UNIVERSIDADE DE SÃO PAULO

Instituto de Física de São Carlos

Projeto 02

Sistemas aleatórios

Jefter Santiago Mares
n° USP:12559016

08 de outubro de 2022

Conteúdo

1 (A) Momentos de distribuição **2**

2 (B) Andarilhos aleatório em uma dimensão **3**

3 (C) Andarilho aleatório em 2 dimensões **8**

4 (D) Cálculo da entropia **12**

São Carlos
2021

(A) Momentos de distribuição

Seja $f(x) = x^n$ uma função $f : \mathbb{R} \mapsto \mathbb{R}$, a média dessa função pode ser definida por

$$\langle f \rangle = \frac{1}{b-a} \int_a^b f(x)dx = \frac{1}{b-a} \cdot \left. \frac{x^{n+1}}{n+1} \right|_a^b$$

estamos interessados no intervalo $[0, 1]$, portanto a média deve ser

$$\langle f \rangle = \frac{1}{1-0} \cdot \left. \frac{x^{n+1}}{n+1} \right|_0^1 = \frac{1}{n+1}$$

$$\langle f \rangle = \frac{1}{n+1} \tag{1}$$

Portanto, para cada caso $n = 1, 2, 3$ e 4 temos

Tabela 1: Valores esperados para o cálculo de $f(x)$ para cada n .

n	$\langle f \rangle$
1	1/2
2	1/3
3	1/4
4	1/5

Código e resultados

O código abaixo executa o cálculo dessa média para qualquer N.

```fortran
C     Tarefa1: Cálcular a média de um número para um N
      read(*, *) N
      sum1 = 0.0e0
      sum2 = 0.0e0
      sum3 = 0.0e0
      sum4 = 0.0e0

      x = rand(iseed)
      do i = 1, N
         x = rand()
         sum1 = sum1 + x
         sum2 = sum2 + x**2
         sum3 = sum3 + x**3
         sum4 = sum4 + x**4
      end do

      sum1 = sum1 / N
      sum2 = sum2 / N
      sum3 = sum3 / N
      sum4 = sum4 / N

      write(*, *) "<x1> = ", sum1
      write(*, *) "<x2> = ", sum2
      write(*, *) "<x3> = ", sum3
      write(*, *) "<x4> = ", sum4
      end
```

	$N = 100$	$N = 10000$	$N = 100000$	$N = 1000000$
x	0.523797512	0.501901746	0.500286758	0.500028610
x^2	0.357666671	0.335528523	0.333478957	0.333151519
x^3	0.271176100	0.252268225	0.249980465	0.249754101
x^4	0.218358591	0.202313691	0.199869826	0.199709803

Esses resultados estão dentro do esperado, já que são valores muito próximo das frações listadas na tabela (1). Note que para N cada vez maior, melhor a aproximação.

(B) Andarilhos aleatório em uma dimensão

(B.1) Muitos andarilhos com $p = \frac{1}{2}$

O programa abaixo cálcula para M andarilhos a posição final após $N = 10000$ passos dados. Então é calculado os valores de $\langle x \rangle$ e $\langle x^2 \rangle$ e além disso foi feito um histograma para visualizarmos o resultado final.

Esse trecho de código é responsável pelo cálculo de $\langle x \rangle$ e $\langle x^2 \rangle$.

```
      parameter (n = 1000)
      parameter (m = 1000)

      parameter (nbin = 31)

      dimension nwalker(m)
      dimension istep(2)

      iStep(1) = -1
      iStep(2) = 1

      x = 0e0
      x2 = 0e0

      r = rand(iseed)
      do i = 1, m
         nSteps = 0
         do j = 1, n
            r = rand() * 2
!           i está no intervalo [1, 2]
            k = int((r + 1) / 2) + 1
!           Quantidade de passos dada por andarilho.
            nSteps = nSteps + iStep(k)
         end do
         nWalker(i) = nSteps
         x = x + nSteps
         x2 = x2 + nSteps ** 2
      end do

      xm = x / m
      x2m = x2 / m

      print *,"<x> =", xm
      print *,"<xš> =", x2m
```

