Instituto de Física de São Carlos
Universidade de São Paulo

[7600017] - Introdução à Física Computacional

Projeto 2

Docente:
Francisco Castilho Alcaraz

Aluno:
Breno Henrique Pelegrin da Silva (13687303)

28 de setembro de 2023

Sumário

Lista de Figuras

1 Contexto

Nesse projeto, foram desenvolvidas tarefas relacionadas à distribuições aleatórias e movimentos aleatórios, que são tópicos extremamente importantes para a Física Estatística, que estuda fenômenos de natureza estocástica. A execução desse projeto contribui para um melhor entendimento desses fenômenos, visto que é possível simulá-los de forma extremamente simples utilizando a linguagem *Fortran 77*.

Todos os programas elaborados nesse projeto foram compilados e testados no ambiente *Linux* do servidor *basalto.ifsc.usp.br*, utilizando o compilador *gfortran*. Um *bash script*, denominado *run.sh*, foi criado para automatizar a compilação, o processo de limpeza dos arquivos de entrada e saída e a medição do tempo de execução. Além disso, todos os gráficos foram gerados utilizando a biblioteca *matplotlib* da linguagem *Python*.

2 Tarefa A

Nessa tarefa, gerou-se diversos momentos de distribuição aleatória $\left\langle x^k \right\rangle$ para k de 1 à 4, utilizando o gerador de números aleatórios, *rand()* do *Fortran 77*. Para isso, foram foram gerados N números r aleatórios, e a média deles foi calculada utilizando a equação 1. A média calculada é uma aproximação para $\left\langle x^k \right\rangle$, de modo que, no limite em que $N \to \infty$, o valor calculado será igual ao valor analítico.

$$\left\langle x^k \right\rangle = \lim_{N \to \infty} \sum_{i=1}^{N} r_i^k \tag{1}$$

Analiticamente, $\left\langle x^k \right\rangle$ é a área sobre a curva x^k no intervalo de $x = 0$ a $x = 1$, ou seja, $\left\langle x^k \right\rangle$ pode ser encontrado utilizando a integral da equação 2.

$$\left\langle x^k \right\rangle = \int_0^1 x^k dx = \left. \frac{x^{k+1}}{k+1} \right|_{x=0}^{x=1} = \frac{1}{k+1} \tag{2}$$

Assim, os valores analíticos de $\left\langle x^k \right\rangle$ para $k = 1, 2, 3$ e 4 calculados utilizando a equação 2 são, respectivamente, $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}$ e $\frac{1}{5}$.

O código desenvolvido para essa tarefa está exibido na figura 1 e a explicação para ele é dada no verbete 1.

Figura 1: Código-fonte da tarefa A

```
      program Distribuicao
         write(*,100)
100      format("Digite o número N: ", $)
         read(*,*) n

         do j=1, 4, 1
            soma = 0
            do i=1, n, 1
               soma = soma + rand()**j
            end do
            write(*, 101) j, soma/n
101         format("<x^", (I0), "> = ", (F5.4))
         end do
      end program
```

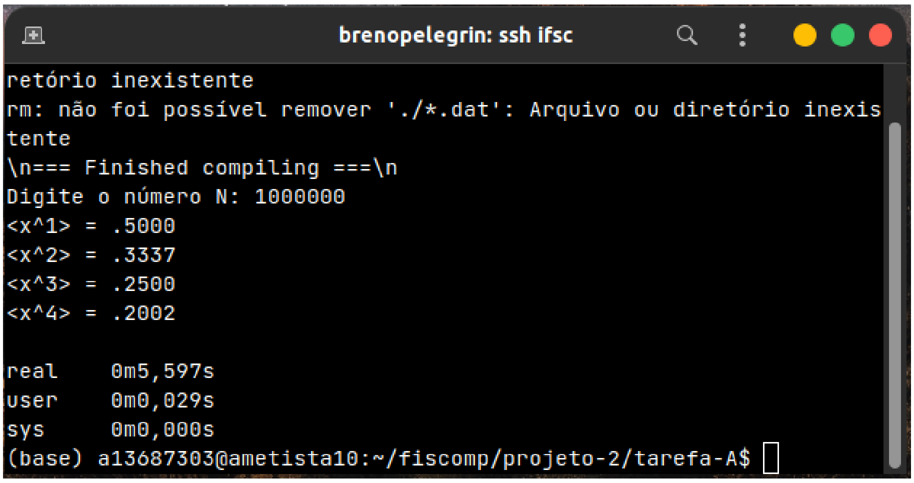
Verbete 1: Explicação do código da tarefa A

Inicialmente, pergunta-se ao usuário a quantidade N de números aleatórios a serem gerados. Em seguida, é realizado um laço de repetição de $j = 1$ a $j = 4$, onde j será o expoente do número aleatório.
Dentro do laço j, define-se uma variável $soma = 0$, e inicia-se um novo laço de repetição de $n = 1$ a $n = N$. Dentro desse laço interior, serão gerados números aleatórios r, que serão elevados à j e somados à variável soma. Ao final do laço n, escreve-se na tela a média calculada pela equação 1.

