ", dmoment(4) / real(N, 8)

end program tarefa1
```

1.3 Resultados

A tabela 1.3 contém os resultados para o programa, utilizando os valores para N indicados. Os valores foram aredondados para 3 casas decimais.

$\langle x \rangle$	$\langle x^2 \rangle$	$\langle x^3 \rangle$	$\langle x^4 \rangle$	N
0,518	0,355	0,270	0,218	100
0,498	0,327	0,241	0,189	1000
0,502	0,335	0,252	0,202	10000
0,500	0,333	0,250	0,200	100000

Tabela 1: Resultados obtidos para a primeira tarefa.

2 Tarefa 2 - O andarilho aleatório

2.1 Introdução teórica

Essa atividade consiste em resolver o problema do andarilho aleatório para uma dimensão, calculando a média e a distribuição de viajantes no espaço, após N passos. Supondo um andarilho que pode andar somente para a direita ou esquerda, sem opção de ficar parado, por N interações (passos), o objetivo é calcular sua posição final, considerando p a probabilidade de andar para a direita (por complementariedade, $q = 1 - p$ é a de andar para esquerda).

Para calcular o valor esperado, precisamos modelar o problema de outra forma, como a posição x pode ser calculada pela diferença de passos a direita com os passos a esquerda e a quantidade total de passos sempre é igual a N, podemos considerar o passo a direita (d) como um "sucesso". A probabilidade de um sucesso em N experimentos é dada pela distribuição binomial

$$P(d, N) = \binom{N}{d} p^d q^e. \tag{3}$$

Em nosso modelo todos os viajantes tem o passo de mesmo tamanho $\Delta x = 1$ u.a. por isso $\langle x \rangle = \langle d \rangle - \langle e \rangle = 2\langle d \rangle - N$. Para calcular $\langle d \rangle$, pela definição,

$$\langle d \rangle = \sum_{d=0}^{N} \binom{N}{d} d p^d q^{N-d}, \tag{4}$$

como $p\frac{\partial p^d}{\partial p} = dp^d$, podemos fazer a substituição e chegar em

$$\langle d \rangle = \sum_{d=0}^{N} \binom{N}{d} p \frac{\partial p^d}{\partial p} q^{N-d} = p \frac{\partial}{\partial p} \sum_{d=0}^{N} \binom{N}{d} p^d q^{N-d} \tag{5}$$

que é exatamente a forma do teorema binomial $(a+b)^m = \sum_{k=0}^{m} \binom{N}{n} a^k b^{m-k}$, pois $q = 1-p$. Portanto,

$$\langle d \rangle = p \frac{\partial (p+q)^N}{\partial p} = pN(p+q)^{N-1} = pN, \tag{6}$$

da mesma forma para calcular $\langle x^2 \rangle$ é necessário o cálculo de $\langle x^2 \rangle$, de forma análoga fazemos:

$$\langle d^2 \rangle = \sum_{d=0}^{N} \binom{N}{d} d^2 p^d q^{N-d}, \tag{7}$$

usando agora que $p^2\frac{\partial^2 p^d}{\partial p^2} = d^2 p^d - dp^d$, substituindo em (7) e utilizando novamente o teorema binomial é fácil chegar em

$$\langle d^2 \rangle = p^2 \frac{\partial^2 (p+q)^N}{\partial p^2} + p \frac{\partial (p+q)^N}{\partial p} = p^2 N(N-1) + pN. \tag{8}$$

Finalmente, $\langle x \rangle = 2pN - N = N(2p-1)$ e $\langle x^2 \rangle = \langle (d-e)^2 \rangle = 4\langle d^2 \rangle - 4N\langle d \rangle + N^2$, usando (6) e (8) chegamos em $\langle x^2 \rangle = 4(p^2 N(N-1) + pN) - 4N^2 p + N^2$. Vamos resolver

o andarilho aleatório para $p = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ e a tabela 2.1 mostra os valores esperados para $\langle x^2 \rangle$ e $\langle x \rangle$, considerando $N = 1000$.

p	$\langle x \rangle$	$\langle x^2 \rangle$
1/2	0	1000
1/3	$-1000/3$	112000
1/4	-500	250750
1/5	-600	360640

Tabela 2: Valores esperados para a média de x e de x^2 na tarefa 2a e 2b.

Note que os valores para $\langle x \rangle$ são negativos para $p < 1/2$ fato esperado, pois a probabilidade de dar passos para a direita diminuiu em relação aos passos para a esquerda. Assim, em média, os andarilhos, com condição inicial $x = 0$, param em $x < 0$.

2.2 Estratégias de resolução

2.2.1 Entradas e saídas de dados

Os programas da tarefa 2a e 2b esperam de entrada somente o número M de andarilhos e retornam em arquivos de saída a posição final de cada andarilho.

2.2.2 Desenvolvimento do código

O código consiste de dois *loops* para o número de andarilhos e número de passos, definido com $N = 1000$ no início do programa. Para cada passo, um número aleatório é gerado com a função *RAND* do *Fortran*, então testo se esse número está em $0 \leq x < q$ ou $q \leq x \leq 1$. Simplificando, caso o número seja maior que $q = 1 - p$, então o andarilho dá um passo para a direita, caso contrário, para esquerda. Assim, para $p = 1/3$, por exemplo, resultados de 0 até 2/3 são passos para esquerda e consequentemente, 2/3 até 1 para a esquerda. O código para a tarefa 2a:

```
program tarefa2a

integer*4, parameter :: N = 1000
real*8 :: sumx = 0.e0, sumx2 = 0.e0

open(11, file = "tarefa-2a-saida-1.dat")
call srand(1)

M = 0
p = 0.5e0

write(*,*) "Número de andarilhos (M): "
read(*,*) M

do i = 1, M
```

```
        ix = 0

        do j = 1, N

            if (rand().ge.p) then
               ix = ix + 1
            else
               ix = ix - 1
            end if

        end do

        write(11,'(I7)') ix

        sumx = sumx + ix
        sumx2 = sumx2 + ix**2
      end do

      write(*,*) "<x>=", sumx / real(M, 4)
      write(*,*) "<x^2>=", sumx2 / real(M, 4)

      close(11)

      end program tarefa2a
```

da mesma forma para a tarefa 2b:

```
      program tarefa2b

      integer*8, parameter :: N = 1000
      integer*8 :: ix(3)
      real*8 :: p(3), sumx(3) = 0.e0, sumx2(3) = 0.e0

      open(11, file = "tarefa-2b-saida-1.dat")
      open(12, file = "tarefa-2b-saida-2.dat")
      open(13, file = "tarefa-2b-saida-3.dat")

      call srand(1)

      M = 0

      p(1) = 1.e0 / 3.e0
      p(2) = 1.e0 / 4.e0
      p(3) = 1.e0 / 5.e0

      write(*,*) "Número de andarilhos (M):"
      read(*,*) M
```

```
do i = 1, M

   ix(1) = 0
   ix(2) = 0
   ix(3) = 0

   do j = 1, N

      rand_x = rand()

      do k = 1, 3
         if (rand_x.ge.(1.e0 - p(k))) then
            ix(k) = ix(k) + 1
         else
            ix(k) = ix(k) - 1
         end if
      end do

   end do

   write(11,'(I7)') ix(1)
   write(12,'(I7)') ix(2)
   write(13,'(I7)') ix(3)

   do k = 1, 3
      sumx(k) = sumx(k) + ix(k)
      sumx2(k) = sumx2(k) + ix(k)**2
   end do

end do

do k = 1, 3

   write(*,*) "Para p=", p(k)
   write(*,*) "<x>=", sumx(k) / real(M, 4)
   write(*,*) "<x^2>=", sumx2(k) / real(M, 4)

end do

close(11)
close(12)
close(13)

