Projeto 2

Juliana Naomi Yamauti Costa, nUSP 10260434

01 de Abril de 2020

Este projeto tem o o intuito de estudar sistemas randômicos, desvendando-os através de algorítmos probabilísticos, os quais utilizam a probabilidade como parte de sua lógica. Em particular, estudaremos o problema do 'Caminho aleatório', que tem fundametal importância em Física Estatística por modelar vários fenômenos naturais, mesmo que na realidade alguns deles não sejam puramente aleatórios.

1 Tarefa A

A distribuição normal é um importante tipo de distribuição de probabilidade em Estatística. Isso se deve ao encaixe do modelo em muitos fenômenos naturais e sua ligação a vários conceitos matemáticos, como o Movimento Browniano. Também conhecida como distribuição Gaussiana, corresponde ao efeito agregado de experiências aleatórias independentes quando o número de ensaios é muito alto.

O programa a seguir gera números pseudo-aleatórios através da função *rand()* e, com ele, verificou-se propriedades da distribuição e o comportamento da média conforme a mudança de parâmetros.

tarefa-A-10260434.f90

```
Program A

!Média de números aleatórios

Print*, 'Informe a quantidade de números aleatórios a serem gerados'
read*, i

do n=1, 4                                !expoente do número aleatório
        soma = 0.0
        do j=1, i
        r = (rand())**n                  !número aleatório
        soma = soma + r
        end do
        !Calculando a média
        cmedia = soma/i
        print*, 'Para n =',n, 'e',i,'pontos aleatórios a média é',cmedia
end do

End Program A
```

1.1 Resultados

A função *rand()* fornece número aleatórios no intervalo de 0 a 1, dessa forma, podemos encontrar os valores teóricos de $\langle x^n \rangle$ e comparar com os valores obtidos estatísticamente. A média pode ser calculada pela integral da função:

$$\int_0^1 x^n dx = \frac{1}{(n+1)}$$

n	$\langle x^n \rangle$	Núm. aleatórios	Média obtida
1	0.5	100	0.518424511
		1000	0.497961760
		10000	0.501827717
2	0.33	100	0.314383537
		1000	0.338106215
		10000	0.330953181
3	0.25	100	0.273117125
		1000	0.235080987
		10000	0.247461602
4	0.2	100	0.164195687
		1000	0.209900960
		10000	0.198787957

Tabela 1: Comparação entre os resultados teórico e encontrados na simulação

Observa-se que aumentando a quantidade de números aleatórios, mais precisa se torna a média. Isso decorre do Teorema Central do Limite, um resultado fundamental que afirma que quando o tamanho da amostra aumenta, a distribuição amostral da sua média aproxima-se cada vez mais de uma distribuição normal.

2 Tarefa B

O passeio aleatório é um objeto matemático que descreve um caminho que consiste de uma sucessão de passos aleatórios e, neste problema, estamos explorando o caso unidimensional. Estabelecemos que o andarilho possui probabilidade 'p' de ir para a direita e 'q' de ir para a esquerda (p+q=1), com passo de tamanho 'l', sendo cada passo um evento aleatório independente. A função *rand()* foi utilizada devido a sua distribuição uniforme de pseudo-números, assim, dentro do intervalo entre 0 e 1 todos os números tem igual probabilidade de serem escolhidos. Dessa forma, basta dividir o intervalo de acordo com a probabilidade e utilizar como fator de decisão o número aleatório estar dentro dele ou não.

Como parâmetros fixos, foram aplicados 10000 andarilhos, cada um dando 1000 passos, sendo testadas diferentes probabilidades de passos para a direita (informações destacadas em azul).

tarefa-B1-10260434.f90

```
Program R_walk

parameter (nmax= 10000)
dimension c(nmax), a(nmax)

!Número de andarilhos
iandarilhos = 10000

!Número de passos
Np = 1000

!Distribuição uniforme de probabilidade
p=1.0/2.0                !probabilidade passo para direita

!arquivo para colocar x
open(10, file = 'saida-B1-10260434.dat')

soma1 = 0.0
soma2 = 0.0

!simulação do andarilho
do i=1, iandarilhos
        x = 0          !Posição inicial do andarilho i
        do j=1, Np
                r = rand()                !numero aleatório
                if(r .lt. p) then
                x = x + 1
                else
                x = x - 1
                end if
        end do
        soma1 = soma1 + x                !soma das posições para calcular
 ↪  a média
        soma2 = soma2 + x**2
        write(10,*) i,x                !posição final de cada andarilho
end do
```

```
close(10)

!abrindo novamente para contagem de andarilhos em cada posição
open(10, file = 'saida-B1-10260434.dat')

!arquivo para gerar gráfico n(x) vs x
open(20, file = 'histogramaB1-10260434.dat')

do n=1, nmax
        read(10, *,end=1) c(n), a(n)        !lendo o arquivo
end do
1 continue

do b= -Np, Np        !contagem de andarilhos em cada posição
        cont=0
        do n=1, iandarilhos-1
                if (a(n) .eq. b) then
                cont=cont+1
                end if
        end do
        if (cont .ne. 0) then
        write(20,*) b,cont
        end if
end do

!forma estatística
xmedia = soma1/iandarilhos                    !media de <x>
x2media = soma2/iandarilhos                     !media de <x**2>

!forma analítica
q = 1 - p
xm = (p-q)*Np
x2m= (Np*(p-q))**2 + 4*p*q*Np

print*, 'Resultado estatístico <x>=',xmedia,'e analítico <x>=', xm
print*, 'Resultado estatístico <x**2>=',x2media,'e analítico <x**2>=', x2m

close(10)
close(20)
End Program R_walk
```

2.1 Tarefa B1

Neste problema os andarilhos tem igual probabilidade de irem para esquerda e direita (p=q=1/2). Foi utilizado o código *'tarefa-B1-10260434.f90'* para gerar as posições finais de cada andarilho (armazenados no arquivo *'saída-B1-10260434.dat'*), calcular a média e média quadrática de posições, e contar a quantidade de andarilhos em cada posição (armazenado no arquivo *'histogramaB1-10260434.dat'*).