2.1 Exemplo de funcionamento e resultados

Na figura 2, é exibido um exemplo de funcionamento do programa criado. Note que, tanto nessa tarefa, quanto nas tarefas seguintes, um arquivo *run.sh* é chamado. Esse arquivo é um *bash script* que limpa os arquivos de saída, compila o programa, executa o binário e mede o tempo de execução. Ao final da execução do programa, são exibidos os tempos medidos pelo comando *time* do *Linux*. O programa da tarefa A executou em aproximadamente 5,6 segundos.

Figura 2: Teste de entrada e saída da tarefa A para $N = 10^6$



Comparando os resultados obtidos pelo programa com os resultados analíticos obtidos a partir da equação 2 para os mesmos valores de k, observa-se que os resultados do programa são muito próximos dos analíticos. Quanto maior é o valor de N, mais os valores se aproximarão do caso analítico.

3 Tarefa B

Na tarefa B, o caminhar de M andarilhos no eixo x foi simulado. Cada andarilho andarilho inicia em $x = 0$ e realiza N passos discretos (de 1 em 1) para direita ou esquerda, baseado em uma probabilidade $p = \frac{1}{2}$ de ir para a direita e $q = 1 - p$ de ir para a esquerda. Assim, a cada passo n da simulação, o andarilho tem 50% de chance de ir para a direita e 50% de ir para a esquerda. Quando ele vai para a direita, temos $x_n = x_{n-1} + 1$, e quando vai para esquerda, $x_n = x_{n-1} - 1$, onde 1 é o tamanho do passo e $x_n \in \mathbb{Z}$.

A mesma simulação foi realizada para $p = \frac{1}{2}, p = \frac{1}{3}, p = \frac{1}{4}$ e $p = \frac{1}{5}$. Além disso, para cada p, calculou-se $\langle x \rangle$ e $\left\langle x^2 \right\rangle$ e criou-se um histograma do número de andarilhos por intervalos de espaço. Como os andarilhos andam em passos discretos de tamanho 1, o histograma mede quantos andarilhos acabaram em cada posição $x_f \in [-N, N], x_f \in \mathbb{Z}$, que é o intervalo máximo de posições que um andarilho pode chegar. Assim, o histograma é dividido em *bins* (intervalos) $n \in [-N, N]$, com $n \in \mathbb{Z}$.

Analiticamente, o valor de $\langle x \rangle$ e de $\left\langle x^2 \right\rangle$ são dados pelas equações 3 e 4, respectivamente.

$$\langle x \rangle = N(p - q) = N(2p - 1) \tag{3}$$

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

Calculando analiticamente os valores de $\langle x \rangle$ para $p = \frac{1}{2}, p = \frac{1}{3}, p = \frac{1}{4}$ e $p = \frac{1}{5}$, obtém-se, respectivamente, $0, -333, -500$ e -600. Após, realizando também o cálculo analítico de $\left\langle x^2 \right\rangle$ para os mesmos valores de p, é possível obter, respectivamente, $1000, 112000, 250750$ e 360640.

O código desenvolvido para essa tarefa está exibido na figura 3 e a explicação para ele é dada logo em seguida no verbete 2. Na figura 4, é exibido o histograma gerado para os valores de p especificados anteriormente.

