Universidade de São Paulo
Instituto de Física de São Carlos
7600017 - Introdução à Física Computacional
Docente: Francisco Castilho Alcaraz

Projeto 02: Sistemas Aleatórios

Andressa Colaço
Nº USP: 12610389

São Carlos
2022

Sumário

1 Introdução

Esse projeto tem o objetivo de estudar os sistemas aleatórios através de uma abordagem probabilística. Será estudado o problema dos caminhantes aleatórios em 1 e 2 dimensões e o aumento da entropia deste sistema conforme aumenta-se o número de passos e os viajantes podem se distanciar mais uns dos outros no espaço.

2 Tarefa A

A fim de testar o gerador de números aleatórios, que será utilizado durante todo este projeto, realizamos o cálculo da média $< x^n >$ para $n = 1, 2, 3, 4$. Para tal, foi escrito o seguinte código:

```
      PROGRAM media_aleat
       dimension sum(1:4)

       write(*,*) 'Insira N:'
       read(*,*) N

       do i = 1, 4
           sum(i) = 0.0
       end do

       do j = 1, N
           x = rand()
           do i = 1, 4
               sum(i) = sum(i) + x**i
           end do
       end do

       do i = 1, 4
           sum(i) = sum(i)/N
       end do

      write(*,*) '<x> = ', sum(1)
      write(*,*) '<x²> =', sum(2)
      write(*,*) '<x³> =', sum(3)
      write(*,*) '<x^4> =', sum(4)

      stop
```

```
        end
```

Aqui, o número N deve ser dado pelo usuário - para números de ordem grande, as aproximações das médias serão mais coerentes com o esperado, enquanto, para N pequeno, tende-se a observar um desvio maior deste valor.

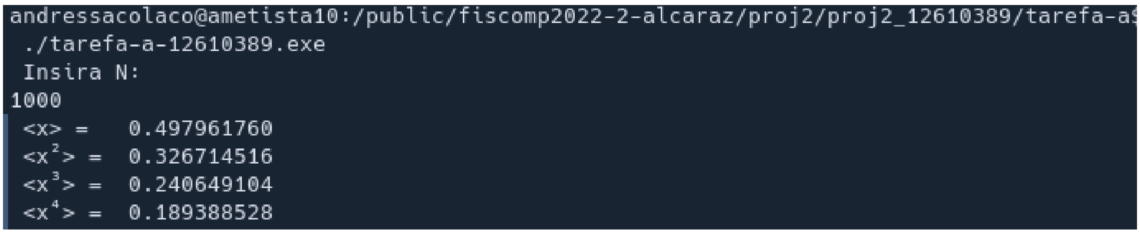
Logo em seguida, é criado um vetor com 4 posições, sendo que cada uma delas armazenará o resultado das somas parciais de x^n conforme os números aleatórios são gerados. O número da posição corresponde ao n de x calculado no momento. Antes de se começar a armazenar dados, é atribuído o valor 0 a cada soma, para evitar somas com valores obtidos no endereço de memória do vetor advindos de outros programas. Essa decisão foi tomada para não precisar chamar a função rand() separadamente para as quatro médias.

É criado um loop de 1 até N, que corresponde ao número de vezes que um número aleatório será gerado. Tal número é obtido e guardado na variável real x. Em seguida, entra-se em um loop interno que soma x^n à sua soma parcial correspondente.

Por fim, ao terminar as somas, divide-se cada valor por N, obtendo-se a média correspondente. Esse valor é então impresso na tela.

A saída para $N = 1000$ é:

Figura 1: Saída do código 1.



Podemos calcular os resultados obtidos estatisticamente com o resultado dado pela integral:

$$< x^n > = \int_0^1 x^n dx = \frac{1}{n+1} \tag{1}$$

Portanto, temos para $N = 1000, 10000, 100000$:

Tabela 1: $< x^n >$

n	$< x^n >$	N	Média obtida	|Erro|
1	0.5	1000	0,497961760	0.00203824
		10000	0.501827717	0.001827717
		100000	0.500281513	0.000281513
2	1/3	1000	0.326714516	0.006618817333
		10000	0.335473716	0.002140382667
		100000	0.333476216	0.0001428826667
3	0.25	1000	0.240649104	0.009350896
		10000	0.252227634	0.002227634
		100000	0.249979049	0.000020951
4	0.2	1000	0.189388528	0.010611472
		10000	0.202283651	0.002283651
		100000	0.199869081	0.00203824

Observamos que, como esperado pela teoria, conforme a amostra N aumenta, mais os valores se aproximam da média calculada com a integral.

3 Tarefa B

Na tarefa B, começamos a estudar o problema dos viajantes alatórios em 1 dimensão. Dado um número N de passos, a cada um deles um andarilho tem uma certa probabilidade p de andar para a direita em uma $q = 1 - p$ de andar para a esquerda. Após este número de passos, o andarilho se encontra em uma posição x.

Dado um número grande de andarilhos, podemos verificar que estes acabam terminando suas trajetórias em pontos diferentes, mas tende-se a notar que há uma maior concentração de andarilhos que terminam suas caminhadas em um determinado ponto e ela diminui conforme afasta-se desta posição.

Para visualizar o problema, foi escrito o seguinte código:

```
      PROGRAM unidimensional_random_walk
      parameter(p = 1.0/2.0)
      parameter(N = 1000)
      parameter(ip_dim = 1/p)
      parameter(iwidth=10)
      parameter(idim_h = (2*N/iwidth)+1)
      dimension ip(1:ip_dim) !vetor de controle de possibilidades
      dimension ihist(1:idim_h) !vetor do histograma

```

```fortran
      ip(1) = 1

      do k = 2, ip_dim
          ip(k) = -1
      enddo

      min_hist = -(idim_h)/2

      do i = 1, idim_h
          ihist(i) = 0
      end do

      open(10, FILE = 'saida-b-histograma1-12610389.dat')

      write(*,*) 'Insira o número M de andarilhos:'
      read(*,*) M

      sum1 = 0 ! <x>
      sum2 = 0 ! <x²>

      do i = 1, M
          ipos = 0 !posicao do andarilho

          do j = 1, N
              ix = (rand()/p) + 1
              ipos = ipos+ip(ix)
          end do

          sum1 = sum1 + ipos
          sum2 = sum2 + ipos**2

          ipos_hist = ipos/iwidth
          local = ipos_hist-min_hist+1
          ihist(local) = ihist(local)+1
      end do

c     Contar andarilhos para cada número de passos

      do i = 1, idim_h
          ipos_hist = i+min_hist-1
```

```
            write(10,*) ipos_hist*iwidth, ihist(i)
        end do

c       Média das posições

        sum1 = sum1/M
        sum2 = sum2/M

c       forma estatistica
        write(*,*) 'Resultado estatístico: '
        write(*,*) '<x> =', sum1
        write(*,*) '<x²> =', sum2

c       forma analitica
        q = 1 - p
        sum_an1 = N*(p-q)
        sum_an2 = (N*(p-q))**2 + 4*N*p*q

        write(*,*) 'Resultado analítico:'
        write(*,*) '<x>=', sum_an1
        write(*,*) '<x^2>=', sum_an2

        close(10)

        stop
        end
```

A lógica empregada foi a seguinte: em primeiro lugar, declaram-se os parâmetros que serão utilizados durante o código. A variável p recebe a probabilidade de se dar um passo para a direita e pode ser ajustada para qualquer valor (desde que $p \leq q$) enquanto N recebe o número de passos que serão simulados. ip_dim e ip são, respectivamente, a dimensão do vetor de possibilidades de caminhada e o vetor em si. Seu uso será explicado na geração dos passos. As variáveis restantes se referem ao histograma que será gerado pelo programa: i_width é a largura dos intervalos deste, a qual é escolhida arbitrariarmente; $idim_h$ é a dimensão do vetor que guarda as informações do histograma, que é igual ao número de intervalos obtidos através da divisão do intervalo $[-N, N]$ pela largura de cada um; e $ihist$ é o vetor em questão.

Logo após a declaração das variáveis, define-se o vetor de possibilidades ip. Ele foi criado de forma com a qual seu número de elementos é $1/p$ e seu primeiro elemento é 1 (passo à direita) e os demais -1 (passo à esquerda), de forma que, gerando números aleatórios no intervalo do tamanho do vetor, o passo à direita tenha probabilidade $1/p$ de ser escolhido. Por exemplo, se $p = 1/2$, é gerado o vetor $\vec{ip} = (1, -1)$ e são gerados números aleatórios que assumem os valores 1 ou 2. Acessando o valor do índice correspondente ao número aleatório gerado, adicionamos ou subtraímos um passo na linha, o que corresponde a andar para a direita ou para a esquerda. A lógica é similar para $p = 1/3, 1/4$ e $1/5$, que terão mais valores -1, ou seja, possuem maior chance de ir para a esquerda.

As declarações foram estruturadas desta maneira para permitir que p assuma o valor que se queira sem precisar criar outros arquivos ou modificar o restante do código. Precisaria-se fazer algumas modificações caso p fosse maior que q, porém, como neste projeto isto não se fez necessário, esta implementação não foi adicionada.