end program tarefa2b
```

2.3 Resultados

O programa foi executado com $M = 100000$. Como visto anteriormente e esperado teoricamente, a média de x e x^2 não depende do número de andarilhos, mas quanto maior M, mais nos aproximamos da média teórica. A tabela 2.3 mostra os resultados com duas casas decimais.

p	$\langle x \rangle$	$\langle x^2 \rangle$
1/2	$0,06$	$998,32$
1/3	$-333,29$	$111966,99$
1/4	$-499,92$	$250673,37$
1/5	$-599,98$	$360614,05$

Tabela 3: Valores obtidos para a média de x e de x^2 na tarefa 2a e 2b.

Comparando as tabelas, fica claro que os resultados foram consonantes. Para uma visualização mais clara, foi feito um histograma para todas as probabilidades p, com os dados dos arquivos de saídas.

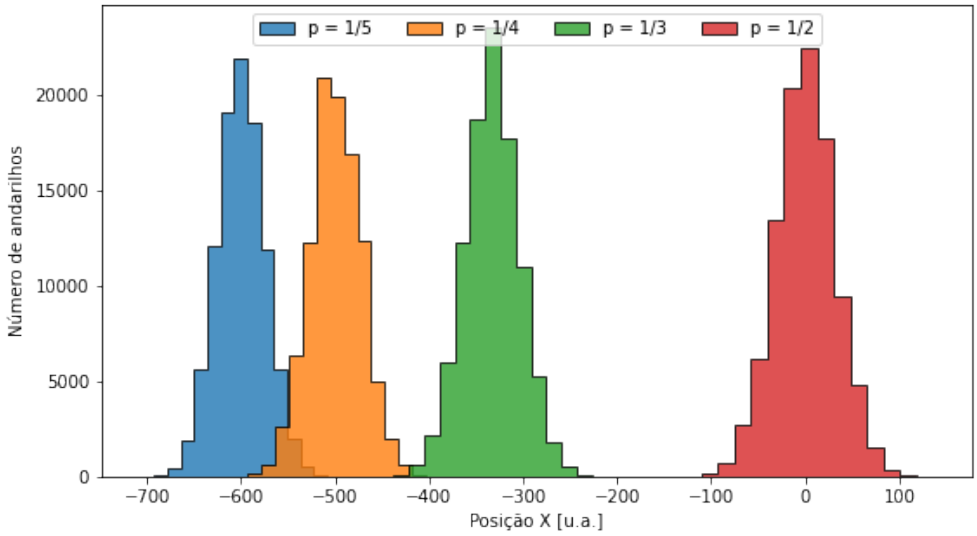

Figura 1: Histograma do número de andarilhos em função da posição x.

3 Tarefa 3 - Andarilho aleatório em 2 dimensões

3.1 Introdução teórica

O problema em duas dimensões consiste basicamente de uma generalização do que já foi abordado não seção anterior. Dessa forma, o movimento de cada andarilho pode ser decomposto para dois movimentos em 1 dimensão. Existe 50% de probabilidade do caminhante andar no eixo x e 50% de andar no eixo y. Supondo que o caminhante andou no eixo x, existe 50% de chance de caminhar para a direita e 50% de caminhar para a

esquerda, ou seja, é o movimento em 1 dimensão com $p = 1/2$, então $\langle x \rangle = 0$ tendo em vista todo o formalismo da questão anterior e os resultados na tabela 2.1. Da mesma forma $\langle y \rangle = 0$. Para $\Delta^2 = \langle (x,y) \cdot (x,y) \rangle - \langle (x,y) \rangle \langle (x,y) \rangle = \langle x^2 \rangle + \langle y^2 \rangle$, pois $\langle (x,y) \rangle \langle (x,y) \rangle = 0$.

3.2 Estratégias de resolução

3.2.1 Entradas e saídas de dados

O programa pede o número N de passos que os $M = 10000$ andarilhos devem dar. Como saída, o arquivo da posição final (x, y) dos andarilhos.

3.2.2 Desenvolvimento do código

O programa é uma generalização do programa da atividade anterior. Um *loop* para cada andarilho e um *loop* para cada passo, então, com a posição iniciada em $(x, y) = (0, 0)$, é gerado um número aleatório com a função *RAND* do *Fortran77*. Como as chances do andarilho andar pada cada direção são as mesmas e a função gera uma distribuição aleatória de 0 até 1, a seguinte condição foi aplicada: **se** $x \leq 0.25$ então o passo é para **cima**, **se** $x \leq 0.5$ então o passo é para **baixo**, **se** $x \leq 0.75$ então o passo é para **direita**, caso contrário o passo é para **esquerda**. O código é como abaixo:

```
program tarefa3

integer*8 :: ix = 0, iy = 0
real*8 :: dmeanx = 0.d0, dmeany = 0.d0, delta = 0.d0

open(11, file = "tarefa-3-saida-1.dat")

call srand(1)

M = 100000
N = 0

write(*,*) "Número de passos (N): "
read(*,*) N

do i = 1, M

   ix = 0
   iy = 0

   do j = 1, N

      rand_x = rand()

      if (rand_x.le.(0.25d0)) then
         iy = iy + 1
      else if (rand_x.le.(0.5d0)) then
```