Esse bloco de código cálcula o histograma para os resultados obtidos na parcela anterior O número de bins ou conjuntos do histograma foi escolhido como $n \sim \sqrt{N}$, onde N é a quantidade de passos.

```fortran
!   Constroí o histograma
    xmin = nWalker(1)
    xmax = xmin

    do i = 2, m
       if(nWalker(i) < xmin) then
          xmin = nWalker(i)
       end if
       if(nWalker(i) > xmax) then
          xmax = nWalker(i)
       end if
    end do

    dx =  (xmax - xmin) / nbin

    open(unit=10,file='output.dat')
    do j = 1, nbin
       nhist = 0
       infLim = xmin + (j - 1) * dx
       supLim = xmin + j * dx

       do i = 1, m
          if(nWalker(i) >= infLim .and. nWalker(i) < supLim) then
            nhist = nhist + 1
          end if
       end do

       write(10, *) supLim , nhist
    end do
    close(10)
```

Com base nos resultados escritos no arquivo `output.dat` é criado o histograma abaixo

Esses resultados correspondem ao esperado, pois analiticamente, a posição final média pode ser calculada pela relação $\langle x \rangle = N(p-q)$, como $p+q=1$ podemos escrever a equação em termos de p, logo $q = 1-p \Rightarrow (p-q) = p-(1-p) = 2p-1$, então

$$\langle x \rangle = N(2p-1) \tag{2}$$

e a deslocamento quadrado médio é dada por $\left\langle x^2 \right\rangle = 4Npq + N^2 - 4N^2pq$, escrevendo em termos de p temos

$$\left\langle x^2 \right\rangle = 4Np(1-p) + N^2 - 4N^2p(1-p) \tag{3}$$

como $p+q=1$ e $p=1/2$ segue que $p=q \Rightarrow p-q=0$ então a posição média esperada para $p=1/2$ é $\langle x \rangle = 0$. Já o termo $\left\langle x^2 \right\rangle = 4 \cdot 1000 \cdot \frac{1}{2} \cdot \frac{1}{2} + (1000)^2 - 4(1000)^2 \cdot \frac{1}{2}\frac{1}{2} = 1000 \Rightarrow \left\langle x^2 \right\rangle = 1000$.

Os valores cálculados são $\langle x \rangle = -0.00913999975$ e $\left\langle x^2 \right\rangle = 1001.44202$.

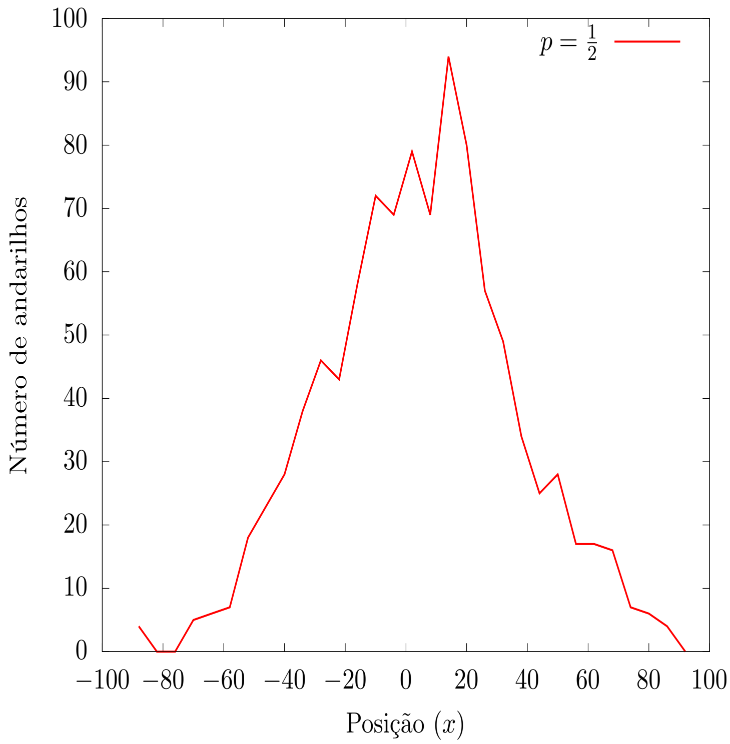

Figura 1: Simulação com $p = 1/2$.

