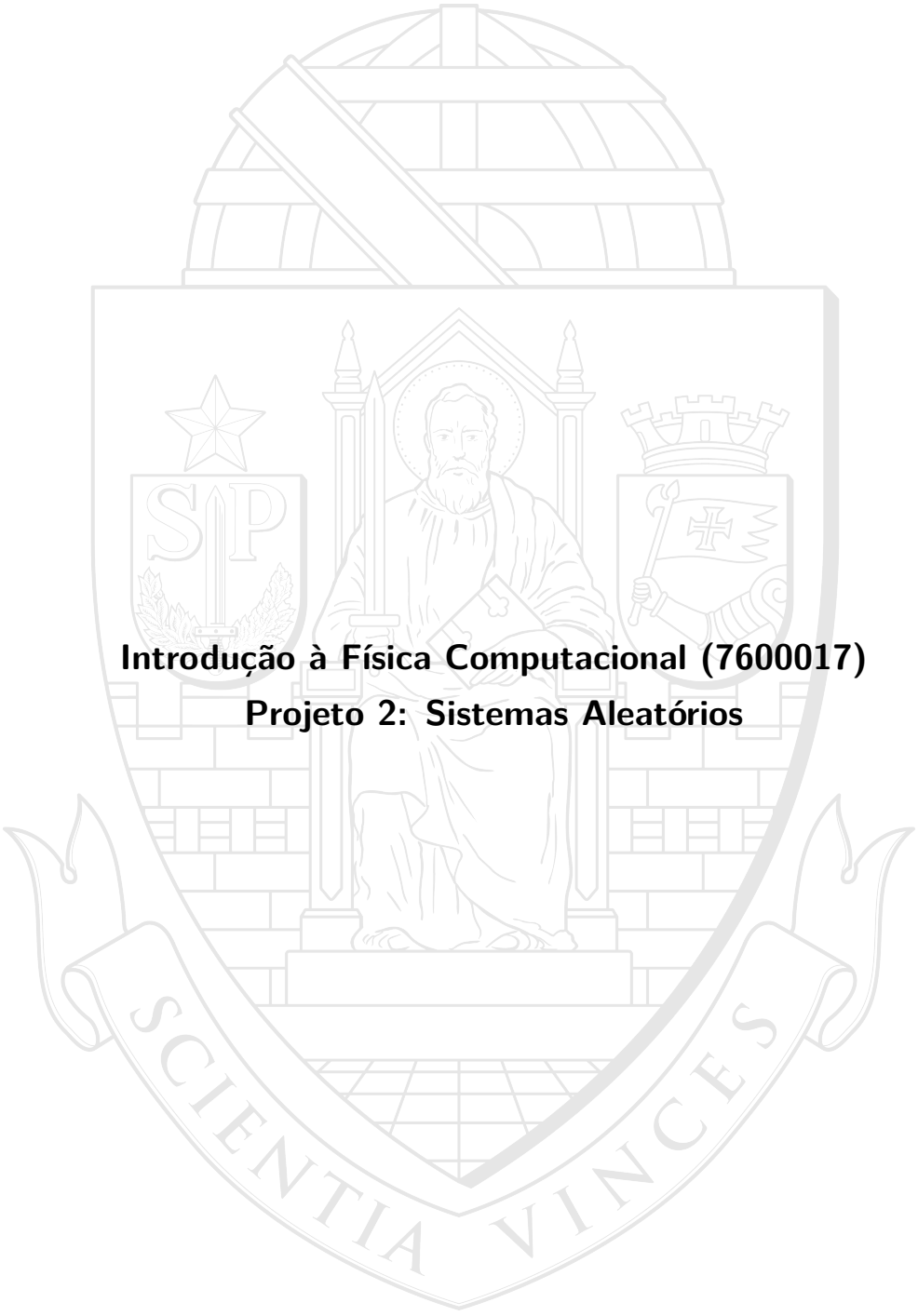
GUSTAVO HENRIQUE FREITAS DE RESENDE

N.º USP: 12345678



Introdução à Física Computacional (7600017)