2.1.1 Resultados

A probabilidade de encontrar um bêbado em determinada posição, quando p=q=1/2 é dada pela equação:

$$P_N(x) = \frac{N!}{[(N+x)/2]![(N-x/2)]!}\left(\frac{1}{2}\right)^N \tag{1}$$

Nesse caso especial, ela assume uma forma simétrica e a distribuição binomial apresenta o seguinte gráfico:

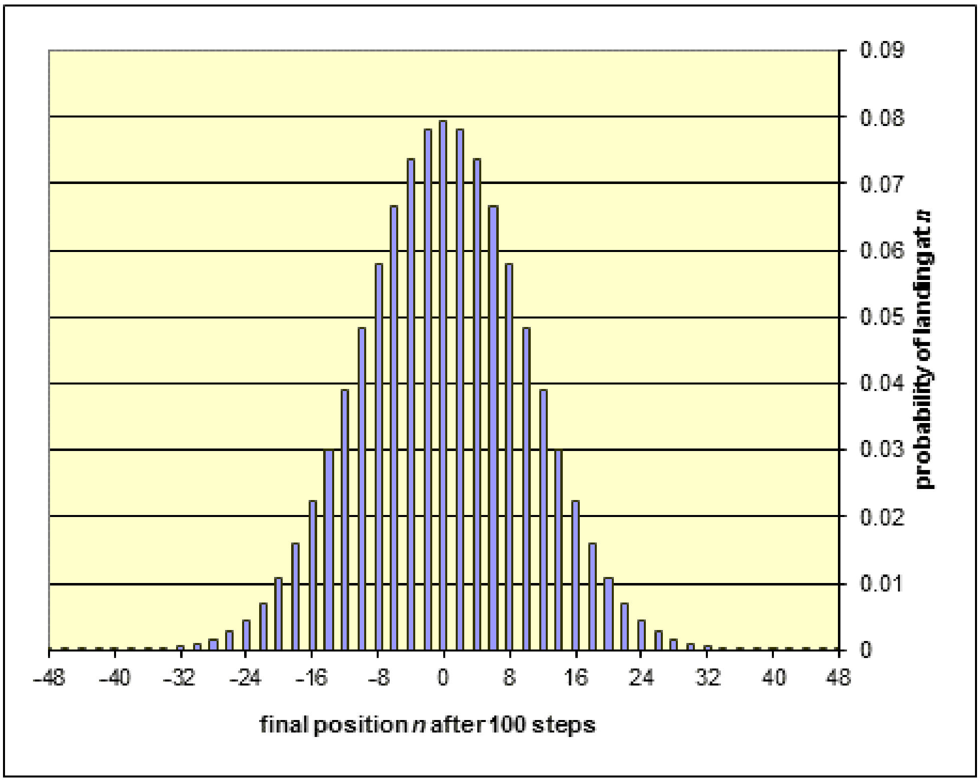


Figura 1: Probabilidade de encontrar um andarilho em cada posição. Fonte: (THE. . . , s.d.)

Observa-se que a probabilidade de encontrar um andarilho N passos distante da origem é muito pequena, enquanto de encontrá-lo nas proximidades da origem é muito maior. Esse resultado assemelha-se com o encontrado, o qual é centrado na origem, simétrico e possui a forma aproximada de uma distribuição normal:

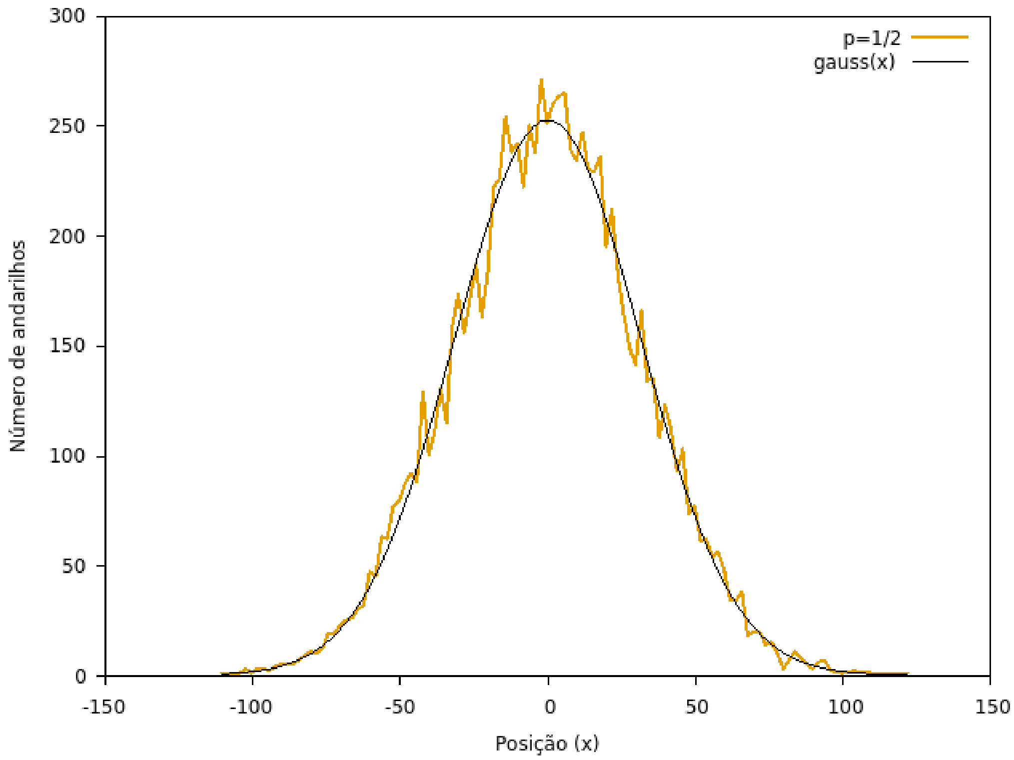


Figura 2: Gráfico obtido a partir da simulação para p=1/2. A curva em preto 'gauss(x)' representa o fitting dos dados a uma curva gaussiana

Isso é possível pois para N grande (quando o intervalo de tempo de um passo tende a zero) a probabilidade tende a uma função Gaussiana:

$$P(x, N\triangle t) = \sqrt{\frac{2}{\pi N}} e^{\frac{-x^2}{2N}} \tag{2}$$

Assim, observa-se que os dados adaptam-se à curva de uma distribuição gaussiana, como mostra o gráfico. Para as médias foram obtidos os seguintes resultados: $\langle \mathbf{x} \rangle = \mathbf{9.16000009.10^{-02}}$ e $\langle \mathbf{x^2} \rangle = \mathbf{999.865601}$ como média e média quadrática de posições, respectivamente. Como discutido anteriormente, esperava-se que a média das posições se aproximasse de 0. Já a média quadrática aumenta linearmente com o tempo, porém, como cada passo foi considerado como um intervalo de tempo, temos que ela se iguala ao número de passos (N=1000), como foi encontrado.

