

7600017 - Introdução à Física Computacional - Projeto 1

Raphael Vieira Moreira da Serra

14 de Agosto de 2025

Primeira Tarefa

Para calcular os juros nessa tarefa, usei a fórmula:

$$V = \frac{Q \cdot AJM}{N}$$

A sua implementação em Fortran foi da seguinte forma:

```
1  PROGRAM JUROS
2
3  WRITE(*,*) 'Valor a ser recebido'
4  READ(*,*) Q
5  WRITE(*,*) 'Quantidade de parcelas'
6  READ(*,*) C
7  WRITE(*,*) 'Juros'
8  READ(*,*) AJM
9
10 V = (Q/C)*AJM
11
12 WRITE(*,*) V
13
14 END
```

Listing 1: Tarefa 01

O valor retornado para o caso pedido foi de $\approx 183,33$.

Segunda Tarefa

Para realizar essa tarefa, usei as fórmulas de área e volume presentes no formulário:

$$A = 4\pi^2 Rr$$

$$V = 2\pi^2 Rr^2$$

Implementadas no Fortran da seguinte forma:

```

1  PROGRAM TORO
2
3  PI = 3.14159
4
5  WRITE(*,*)"Insira o raio interno"
6  READ(*,*) R1
7  WRITE(*,*)"Insira o raio externo"
8  READ(*,*) R2
9
10 AREA = 4 * PI * PI * R1 * R2
11 VOL = 2 * PI * PI * R2 * R1 * R1
12
13 WRITE(*,*)"A REA : ", AREA
14 WRITE(*,*)"O VOLUME : ", VOL
15
16 END

```

Listing 2: Tarefa 02

Terceira Tarefa

Realizei essa tarefa em 3 passos: *Leitura*, *Vetorização* e *Ordenação*. Na etapa de *Leitura*, usei um *loop* para ler o arquivo linha por linha e contar quantas linhas haviam. Na *Vetorização* usei um *loop* para passar todas as linhas do arquivo para um só vetor afim de facilitar sua manipulação.

Já na etapa de *Ordenação*, usei o algoritmo *Selection Sort* para ordenar o vetor criado na etapa de *Vetorização*. O algoritmo, em PORTUGOL é:

```

para i = 1 até n:
  para j = 1 até n:
    se a[i] > a[j]:
      troque(a[i], a[j])

```

A implementação em Fortran foi da seguinte forma:

```

1  PROGRAM ORDEM
2
3  DIMENSION A(100000)
4
5  !Arquivo de entrada
6  OPEN(UNIT=1, FILE='tarefa-3-entrada-1.in', status='old')
7
8  !Leitura
9  N = 0
10 10 READ(1,*,END=13) temp
11      N = N + 1
12      GOTO 10
13 13 REWIND(1)
14  WRITE(*,*) 'N igual a: ', N
15
16  !Cria o do vetor
17  i = 0
18 19 READ(1,*,END=23) tem

```

```

19      i = i+1
20      A(i) = tem
21      GOTO 19
22 23    REWIND(1)
23
24      !Ordena o
25      DO j = 1, N
26          DO k = (j+1), N
27              IF (A(j) .GT. A(k)) THEN
28                  sub = A(j)
29                  A(j) = A(k)
30                  A(k) = sub
31              END IF
32          END DO
33      END DO
34
35      CLOSE(1)
36
37      !Recebimento de M
38      WRITE(*,*) 'Digite uma quantidade de valores menor que N'
39      READ(*,*)M
40
41      !Display dos resultados
42      WRITE(*,*) A(1:M)
43
44      END

```

Listing 3: Tarefa 03

Quarta Tarefa

Para realizar o item A, usei um *loop* com *while* para calcular termos da série até que a diferença entre o valor da série e a da função nativa fossem menores que $\epsilon = 10^{-5}$. Implementado da seguinte forma:

```

1      PROGRAM APPROX
2
3      I = 1 !Contar os termos da s rie
4      FLOG = 0
5      EPS = 1
6
7      WRITE(*,*)"Valor de x a ser calculado"
8      READ(*,*) X
9
10     DO WHILE (EPS .GT. 0.00001)
11         TERMO = ((1 - X)**I)/I
12         FLOG = FLOG - TERMO
13         EPS = ABS(LOG(X) - FLOG)
14         I = I + 1
15     END DO
16
17     WRITE(*,*)'O valor pelo Fortran nativo', LOG(X)
18     WRITE(*,*)'O valor pela s rie', FLOG
19

```

20 **END**

Listing 4: Tarefa 04 item A

Para realizar o item B, mudei as variáveis para dupla precisão e permiti que o usuário decidisse o erro da série. Fazendo isso, consegui valores de ϵ da ordem de 10^{-17} .