(B.2) Muitos andarilhos com $p = \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$

Foi generalizado o código para $p = \frac{1}{2}$ em uma dimensão, temos então esse código

```
!       Tarefa B - Cálcular <x> e <x2>
        parameter (n = 10000)
        parameter (m = 10000)

        parameter (nbin = 100)

        dimension nwalker(m)

        dimension istep(2)

        iStep(1) = -1
        iStep(2) = 1

        write(*,*) "p = "
        read(*, *) p

        x = 0e0
        x2 = 0e0

        r = rand(iseed)
        do i = 1, m
           nSteps = 0
           do j = 1, n
              r = rand() * int((1 / p))
!             i está no intervalo [1, int(1/p)]
              k = int((r + 1) / (1 / p)) + 1
!          Quantidade de passos dada por andarilho.
              nSteps = nSteps + iStep(k)
           end do
           nWalker(i) = nSteps
           x = x + nSteps
           x2 = x2 + nSteps ** 2
        end do

        xm = x / m
        x2m = x2 / m

        print *,"<x> =", xm
        print *,"<x2> =", x2m
```

o código fonte está na pasta `./tarefaB/tarefa-b2.f` e pode ser usado para fazer o cálculo para qualquer valor de `p`. O trecho de código que gera o histograma é o mesmo que para o anterior.

Resultados

Podemos estimar os resultados analiticamente por meio das equações (2) e (3), como $p + q = 1 \Rightarrow q = 1 - p$, com $N = 1000$. Foi compilado abaixo os resultados analíticos e estatísticos:

Abaixo estão os histogramas para as posições finais com $p = \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ e $M = 10000$ andarilhos.

Tabela 2: Valores de $\langle x \rangle$ calculado pelas simulações e analiticamente com $N = 1000$.

p	Simulação	Resultado Analitico
1/3	-333.099609	-333.3337
1/4	-499.925812	-500.0
1/5	-599.990784	-600.0

Tabela 3: Valores de $\langle x^2 \rangle$ calculado pelas simulações e de forma analitica, com $N = 1000$.

p	Simulação	Resultado Analitico
1/3	111835.586	111999.9998
1/4	250667.750	250750.0
1/5	360620.500	360640.0