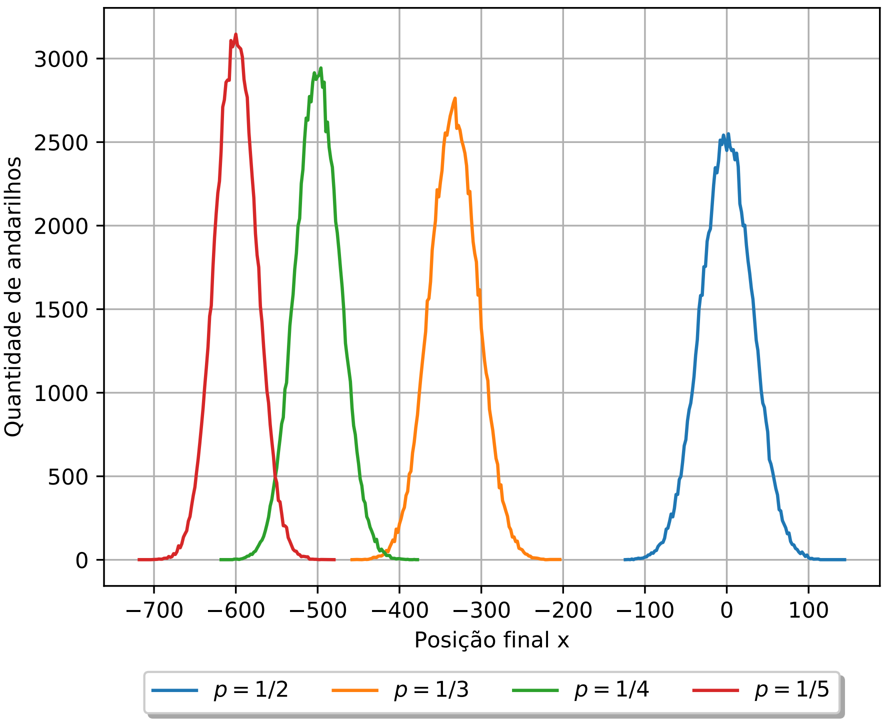
Figura 3: Código-fonte da tarefa B

```
      program Andarilho
         parameter (iseed = 123)
         parameter (mand = 100000)
         parameter (npassos = 1000)
         dimension ihist(-npassos:npassos)
         character fname*50
         call srand(iseed)

         do ifrac=2, 5, 1
            ihist = 0
            p = 1.0e0/ifrac
            xsoma = 0
            x2soma = 0
            do j=1, mand, 1
               ix = 0
               do i=1, npassos, 1
                  aleatorio = rand()
                  if(aleatorio .lt. (1-p)) then
                     ! x esquerda
                     ix = ix - 1
                  else
                     ! x direita
                     ix = ix + 1
                  end if
               end do
               xsoma = xsoma + ix
               x2soma = x2soma + ix**2
               ihist(ix) = ihist(ix) + 1
            end do

            write(*,*) "p, <x>  = ", p, ",", xsoma/mand
            write(*,*) "p, <x^2> = ", p, ",",  x2soma/mand

            !gera arquivo de histograma
            write(fname, 100) ifrac
100         format("saida-b-13687303-hist-p1over" (I0) ".dat")
            open(unit=50, file=fname)
            do i=-npassos, npassos, 1
               write(50,*) i, ihist(i)
            end do
            close(unit=50)
         end do

      end program
```

Verbete 2: Explicação do código da tarefa B

Inicialmente, define-se os parâmetros da simulação: número de andarilhos *mand* (M), número de passos *npassos* (N), a semente para o gerador aleatório e o tamanho do vetor de histograma. Além disso, a fim de imprimir arquivos com nomes diferentes para cada p, define-se uma cadeia de caracteres *name*, que armazenará o nome do arquivo a ser salvo.
Em seguida, é realizado um laço *ifrac* de 2 a 5, onde *ifrac* é o denominador da fração de p. Assim, p é calculado como $1/ifrac$. Dentro desse laço, o vetor histograma e as variáveis de soma de x e x^2 são zeradas. Após, realiza-se um laço interior de $j = 1$ a $j = M$, e simula-se os N passos de cada andarilho j. Ao fim da simulação dos passos de cada andarilho, soma-se a posição final x_f e a posição final quadrada x_f^2 às variáveis *xsoma* e *x2soma*, respectivamente. Além disso, incrementa-se o vetor histograma no índice x_f em 1, ou seja, quando o andarilho para na posição x_f, o bin $n = x_f$ do histograma tem seu valor de andarilhos incrementado por 1.
Ao fim do loop j, calcula-se as médias $\langle x \rangle$ e $\langle x^2 \rangle$ utilizando a equação 1, imprimindo-as na tela e, também, imprime-se o vetor histograma em um arquivo de saída específico para cada p.

Figura 4: Histograma da tarefa B para diferentes valores de p



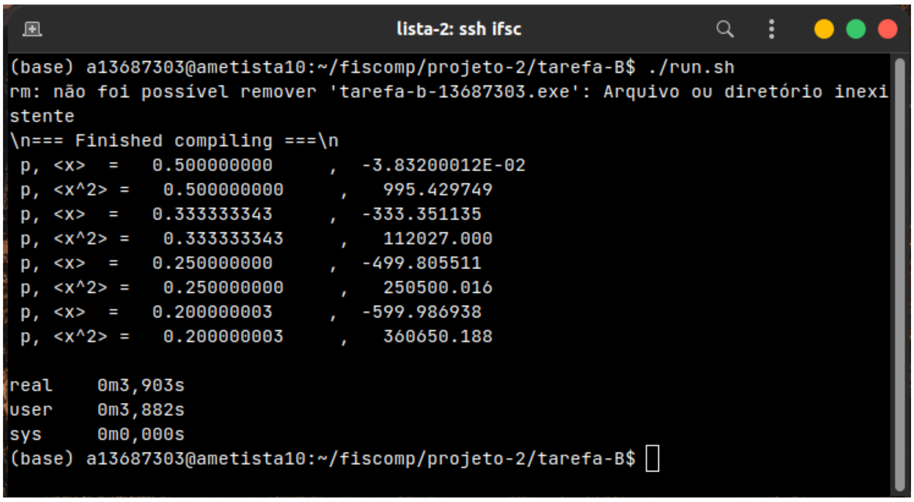
No histograma da figura 4, é possível notar que a curva gaussiana de centro em $x = 0$ é aquela gerada a partir de $p = \frac{1}{2}$, ou seja, quando o andarilho tem iguais probabilidades de ir para a direita e para a esquerda. Nos outros casos, temos $p \neq q$, o que gera uma movimentação do centro da gaussiana para a esquerda, já que, conforme p diminui, os andarilhos tendem a ir para a esquerda pois a probabilidade q é maior.