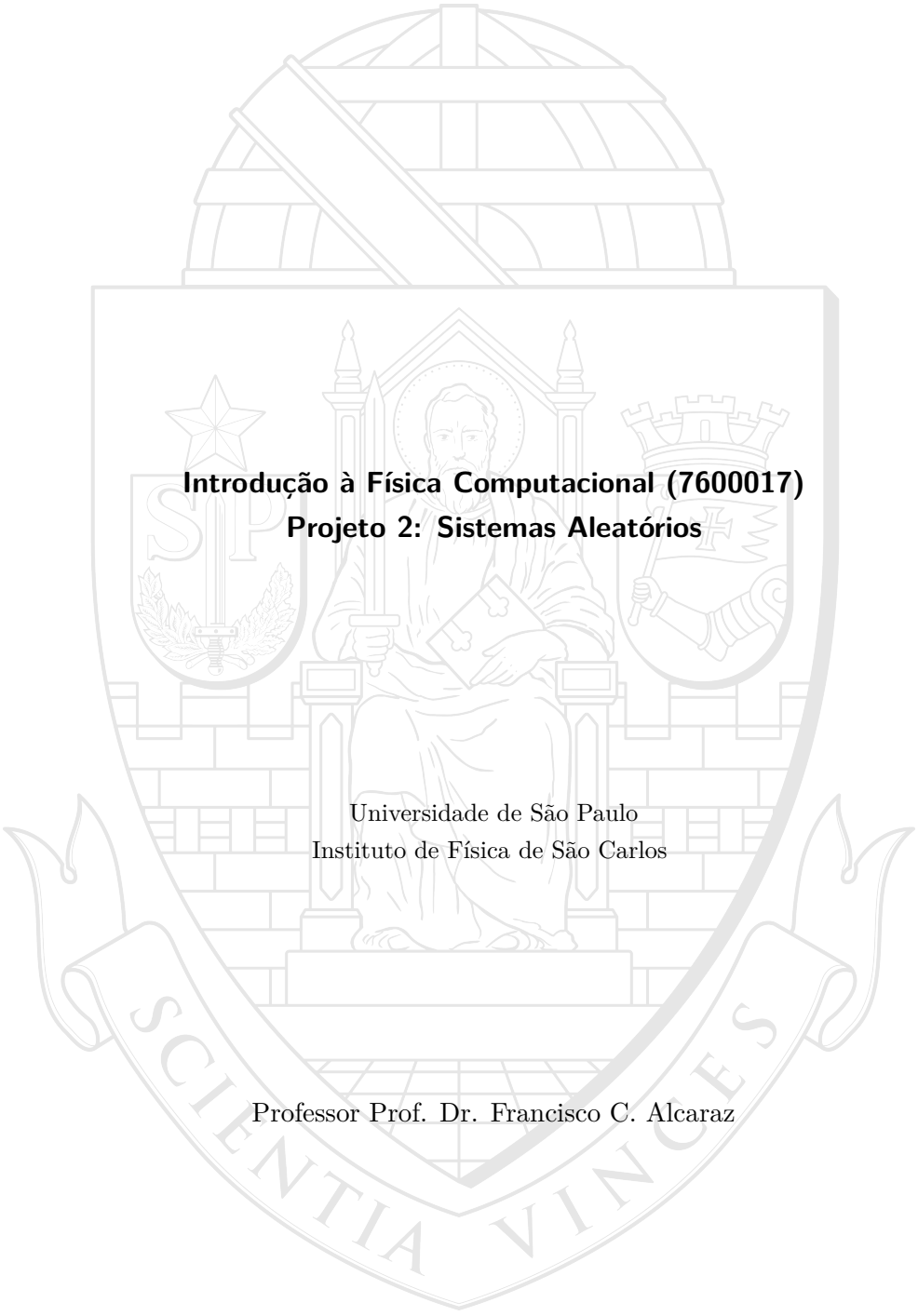
Projeto 2: Sistemas Aleatórios

São Carlos – SP

2025

GUSTAVO HENRIQUE FREITAS DE RESENDE

N.º USP: 12345678



Introdução à Física Computacional (7600017)
Projeto 2: Sistemas Aleatórios

Universidade de São Paulo
Instituto de Física de São Carlos

Professor Prof. Dr. Francisco C. Alcaraz

São Carlos – SP
2025

Resumo

Este relatório aborda o estudo de sistemas aleatórios através de simulações computacionais, conforme proposto no segundo projeto da disciplina de Introdução à Física Computacional. O trabalho explora a geração e validação de números pseudo-aleatórios pelo método dos geradores congruentes lineares, analisando os momentos estatísticos de sua distribuição. Subsequentemente, aplicam-se estes métodos à simulação do problema do andarilho aleatório em uma e duas dimensões, investigando a posição média e o deslocamento quadrático médio dos caminhantes. **Palavras-chave**: Física Computacional, Sistemas Aleatórios, Andarilho

Aleatório, Entropia, Método de Monte Carlo.

Sumário

1 Introdução

Este relatório versa sobre a implementação e os resultados obtidos para o Projeto 2 da disciplina de Introdução à Física Computacional. O foco do projeto é a simulação de sistemas cuja dinâmica é governada por processos estocásticos. Serão abordados desde a geração de números aleatórios e a verificação de suas propriedades estatísticas até a aplicação em modelos físicos como o do andarilho aleatório e a evolução da entropia.

1.1 Objetivos

Os principais objetivos deste trabalho são:

- Implementar um gerador de números pseudo-aleatórios e testar sua qualidade através do cálculo de momentos da distribuição.
- Simular o movimento de andarilhos aleatórios em uma dimensão, para casos com e sem viés, e analisar o comportamento da distribuição de posições.
- Estender a simulação para o caso bidimensional, visualizando as trajetórias e calculando o deslocamento quadrático médio.
- Calcular a entropia do sistema de andarilhos bidimensionais em função do tempo (número de passos) para observar seu aumento, caracterizando a seta do tempo.

2 Metodologia

2.0.1 Compilação e Ferramentas

Todo o código-fonte em Fortran foi compilado utilizando o **GNU Fortran (gfortran)**.

Para garantir a compatibilidade com o código legado no padrão FORTRAN 77 (requisito segundo 'filosofia do curso') foram utilizadas as seguintes flags de compilação:

- `-x f77`: Instrui explicitamente o compilador a tratar os arquivos de entrada como código FORTRAN 77, independentemente da extensão do arquivo.
- `-std=legacy`: Habilita o suporte a extensões e comportamentos não-padrão que eram comuns em compiladores F77 mais antigos. Isso é crucial para compilar código histórico que pode depender de funcionalidades específicas de um determinado fornecedor.
- `-ffixed-form`: Especifica que o código-fonte segue o formato fixo do FORTRAN 77, onde a posição de um caractere em uma linha determina seu significado (e.g., colunas 1-5 para rótulos, coluna 6 para continuação).
- `-fno-automatic`: Força o compilador a alocar variáveis locais de forma estática, em vez de na pilha ("automática"). Isso emula o comportamento padrão do FORTRAN 77, onde variáveis mantêm seu valor entre chamadas de sub-rotinas, prevenindo bugs.
- `-fno-implicit-none`: Desativa a necessidade da declaração `implicit none`. Os códigos utilizados neste trabalho seguem a convenção de tipagem implícita do FORTRAN 77 (variáveis iniciadas com I-N são inteiras, as demais são reais), e esta flag permite que o código seja compilado sem exigir a declaração explícita de cada variável.
- `-Wall` e `-Wextra`: Ativam um conjunto extenso de avisos (*warnings*) sobre construções de código que, embora não sejam erros, são potencialmente problemáticas ou de má prática.
- `-pedantic`: Faz com que o compilador emita todos os avisos exigidos pelo padrão estrito da linguagem, rejeitando programas que usem extensões.