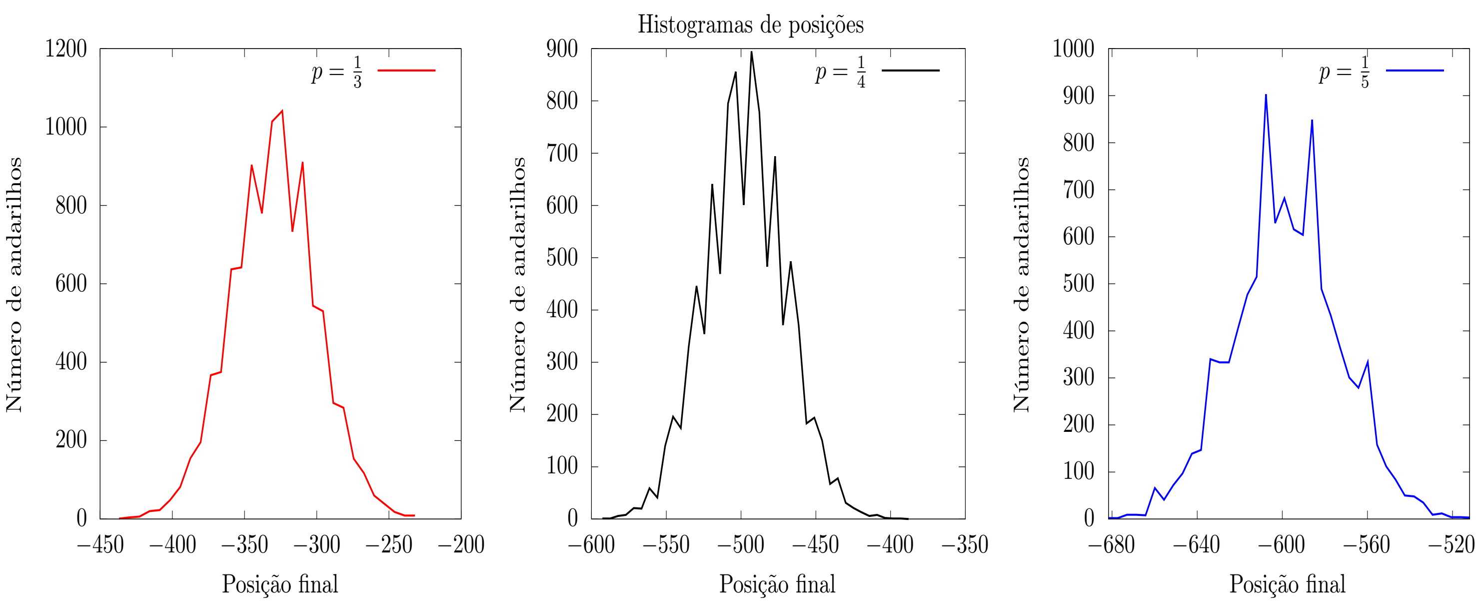

Figura 2: Simulações para $p = \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$, com $N = 1000$.

(C) Andarilho aleatório em 2 dimensões

Generalizando o algoritmo para cálculo do andarilhos em uma dimensão para duas, com probabilidades iguais de passos à norte, sul, leste e oeste, ou seja, $p = \frac{1}{4}$, temos o código abaixo.

```fortran
! Tarefa C - andarilhos em duas dimensoes
    parameter (m = 1000)

    dimension walker(m, 2)
    dimension r(m)

    dimension iStep(2)

    write(*, *) "N"
    read(*, *) n

    iStep(1) = -1
    iStep(2) = 1

    rMean = 0e0
    r2Mean  = 0e0

    rX = 0e0
    rY = 0e0

    rnd = rand(iseed)
    open(unit=10,file='saida-tarefa-c.dat')
    do i = 1, m

       walker(i, 1) = 0
       walker(i, 2) = 0

       do j = 1, n
          k = rand() * 2
          k = int((k + 1) / 2) + 1
          l = rand() * 2
          l = int((l + 1) / 2) + 1
          walker(i, k) = walker(i, k) + iStep(l)
       end do

       write(10, *) walker(i, 1), walker(i, 2)

       rX = rX + walker(i, 1)
       rY = rY + walker(i, 2)
       r2Mean = r2Mean + walker(i, 1) ** 2  + walker(i, 2) ** 2

    end do
    close(10)

    rMean = sqrt(rX**2 + rY**2) / m
    print *,"<r> = ", rMean

    r2Mean = r2Mean / m
```

```
    rDelta = r2Mean - rMean**2
    print *,"2 = ", rDelta
    end

```

Foi usado esse código para cálcular $\langle r \rangle$ e $\Delta^2 = \left\langle r^2 \right\rangle - \langle r \rangle^2$ e para gerar o gráfico de difusão com $N = 10, 10^2, ..., 10^6$, que foi feito a partir a partir das posições finais armazenadas em um arquivo. Abaixo estão os resultados:

Tabela 4

N	$\langle r \rangle$	Δ^2
10	0.142056316	10.2558203
10^2	0.427055031	104.869629
10^3	0.539230943	980.699219
10^4	1.50345862	10793.6309
10^5	13.0313272	99725.5781
10^6	43.1298370	1009048.38

Nota-se que, para valores cada vez maiores de N o vetor médio $\langle r \rangle$ tende a ficar mais longe da origem, ou seja, tem uma maior dispersão e assim como no caso unidimensional a variação Δ^2 é próxima do número de passos.