Além disso, o fato de o histograma ser uma distribuição gaussiana (normal) é justificado pelo teorema do limite central: a soma de N variáveis aleatórias independentes com qualquer distribuição e variâncias semelhantes, é uma variável com distribuição normal quando N tende ao infinito.

3.1 Exemplo de funcionamento e resultados

Na figura 5, é exibido um exemplo de funcionamento do programa criado. O programa finalizou após aproximadamente $3,9$ segundos.

Figura 5: Teste de entrada e saída do terminal na tarefa B



Fonte: Autoria própria

Comparando com os valores analíticos obtidos anteriormente, observa-se que o programa fornece resultados muito próximos do esperado e executa em um tempo razoavelmente rápido.

4 Tarefa C

Nessa tarefa, o algoritmo da tarefa B foi generalizado para simular M andarilhos caminhando aleatoriamente em um espaço bidimensional, ou seja, em duas direções: x e y. Cada passo do andarilho, para cima e para baixo no eixo y, e para direita e para esquerda no eixo x, possuem igual probabilidade $p = \frac{1}{4}$. Assim, o andarilho percorrerá N passos no total, restritos a dois eixos, ou seja, sua posição é determinada por um vetor $r_n = x_n\hat{x} + y_n\hat{y}$.

A cada passo, sorteia-se um número aleatório r. Caso $r \in [0, \frac{1}{4})$, é dado um passo na direção $-x$, caso $r \in [\frac{1}{4}, \frac{1}{2})$, é dado um passo na direção $+x$, caso $r \in [\frac{1}{2}, \frac{3}{4})$, é dado um passo na direção $-y$, e por fim, caso $r \in [\frac{3}{4}, 1)$, é dado um passo na direção

$+y$. O caso onde $r = 1$ não é considerado pois a função *rand* do *Fortran 77* retorna números no intervalo $[0, 1)$. Além disso, também calculou-se $\langle \vec{r} \rangle$, $\langle \vec{r}^2 \rangle$ e Δ^2 ao final da simulação, utilizando as equações 5, 6 e 7, respectivamente.

$$\langle \vec{r} \rangle = \langle x \rangle \hat{x} + \langle y \rangle \hat{y} \tag{5}$$

$$\langle \vec{r}^2 \rangle = \langle x^2 \rangle + \langle y^2 \rangle \tag{6}$$

$$\langle \vec{r} \rangle^2 = \langle x \rangle^2 + \langle y \rangle^2$$

$$\Delta^2 = \langle \vec{r}^2 \rangle - \langle \vec{r} \rangle^2 \tag{7}$$

O código elaborado para essa tarefa está exposto na figura 6, e sua explicação é dada no verbete 3. Na figura 7, é exposto um gráfico das posições finais $\vec{r}_f$ de cada andarilho, para diferentes valores de passos $N = 10^1, N = 10^2, N = 10^3, N = 10^4, N = 10^5$ e $N = 10^6$. Na figura 8, são exibidas as mesmas posições para diferentes valores de passos, mas em escala log-log. Nos gráficos, os pontos (r_x, r_y) possuem transparência, possibilitando ver a concentração de pontos no mesmo local através da intensidade da cor.