Em sequência a este vetor, foi também definida a variável min_hist, que demarca o valor mínimo do eixo x do histograma. Ela auxiliará na escrita dos dados, já que vetores têm sua indexação iniciada do 1 e teremos valores negativos sendo armazenados no começo deles.

Também, ainda antes de começar a gerar as posições, é lido do usuário o número M de andarilhos que se queira simular e as variáveis $sum1$ (que guardará a soma de $< x >$) e $sum2$ (que guardará a soma de $< x^2 >$) são inicializadas em 0.

Finalmente, é feito um laço que vai de 1 até M, ou seja, roda a geração das posições para cada andarilho. Entrando nesse laço, a posição *ipos* do caminhante é inicialmente 0 e, logo a seguir, gera-se os N passos definidos para ele da seguinte maneira: para cada passo, é gerado um número pseudo-aleatório (no intervalo marcado por $1/p$) e adiciona-se 1 a este valor já que *ip* começa em 1; em sequência, é adicionado ou subtraído 1 à posição do caminhante conforme o número pseudo-aleatório gerado. Ao final, o caminhante se encontra em uma determinada posição *ipos*. Essa posição, que também pode ser chamada de x, é então somada à soma que dará origem à média $< x >$ e seu quadrado é adicionado à soma que dará origem à média $< x^2 >$. Por fim, deve-se adicionar este caminhante ao intervalo do histograma no qual sua posição final caiu. Para isso, é feita uma divisão da coordenada x pela largura do intervalo e pega a parte inteira; em sequência, adiciona-se o caminhante ao intervalo, através do vetor que guarda suas informações. Os *local* é o índice onde o caminhante deve ser adicionado e está ajustado considerando que *ihist* vai de $-N/largura$ até $N/largura$.

Ao terminar de gerar os passos dos M andarilhos, precisamos determinar quantos deles terminaram em cada intervalo. Essas informações já estão prontas no vetor, então apenas basta escrever o intervalo dos passos e sua respectiva contagem em um arquivo.

Por fim, as somas mencionadas anteriormente são divididas pelo número de andarilhos, gerando as médias estatísticas $< x >$ e $< x^2 >$, que são impressas na tela. Por fim, as médias também são calculadas da forma analítica:

$$< x >= N(p - q) \tag{2}$$

e

$$< x^2 >= (N(p-q))^2 + 4Npq \tag{3}$$

As saídas do programa serão discutidas nos items 3.1 e 3.2.

3.1 Tarefa B1

Neste primeiro item, consideraremos o caso onde $p = q = 1/2$. O código apresentado na seção B é ajustado para $p = 1/2$ e é escolhido o número de andarilhos $M = 10000$. Executando-o, temos no terminal as médias obtidas estatistica e analiticamente:

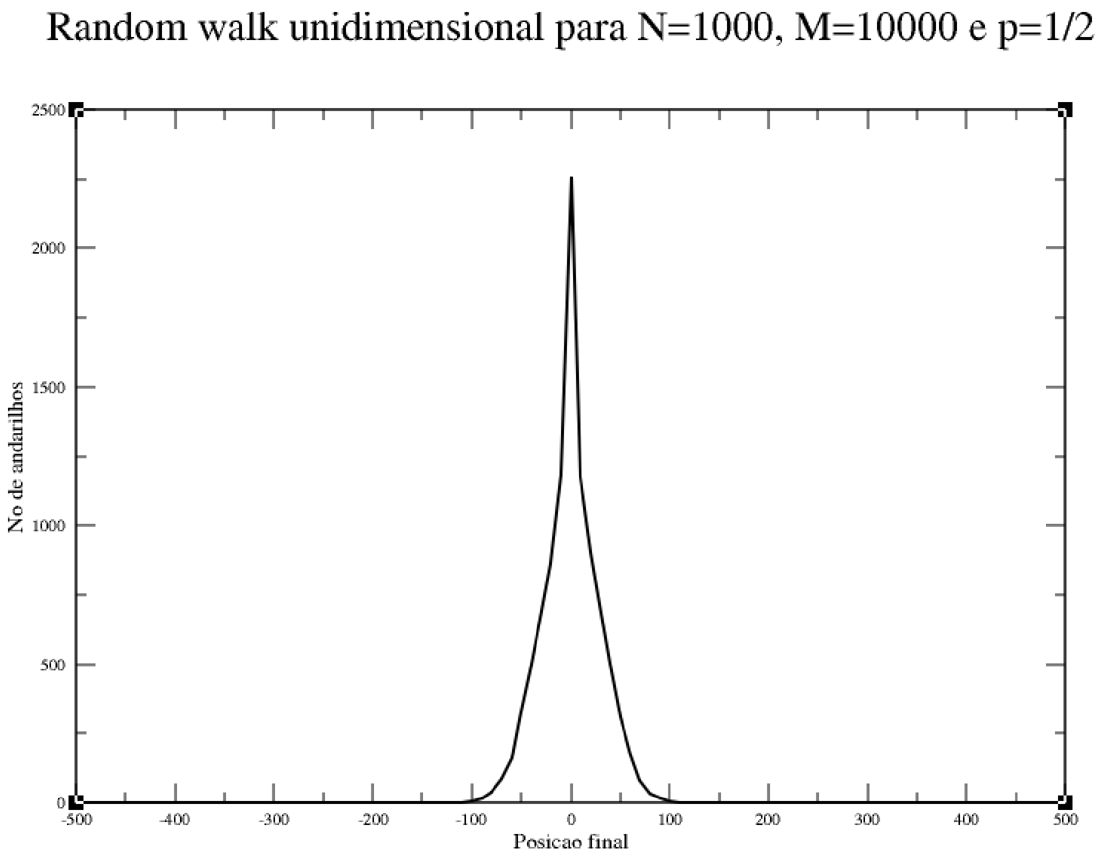
Figura 2: Saída do código B1.

```
andressacolaco@ametista10:/public/fiscomp2022-2-alcaraz/proj2/proj2_12610389/tarefa-b$
 ./tarefa-b-12610389.exe
 Insira o número M de andarilhos:
10000
 Resultado estatístico:
 <x> =   9.16000009E-02
 <x^2> =   999.865601
 Resultado analítico:
 <x>=   0.00000000
 <x^2>=   1000.00000
```

Ambas se apresentam em concordância, conforme esperado a partir do teste realizado na seção A. Vemos então que a primeira média é idealmente próxima a 0 e a segunda, ao número de passos simulados.

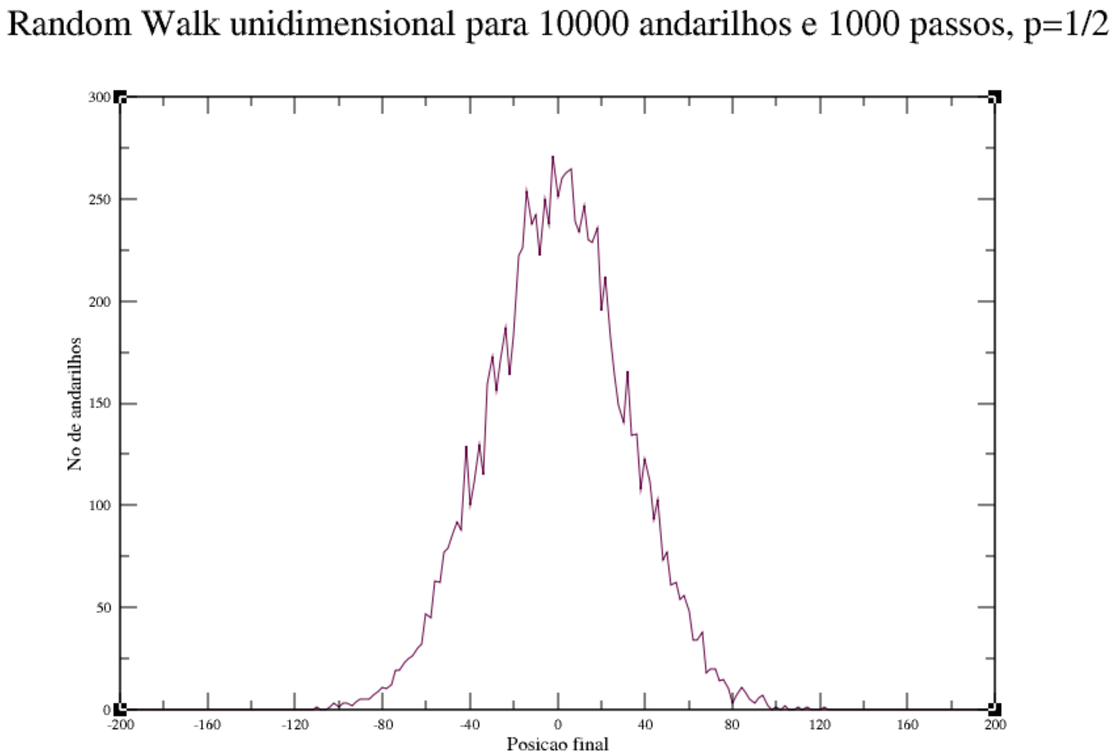
A outra saída do programa se encontra no arquivo *'histograma-andarilhos-12610389.dat'*. Utilizando o *grace* para realizar o plot dos dados deste arquivo, obtemos o seguinte gráfico:

Figura 3: Plot do random walk para N=1000, M=10000 e $p = 1/2$.