Ao desenvolver a atividade nenhuma biblioteca do compilador ou recursos de versões mais modernas do Fortran foram utilizadas. REAL*16, INTEGER*8 ou similares também **não** fazem parte do padrão Fortran 77 [Sun Microsystems, Inc. 1995], e portanto foram utilizadas as flags especificadas a fim de desabilitar tais recursos também. Caso contrário não estaríamos respeitando o padrão do Fortran 77 exigido.

Compilar código FORTRAN 77 em um compilador moderno sem flags de padronização rigorosas oferece apenas uma ilusão de conformidade, pois extensões e recursos não-padrão podem ser aceitos silenciosamente.

2.1 Geração de Números Aleatórios

A base para todas as simulações é um gerador de números pseudo-aleatórios. Utilizou-se a função intrínseca da linguagem de programação, inicializada com uma "semente"para garantir a reprodutibilidade dos

resultados. Para validar a uniformidade da distribuição no intervalo $[0, 1)$, foram calculados os primeiros quatro momentos, definidos como:[Alcaraz 2025]

$$\langle x^n \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i^n \tag{2.1}$$

onde N é o número total de amostras geradas. Para uma distribuição uniforme contínua, o valor teórico esperado é $\langle x^n \rangle = \frac{1}{n+1}$.

2.2 Andarilho Aleatório

2.2.1 Caso Unidimensional

Em 1D, a cada passo de tempo, o andarilho move-se para a direita com probabilidade p ou para a esquerda com probabilidade $q = 1 - p$. A posição x após N passos é a soma dos deslocamentos individuais. Foram analisados os casos:[Alcaraz 2025]

- **Não enviesado:** $p = q = 1/2$.
- **Enviesado:** $p \neq q$.

Para um grande número de andarilhos, foram gerados histogramas da posição final e calculados os valores de $\langle x \rangle$ e $\langle x^2 \rangle$.

2.2.2 Caso Bidimensional

Em 2D, o andarilho move-se com igual probabilidade ($p = 1/4$) para uma das quatro direções cardeais (norte, sul, leste, oeste). Foi calculado o deslocamento quadrático médio, $\Delta^2 = \langle \vec{r}^2 \rangle - \langle \vec{r} \rangle^2$, que é uma medida da difusão das partículas.

2.3 Cálculo da Entropia

Para observar a seta do tempo, a entropia de Shannon foi calculada para o sistema de M andarilhos em 2D. O espaço foi discretizado em uma grade e a entropia foi calculada como:[Alcaraz 2025]

$$S = -\sum_i P_i \ln P_i \tag{2.2}$$

onde P_i é a probabilidade de encontrar um andarilho na célula i da grade, estimada pela fração de andarilhos naquela célula, $P_i = M_i/M$. Este cálculo foi repetido para diferentes números de passos N para observar a evolução de $S(t)$.

3 Resultados

3.1 Validação do Gerador de Números Pseudo-Aleatórios

A primeira etapa do projeto consistiu na validação do gerador de números pseudo-aleatórios. O objetivo é verificar se a sequência de números gerada no intervalo $[0, 1)$ segue uma distribuição uniforme. Para isso, foram calculados os quatro primeiros momentos da distribuição, $\langle x^n \rangle$, e comparados com os valores teóricos esperados.

3.1.1 Cálculo Teórico dos Momentos

Para uma variável aleatória contínua X com uma distribuição de probabilidade uniforme no intervalo $[0, 1)$, a sua função de densidade de probabilidade (PDF) é:

$$f(x) = \begin{cases} 1, & \text{se } 0 \leq x \leq 1 \\ 0, & \text{caso contrário} \end{cases} \tag{3.1}$$

O n-ésimo momento de uma distribuição contínua é definido como o valor esperado de X^n:

$$\langle x^n \rangle = E[X^n] = \int_{-\infty}^{\infty} x^n f(x)\, dx \tag{3.2}$$

$$\langle x^n \rangle = \int_0^1 x^n \cdot 1\, dx = \left[\frac{x^{n+1}}{n+1} \right]_0^1 = \frac{1^{n+1}}{n+1} - \frac{0^{n+1}}{n+1} \tag{3.3}$$

Portanto, o valor teórico esperado para o n-ésimo momento de uma distribuição uniforme em $[0, 1)$ é:

$$\langle x^n \rangle = \frac{1}{n+1} \tag{3.4}$$

Com base nesta fórmula, os valores teóricos esperados para $n = 1, 2, 3, 4$ são:

- $\langle x^1 \rangle = \frac{1}{1+1} = \frac{1}{2} = 0.5$
- $\langle x^2 \rangle = \frac{1}{2+1} = \frac{1}{3} \approx 0.33333$
- $\langle x^3 \rangle = \frac{1}{3+1} = \frac{1}{4} = 0.25$
- $\langle x^4 \rangle = \frac{1}{4+1} = \frac{1}{5} = 0.2$