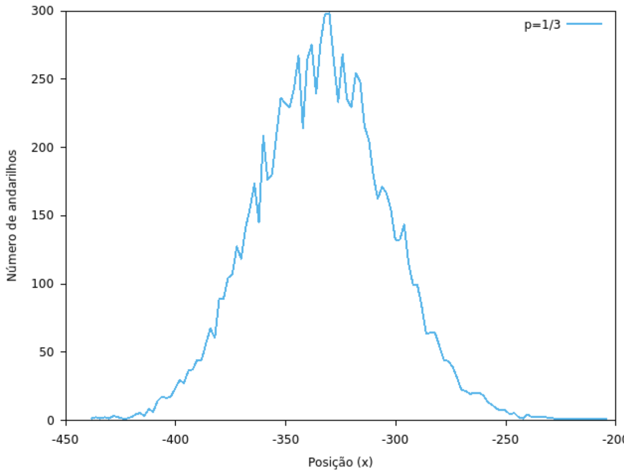
2.2 Tarefa B2

Os resultados analíticos foram comparados com os estatísticos obtidos a partir do programa *'tarefa-B1-10260434.f90'* e os gráficos foram obtidos da mesma forma que a tarefa anterior. Analíticamente, derivamos as seguintes expressões das médias:

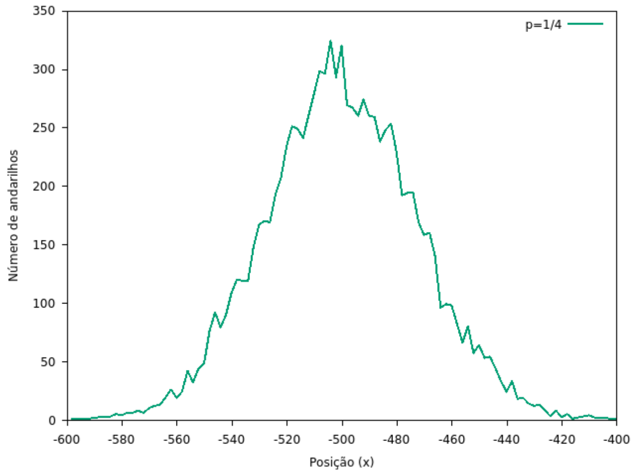
$$\langle x \rangle = N(p - q) \tag{3}$$

$$\langle x^2 \rangle = (N(p - q))^2 + 4pqN \tag{4}$$

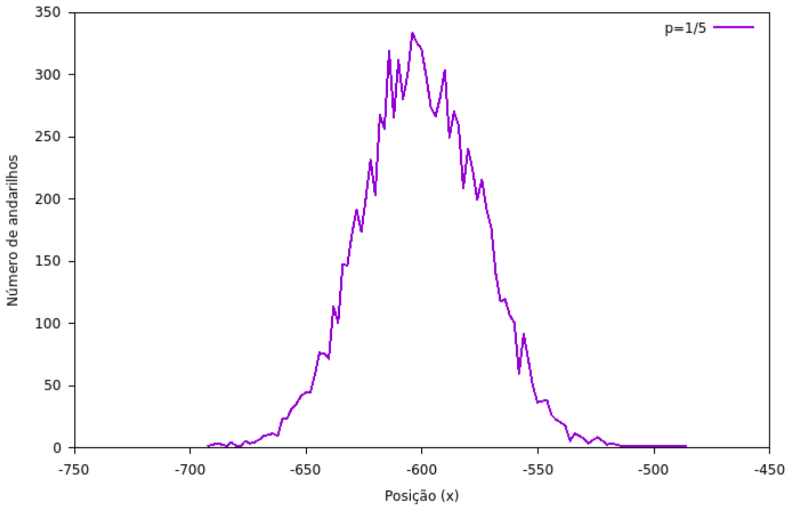
2.2.1 Resultados



(a) Deslocamento para a esquerda, com p=1/3.



(b) Deslocamento para a esquerda, com p=1/4.



(c) Deslocamento para a esquerda, com p=1/5.

Figura 3: Gráficos de distribuição de andarilhos para diferentes probabilidades.

Observa-se o deslocamento do gráfico para posições à esquerda, como esperado, uma vez que diminuindo a probabilidade dos andarilhos darem um passo para direita, mais andarilhos aparecerão em posições à esquerda.

p	$\langle x \rangle$		$\langle x \rangle^2$	
	Analítico	Estatístico	Analítico	Estatístico
1/3	-333.333282	-333.355194	112014.898	111999.969
1/4	-500.000000	-499.931000	250687.797	250750.000
1/5	-600.000000	-599.992798	360641.688	360640.000

Tabela 2: Comparação entre os resultados analíticos e estatísticos das médias

O caminho aleatório 1D pode ser reescrito como a equação de difusão contínua macroscópica, se 'l' e $\triangle t$ tendem a zero. Quando as probabilidade são desiguais, adiciona-se um termo extra que pode interpretado como velocidade. É possível analisar esse fenômeno ao observar as posições dos andarilhos e, com auxílio de algumas bibliotecas do Python foi obtido o seguinte gráfico:

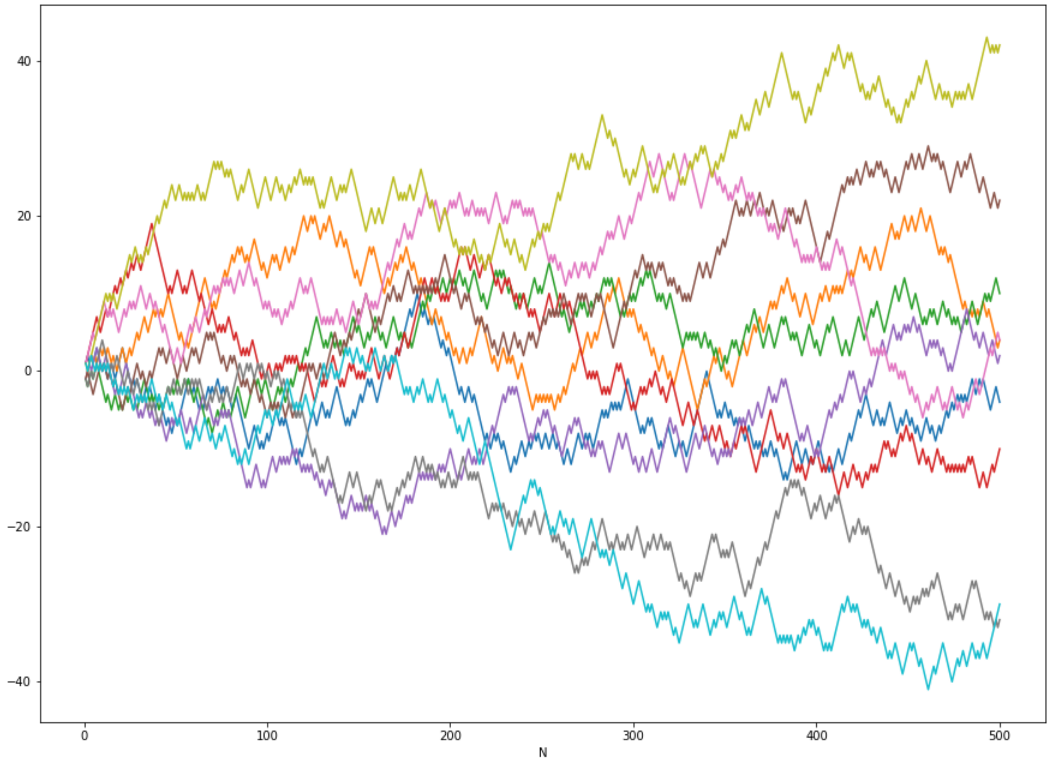


Figura 4: Gráfico de x (posição) *vs* N (passo) para 10 andarilhos e 500 passos.

Verifica-se que com o aumento do número de passos há uma expansão desse 'cone' para posições mais afastadas da origem.

3 Tarefa C

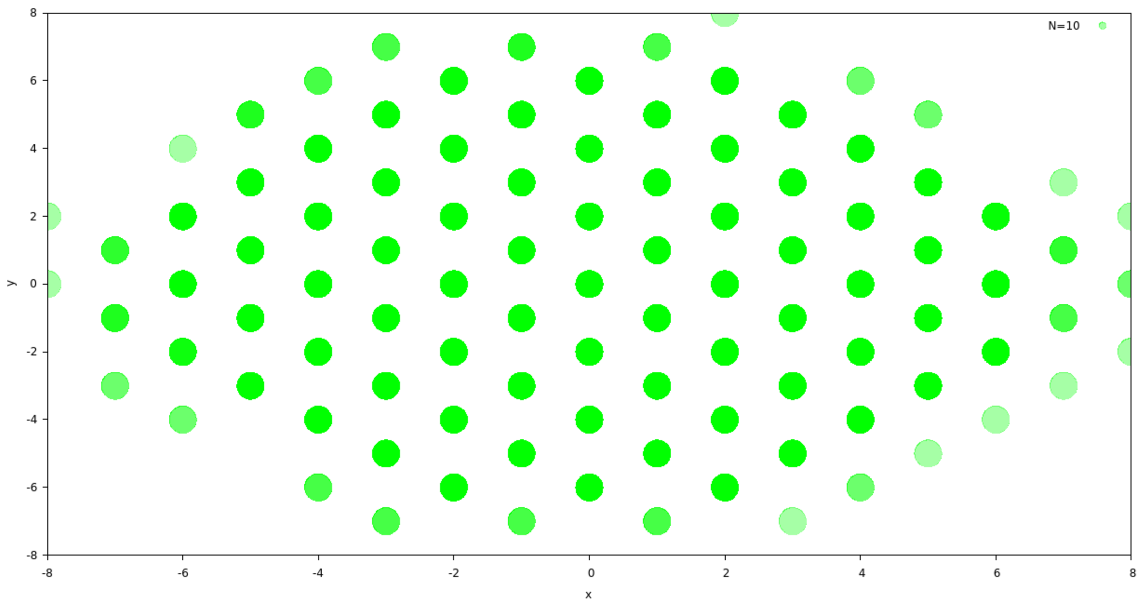
Analisando agora o caso bidimensional do caminho aleatório, a função *rand()* foi utilizada da mesma maneira, porém o intervalo de 0 a 1 foi dividido em quatro, cada um representando a mesma probabilidade do andarilho ir para uma direção. Todos os testes foram feitos com 10000 andarilhos, com início na origem, variando o número de passos. Além disso, como as médias envolvem vetores, as somas foram feitas individualmente para cada componente.