```
            iy = iy - 1
        else if (rand_x.le.(0.75d0)) then
            ix = ix + 1
        else
            ix = ix - 1
        end if

    end do

    dmeanx = dmeanx + ix
    dmeany = dmeany + iy
    delta = delta + ix**2 + iy**2

    write(11,'(2I7)') ix, iy

end do

dmeanx = dmeanx / real(M, 8)
dmeany = dmeany / real(M, 8)
delta = delta / real(M, 8)

write(*,*) "<x>=", dmeanx
write(*,*) "<y>=", dmeany
write(*,*) "<delta^2>=", delta - dmeanx**2 - dmeany**2

close(11)

end program tarefa3
```

3.3 Resultados

O programa foi executado com $M = 100000$, para $N = 10, 100, 1000, 10000, 100000$ e 1000000. Os resultados para $\langle \vec{r} \rangle$ e para Δ^2 estão na tabela 4. Percebe-se que os resultados foram muito satisfatórios e com o aumento do número de andarilhos é esperado que a média se aproxime cada vez mais do valor teórico.

N	$\langle \vec{r} \rangle$	Δ^2
10	$(0.008, 0.002)$	$9,96$
100	$(-0.01, 0.002)$	$100,60$
1000	$(-0.04, -0.04)$	$998,40$
10000	$(-0.14, -0.21)$	$10015,47$
100000	$(-0.20, -0.13)$	$99817,88$
1000000	$(0.10, 0.09)$	$998816,95$

Tabela 4: Resultados obtidos para $\langle \vec{r} \rangle$ e Δ^2 na tarefa 4.

Para visualizar a distribuição espacial dos andarilhos foram realizados histogramas em

função da variável x e y como mapas de calor. O histograma mais comum também foi feito para a variável x e y de maneira independente e se encontram ao lado do respectivo mapa de calor que o representam.

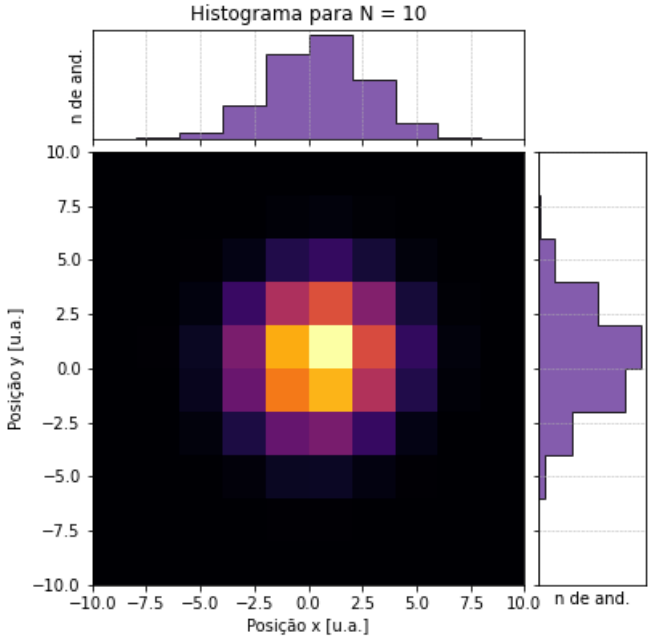

Figura 2: Simulação para $N = 10$.

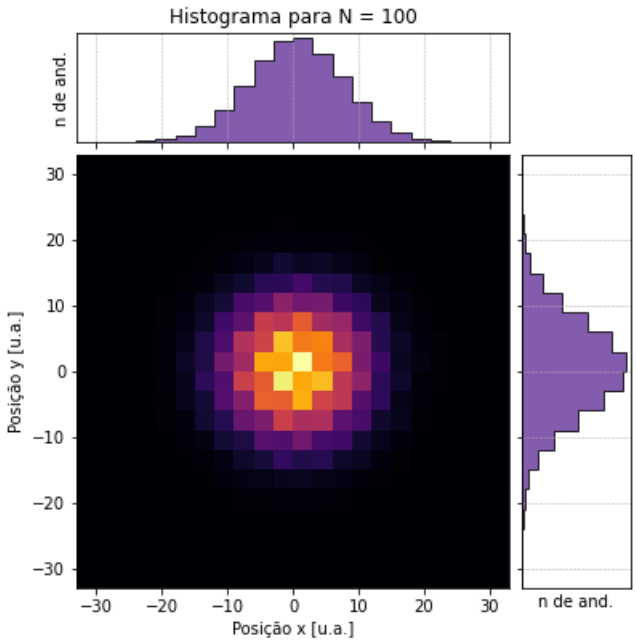

Figura 3: Simulação para $N = 100$.