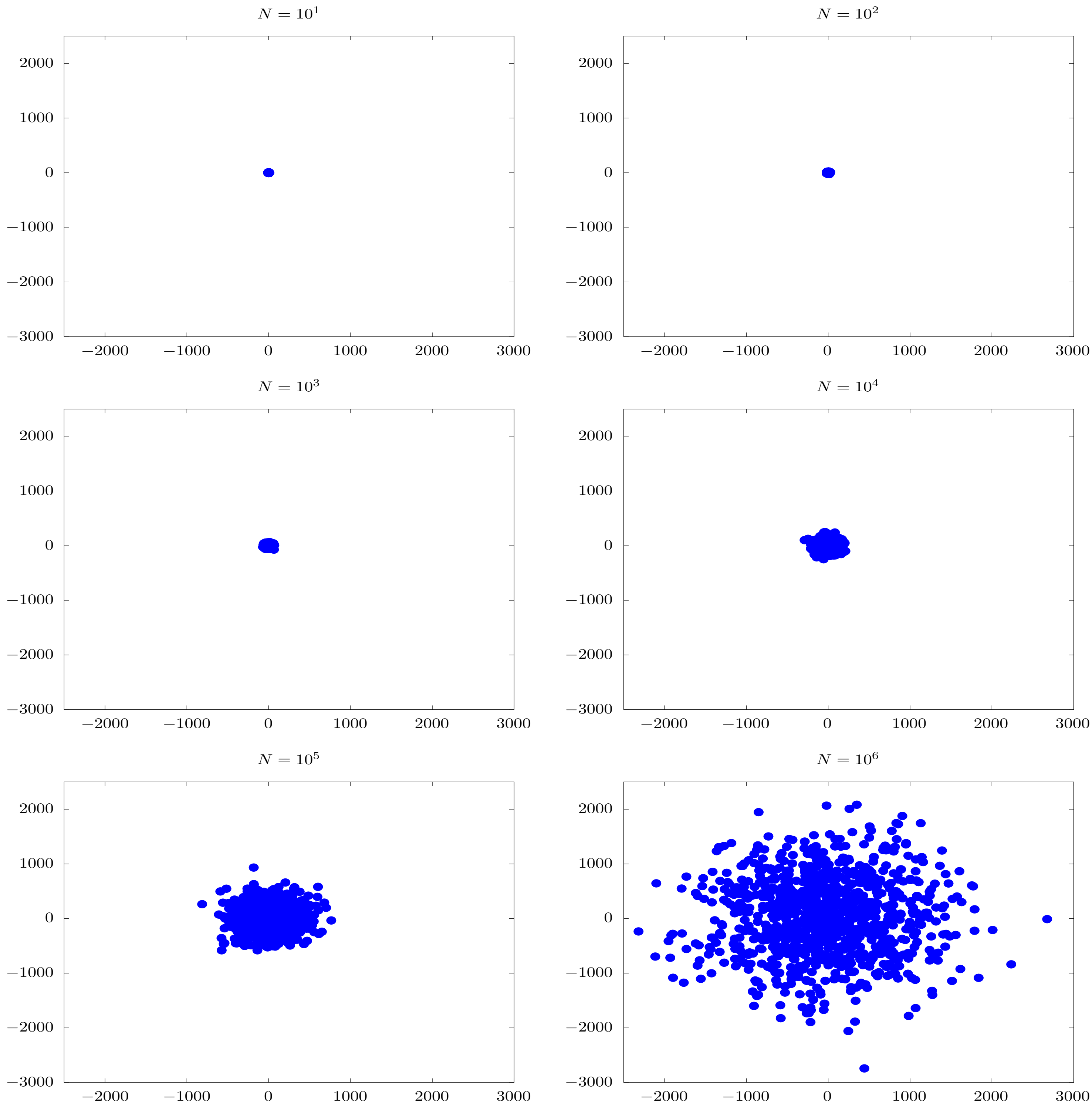

Figura 3: Gráficos de difusão de $M = 1000$ andarilhos em duas dimensões (X, Y).

Foi feito também algumas alterações no código para duas dimensões com objetivo de visualizar as trajetórias traçadas por cada andarilho para N grande, assim podemos ter uma relação espacial a cerca da difusão em 2D.