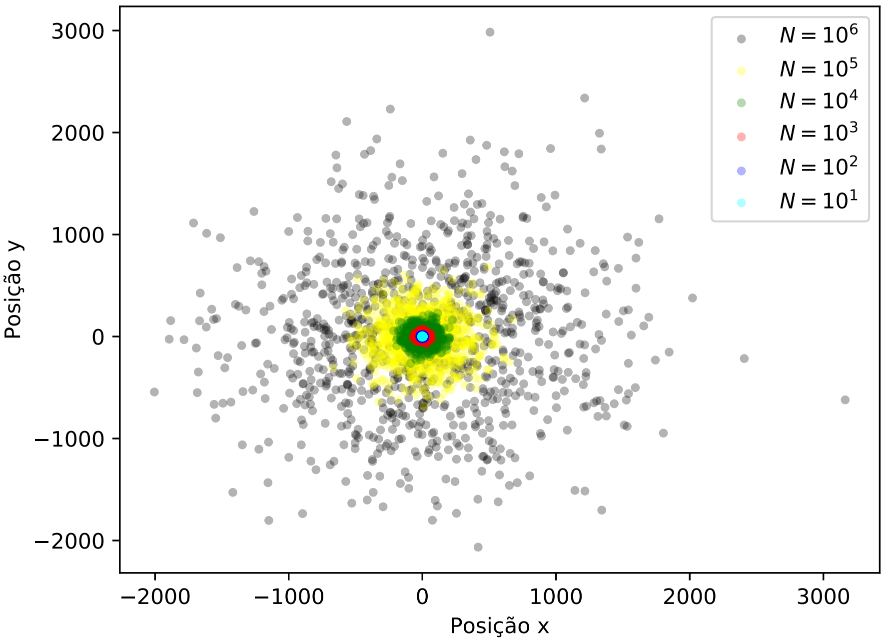
Figura 6: Código-fonte da tarefa C

```
      program Andarilho2D
         parameter (iseed = 123)
         parameter (mand = 1000)
         character fname*50
         call srand(iseed)
         do l=1, 6, 1
            npassos = 10**l
            xsoma = 0
            ysoma = 0
            x2soma = 0
            y2soma = 0
            write(fname, 101) l
 101        format("saida-c-13687303-n1e", (I0), ".dat")
            open(50, file=fname)
            do j=1, mand, 1
             ix = 0
             iy = 0
              do i=1, npassos, 1
                prob = rand()
                ! -x
                if(prob .lt. 0.25e0) ix = ix - 1
                ! +x
                if(prob .ge. 0.25e0 .and. prob .lt. 0.5e0) ix = ix + 1
                ! -y
                if(prob .gt. 0.5e0 .and. prob .lt. 0.75e0) iy = iy - 1
                ! +y
                if(prob .ge. 0.75e0 .and. prob .lt. 1.0e0) iy = iy + 1
              end do
              xsoma = xsoma + ix
              ysoma = ysoma + iy
              x2soma = x2soma + ix**2
              y2soma = y2soma + iy**2
              write(50, *) ix, iy
            end do
            close(50)
            ravgx = real(xsoma)/mand
            ravgy = real(ysoma)/mand
            r2avg = (real(x2soma)/mand + real(y2soma)/mand)
            d2 =  r2avg - (ravgx**2 + ravgy**2)
            write(*,*) "==="
            write(*,100) l
 100        format("npassos = 1E+" (I0))
            write(*,*) "<(rx, ry)> =", ravgx, ", ", ravgy
            write(*,*) "<(rx,ry)^2> =", r2avg
            write(*,*) "Delta^2 =", d2
         end do
         write(*,*) "==="
      end program
```

Verbete 3: Explicação do código da tarefa C

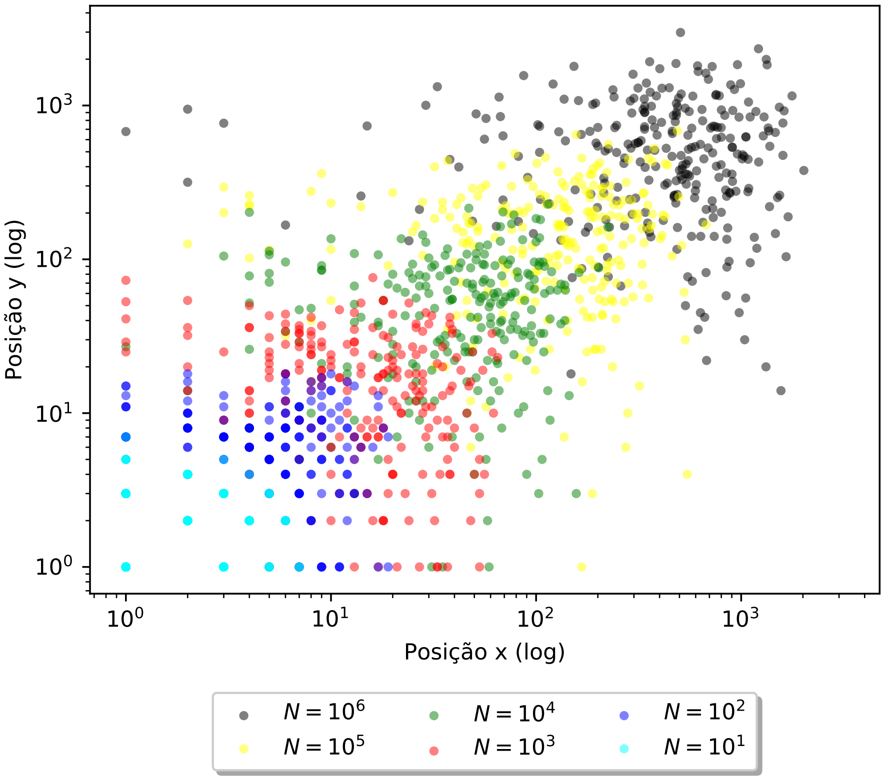
Este código é muito semelhante à tarefa B, já que apenas são adicionadas mais direções de movimento. Inicialmente, define-se os parâmetros da simulação e uma cadeia de caracteres *fname* é criada para armazenar o nome do arquivo de saída, que será diferente para cada N.
Em seguida, um laço de $l = 1$ a $l = 6$ é iniciado, de forma que $N = 10^l$. Dentro desse laço, é executado outro laço de $j = 1$ a $j = M$, ou seja, uma iteração para cada andarilhos. Dentro do laço j, é executado o laço de $i = 1$ a $i = N$, para simular os passos de cada andarilho. Ao final do laço i, as variáveis de soma são incrementadas com os valores calculados e as posições de r_x e r_y finais de cada andarilho são escritas em um arquivo de saída.
Ao final do laço j, calcula-se $\langle \vec{r} \rangle$, $\langle \vec{r}^2 \rangle$ e Δ^2 utilizando as equações 5, 6 e 7 respectivamente, e os valores são impressos na tela.