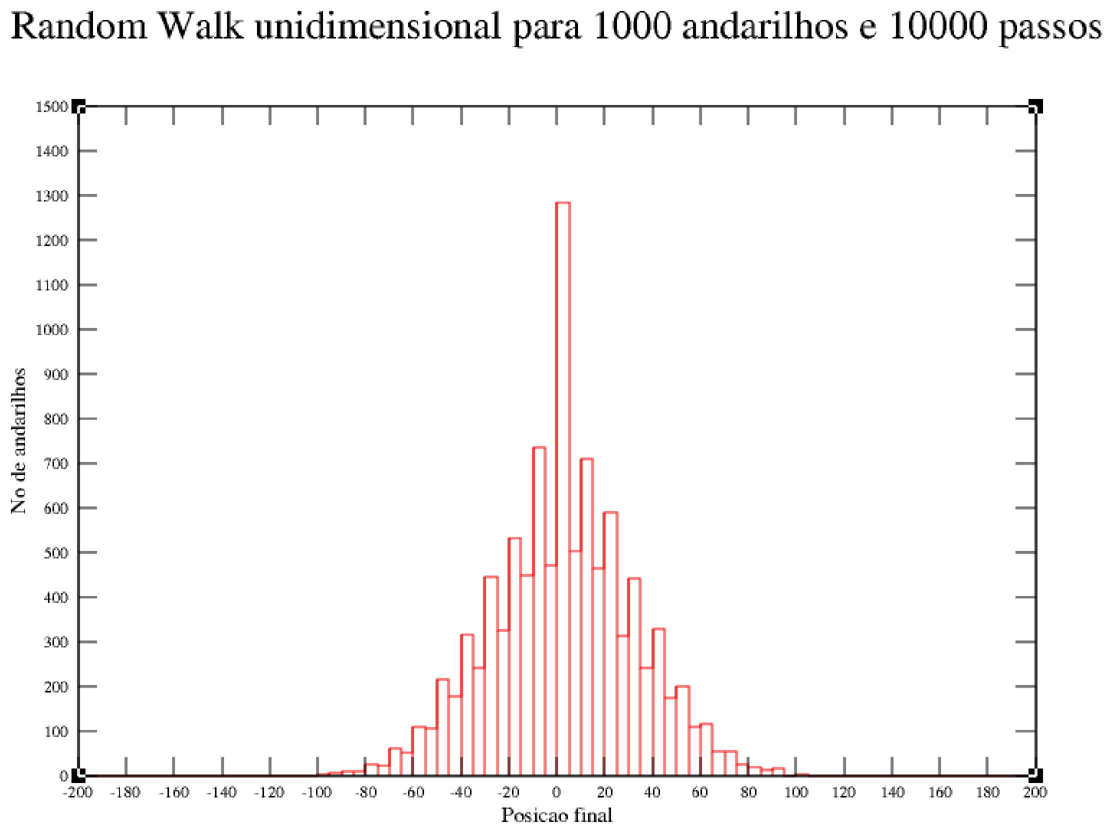
Podemos ver através deste gráfico que a curva para a qual ele se aproxima tem a forma de uma distribuição gaussiana centrada na origem, conforme esperado. Por conta da escolha do intervalo e do plot, vemos que o formato tem muitas linhas retas. Se diminuído o intervalo para 2, conseguimos plots como o seguinte:

Figura 4: Plot do random walk para N=1000, M=10000 e $p = 1/2$.



Que evidencia o formato da gaussiana. A título de visualização e experimentação com o código, também colocou-se a disposição de barras para intervalos de tamanho 5, que gerou o seguinte gráfico:

Figura 5: Histograma do random walk para N=1000, M=10000 e $p = 1/2$.



Na pasta entregue, os gráficos estão nomeados na ordem em que aparecem no relatório. Na próxima tarefa seguirá-se com o padrão do primeiro gráfico.

3.2 Tarefa B2

Utilizando os valores de $p = 1/3, 1/4$ e $1/5$ repete-se o procedimento descrito no item anterior. As saídas no terminal do programa são:

Para $p = 1/3$:

Figura 6: Saída do código B2, $p = 1/3$.

```
andressacolaco@ametista10:/public/fiscomp2022-2-alcaraz/proj2/proj2_12610389/tarefa-b$
 ./tarefa-b-12610389.exe
 Insira o número M de andarilhos:
10000
 Resultado estatístico:
 <x> =  -333.355194
 <x²> =   112014.898
 Resultado analítico:
 <x>=  -333.333282
 <x^2>=   111999.969
```

Para $p = 1/4$:

Figura 7: Saída do código B2, $p = 1/4$.

```
andressacolaco@ametista10:/public/fiscomp2022-2-alcaraz/proj2/proj2_12610389/tarefa-b$
 ./tarefa-b-12610389.exe
 Insira o número M de andarilhos:
10000
 Resultado estatístico:
 <x> =  -499.931000
 <x²> =   250687.797
 Resultado analítico:
 <x>=  -500.000000
 <x^2>=   250750.000
```

E para $p = 1/5$:

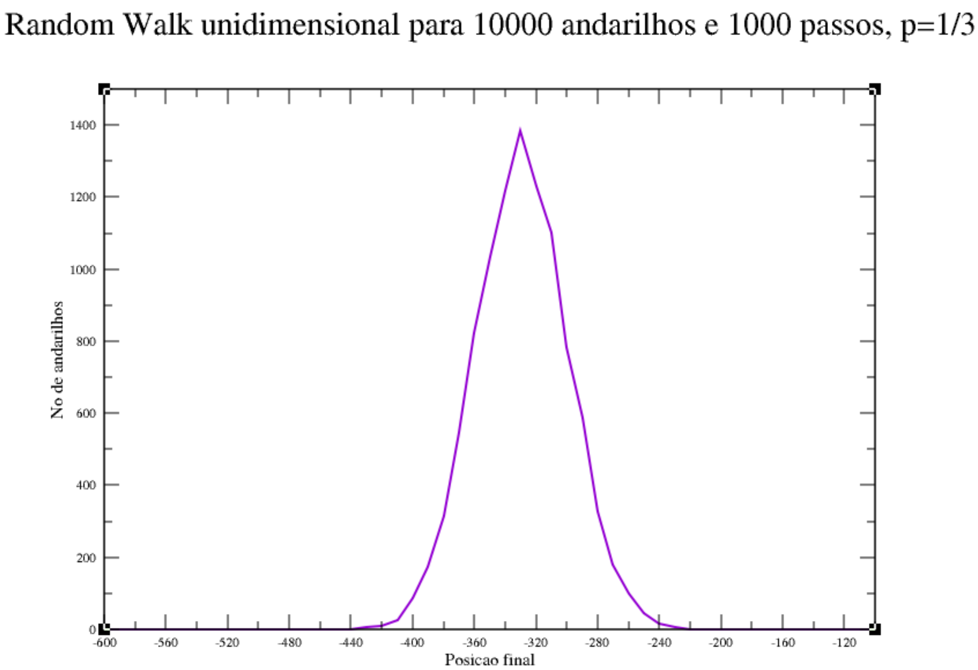
Figura 8: Saída do código B2, $p = 1/5$.

```
andressacolaco@ametista10:/public/fiscomp2022-2-alcaraz/proj2/proj2_12610389/tarefa-b$
 ./tarefa-b-12610389.exe
 Insira o número M de andarilhos:
10000
 Resultado estatístico:
 <x> =  -599.992798
 <x²> =   360641.688
 Resultado analítico:
 <x>=  -600.000000
 <x^2>=   360640.000
```

Podemos notar que as médias calculadas estão muito próximas das médias obtidas pelo método de gerar números aleatórios. Para visualizar as distribuições, os gráficos gerados são:

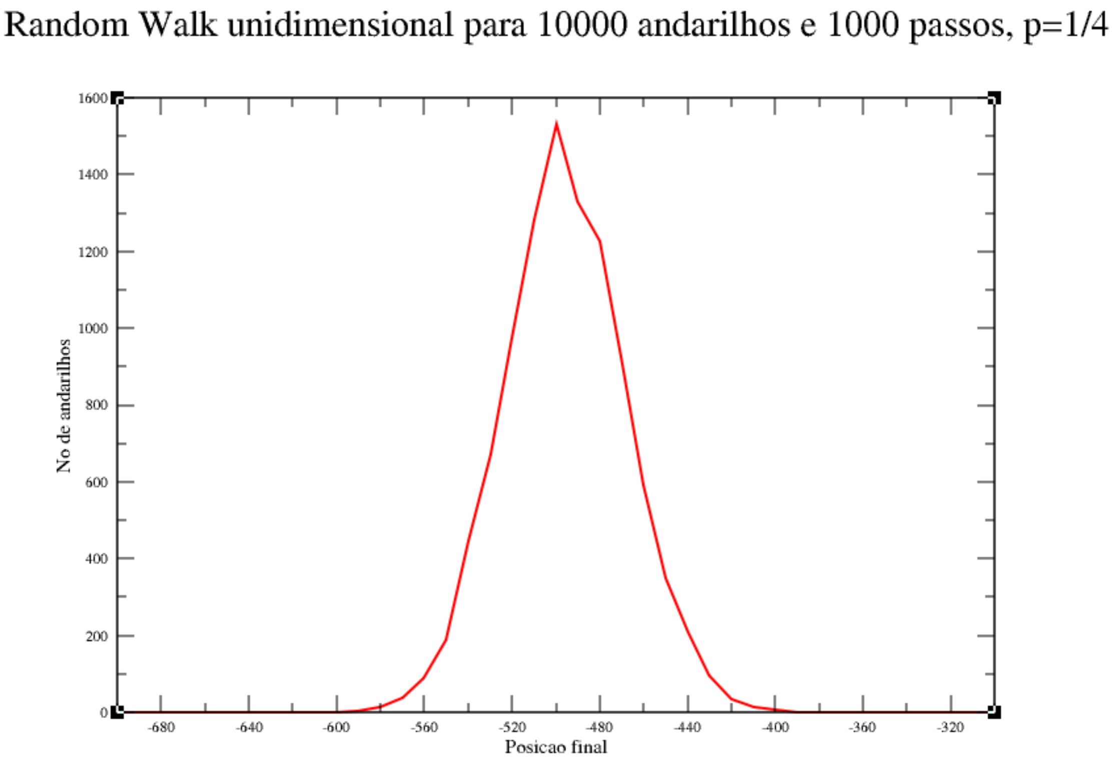
Para $p = 1/3$:

Figura 9: Plot do random walk para N=1000, M=10000 e $p = 1/3$.



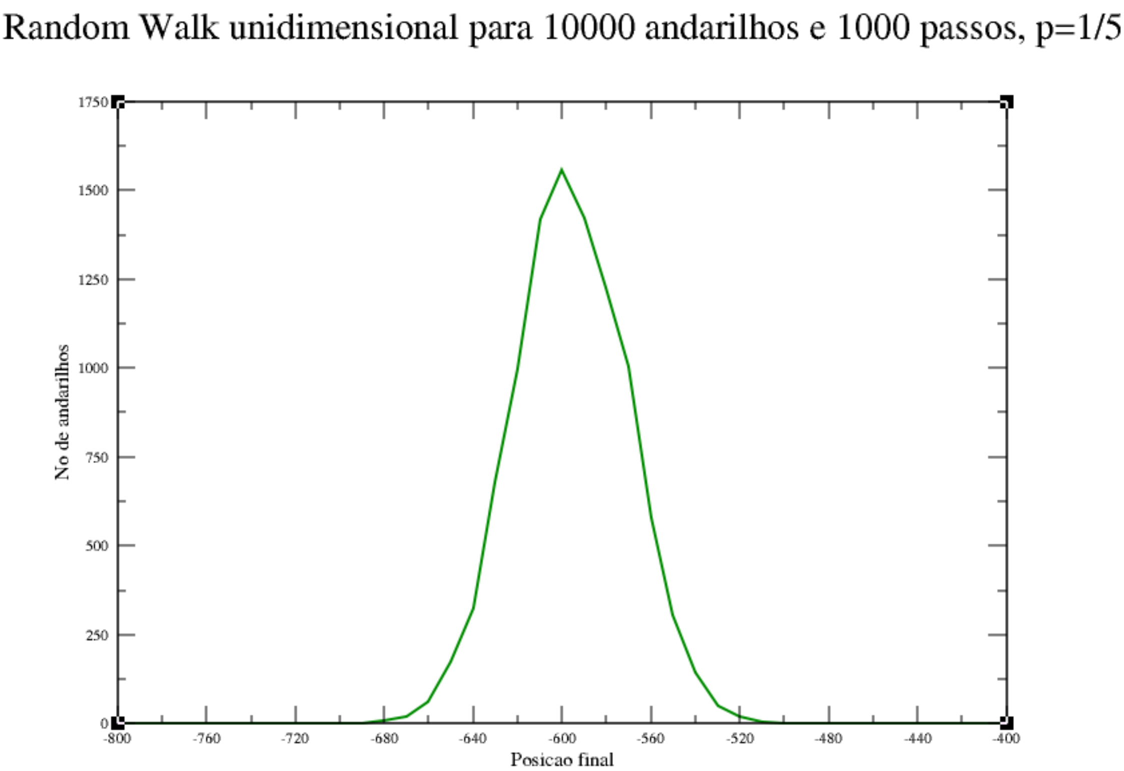
Para $p = 1/4$:

Figura 10: Plot do random walk para N=1000, M=10000 e $p = 1/4$.



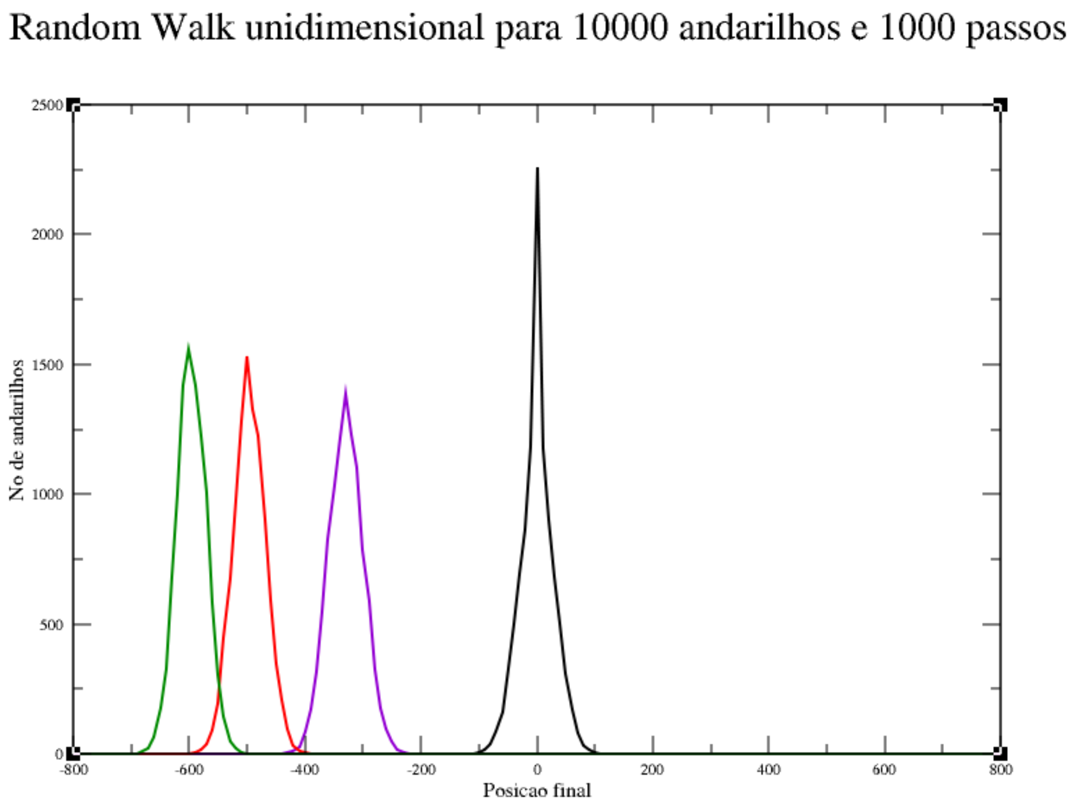
E para $p = 1/5$:

Figura 11: Plot do random walk para N=1000, M=10000 e $p = 1/5$.



Nota-se que estes gráficos possuem a mesma forma do primeiro, no item B1, porém estão deslocados da origem para posições mais à esquerda. Colocando-o todos os casos em um plot só, podemos visualizar melhor esta diferença:

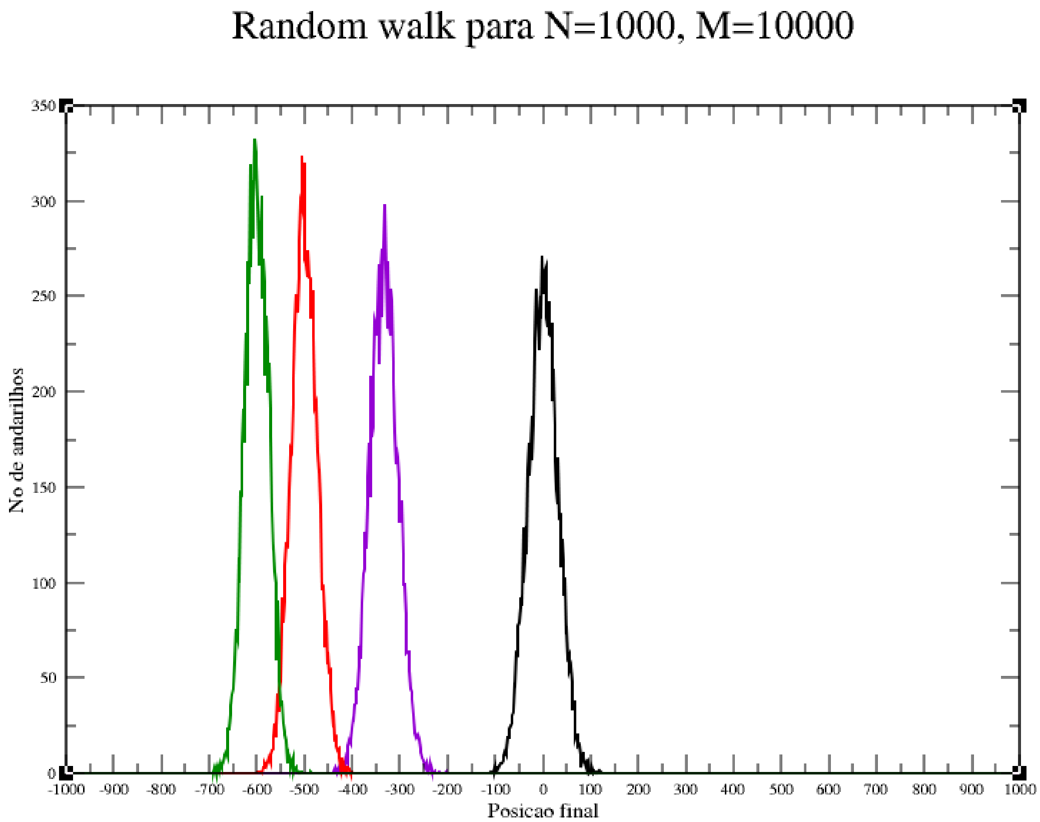
Figura 12: Plot do random walk para $p = 1/2$ (preto), $p = 1/3$ (roxo), $p = 1/4$ (vermelho) e $p = 1/5$ (verde).