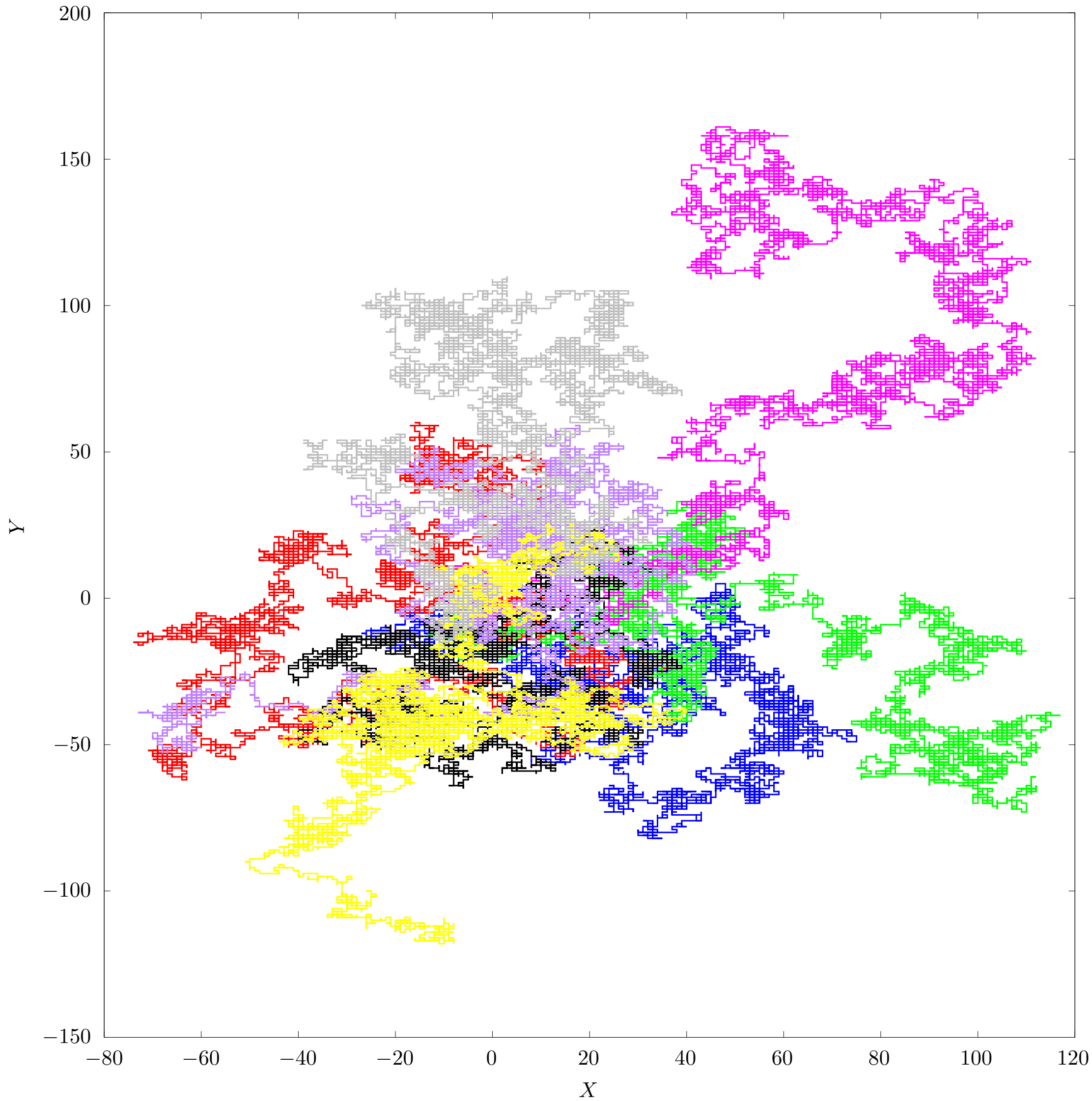

Figura 4: Gráfico da trajetória (X, Y) de 8 andarilhos para $N = 10000$ passos.

Note que no gráfico há um agrupamento grande próximo da origem e conforme o número de passos aumenta o sistema tende a ficar mais complexo, com vários andarilhos se espalhando de forma aleatória.

O código usado para gerar o gráfico das trajetórias é o `./tarefaC/tarefa-c-trajetoria.f`.

(D) Cálculo da entropia

Com intuito de calcular a entropia em presente no sistema de andarilhos aleatórios em duas dimensões, podemos utilizar a seguinte fórmula

$$S = -\sum_{i}^{N} P_i \ln(P_i) \tag{4}$$

onde P_i é a propriedade de se encontrar um micro-estado específico para i, foi dividido o espaço em um grid de tamanho arbitrário escolhido por mim $w = 5$, ou seja, vários quadrados de comprimento w e então calculado a probabilidade P_i desse micro-estado fazendo

$$P_i = \frac{\text{Quantidade de pontos no quadrado}}{\text{Quantidade total de andarilhos}} \tag{5}$$

a partir dessa relação podemos cálcular a entropia.

O cálculo envolvido nessa etapa foi apenas uma derivação do que já tinha sido feito na tarefa anterior, onde eram calculadas as posições finais de andarilhos em duas dimensões e assim mostrada