tarefa-C-10260434.f90

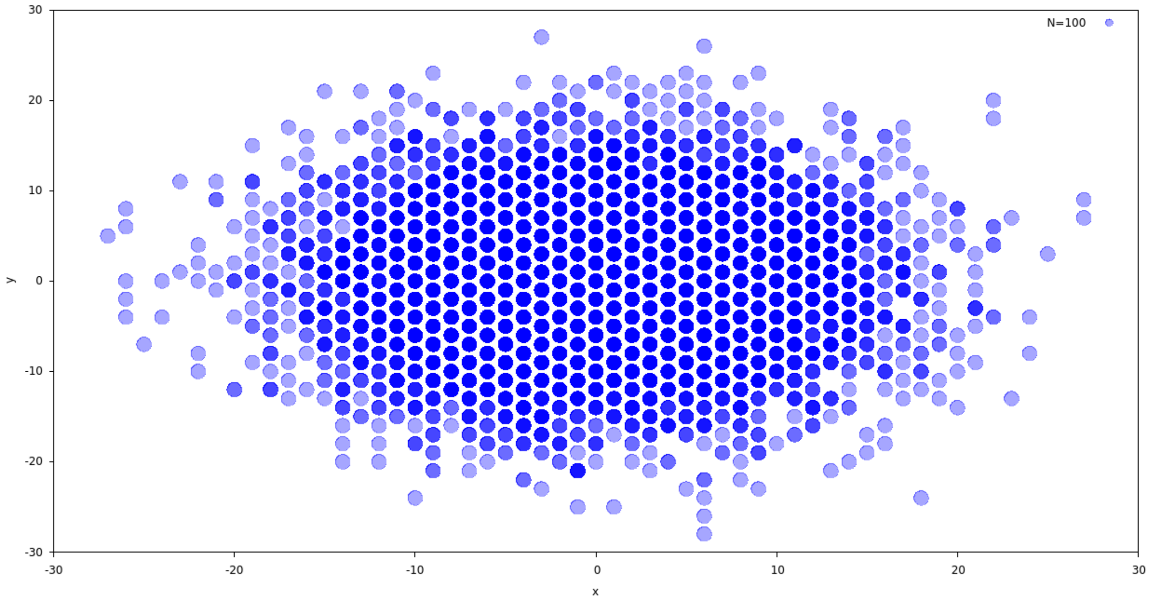
```fortran
Program C

!Random walk, gerando coordenada final de cada andarilho
open(10, file = 'saida-C(10.1)-10260434.dat')

!Número de andarilhos
iandarilhos = 10000

!Passos
N = 10

xsoma = 0.0
ysoma = 0.0
x2soma = 0.0
y2soma = 0.0

!simulação do andarilho
do i=1, iandarilhos
        !Posição inicial do andarilho i
        x = 0
        y = 0
        do j=1, N
                r = rand()                !numero aleatório
                if(r .lt. 0.25 ) then                !Distribuição
                 ↪  uniforme de probabilidade
                x = x - 1
                elseif((r .gt. 0.25) .and. (r .lt. 0.5)) then
                x = x + 1
                elseif ((r .gt. 0.5) .and. (r .lt. 0.75)) then
                y = y + 1
                elseif ((r .gt. 0.75) .and. (r .lt. 1)) then
                y = y - 1
                end if
                end do
        write(10,*) x,y
        !calculando desvio e médias
        xsoma = xsoma + x
        x2soma = x2soma + x**2
        ysoma = ysoma + y
        y2soma = y2soma + y**2
end do
```

```
!medias de cada componente
vxmedia = xsoma/iandarilhos
vymedia = ysoma/iandarilhos
print*, '<r> = (',vxmedia,vymedia,')'

!r*r
vr2 = x2soma/iandarilhos + y2soma/iandarilhos

!produto escalar de <r>*<r>
pescalar = vxmedia*vxmedia + vymedia*vymedia

!dispersão**2
disp =  vr2 - pescalar
print*, 'disp**2 = ', disp

close(10)
End Program C
```

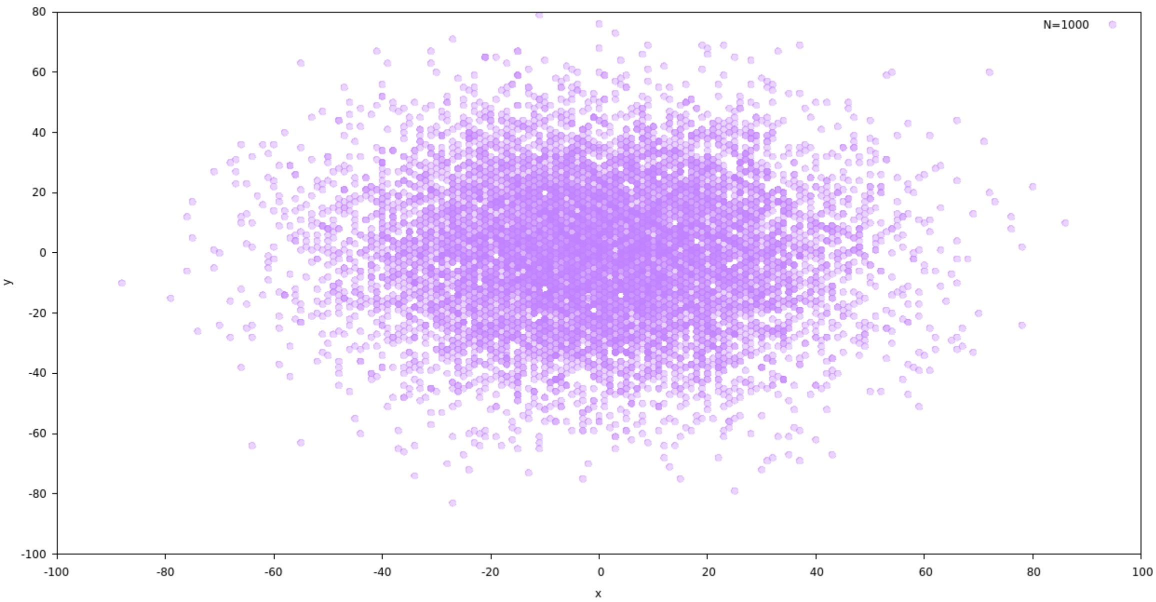
Os gráficos a seguir foram feitos com pontos semi-transparentes, assim, é possível visualizar onde há maior concentração de andarilhos em determinadas posições. Verifica-se que há uma aglomeração nas proximidades da origem, porém quanto maior o número de passos, mais longe os andarilhos podem ser encontrados, mesmo que todas as direções tenham igual probabilidade. Isso indica que com o aumento da dimensão, menor se torna a probabilidade do andarilho voltar à posição inicial.



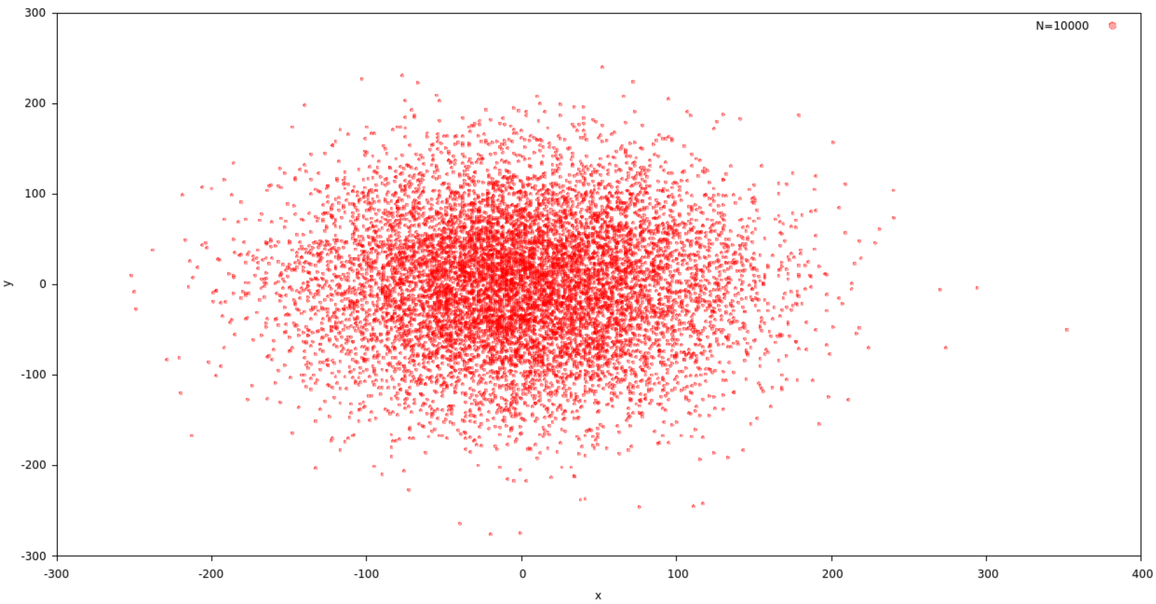
(a) Posições finais dos andarilho para simulação bidimensional. N=10.