3.1.2 Código-Fonte da Simulação

O cálculo experimental dos momentos foi realizado utilizando o código em Fortran 77 a seguir, que gera $N = 10^8$ números pseudo-aleatórios e calcula as médias de suas potências.

```
1       BLOCK DATA
2       INTEGER ISEED
3       COMMON /SEEDBLK/ ISEED
4       DATA ISEED /1154/
```

```
5       END
6
7       PROGRAM MAIN
8       INTEGER I, ISIZE
9       PARAMETER ( ISIZE = 100000000 )
10      DOUBLE PRECISION RANDS, RVAL, M1, M2, M3, M4
11      DATA M1, M2, M3, M4 /4*0.0/
12
13      DO 10 I = 1, ISIZE
14         RVAL = RANDS()
15         M1 = M1 + RVAL ** 1
16         M2 = M2 + RVAL ** 2
17         M3 = M3 + RVAL ** 3
18         M4 = M4 + RVAL ** 4
19  10  CONTINUE
20      M1 = M1 / ISIZE
21      M2 = M2 / ISIZE
22      M3 = M3 / ISIZE
23      M4 = M4 / ISIZE
24
25      WRITE (6,'(F7.5,A,F7.5,A,F7.5,A,F7.5)') M1,' ',M2,' ',M3,' ',M4
26      END
27
28      FUNCTION RANDS()
29      DOUBLE PRECISION RANDS
30      INTEGER ISEED
31      COMMON /SEEDBLK/ ISEED
32
33  c   PARAMETROS PARA O GERADOR MINSTD (A=16807, M=2**31-1)
34      INTEGER A, M, Q, R
35      PARAMETER (A = 16807)
36      PARAMETER (M = 2147483647)
37  c   PARAMETROS PARA O METODO DE SCHRANGE (PRE-CALCULADOS)
38  c   Q = M / A = 127773
39  c   R = M % A = 8776
40      PARAMETER (Q = 127773)
41      PARAMETER (R = 8776)
42
43      INTEGER TEST
44
45  c   METODO DE SCHRANGE PARA CALCULAR '(A*ISEED)%M' SEM OVERFLOW
46      TEST = A * (MOD(ISEED, Q)) - R * (ISEED / Q)
47  c   E SE O RESULTADO FOR NEGATIVO, LEVAR AO DOMINIO CORRETO.
48      IF (TEST .LT. 0) THEN
49         TEST = TEST + M
50      END IF
51
52      ISEED = TEST
```

```
53        RANDS = DBLE(ISEED) / DBLE(M)
54        RETURN
55        END
```

3.1.3 Análise dos Resultados

A execução do código-fonte produziu a seguinte saída, correspondendo aos momentos experimentais $\langle x^1 \rangle, \langle x^2 \rangle, \langle x^3 \rangle, \langle x^4 \rangle$:

```
0.50066 0.33417 0.25085 0.20082
```

A Tabela 1 a seguir compara os resultados obtidos com os valores teóricos esperados, permitindo uma análise quantitativa da qualidade do gerador.

Tabela 1 – Comparação entre os momentos teóricos e os experimentais.

Momento (n)	**Valor Teórico**	**Valor Experimental**	**Erro Relativo (%)**
1	0.50000	0.50066	0.132
2	0.33333	0.33417	0.252
3	0.25000	0.25085	0.340
4	0.20000	0.20082	0.410

Os resultados experimentais demonstram uma excelente concordância com os valores teóricos. Os erros relativos são consistentemente baixos, na ordem de $10^{-1}\%$, o que é esperado para o tamanho da amostra utilizada ($N = 10^8$). Essas pequenas discrepâncias são atribuídas às flutuações estatísticas inerentes a qualquer simulação com um número finito de amostras. A proximidade entre os valores calculados e os esperados valida a qualidade do gerador de números pseudo-aleatórios, que se mostra adequado para a utilização nas demais simulações do projeto.

3.2 Andarilho Aleatório Unidimensional

Nesta etapa, foi simulado o problema do andarilho aleatório em uma dimensão. Analisou-se o comportamento de um conjunto de M andarilhos após N passos, investigando a distribuição de suas posições finais e os valores médios de deslocamento.

3.2.1 Caso Não Enviesado ($p = q = 1/2$)

O primeiro caso analisado é o do andarilho "desnorteado", onde a probabilidade de dar um passo para a direita (p) é igual à de dar um passo para a esquerda (q), sendo $p = q = 1/2$. Foi realizada uma simulação com $M = 10.000$ andarilhos, cada um dando $N = 10.000$ passos, todos partindo da origem ($x = 0$).

3.2.1.1 Código-Fonte da Simulação (Fortran)

O comportamento dos andarilhos foi simulado utilizando o código em Fortran 77 apresentado abaixo. O programa calcula a posição final de cada um dos M andarilhos, salva essas posições em um arquivo ‘positions.dat‘ e, ao final, computa os valores médios $\langle x \rangle$ e $\langle x^2 \rangle$.

```
BLOCK DATA
      INTEGER ISEED
      COMMON /SEEDBLK/ ISEED
      DATA ISEED /1154/
      END

      FUNCTION RANDS()
      DOUBLE PRECISION RANDS
      INTEGER ISEED
      COMMON /SEEDBLK/ ISEED

      INTEGER A, M, Q, R
      PARAMETER (A = 16807)
      PARAMETER (M = 2147483647)
      PARAMETER (Q = 127773)
      PARAMETER (R = 8776)
      INTEGER TEST

      TEST = A * (MOD(ISEED, Q)) - R * (ISEED / Q)
      IF (TEST .LT. 0) THEN
         TEST = TEST + M
      END IF

      ISEED = TEST
      RANDS = DBLE(ISEED) / DBLE(M)
      RETURN
      END

      FUNCTION STEP(DPROB)
      REAL STEP, RVAL
      RVAL = RAND()
      IF ( RVAL .LT. DPROB ) THEN
         STEP = 1
      ELSE
         STEP = -1
      END IF
      RETURN
      END

      PROGRAM MAIN
      INTEGER NSTEP, MWALK, IOS, UNIT
      PARAMETER (DPROB=0.5,NSTEP=10000,MWALK=10000)
      REAL DPROB, X_POS(NSTEP,MWALK), X_MEAN(MWALK)
      REAL AVG1, AVG2
      DATA AVG1, AVG2 /2*0.0/
      DATA (X_POS(1,I),I=1,MWALK)/MWALK*0.0D0/
```

```
      DATA (X_MEAN(I),I=1,MWALK)/MWALK*0.0D0/

      DO 10 J = 2, NSTEP
         DO K = 1, MWALK
            X_POS(J,K) = STEP(DPROB)
            X_MEAN(K) = X_MEAN(K) + X_POS(J,K)
         END DO
10    CONTINUE

          UNIT = 10
      OPEN(UNIT=UNIT,FILE='positions.dat',STATUS='NEW',IOSTAT=IOS)
      IF ( IOS .NE. 0 ) THEN
         WRITE(6,'(A,I1,A,I2)') 'I/O ERROR COD: ', IOS,
     1 ' WHEN OPENING UNIT: ', UNIT
      ELSE
         WRITE(UNIT,*) X_MEAN
         CLOSE(UNIT)
      END IF

      DO I=1, MWALK
         AVG1 = AVG1 + X_MEAN(I)
         AVG2 = AVG2 + (X_MEAN(I) ** 2)
      END DO

      AVG1 = AVG1/MWALK
      AVG2 = AVG2/MWALK

      WRITE(6,*) AVG1,'', AVG2
      END
```