a difusão, com alguns ajustes no código foi calculado a entropia para cada passo dado pelos M andarilhos e então armazenado no arquivo de saída `./tarefaD/saida-tarefa-d.dat`. A partir desse arquivo de saída foi feito o gráfico [grafioc]]. Fiz algumas tentativas de otimização do código, como por exemplo calcular os pontos dentro dos quadrados do grid a partir dos valores x e y mínimos cálculados a cada passo para todos andarilhos.

Segue abaixo o código criado para o cálculo da entropia desse sistema

Essa estrutura é semelhante à das tarefas anteriores

```fortran
!      Tarefa D - Cálcular entropia de caminhantes aleatorios em duas dimensoes
       parameter (m = 1000)
       parameter (n = 1000)

       dimension nWalker(m, 2)

       dimension iStep(2)

       iStep(1) = -1
       iStep(2) = 1

       rnd = rand(iseed)

       do j = 1, m
          nWalker(j, 1) = 0
          nWalker(j, 2) = 0
       end do
```

A partir desse trecho temos as mudanças principais feitas no código

```fortran
open(unit=10,file='saida-tarefa-d.dat')
do i = 1, n

   do j = 1, m
      k = rand() * 2
      k = int((k + 1) / 2) + 1
      l = rand() * 2
      l = int((l + 1) / 2) + 1
      nWalker(j, k) = nWalker(j, k) + iStep(l)
   end do