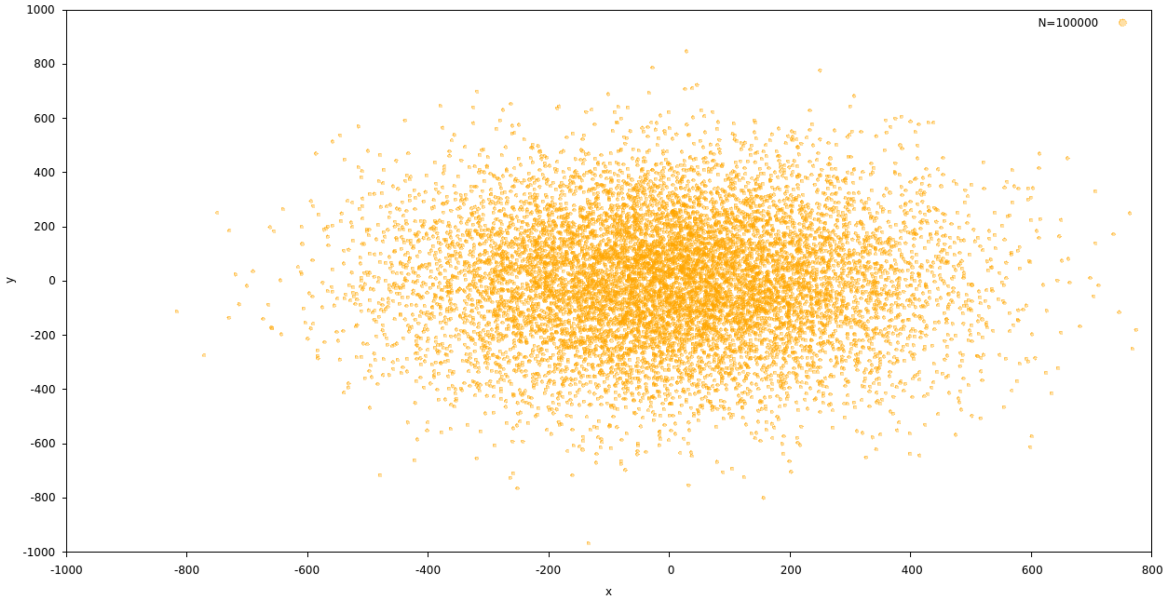
(b) Posições finais dos andarilho para simulação bidimensional. N=100.



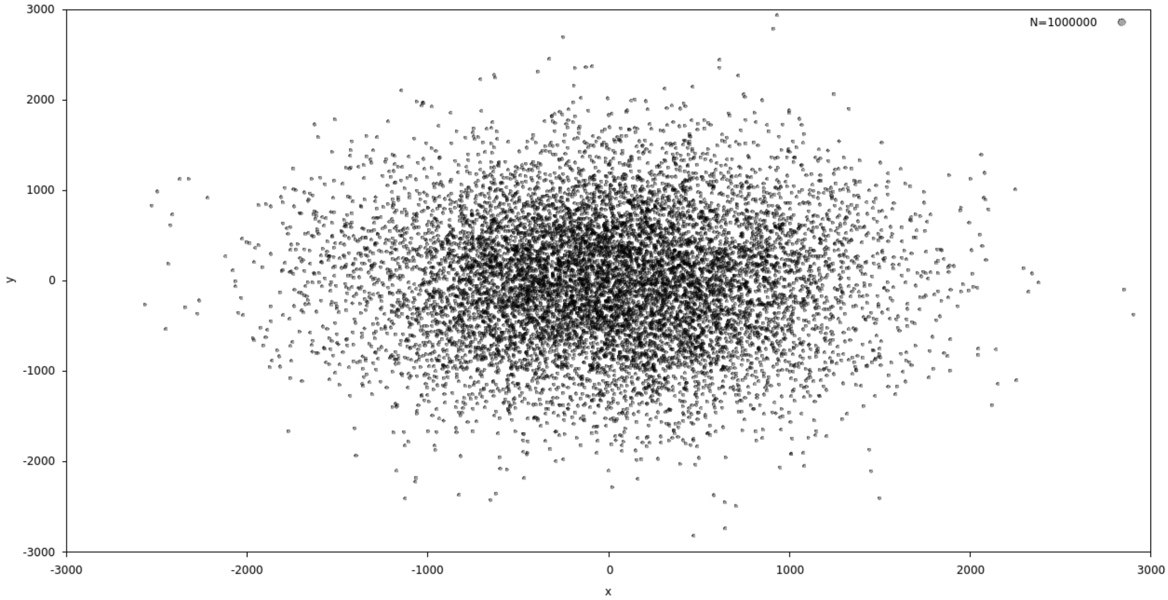
(c) Posições finais dos andarilho para simulação bidimensional. N=1000.



(d) Posições finais dos andarilho para simulação bidimensional. N=10000.



(e) Posições finais dos andarilho para simulação bidimensional. N=100000.



(f) Posições finais dos andarilho para simulação bidimensional. N=1000000.

Figura 3: Gráficos das posições finais dos andarilhos com variação do número de passos N.