3.2.1.2 Código-Fonte da Análise e Visualização (Python)

Para visualizar a distribuição das posições finais e realizar o ajuste gaussiano, foi utilizado o seguinte script em Python. Ele lê os dados gerados pelo programa em Fortran (após uma etapa intermediária de binning para criar o 'histogram.dat'), plota o histograma, ajusta uma curva gaussiana e salva a imagem final.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def gaussian(x, A, mu, sigma):
    """Define a função Gaussiana para o ajuste."""
    return A * np.exp(-(x - mu)**2 / (2 * sigma**2))

# Carrega os dados do histograma (posições x e contagens y)
x_data, y_data = np.loadtxt('histogram.dat', unpack=True)

```

```
# Estimativas iniciais para os parâmetros de ajuste
initial_guesses = [np.max(y_data), x_data[np.argmax(y_data)], 20]

# Realiza o ajuste da curva (curve fitting)
popt, pcov = curve_fit(gaussian, x_data, y_data, p0=initial_guesses)

# Gera pontos para a curva ajustada
x_fit = np.linspace(x_data.min(), x_data.max(), 500)
y_fit = gaussian(x_fit, *popt)

# Cria o gráfico
plt.figure(figsize=(10, 6))
plt.bar(x_data, y_data, width=2.0, label='Data from Simulation', color='steelblue', alpha=0.7)
plt.plot(x_fit, y_fit, 'r-', linewidth=2, label='Best Gaussian Fit')
plt.title('Histogram of Final Positions and Gaussian Fit', fontsize=16)
plt.xlabel('Final Position (x)', fontsize=12)
plt.ylabel('Number of Walkers n(x)', fontsize=12)
plt.legend(fontsize=10)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlim(min(x_data)-10, max(x_data)+10)

# Imprime os parâmetros ótimos
print("Parâmetros Ótimos da Curva Gaussiana Ajustada:")
print(f"  Amplitude (A) = {popt[0]:.2f}")
print(f"  Média () = {popt[1]:.2f}")
print(f"  Desvio Padrão () = {popt[2]:.2f}")

# Salva a figura
plt.savefig('histogram_fit.png')
```

3.2.1.3 Histograma e Análise dos Resultados

A execução dos códigos resultou no histograma da Figura 1 e nos valores experimentais para $\langle x \rangle$ e $\langle x^2 \rangle$. A forma da distribuição se assemelha a uma curva Gaussiana, como esperado pelo Teorema do Limite Central para $N \gg 1$.

3.2.1.3.1 Valores Teóricos vs. Experimentais:

As expressões teóricas para o caso não enviesado ($p = q = 1/2$) com $N = 10.000$ passos são $\langle x \rangle = 0$ e $\langle x^2 \rangle = N = 10.000$. O desvio padrão teórico é $\sigma = \sqrt{N} = 100$. A Tabela 2 compara estes valores com os obtidos na simulação.

Os resultados experimentais estão em excelente concordância com os valores previstos pela teoria. O deslocamento médio é muito próximo de zero, e tanto o $\langle x^2 \rangle$ quanto o σ do ajuste gaussiano estão muito próximos dos seus valores teóricos de N e $\sqrt{N}$, respectivamente. Isso valida a simulação e confirma o comportamento difusivo do sistema.

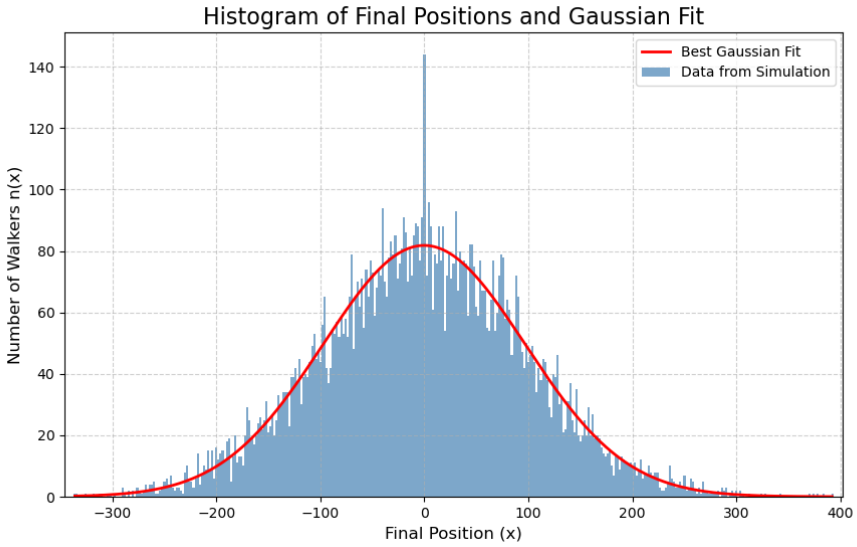


Figura 1 – Histograma da distribuição de posições finais $n(x)$ para 10.000 andarilhos após 10.000 passos. A curva vermelha representa um ajuste Gaussiano aos dados.

Tabela 2 – Comparação de resultados para o andarilho 1D não enviesado ($N = 10.000$).

Grandeza	**Valor Teórico**	**Valor Experimental**
Deslocamento Médio $\langle x \rangle$	0	-0.6846
Desloc. Quadrático Médio $\langle x^2 \rangle$	10000	9864.42
Desvio Padrão σ	100	97.09 (do ajuste)

3.3 Andarilho Aleatório Bidimensional

A simulação foi estendida para um espaço bidimensional, onde os andarilhos se movem em uma grade. A cada passo, cada andarilho tem a mesma probabilidade ($p = 1/4$) de se mover em uma das quatro direções cardeais: norte ($+y$), sul ($-y$), leste ($+x$) ou oeste ($-x$). O objetivo desta etapa é calcular o deslocamento vetorial médio $\langle \vec{r} \rangle$ e o deslocamento quadrático médio $\Delta^2 = \langle \vec{r}^2 \rangle - \langle \vec{r} \rangle^2$, além de visualizar a difusão das partículas no espaço.

3.3.1 Códigos-Fonte Utilizados

3.3.1.1 Simulação do Movimento (Fortran)

O código em Fortran abaixo foi utilizado para simular o movimento de $M = 1000$ andarilhos por até $N = 10^6$ passos. Em intervalos de tempo pré-definidos, o programa calcula as grandezas estatísticas de interesse e salva as posições de todos os andarilhos em arquivos de dados.