É notável que, ao diminuir a probabilidade de se dar um passo à direita, os caminhantes tendem a terminar em posições mais afastadas da origem, à esquerda. Quanto menor p, a distribuição se centra em valores mais negativos. Também é notável que há mais caminhantes no centro da distribuição de $p = 1/2$, o que indica que as outras se distribuem mais no espaço do que a primeira.

A título de visualização, também diminuiu-se o intervalo das janelas para 2, a fim de observar ainda melhor o formato das curvas. Obteve-se o seguinte gráfico:

Figura 13: Plot do random walk para $p = 1/2$ (preto), $p = 1/3$ (roxo), $p = 1/4$ (vermelho) e $p = 1/5$ (verde).



4 Tarefa C

Nesta seção, vamos trabalhar com o problema dos caminhantes aleatórios em 2 dimensões. O código implementado para esta questão é muito similar ao unidimensional, porém alguns detalhes precisaram ser adaptados para a adição de 1 dimensão. Além disso, por conta do objetivo da tarefa, de mostrar a distribuição das posições finais dos andarilhos no espaço após N passos (onde N assume os valores $N = 10, 100, 1000, 10^5$ e 10^6), foi definido um loop onde os dados dos passos de cada um são registrados em um arquivo correspondente a N para cada um dos valores que esta variável assume. A implementação pode ser vista a seguir, com sua explicação em detalhes após o código:

```
        PROGRAM bidimensional_random_walk
        parameter(p = 1.0/4.0)
        parameter(M=10000)
        dimension ipx(1:4), ipy(1:4) !vetores de controle de possibilidades
        parameter(ipx=(/1 , 0, -1, 0/))
        parameter(ipy=(/0, 1, 0, -1/))

        open(10, FILE = 'saida-c-1-12610389.dat')
```

```fortran
      open(20, FILE = 'saida-c-2-12610389.dat')
      open(30, FILE = 'saida-c-3-12610389.dat')
      open(40, FILE = 'saida-c-4-12610389.dat')
      open(50, FILE = 'saida-c-5-12610389.dat')
      open(60, FILE = 'saida-c-6-12610389.dat')

      write(*,*) 'Número de andarilhos: ', M

      do k = 1, 6
          N = 10**k

          xsum = 0.0
          ysum = 0.0
          xsum2 = 0.0
          ysum2 = 0.0

          do i = 1, M
              ipos_x = 0 !posicao do andarilho
              ipos_y = 0

              do j = 1, N
                  ir = (rand()/p)+1
                  ipos_x = ipos_x + ipx(ir)
                  ipos_y = ipos_y + ipy(ir)
              end do

               io = k*10
              write(io,*) ipos_x, ipos_y

              xsum = xsum + ipos_x
              ysum = ysum + ipos_y

              xsum2 = xsum2 + ipos_x**2
              ysum2 = xsum2 + ipos_y**2
          end do

c     Média das posições
          r_norma = sqrt((xsum/M)**2+(ysum/M)**2)

```

```
c       Δ**2
            pescalar = (xsum/M)**2+(ysum/M)**2
            disp = (xsum2/M)+ (ysum2/M) - pescalar

            write(*,*) 'N = ', N
            write(*,*) '<r> =', r_norma
            write(*,*) '<Δ²> =', disp
            write(*,*) '-----------------'

        end do
        close(10)
        close(20)
        close(30)
        close(40)
        close(50)
        close(60)

        stop
        end
```

A primeira mudança feita em relação ao código anterior é que agora consideraremos apenas um valor para p e por isso variáveis como o vetor de possibilidades agora são definidas diretamente. Sendo $p = 1/4$ (possibilidades iguais de caminhada para cada ponto cardeal), teremos agora dois vetores cujas posições serão acessadas a fim de adicionar ou subtrair um passo da posição atual do andarilho. No momento em que os números pseudo-aleatórios forem gerados (entre 1 e 4), teremos uma variável controlando a posição x e outra a posição y do andarilho. Para definir qual número corresponde a cada coordenada, utiliza-se os quatro quadrantes sendo que: 1 será um passo à direita em x; 2 será um passo para cima; 3 para a esquerda e 4 para baixo. Seguindo os índices dos vetores, sabemos então que $\vec{ipx} = (1, 0, -1, 0)$ (pois em 2 e 4 não há movimentação em x) e $\vec{ipy} = (0, 1, 0, -1)$ (em 1 e 3 não há movimentação em y).

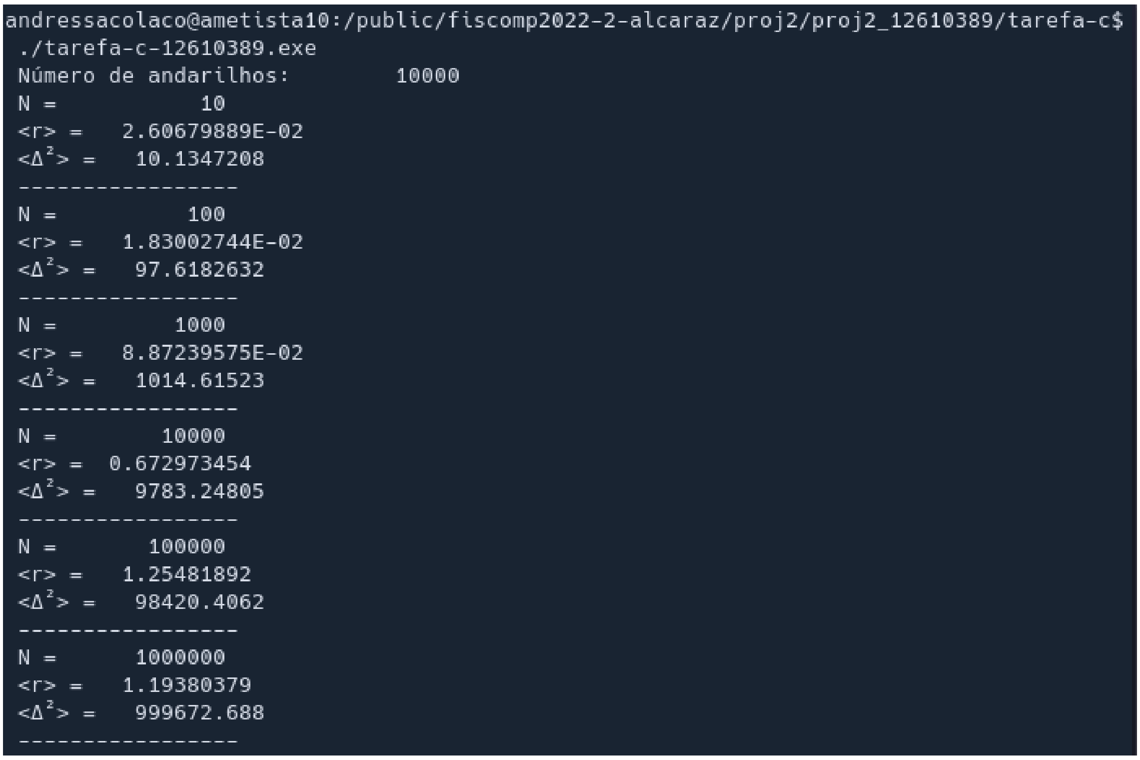
A seguir, são abertos os arquivos que armazenarão cada grupo de informações para N. É requerido o número M de andarilhos e então começa-se os procedimentos para cada N.

Entrando no loop, definimos N e então inicializamos as somas das cooordenadas que darão origem às médias $< \vec{r} >$ e Δ^2 (sendo $\Delta^2 =< \vec{r}^2 > - < \vec{r} > \cdot < \vec{r} >$). Da mesma maneira que no código unidimensional, entramos em um loop que calcula as posições finais de cada andarilho gerando números aleatórios para cada um dos N passos e adicionando ou subtraíndo um passo da coordenada correspondente de acordo com o definido no parágrafo anterior. Ao final da geração dos passos, escreve-se a posição final do andarilho no arquivo correspondente, soma-se as coordenadas à primeira soma e soma-se o quadrado delas à segunda soma.