Com a tabela a seguir é possível verificar o que foi discutido acima. Observando o vetor médio de cada variação de N, nota-se que quanto maior o valor de N, menos próximo ele se localiza da origem - correspondente a maior dispersão de andarilhos apresentada nos gráficos. Também podemos perceber que $\triangle^2$ aproxima-se do número de passos, da mesma forma que o caso unidimensional.

Andarilhos	N	$\langle \vec{r} \rangle$		$\triangle^2$
		x	y	
10000	10	-1.77999996E-02	1.99999996E-02	9.83248329
10000	10^2	-2.20999997E-02	7.93000013E-02	100.971024
10000	10^3	-2.34999992E-02	-0.105899997	1015.22443
10000	10^4	0.358999997	-0.418199986	9863.49023
10000	10^5	2.13569999	-1.42120004	100666.266
10000	10^6	1.29030001	-2.28049994	994600.625

Assim como problema 1D, a análise bidimensional também pode ser relacionada a equação de difusão, sendo mais uma possibilidade de verificar como mecanismos microscópicos podem chegar ao limite macroscópico ao aumentar o número de andarilhos ao infinito e utilizar escalas maiores. Um código alternativo foi criado para verificar o caminho dos andarilhos ao longo do espaço (*'tarefa-C1-10260434.f90'*) e, com auxílio de algumas bibliotecas do Python foi obtido o seguinte gráfico:

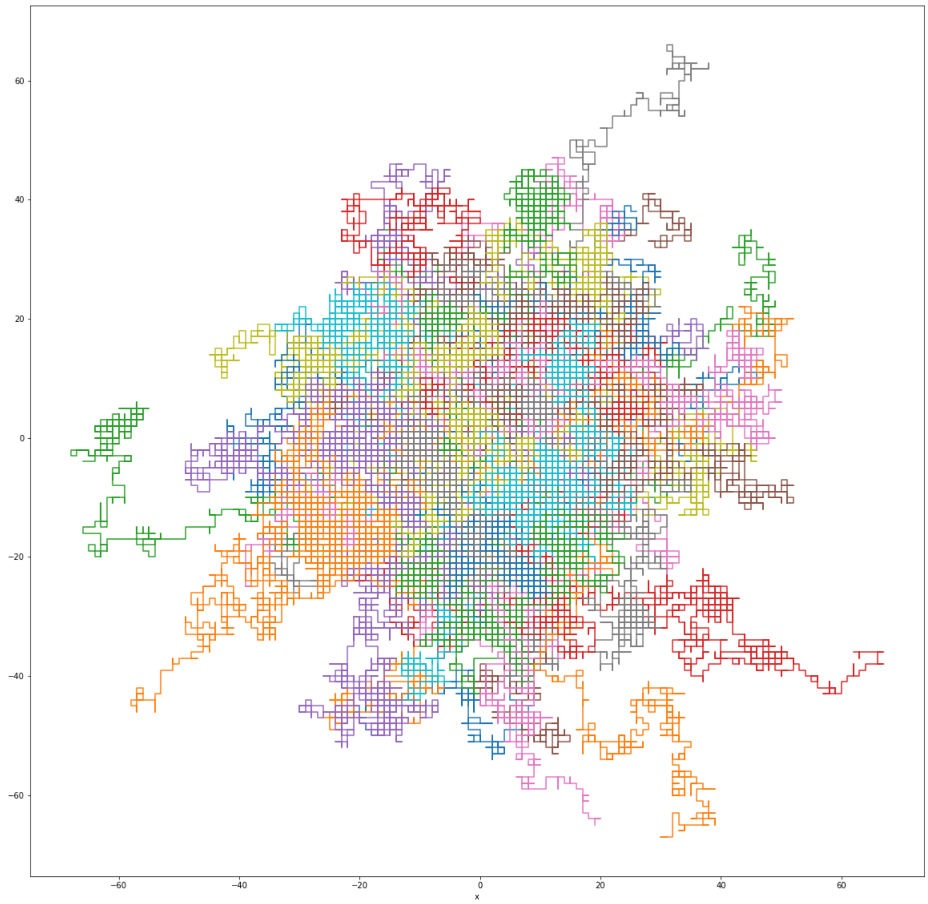


Figura 4: Posição dos andarilhos no espaço 2D. x *vs* y. (100 andarilhos, 100 passos)

4 Tarefa D

Esta tarefa foi abordada de maneira diferente para facilitar a forma como o problema é lidado, assim, ao invés de usar arquivos que armazenam as posições, foram utilizados vetores. Para calcular a entropia, foram utilizados 2000 andarilhos, cada um dando 10000 passos e com posição inicial na origem. A probabilidade de encontrar o sistema em certo microestado 'i' foi calculada dividindo a quantidade de andarilhos presente em um reticulado.

$$S = -\sum_{i=1}^{N} P_i ln P_i \tag{5}$$

Portanto, o espaço 2D foi dividindo em quadrantes de lado=30, adicionando-se um contador de andarilhos para cada quadrante após cada passo. Além disso, cada passo é considerado como um instante de tempo e o espaço analisado foi delimitado a um valor máximo pmax=1200, tornando a análise constante a cada passo.

tarefa-D-10260434.f90

```fortran
Program Entropia

parameter (nmax= 100000)
dimension ix(nmax), iy(nmax)

iandarilhos = 1000
N = 30000

do i=1, iandarilhos
        ix(i) = 0
        iy(i) = 0
end do

!simulação do andarilho por vetores
p = 0.25                !probabilidade

do i = 1, N !cada passo será contado como um instante de tempo
        do j = 1, iandarilhos         !todos os andarilho dão um passo ao
         ↪  mesmo tempo
                r = rand()
                if (r .lt. 0.25) then
                ix(j) = ix(j) + 1
                elseif(r .gt. 0.25 .and. r .lt. 0.5) then
                ix(j) = ix(j) - 1
                elseif (r .gt. 0.5 .and. r .lt. 0.75) then
                iy(j) = iy(j) + 1
                else
                iy(j) = iy(j) -1
                end if
        end do

!'do' inicial não é fechado

!contagem de andarilhos em subdivisões do espaço 2D p/ calcular
```

```
!a probabilidade de encontrá-los naquele microestado

open(20, file = 'saida-D-entrop-10260434.dat')

stotal = 0.0        !somatório da entropia p/ cada microestado
lado = 30               !tamanho do quadrante
pmax = 1200               !tamanho do espaço analisado

!divisão do espaço 2D em quadrantes ladoxlado
!percorre um quadrante de tamanho= 'lado' em x e todas as possibilidade em y

        do k=-pmax,pmax-lado,lado        !eixo x
                do l=-pmax,pmax-lado,lado        !eixo y
                        cont = 0.0                   !contador de andarilhos dentro do quadrante
                        do m=1, iandarilhos                     !condição para estar dentro do quadrante
                                if ((ix(m) .le. k+lado) .and. (ix(m) .ge. k) .and. &
                                & (iy(m) .le. l+lado ) .and. (iy(m) .ge. l)) then
                                cont= cont+1
                        end if
                        end do
                if (cont .ne. 0) then        !evitar log(0)
                        prob = cont/iandarilhos
                        s = prob*log(prob)         !entropia
                        stotal = stotal - s
                end if
                end do
        end do
        write(20,*) i, stotal
end do
close(20)

End Program Entropia
```