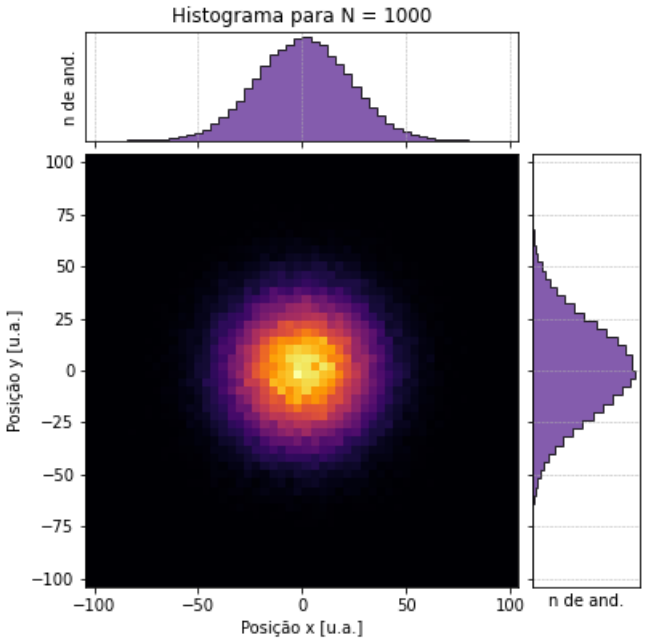

Figura 4: Simulação para $N = 1000$.

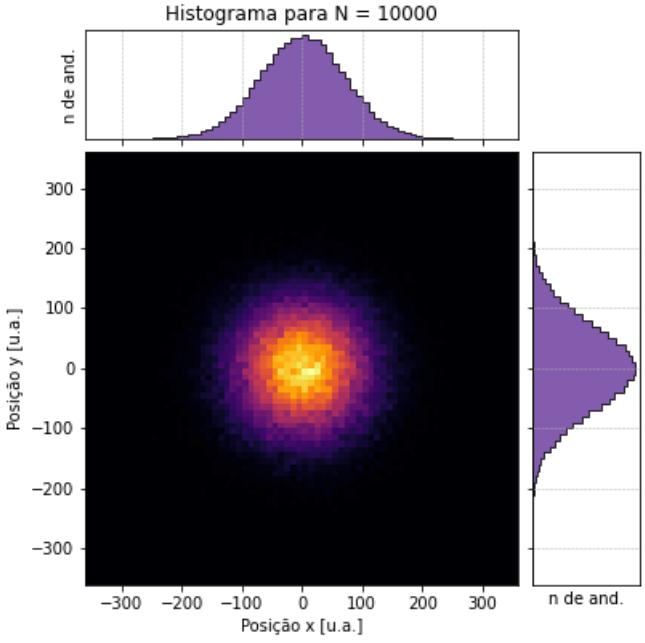

Figura 5: Simulação para $N = 10000$.

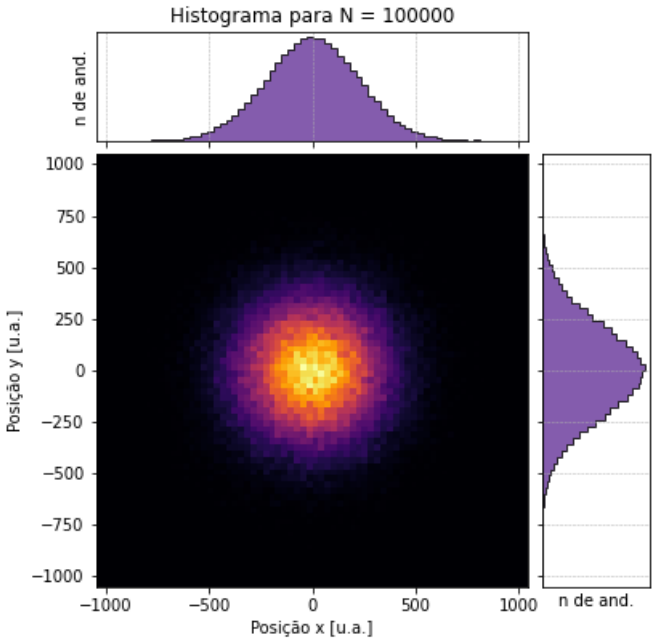

Figura 6: Simulação para $N = 1000000$.

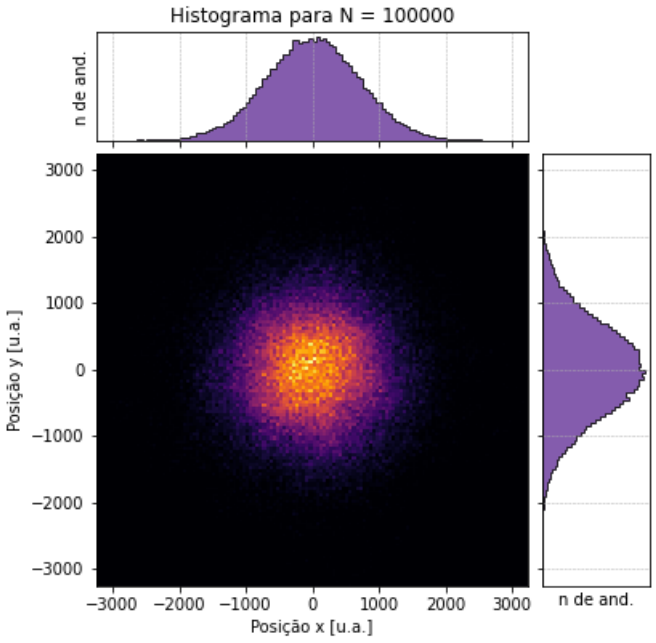

Figura 7: Simulação para $N = 1000000$.

4 Tarefa 4 - Entropia

4.1 Introdução teórica

O objetivo da tarefa 4 é calcular como a entropia do sistema evolui no tempo que no caso é proporcional ao número de passos. Para isso vamos definir reticulados no espaço,

muito maiores que o tamanho médio do passo, constante e igual a 1 u.a. A entropia do sistema é definida como

$$S = -\sum_i P_i \ln(P_i), \tag{9}$$

onde P_i é a probabilidade de se encontrar o sistema em um certo micro-estado i.

4.2 Estratégias de resolução

4.2.1 Entradas e saídas de dados

Não é necessário nenhuma entrada, o programa retorna como saída um arquivo contendo a entropia em função do número de passos.

4.2.2 Desenvolvimento do código

Para resolver esse problema realizei algumas mudanças no programa da tarefa anterior. Para gerar a entropia em função do número de passos, o primeiro *loop* foi feito para o número de passos e em cada iteração todos o andarilhos caminham uma unidade. Depois, o código realiza a contagem de quantos andarilhos estão no intervalo $x < x_i < x + a$, onde a é o tamanho do *grid* e x varia por todas as posições que os andarilhos podem assumir $-N$ até $N - a$, a mesma condição é feita para y, então todo o espaço é mapeado. Após realizar a contagem e aplicar a equação (9), temos a entropia para cada passo.