Ao terminar o loop dos caminhantes, são calculadas $<\vec{r}>$ e Δ^2. Para $<\vec{r}>$, calcula-se a norma do vetor obtido com as médias das posições. Para Δ^2, é calculado o produto escalar $<\vec{r}>\cdot<\vec{r}>$, o qual é subtraído de $<\vec{r}^2>$ (onde $<\vec{r}^2>=<\vec{r}\cdot\vec{r}>$). Esses valores são impressos na tela e encerra-se o loop externo e o programa.

A saída no terminal para $M = 10000$ é:

Figura 14: Saída do código C.

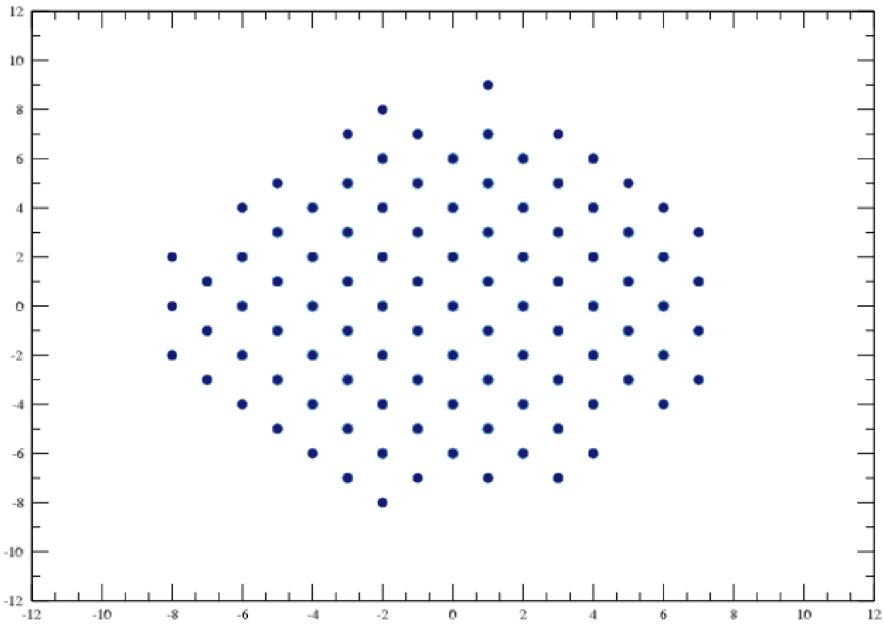


É notável que as médias das distribuições ficam todas muito próximas de 0, conforme esperado considerando o caso unidimensional. Também é notável que Δ^2 se aproxima do número de passos utilizado, fato que também foi observado no caso anterior.

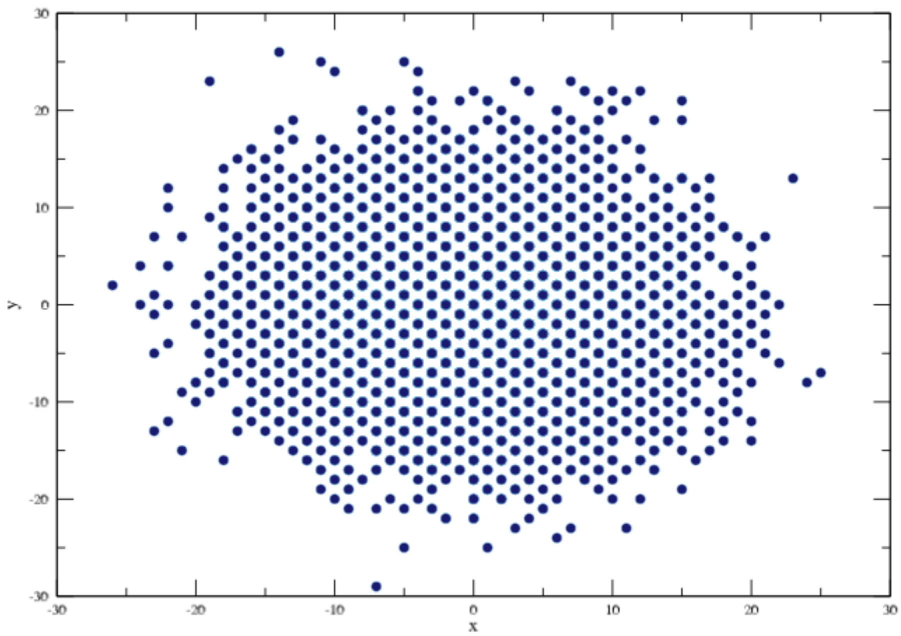
Quanto aos arquivos, estes foram utilizados para criar os seguintes diagramas:

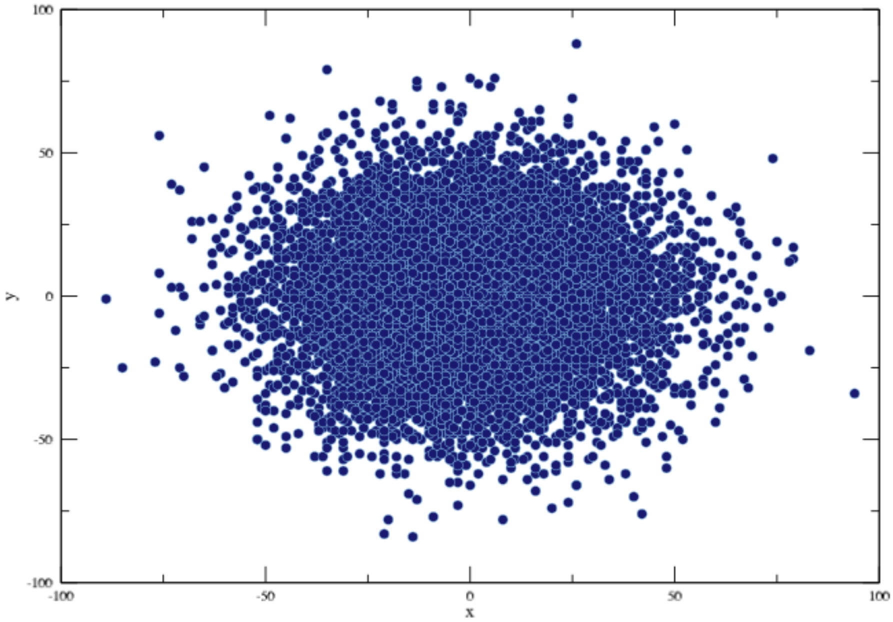
Figura 15: Gráficos de distribuição

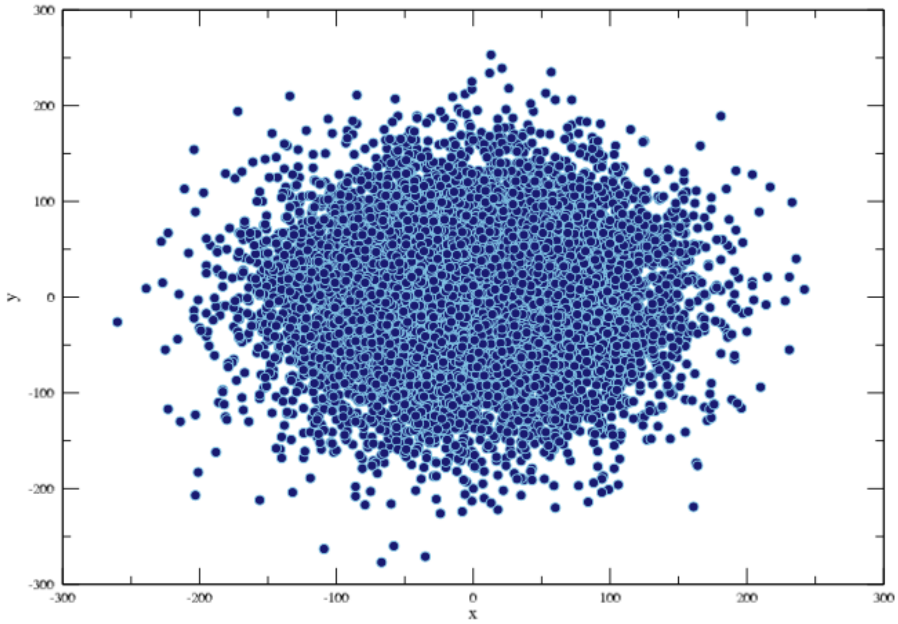
(a) 10^1

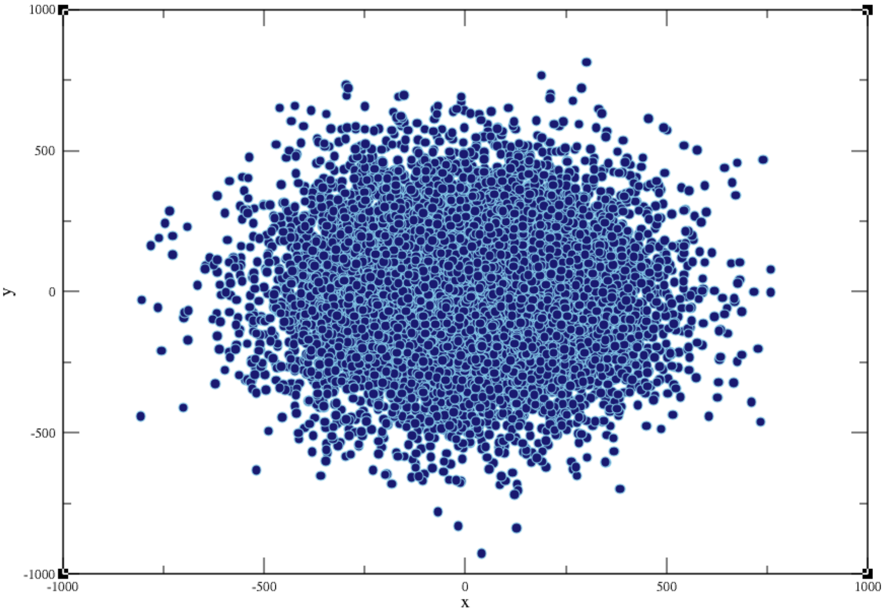


(b) 10^2

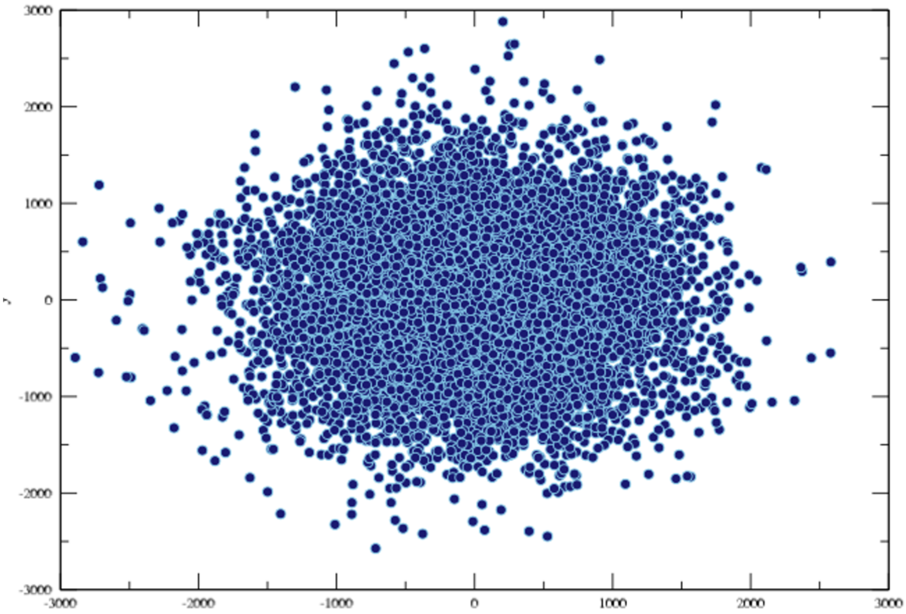



100
50
0
-50
-100
y
-100
-50
0
50
100
x
(c) 10^3


300
200
100
0
-100
-200
-300
y
-300
-200
-100
0
100
200
300
x
(d) 10^4



(e) 10^5



(f) 10^6

Com o auxílio visual, é fácil notar que, conforme o número de passos aumenta, mais os andarilhos (ou, neste caso, podemos considerá-los moléculas) se afastam da origem e se afastam entre si; porém, ainda é notável que haverá uma maior concentração próximo à origem (probabilidades iguais para as quatro direções), o que é verificável olhando para as distâncias médias $<\vec{r}>$ dos caminhantes à origem (muito próximas de 0 em comparação com o tamanho do espaço onde podiam se espalhar).

5 Tarefa D

Notando a forma com a qual as distribuições anteriores são dadas, podemos observar como se dá a variação da entropia em um sistema modelado como o problema dos caminhantes aleatórios em 2 dimensões. Para isso, calcula-se a entropia em diferentes instantes, os quais definiremos como sendo o intervalo necessário para caminhar 100 passos. A equação para encontrar a entropia nessa situação é:

$$S = -\sum_i P_i ln P_i \tag{4}$$

Onde i é um "micro-estado"e P_i a probabilidade de encontrar o sistema naquele micro-estado. Para definir os micro-estados, definimos reticulados de lado $iwidth$ que dividirão o espaço. Contando quantos caminhantes acabam em cada reticulado e dividindo este valor pelo número de caminhantes, encontramos P_i, a cada instante de tempo. Ao final, teremos uma relação entre instante e S que pode ser graficada a fim de melhor investigar a variação de entropia do sistema.

O código implementado se encontra abaixo. Sua descrição detalhada se encontra logo após.

```
      PROGRAM entropy
      parameter(p = 1.0/4.0)
      parameter(Nmax=10000)
      parameter(M=10000)
      parameter(iwidth = 10)

      parameter(idim_m = (2*Nmax/iwidth)+1)
      dimension m_count(idim_m, idim_m)

      parameter(ip_dim = 1/p)
      dimension ipx(ip_dim), ipy(ip_dim)
      parameter(ipx=(/1 , 0, -1, 0/))
      parameter(ipy=(/0, 1, 0, -1/))

      open(10, file = 'saida-d-12610389.dat')

c     Inicializando os vetores em 0
      do i = 1, idim_m
           do j = 1, idim_m
               m_count(i,j) = 0
           end do
       end do
```

```
    min_mat = -((idim_m-1)/2)

    write(10, *) 0, 0

    do Np = 100, Nmax, 100
        S_total = 0.0

        do i = 1, M

            ix = 0
            iy = 0

            do j=1, Np
                ir = (rand()/p)+1
                ix = ix + ipx(ir)
                iy = iy + ipy(ir)
            end do

            ix = ix/iwidth
            ixlocal = ix-min_mat+1

            iy = iy/iwidth
            iylocal = iy-min_mat+1

            m_count(ixlocal,iylocal) = m_count(ixlocal,iylocal) + 1
        end do

        do k = 1, idim_m
            do l = 1, idim_m
                if (m_count(k,l) .ne. 0) then
                    Pi = m_count(k,l)*(1.0)/M
                    S = Pi*log(Pi)
                    S_total = S_total - S
                    m_count(k,l) = 0
                end if
            end do
        end do

        write(10,*) Np, S_total
```

```
        end do

        close(10)

        end program
```

A lógica seguida neste código é muito parecida com a criação dos histogramas, na tarefa b, e com a implementação dos vetores de possibilidades de caminhada da tarefa c. Sendo assim, começamos com $p = 1/4$, e uma escolha para N e M (preferencialmente grande, pois os valores de entropia são relativamente pequenos em comparação e teremos melhores resultados aumentando estes valores), sendo N o valor máximo de passos, além de uma escolha para o lado dos retículos. Em seguida, é definido o tamanho da matriz que armazenará o número de andarilhos por retículo. Seu funcionamento é igual ao do histograma, assim como a definição de suas dimensões, mas agora temos uma dimensão a mais para cuidar. Então, definimos os vetores de possibilidades tal qual na tarefa C e abrimos um arquivo que armazenará os dados que gerarão o gráfico de entropia.

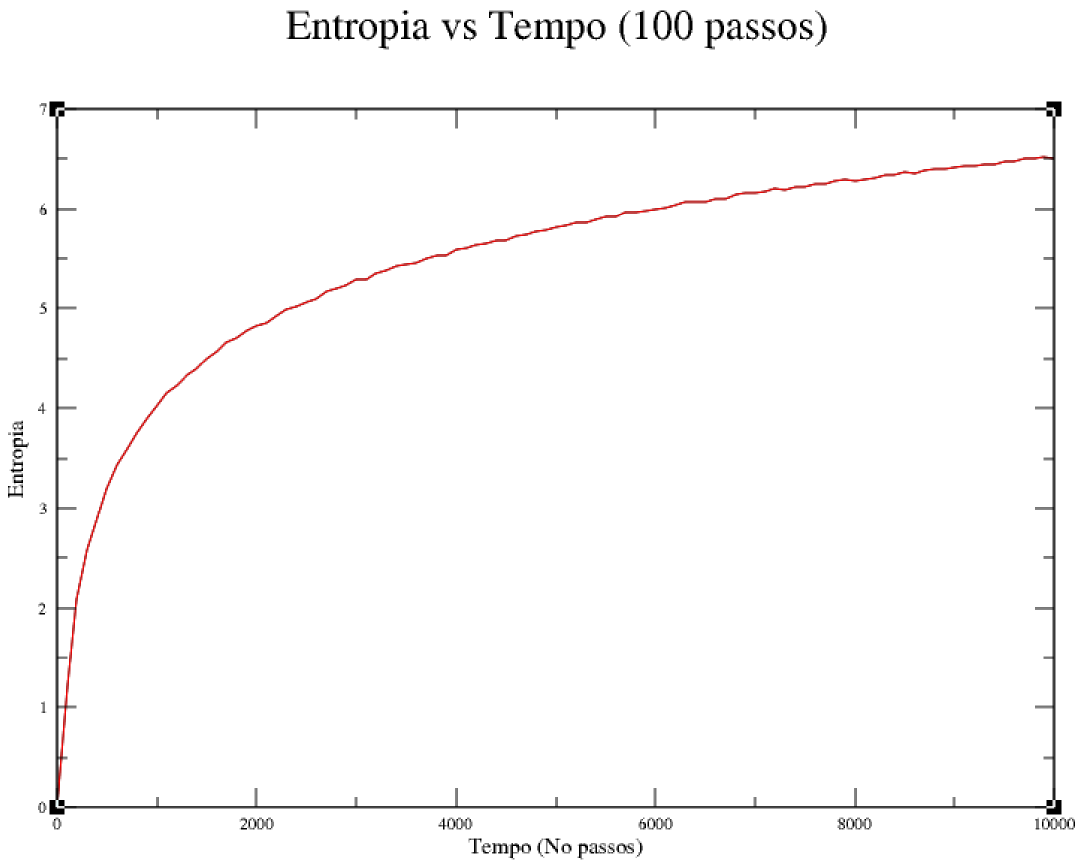
Logo em sequência, a matriz m_{count}, que armazena a contagem de andarilhos, é inicializada em 0 e é definida uma variável min_{mat} que é o valor mínimo assumido pelas divisões de N, declarado para ajustar os índices da matriz, assim como feito para o vetor histograma na tarefa B. Também se escreve os valores 0 como tempo inicial e valor de entropia inicial do sistema no arquivo antes de começar as iterações.