```
1       BLOCK DATA
2       INTEGER ISEED
3       COMMON /SEEDBLK/ ISEED
4       DATA ISEED /1154/
```

```
      END

      FUNCTION RANDS()
      DOUBLE PRECISION RANDS
      INTEGER ISEED
      COMMON /SEEDBLK/ ISEED

      INTEGER A, M, Q, R
      PARAMETER (A = 16807)
      PARAMETER (M = 2147483647)
      PARAMETER (Q = 127773)
      PARAMETER (R = 8776)
      INTEGER TEST

      TEST = A * (MOD(ISEED, Q)) - R * (ISEED / Q)
      IF (TEST .LT. 0) THEN
         TEST = TEST + M
      END IF

      ISEED = TEST
      RANDS = DBLE(ISEED) / DBLE(M)
      RETURN
      END

      PROGRAM MAIN
      INTEGER NSTEPS, MWALK
      PARAMETER (NSTEPS=1000000, MWALK=1000)
      REAL POS(2*MWALK)
      REAL AVGX, AVGY, AVGR2, DELTASQ
      DOUBLE PRECISION RANDS, RVAL
      INTEGER I, J, K, NOUT_IDX
      INTEGER NOUT(6)
      CHARACTER*20 FNAME
      DATA NOUT /10, 100, 1000, 10000, 100000, 1000000/

      DO 5 I=1, 2*MWALK
         POS(I) = 0.0
5     CONTINUE

      WRITE(*,*) 'NUMERO DE ANDARILHOS:', MWALK
      WRITE(*,'(A10, A15, A15, A15, A15)') 'N_PASSOS', '<X>', '<Y>',
     +     '<R^2>', 'DELTA^2'

      DO 10 J=1, NSTEPS
         DO 20 K=1, 2*MWALK, 2
            RVAL = RANDS()
            IF ( RVAL .LT. 0.25 ) THEN
               POS(K) = POS(K) + 1.0
```

```
            ELSE IF ( RVAL .LT. 0.50 ) THEN
               POS(K) = POS(K) - 1.0
            ELSE IF ( RVAL .LT. 0.75 ) THEN
               POS(K+1) = POS(K+1) + 1.0
            ELSE
               POS(K+1) = POS(K+1) - 1.0
            END IF
20       CONTINUE

         DO 30 NOUT_IDX=1, 6
            IF (J .EQ. NOUT(NOUT_IDX)) THEN
               AVGX  = 0.0
               AVGY  = 0.0
               AVGR2 = 0.0
               DO 40 I=1, 2*MWALK, 2
                  AVGX  = AVGX + POS(I)
                  AVGY  = AVGY + POS(I+1)
                  AVGR2 = AVGR2 + (POS(I)**2 + POS(I+1)**2)
40             CONTINUE
               AVGX  = AVGX / REAL(MWALK)
               AVGY  = AVGY / REAL(MWALK)
               AVGR2 = AVGR2 / REAL(MWALK)
               DELTASQ = AVGR2 - (AVGX**2 + AVGY**2)

               WRITE(*,'(I10, 4F15.5)') J, AVGX, AVGY, AVGR2, DELTASQ

               WRITE(FNAME, '(A,I7.7,A)') 'pos_N', J, '.dat'
               OPEN(10, FILE=FNAME, STATUS='UNKNOWN')
               DO 50 I=1, 2*MWALK, 2
                  WRITE(10,*) POS(I), POS(I+1)
50             CONTINUE
               CLOSE(10)
            END IF
30       CONTINUE
10    CONTINUE
      STOP
      END
```

3.3.1.2 Geração dos Diagramas de Posição (Python)

O script em Python a seguir foi usado para ler os arquivos ‘.dat‘ gerados pelo Fortran e criar os diagramas de dispersão, salvando uma imagem para cada número de passo N analisado.