Segue a implementação em Fortran:

```
1  PROGRAM APPROX
2  IMPLICIT REAL*8(A-H, O-Z)
3
4  I = 1 !Contar os termos da s rie
5  FLOG = 0
6  EPS = 1
7
8  WRITE(*,*)"Valor de x a ser calculado"
9  READ(*,*) X
10
11 WRITE(*,*)"Escolha o valor de Epsilon"
12 READ(*,*) EPSILON
13
14 DO WHILE (EPS .GE. EPSILON)
15   TERMO = ((1 - X)**I)/I
16   FLOG = FLOG - TERMO
17   EPS = ABS(DLOG(X) - FLOG)
18   I = I + 1
19 END DO
20
21 WRITE(*,*)"0 valor pelo Fortran nativo", DLOG(X)
22 WRITE(*,*)"0 valor pela s rie", FLOG
23 WRITE(*,*)"0 valor de epsilon", EPS
24
25 END
```

Listing 5: Tarefa 04 item B

Quinta Tarefa

Para calcular a permutação $N+1$ a partir de um arquivo contendo a permutação N , usei um algoritmo que atuava da seguinte forma: Para cada linha do arquivo, "abrir um dos espaços" e colocar $N+1$ nesse espaço. Posteriormente, calculei a paridade usando a seguinte equação:

$$P_{N+1} = P \cdot (-1)^{N+1-k}$$

Onde 'k' é a posição onde $N+1$ foi colocado.

. O código FORTRAN foi o seguinte:

```
1  INTEGER FUNCTION FATORIAL(m)
2  FATORIAL = 1
3  DO i = 1, m
4    FATORIAL = FATORIAL * i
5  END DO
```

```

6      END
7
8      PROGRAM PERMUTACAO
9      !Tipos
10     INTEGER MATRIZ(1040, 1041), NOVA_MATRIZ(2080, 2082)
11     INTEGER FATORIAL
12     INTEGER arquivo, linhas, colunas, p
13     CHARACTER nome*5
14
15     !Parametros
16     arquivo = 3
17
18     !Leitura e Escrita dos valores do arquivo numa matriz
19     WRITE(*,*) 'Diga a quantidade de n meros que voc permutou:
20
21     READ(*,*) N
22     linhas = FATORIAL(N)
23     colunas = N + 1
24
25     WRITE(*,*) 'Diga aonde est o seus arquivos'
26     READ(*,*) nome
27     OPEN(UNIT=arquivo, FILE=nome, STATUS='OLD')
28
29     DO i=1, linhas
30         READ(arquivo,*) (MATRIZ(i,j), j=1, colunas)
31     END DO
32
33     CLOSE(arquivo)
34     !Fim da Leitura e Escrita
35
36     !Uma matriz com (N+1) c pias da matriz original
37     nova_linhas = FATORIAL(N + 1)
38     nova_colunas = N + 2
39     DO i=1, linhas
40         DO j = 0, N
41             DO k = 1, N+1
42                 NOVA_MATRIZ((colunas*i) - j, k) = MATRIZ(i, k)
43             END DO
44         END DO
45     END DO
46     !Fim das c pias
47
48     !Criar permuta es
49     DO i=1, nova_linhas
50         i_sub_matriz = MOD(i, (N+1))
51         p = NOVA_MATRIZ(i, (N+1))
52         IF (i_sub_matriz .EQ. 0) THEN
53             NOVA_MATRIZ(i, (N+1)) = N + 1
54             NOVA_MATRIZ(i, (N+2)) = p
55         ELSE
56             DO j = (N+1), i_sub_matriz, -1
57                 NOVA_MATRIZ(i, j+1) = NOVA_MATRIZ(i, j)
58                 NOVA_MATRIZ(i,j) = (N + 1)
59             END DO
60             NOVA_MATRIZ(i, (N+2)) = p * (-1) ** (N + 1 - i_sub_matriz
)