Figura 7: Posição final dos andarilhos para diferentes N



Fonte: Autoria própria

Figura 8: Posição final dos andarilhos para diferentes N (log-log)



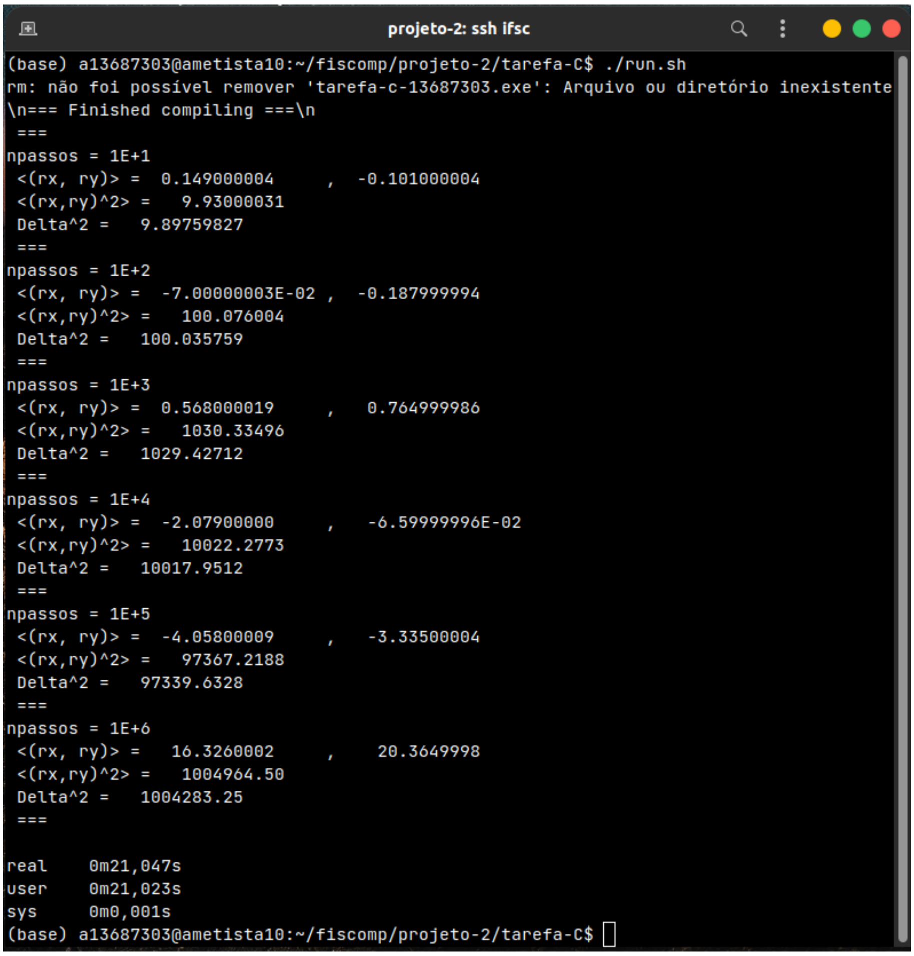
Fonte: Autoria própria

Analisando as figuras 7 e 8, observa-se que para uma quantidade pequena de passos, de $N = 10^1$ a $N = 10^3$, os andarilhos estão muito concentrados em torno de $(0, 0)$, mas, conforme são dados mais passos, eles tendem a se difundir radialmente pelo espaço bidimensional. Assim, esse tipo de movimento pode representar, por exemplo, a difusão das moléculas de um pingo de leite em um delicioso copo de café quente.