Como dito, são definidos intervalos de tempo como sendo 100 passos. Desta maneira, é feito um loop que calcula o valor S para cada Np em um intervalo começando por 100 e incrementando a contagem a cada iteração por esse mesmo valor. S é inicializada em 0 e então, entra-se em um loop que calcula os passos para cada andarilho e dá suas posições finais nas variáveis ix e iy. Esses valores são divididos pela largura do retículo considerado, cuja parte inteira indica em qual retículo o caminhante parou. *ixlocal* e *iylocal* são os índices de cada retículo na matriz e são definidos de forma a ajustar onde os os caminhantes devem ser adicionados, passo executado na linha seguinte.

Ao terminar as trajetórias dos caminhantes, é iniciado um loop para calcular o valor de S para cada retículo, subtraindo-o do valor total. O espaço é então resetado para 0 para ser utilizado na próxima iteração.

Por fim, o intervalo considerado e o seu valor de entropia são escritos em um arquivo. Com este, podemos gerar o seguinte gráfico:

Figura 16: Entropia vs Tempo (100 passos).

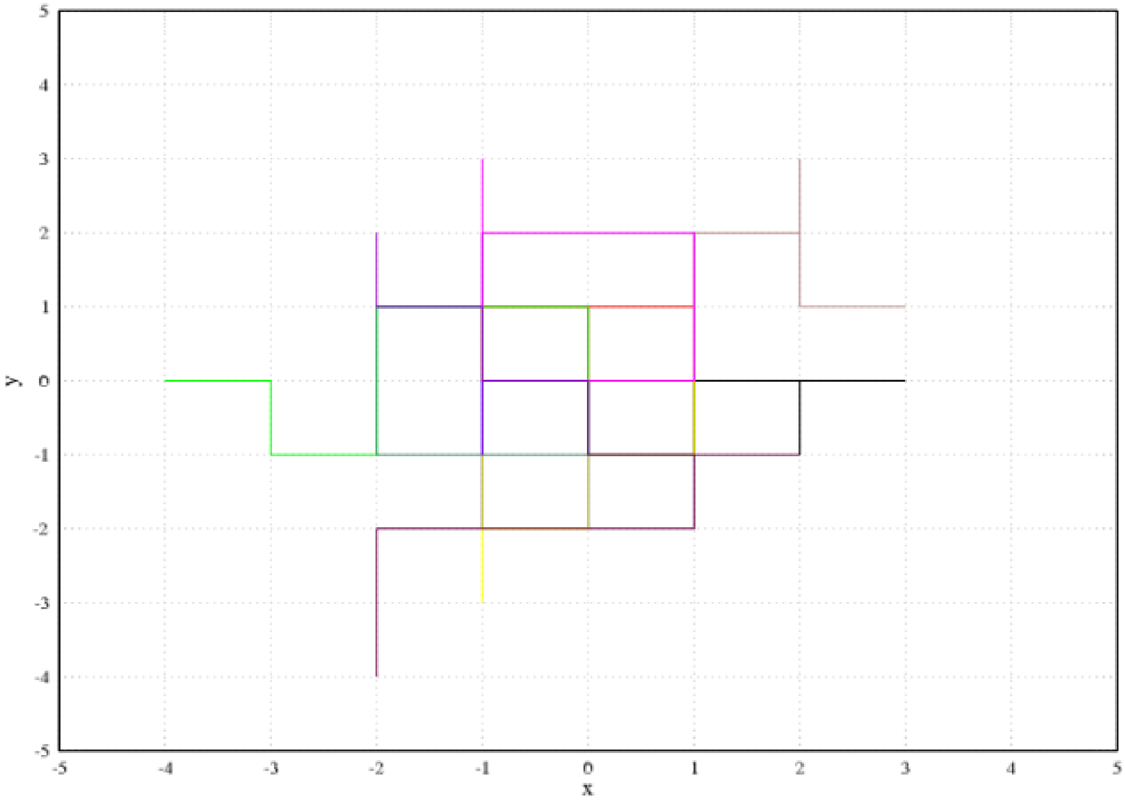


O qual evidencia o aumento da entropia com a evolução do sistema e o seu comportamento logarítmico.

6 Apêndices

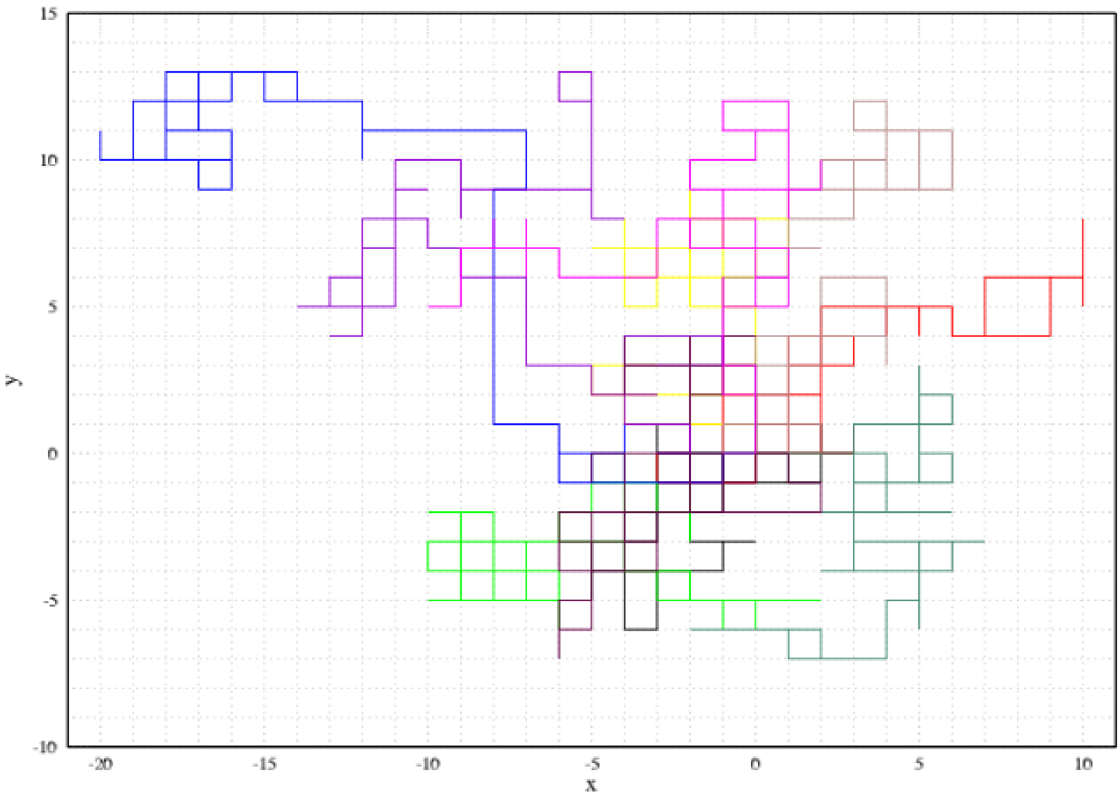
A título de visualizar como as trajetórias dos caminhantes se parecem no espaço bidimensional, adaptou-se o código da Tarefa C para gerar e armazenar em um arquivo as coordenadas de cada passo para cada andarilho em uma trajetória de 10 passos. Foram considerados 10 andarilhos, e estas informações foram reunidas em um diagrama que mostra, em diferentes cores, as trajetórias feitas por cada um:

Figura 17: Trajetórias de 10 andarilhos em um total de 10 passos.



Também foi feito o diagrama para $N = 100$ passos:

Figura 18: Trajetórias de 10 andarilhos em um total de 100 passos.



E, finalmente, para $N = 1000$ passos:

Figura 19: Trajetórias de 10 andarilhos em um total de 1000 passos.