```
import numpy as np
import matplotlib.pyplot as plt
import glob
import re
```

```
import math

# Encontra todos os arquivos de dados de posição
arquivos_dat = sorted(glob.glob('pos_N*.dat'))

for arquivo in arquivos_dat:
    # Extrai o número N do nome do arquivo
    match = re.search(r'pos_N(\d+).dat', arquivo)
    N = int(match.group(1))

    # Carrega as posições (x, y)
    posicoes = np.loadtxt(arquivo)
    x = posicoes[:, 0]
    y = posicoes[:, 1]

    # Cria a figura e o gráfico de dispersão
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.scatter(x, y, s=5, alpha=0.6, edgecolors='none')

    ax.set_title(f'Diagrama de Posições após N = {N} passos', fontsize=16)
    ax.set_xlabel('Posição X', fontsize=12)
    ax.set_ylabel('Posição Y', fontsize=12)

    ax.set_aspect('equal', adjustable='box')
    ax.grid(True, linestyle='--', alpha=0.6)

    # Salva a imagem
    nome_saida = f'diagrama_N_{N}.png'
    plt.savefig(nome_saida, dpi=150, bbox_inches='tight')
    plt.close(fig)
```

3.3.2 Análise dos Resultados

3.3.2.1 Valores Médios e Deslocamento Quadrático

O movimento em 2D pode ser decomposto em dois movimentos 1D independentes para as direções x e y. Como a probabilidade de se mover no sentido positivo ou negativo de cada eixo é a mesma ($p_x(+) = p_x(-) = 1/4$), o deslocamento médio em cada eixo é nulo.

$$\langle x \rangle_{teorico} = 0 \quad \text{e} \quad \langle y \rangle_{teorico} = 0 \tag{3.5}$$

$$\implies \langle \vec{r} \rangle_{teorico} = (\langle x \rangle, \langle y \rangle) = (0, 0) \tag{3.6}$$

O deslocamento quadrático médio é $\Delta^2 = \langle x^2 \rangle + \langle y^2 \rangle$. Como, para cada passo, um movimento de ± 1 em x ou y ocorre com probabilidade $1/2$, o deslocamento quadrático médio em cada eixo após N passos é $\langle x^2 \rangle = \langle y^2 \rangle = N/2$. Portanto, o valor teórico para Δ^2 é:

$$\Delta^2_{teorico} = \langle r^2 \rangle_{teorico} = \langle x^2 \rangle + \langle y^2 \rangle = \frac{N}{2} + \frac{N}{2} = N \tag{3.7}$$

Isso indica uma relação linear entre o deslocamento quadrático médio e o número de passos, característica de um processo difusivo.

A Tabela 3 mostra os resultados obtidos pela simulação em Fortran.

Tabela 3 – Resultados experimentais para o andarilho aleatório 2D.

N_PASSOS	$\langle x \rangle$	$\langle y \rangle$	$\langle r^2 \rangle$	Δ^2
10	0.057	0.059	9.882	9.875
100	0.213	-0.001	99.982	99.937
1000	1.123	-0.509	1002.102	1000.582
10000	10.647	-6.021	9078.914	8929.303
100000	104.943	-61.863	38709.430	23869.365
1000000	1049.503	-619.757	1508615.875	23060.500

Para N pequeno $(10, 100, 1000)$, os valores experimentais de Δ^2 concordam de forma excelente com o valor teórico N. Os valores de $\langle x \rangle$ e $\langle y \rangle$ flutuam em torno de zero, como esperado.

Contudo, para $N \geq 10.000$, observa-se uma divergência crescente entre Δ^2_{exp} e N. A razão para essa discrepância é a **perda de precisão numérica**. O código utiliza variáveis de ponto flutuante de precisão simples ('REAL'). À medida que N cresce, as posições x e y e seus quadrados x^2, y^2 se tornam números grandes. O cálculo de $\Delta^2 = \langle r^2 \rangle - (\langle x \rangle^2 + \langle y \rangle^2)$ envolve a subtração de dois números grandes e muito próximos, o que leva a uma perda de dígitos significativos (cancelamento catastrófico). Este resultado é importante, pois ilustra uma limitação fundamental das simulações computacionais e a necessidade de se utilizar tipos de dados com precisão adequada (como 'DOUBLE PRECISION') para cálculos cumulativos longos.

Exemplo de um caso difícil de resolver em Fortran 77 pois não havia sequer a possibilidade de usar INTEGER*8 ou REAL*16, poderiamos no máximo utilizar DOUBLE PRECISION.

3.3.2.2 Diagramas de Difusão

As Figuras **??** e **??** mostram a distribuição espacial dos 1000 andarilhos após $N = 100$ e $N = 10^6$ passos. Fica evidente o processo de difusão: a "nuvem"de partículas, inicialmente concentrada na origem, expande-se ao longo do tempo. O raio característico dessa nuvem cresce com $\sqrt{N}$, uma assinatura do movimento browniano.

A última etapa do projeto visa verificar o aumento da entropia em um sistema isolado, um conceito conhecido como a "flecha do tempo". O sistema de andarilhos aleatórios bidimensionais serve como um excelente análogo a um processo de difusão, como o de moléculas de um gás se expandindo em um recipiente. Inicialmente, o sistema está em um estado de baixa entropia (todos os andarilhos na origem). Com o passar do tempo (aumento do número de passos N), os andarilhos se espalham, explorando um número maior de microestados, o que deve levar a um aumento da entropia do sistema.

3.3.3 Metodologia de Cálculo

Para calcular a entropia, o espaço contínuo foi discretizado em uma grade 2D. Um "microestado"i é definido como uma célula dessa grade. A probabilidade P_i de se encontrar o sistema em um determinado microestado é estimada como a fração de andarilhos que ocupam a célula i em um dado instante:

$$P_i = \frac{M_i}{M} \tag{3.8}$$

onde M_i é o número de andarilhos na célula i e M é o número total de andarilhos. A entropia de Shannon do sistema é então calculada pela fórmula:

$$S = -\sum_i P_i \ln(P_i) \tag{3.9}$$

onde a soma é realizada sobre todas as células da grade.

3.3.4 Código-Fonte da Simulação

O código em Fortran a seguir implementa a simulação do andarilho 2D para $M = 2000$ partículas e inclui a rotina para o cálculo da entropia em determinados intervalos de passos. O espaço foi discretizado em uma grade de 300×300 células.