```
program tarefa4

integer, parameter :: N = 1000
integer, parameter :: M = 1000
integer*8 :: ipos(M, 2) = 0

open(11, file = "tarefa-4-saida-1.dat")
call srand(1)

do i = 1, N

   do j = 1, M
      rand_x = rand()

      if (rand_x.le.(0.25d0)) then
         ipos(j, 2) = ipos(j, 2) + 1
      else if (rand_x.le.(0.5d0)) then
         ipos(j, 2) = ipos(j, 2) - 1
      else if (rand_x.le.(0.75d0)) then
         ipos(j, 1) = ipos(j, 1) + 1
      else
         ipos(j, 1) = ipos(j, 1) - 1
      end if
   end do
```

```
      S = 0.e0

      do ix = -N, N-5, 5
         do iy = -N, N-5, 5

            and = 0.e0

            do k = 1, M

               x_and = ipos(k, 1)
               y_and = ipos(k, 2)

               if(((ix.le.x_and).and.(x_and.le.(ix + 5)))
     *            .and.((iy.le.y_and).and.(y_and.le.(iy + 5)))) then

                  and = and + 1.e0

               end if
            end do

            if (and.ne.(0.e0)) then

               pi = and / real(M, 8)
               S = S - pi * log(pi)
            end if
         end do
      end do

      write(11,*) i , S
   end do

   close(11)

   end program tarefa4
```

4.3 Resultados

Como resultado, uma curva da entropia em função do número de passos. O código gerou essa saída para $N = 1000$ e número de andarilhos $M = 1000$. A malha utilizada foi com $a = 5$.

Figura 8: Entropia em função do número de passos.