4.1 Exemplo de funcionamento e resultados

Na figura 9, é exibido um exemplo de uso do programa, com a entrada e saída do terminal, exibindo os valores calculados de $\langle \vec{r} \rangle$, $\langle \vec{r}^2 \rangle$ e Δ^2 para cada configuração de passos N. O programa executou em aproximadamente 21 segundos, que é um tempo razoável, tendo em vista a grande quantidade de passos simulados.

Figura 9: Teste de entrada e saída do terminal na tarefa C



```
projeto-2: ssh ifsc
(base) a13687303@ametista10:~/fiscomp/projeto-2/tarefa-C$ ./run.sh
rm: não foi possível remover 'tarefa-c-13687303.exe': Arquivo ou diretório inexistente
\n=== Finished compiling ===\n
 ===
npassos = 1E+1
 <(rx, ry)> =  0.149000004     ,  -0.101000004
 <(rx,ry)^2> =   9.93000031
 Delta^2 =   9.89759827
 ===
npassos = 1E+2
 <(rx, ry)> =  -7.00000003E-02 ,  -0.187999994
 <(rx,ry)^2> =   100.076004
 Delta^2 =   100.035759
 ===
npassos = 1E+3
 <(rx, ry)> =  0.568000019     ,   0.764999986
 <(rx,ry)^2> =   1030.33496
 Delta^2 =   1029.42712
 ===
npassos = 1E+4
 <(rx, ry)> =  -2.07900000     ,   -6.59999996E-02
 <(rx,ry)^2> =   10022.2773
 Delta^2 =   10017.9512
 ===
npassos = 1E+5
 <(rx, ry)> =  -4.05800009     ,   -3.33500004
 <(rx,ry)^2> =   97367.2188
 Delta^2 =   97339.6328
 ===
npassos = 1E+6
 <(rx, ry)> =   16.3260002     ,    20.3649998
 <(rx,ry)^2> =   1004964.50
 Delta^2 =   1004283.25
 ===

real    0m21,047s
user    0m21,023s
sys     0m0,001s
(base) a13687303@ametista10:~/fiscomp/projeto-2/tarefa-C$
```

Fonte: Autoria própria

Analisando os resultados obtidos para $\langle \vec{r} \rangle$, pode-se concluir que eles estão dentro do esperado, já que, para N pequeno, as posições tendem a se concentrar em $(0, 0)$, então a média deve ser próxima de 0, e para N grande, as posições se dispersam, fazendo com que a média seja diferente de zero.

5 Tarefa D

Nessa tarefa, assim como na tarefa C, o movimento de andarilhos em um espaço bidimensional foi simulado. Todavia, a cada passo dado, calcula-se a entropia do sistema. Assim, ao final da execução do programa, é possível obter uma curva da evolução da entropia do sistema em função do número de passos (ou tempo).

Para isso, define-se um reticulado no espaço bidimensional, ou seja, divide-se o espaço em quadrados de lado l. Assim, a entropia do sistema é dada pela equação 8, onde P_i é a probabilidade de um andarilho estar no quadrado i.

$$S = -\sum_i P_i \cdot ln(P_i) \tag{8}$$

O valor de P_i em cada passo pode ser estimado da seguinte forma: para cada quadrado i, conta-se quantos andarilhos tem $\vec{r}$ dentro dos limites do quadrado i, e divide-se o número de andarilhos dentro do quadrado pelo número M de andarilhos totais do sistema. Quanto maior o lado l do quadrado, maior é a chance de um andarilho estar dentro dele.

O código elaborado para essa tarefa está exposto nas figura 10 (parte 1) e 11, e sua explicação é dada no verbete 4. Na figura 12, é exposto um gráfico da evolução da entropia do sistema em função dos passos N, ou seja, em função do tempo.

Figura 10: Código-fonte da tarefa D (parte 1)

```fortran
      program AndarilhoEntropia
         parameter (iseed = 123)
         parameter (nand = 1000)
         parameter (lado = 10)
         parameter (mpassos = 2000)
         dimension irxand(nand)
         dimension iryand(nand)

         call srand(iseed)
         irxand = 0
         iryand = 0

         open(50, file='saida-d-13687303.dat')
         do i=1, mpassos, 1
            do j=1, nand, 1
              prob = rand()
            if(prob .lt. 0.25e0) then
                  ! esquerda x
                  irxand(j) = irxand(j) - 1
               end if
               if(prob .ge. 0.25e0 .and. prob .lt. 0.5e0) then
                  ! direita x
                  irxand(j) = irxand(j) + 1
               end if
               if(prob .gt. 0.5e0 .and. prob .lt. 0.75e0) then
                  ! esquerda y
                  iryand(j) = iryand(j) - 1
               end if
               if(prob .ge. 0.75e0 .and. prob .lt. 1.0e0) then
                  ! direita y
                  iryand(j) = iryand(j) + 1
               end if
            end do
```