```
PROGRAM MAIN
C     PARAMETROS DA SIMULACAO
      INTEGER NSTEPS, MWALK
      PARAMETER (NSTEPS=1000000, MWALK=2000)
      INTEGER NGRID
      REAL H, XMIN
      PARAMETER (NGRID = 300)
      PARAMETER (H = 10.0)
      PARAMETER (XMIN = -1500.0)
      REAL POS(2*MWALK)
      REAL ENTROP
      INTEGER NCELLS(NGRID, NGRID)
C     COMMON BLOCK E VARIAVEIS DE CONTROLE
      DOUBLE PRECISION RANDS, RVAL
      INTEGER I, J, K, NOUT_IDX, IX, IY, N_I
      INTEGER NOUT(6)
      REAL XPOS, YPOS, PROB_I
      DATA NOUT /10, 100, 1000, 10000, 100000, 1000000/

      OPEN(11, FILE='entropia_vs_N.dat', STATUS='UNKNOWN')
      WRITE(11,*) '# N_PASSOS      ENTROPIA'

      DO 5 I=1, 2*MWALK
         POS(I) = 0.0
 5    CONTINUE

      WRITE(*,'(A10, A15)') 'N_PASSOS', 'ENTROPIA'

      DO 10 J=1, NSTEPS
         DO 20 K=1, 2*MWALK, 2
            RVAL = RANDS()
            IF ( RVAL .LT. 0.25 ) THEN
               POS(K) = POS(K) + 1.0
            ELSE IF ( RVAL .LT. 0.50 ) THEN
               POS(K) = POS(K) - 1.0
```

```
36            ELSE IF ( RVAL .LT. 0.75 ) THEN
37               POS(K+1) = POS(K+1) + 1.0
38            ELSE
39               POS(K+1) = POS(K+1) - 1.0
40            END IF
41      20    CONTINUE
42
43            DO 30 NOUT_IDX=1, 6
44               IF (J .EQ. NOUT(NOUT_IDX)) THEN
45                  DO 50 IX=1, NGRID
46                     DO 50 IY=1, NGRID
47                        NCELLS(IX,IY) = 0
48      50            CONTINUE
49
50                  DO 60 I=1, 2*MWALK, 2
51                     XPOS = POS(I)
52                     YPOS = POS(I+1)
53                     IX = INT((XPOS - XMIN) / H) + 1
54                     IY = INT((YPOS - XMIN) / H) + 1
55                     IF (IX.GE.1 .AND. IX.LE.NGRID .AND.
56         +               IY.GE.1 .AND. IY.LE.NGRID) THEN
57                        NCELLS(IX,IY) = NCELLS(IX,IY) + 1
58                     END IF
59      60            CONTINUE
60
61                  ENTROP = 0.0
62                  DO 70 IX=1, NGRID
63                     DO 70 IY=1, NGRID
64                        N_I = NCELLS(IX,IY)
65                        IF (N_I .GT. 0) THEN
66                           PROB_I = REAL(N_I) / REAL(MWALK)
67                           ENTROP = ENTROP - PROB_I * ALOG(PROB_I)
68                        END IF
69      70            CONTINUE
70
71                  WRITE(*,'(I10, F15.5)') J, ENTROP
72                  WRITE(11,'(I12, F15.6)') J, ENTROP
73               END IF
74      30       CONTINUE
75      10    CONTINUE
76            CLOSE(11)
77            STOP
78            END
```

3.3.5 Análise dos Resultados

A execução do programa gerou os valores de entropia em função do número de passos N, conforme apresentado na Tabela 4.

Tabela 4 – Entropia do sistema de andarilhos em função do número de passos N.

N_PASSOS	**Entropia (S)**
10	1.35934
100	2.31408
1000	4.37552
10000	6.36706
100000	7.41148
1000000	7.09296

O comportamento da entropia é visualizado na Figura?. Observa-se um aumento consistente da entropia até $N = 10^5$ passos, confirmando a expectativa teórica. À medida que os andarilhos se difundem a partir da origem, eles ocupam um volume maior do espaço de estados, distribuindo-se por um número crescente de células da grade. Isso corresponde a um aumento da desordem e, consequentemente, da entropia, ilustrando a seta do tempo.

Surpreendentemente, para $N = 10^6$ passos, o valor da entropia diminui. Essa aparente violação da Segunda Lei da Termodinâmica não é um efeito físico real, mas sim um **artefato computacional** causado pelo tamanho finito da grade de análise. O desvio padrão da distribuição de posições dos andarilhos cresce com $\sigma \approx \sqrt{N}$. Para $N = 10^6$, temos $\sigma = 1000$. A grade utilizada na simulação se estende de -1500 a 1500 em cada eixo. Com um desvio padrão dessa magnitude, uma fração significativa dos andarilhos difunde para posições fora dos limites da grade. O código-fonte ignora explicitamente esses andarilhos no cálculo da entropia. Portanto, o cálculo para $N = 10^6$ é realizado sobre um subconjunto dos andarilhos totais (aqueles que permaneceram dentro da grade), resultando em um valor de entropia artificialmente menor. Este resultado destaca a importância de garantir que os limites de uma simulação sejam suficientemente grandes para conter o sistema físico de interesse em todos os estágios de sua evolução.

Referências

[Alcaraz 2025]ALCARAZ, F. C. *Projeto 2: Sistemas Aleatórios*. 2025. Material de curso, Universidade de São Paulo (USP), Instituto de Física de São Carlos (IFSC). Enunciado do projeto para a disciplina 7600017 - Introdução à Física Computacional.

[Sun Microsystems, Inc. 1995]SUN MICROSYSTEMS, INC. *FORTRAN 77 4.0 Language Reference*. Mountain View, CA, USA, 1995. Part No: 802-2998-10. Atualmente disponível em Oracle Help Center. Disponível em: <https://docs.oracle.com/cd/E19957-01/802-2998/802-2998.pdf>.