```

```
   ixmin = nWalker(1, 1)
   ixmax = ixmin
   iymin = nWalker(1, 2)
   iymax = iymin

   do j = 2, m
      if(nWalker(j, 1) < ixmin) then
         ixmin = nWalker(j, 1)
      end if
      if(nWalker(j, 1) > ixmax) then
         ixmax = nWalker(j, 1)
      end if
      if(nWalker(j, 2) < iymin) then
         iymin = nWalker(j, 2)
      end if
      if(nWalker(j, 2) > iymax) then
         iymax = nWalker(j, 2)
      end if
    end do
     entropy = 0e0
     isqLen = 5

       do ix = ixmin, ixmax, isqLen
          do iy = iymin, iymax, isqLen
             count = 0e0
             do j = 1, m
                iposX = nWalker(j,1)
                iposY = nWalker(j,2)
                if(((iposX <= ix + isqLen) .and. (iposX >= ix)) .and.
   $                 (iposY <= iy + isqLen .and. iposY >= iy)) then
                   count = count + 1
                end if
             end do
             if(count .ne. 0) then
                prob = count / m
                entropy = entropy - prob * log(prob)
             end if
          end do
       end do
       write(10, *) i, entropy
    end do
    close(10)
    end
```

A partir dos pontos cálculados e armazenados no arquivo `./tarefaD/saida-tarefa-d.dat` foi feito o gráfico abaixo

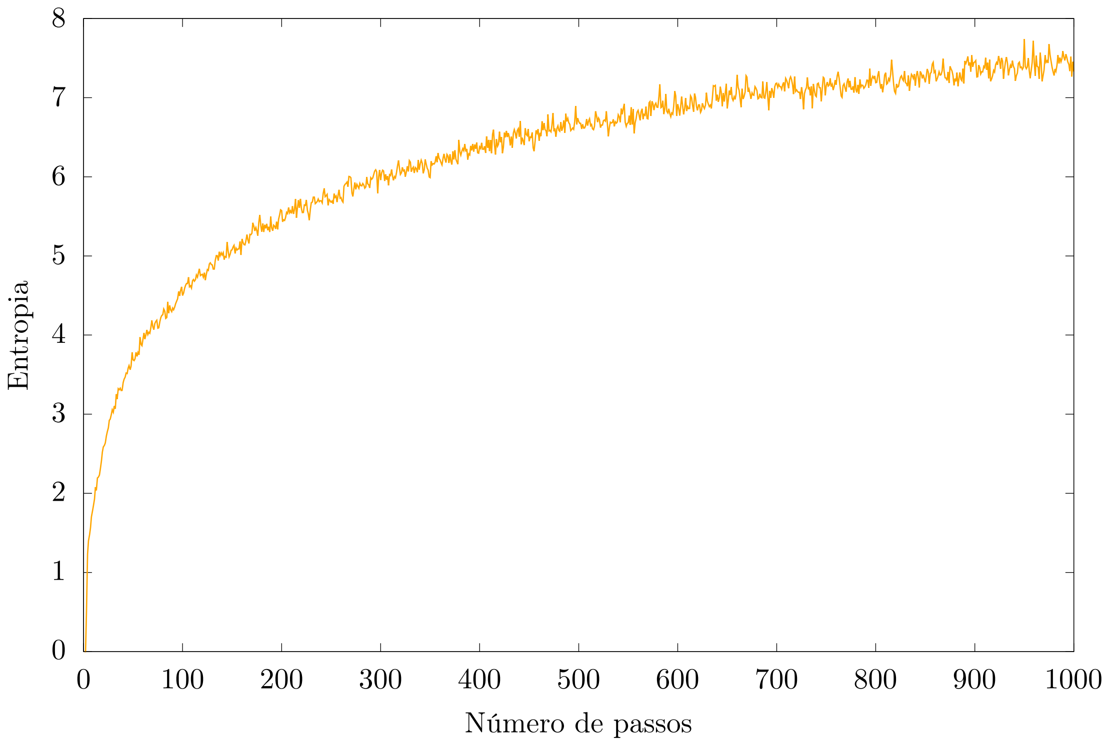

Figura 5: Gráfico de entropia para $M = 1000$ andarilhos.