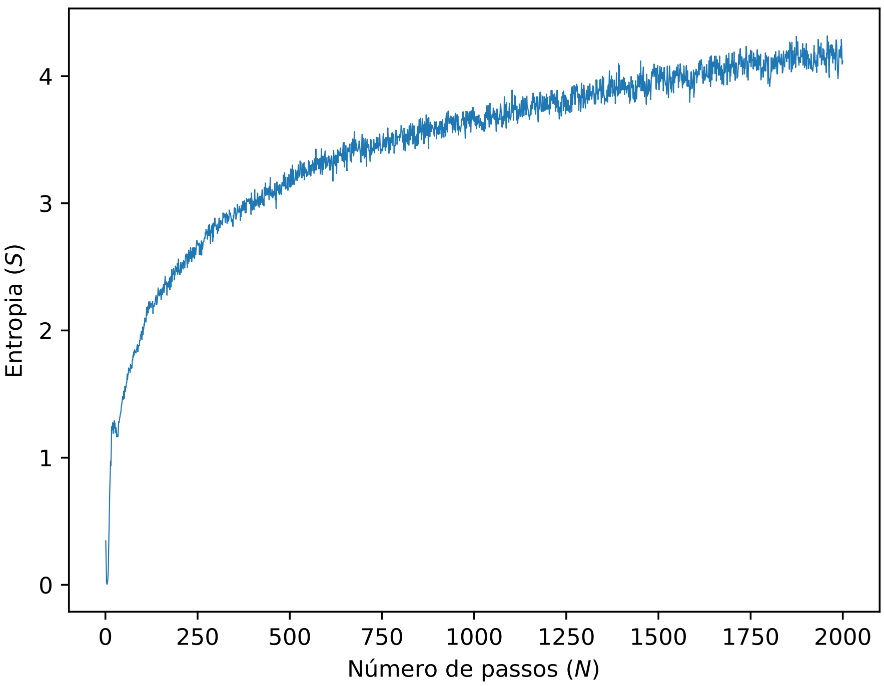
Figura 11: Código-fonte da tarefa D (parte 2)

```
            ixmin = 0
            ixmax = 0
            iymin = 0
            iymax = 0

            do j=1, nand, 1
c              O intuito disso é otimizar o programa, encontrando
c              o limite máximo onde devemos percorrer o reticulado
               if(irxand(j) .lt. ixmin) ixmin = irxand(j)
               if(irxand(j) .gt. ixmax) ixmax = irxand(j)
               if(iryand(j) .lt. iymin) iymin = iryand(j)
               if(iryand(j) .gt. iymax) iymax = iryand(j)
            end do

            s = 0.0e0
            do ix=ixmin, ixmax, lado
               do iy=iymin, iymax, lado
c                 Percorre cada quadrado i do reticulado e conta
c                 o número de andarilhos, calculando S_i
                  icount = 0
                  do j=1, nand, 1
                     if( irxand(j) .gt. ix .and. irxand(j) .lt. ix+lado
     &.and. iryand(j) .gt. iy .and. iryand(j) .lt. iy+lado) then
                        icount = icount + 1
                     end if
                  end do
                  if(icount .gt. 0) then
                     probi = real(icount)/nand
                     s = s - (probi)*log(probi)
                  end if
               end do
            end do
            write(50,*) i, s
         end do
         close(50)
      end program
```

Verbete 4: Explicação do código da tarefa D

Este código é muito semelhante à tarefa C, com a diferença de que, em cada passo, é calculada a entropia do sistema. Inicialmente, define-se os parâmetros da simulação. Depois, inicia-se um laço i de 1 a N, e dentro dele, um laço j de 1 a M. Assim, ao final do laço j, todos os andarilhos terão dado um passo. Depois de terminado o laço j, encontra-se as mínimas e máximas posições x e y dos andarilhos, para otimizar a contagem de andarilhos por reticulado. Em seguida, é realizado um laço para x de x_{min} a x_{max} e, dentro dele, um laço para y de y_{min} a y_{max}, contando quantos andarilhos estão dentro de cada quadrado.
Após realizada a contagem em todo o reticulado, calcula-se a entropia do sistema naquele passo utilizando a equação 8 e o resultado é salvo em um arquivo de saída. Em seguida, o processo se repete até $i = N$.

Figura 12: Entropia do sistema em função de N



Fonte: Autoria própria

Ao analisar a figura 12, observa-se que a entropia do sistema segue uma curva logarítmica, ou seja, ela cresce rapidamente até $N = 200$ e depois, cresce lentamente até começar a entrar em equilíbrio para $N > 1500$.

5.1 Exemplo de funcionamento e resultados

Na figura 13, é exibido um teste de entrada e saída no terminal para a tarefa D. Observa-se que o programa executou em aproximadamente 1,4 segundos, que é um tempo razoavelmente pequeno dado o número de iterações necessárias para percorrer o reticulado inteiro.

Figura 13: Teste de entrada e saída do terminal na tarefa D

Fonte: Autoria própria