4.0.1 Resultados

O seguinte gráfico foi obtido para a entropia:

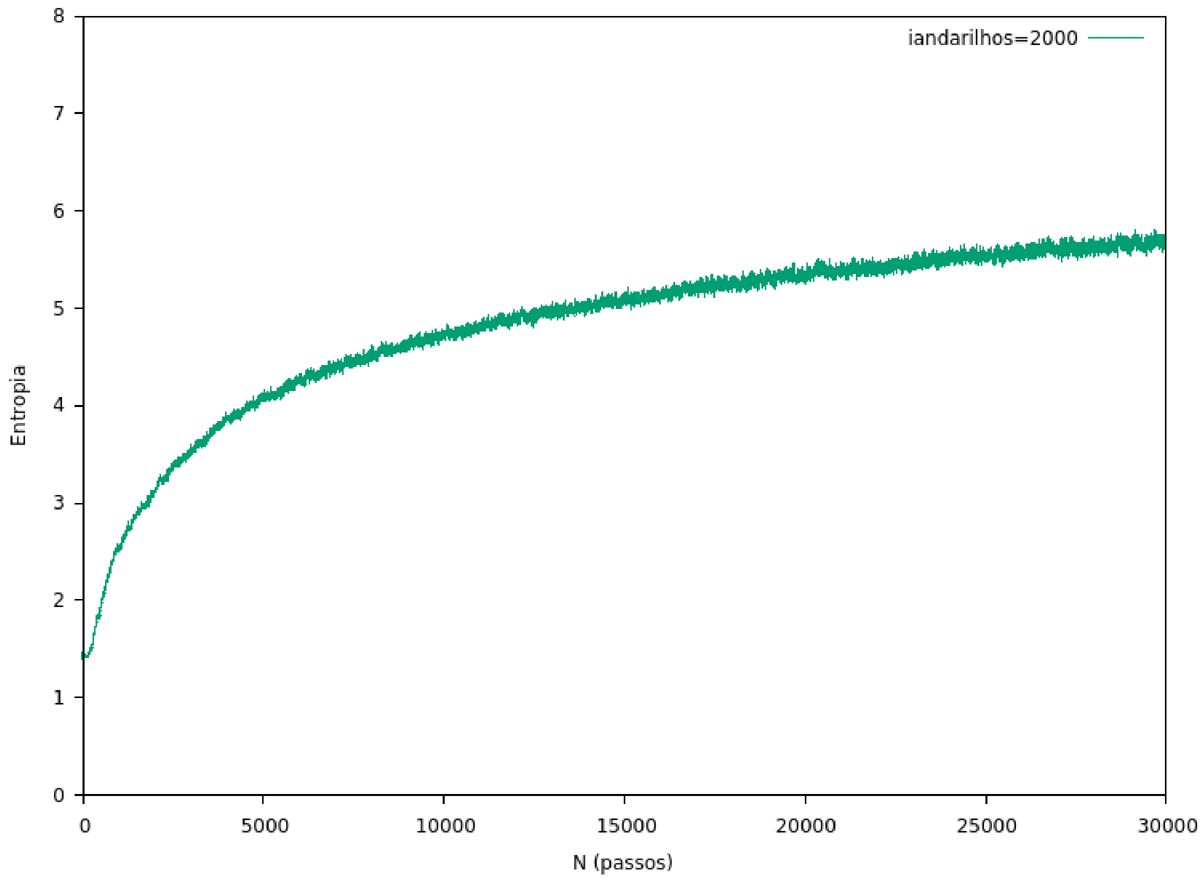


Figura 5: Evolução da entropia no sistema.

Podemos observar que a entropia cresce com o tempo e existem flutuações, as quais se devem devido ao tamanho finito das células. Como esperado, a curva tem comportamento logarítmico e observa-se que, apesar do crescimento constante, há diminuição na taxa de crescimento, indicando que o sistema está se aproximando do estado de equilíbrio.

Todos os gráficos foram verificados inicialmente pelo xmgrace, mas por fins de visualização, os que estão sendo apresentados foram processados no Gnuplot.

Referências

DESCONHECIDO. **Distribuição normal**. Disponível em: ¡https://pt.wikipedia.org/wiki/Distribui%C3%A7%C3%A3o_normal¿. (Acessado: 31.03.2020).

________. **Passeio aleatório**. Disponível em: ¡https://pt.wikipedia.org/wiki/Passeio_aleat%C3%B3rio#Dimens%C3%B5es_superiores¿. (Acessado: 31.03.2020).

NORDLUND, Kai. **Random Walks**. Disponível em: ¡http://beam.helsinki.fi/~knordlun/mc/mc7nc.pdf¿. (Acessado: 30.03.2020).

PERESSI, M. **Random Walks and Diffusion**. Disponível em: ¡http://www-dft.ts.infn.it/~peressi/rw.pdf¿. (Acessado: 31.03.2020).

REIF, Frederick. **Fundamentals of statistical and thermal physics**. [S.l.]: Waveland Press, 2009.

SALINAS, Silvio RA. **Introdução a física estatística vol. 09**. [S.l.]: Edusp, 1997.

THE One-Dimensional Random Walk. Disponível em: ¡http://galileo.phys.virginia.edu/classes/152.mf1i.spring02/RandomWalk.htm¿. (Acessado: 31.03.2020).

THEUNS, Tom. **Lecture 6: Random Walks**. Disponível em: ¡http://star-www.dur.ac.uk/~tt/MSc/Lecture6.pdf¿. (Acessado: 28.03.2020).