```

```

61      END IF
62    END DO
63    !Fim da cria o
64
65    !Escrevendo num arquivo
66    OPEN(UNIT=20, FILE='neu.txt', STATUS='NEW')
67
68    DO i =1, nova_linhas
69      DO j = 1, N + 1
70        WRITE(20, '(I2, 1X)', ADVANCE='NO') NOVA_MATRIZ(i, j)
71      END DO
72      WRITE(20, '(I3)') NOVA_MATRIZ(i, nova_colunas)
73    END DO
74
75    CLOSE(20)
76    !Fim da escritura
77
78    !Debugging
79    DO i = 1, linhas
80      WRITE(*,*) 'Vetor n mero ', i
81      DO j = 1, colunas
82        WRITE(*,*) MATRIZ(i,j)
83      END DO
84    END DO
85
86    DO i=1, nova_linhas
87      WRITE(*,*) 'Vetor n mero ', i
88      DO j=1, nova_colunas
89        WRITE(*,*) NOVA_MATRIZ(i,j)
90      END DO
91    END DO
92  END PROGRAM

```

Listing 6: Tarefa 05 item A

Com isso, gerei as permutações $N = 4, 5, 6$, necessárias para o item seguinte. Para calcular o determinante de uma matriz qualquer a partir de sua permutação, usei esses arquivos com permutações como "funções" para calcular o somatório do Método de Leibnitz

A implementação foi assim:

```

1  INTEGER FUNCTION FATORIAL(m)
2      FATORIAL = 1
3      DO i = 1, m
4          FATORIAL = FATORIAL * i
5      END DO
6  END FUNCTION
7
8
9  PROGRAM DETERMINANTE
10     !Tipos
11     INTEGER PERMUTA(1040, 1041), MATRIZ(10, 10)
12     INTEGER FATORIAL
13     INTEGER linhas, colunas, p, m, termo
14     CHARACTER nome*5, nomee*20
15
16     !Parametros

```

```

17 p = 3
18 m = 4
19
20 !Leitura e Escrita dos valores do arquivo de permuta o
21 WRITE(*,*) 'Diga o n mero de linhas na sua matriz'
22 READ(*,*) N
23 linhas = FATORIAL(N)
24 colunas = N + 1
25
26 WRITE(*,*) 'Diga aonde est sua matriz de refer ncia'
27 READ(*,*) nome
28 OPEN(UNIT=p, FILE=nome, STATUS='OLD')
29
30 DO i=1, linhas
31 READ(p,*) (PERMUTA(i,j), j=1, colunas)
32 END DO
33
34 CLOSE(p)
35 !Fim da Leitura e Escrita
36
37 !Leitura e Escrita dos valores da matriz
38 WRITE(*,*) 'Diga aonde est a matriz a ser calculada'
39 READ(*,*) nomee
40 OPEN(UNIT=m, FILE=nomee, STATUS='OLD')
41
42 DO i=1, N
43 READ(m,*) (MATRIZ(i,j), j=1, N)
44 END DO
45
46 CLOSE(m)
47 !Fim da Leitura e Escrita
48
49 !C lculo do determinante
50 soma = 0
51 DO i=1, linhas
52 termo = PERMUTA(i, (N+1))
53 DO j=1, N
54 termo = termo * (MATRIZ(j, PERMUTA(i,j)))
55 END DO
56 WRITE(*,*) termo
57 soma = soma + termo
58 END DO
59 WRITE(*,*) soma
60 !Fim do c lculo do determinante
61
62 !Debugging
63 !DO i=1, linhas
64 !WRITE(*,*) 'Vetor n mero ', i
65 ! DO j=1, colunas
66 ! WRITE(*,*) PERMUTA(i,j)
67 ! END DO
68 !END DO
69
70 !DO i=1,N
71 !WRITE(*,*) 'Linha n mero ', i
72 ! DO j=1, N
73 ! WRITE(*,*) MATRIZ(i,j)

```

```

74      ! END DO
75      !END DO
76      END PROGRAM

```

Listing 7: Tarefa 05 item B

Para testar meu código, calculei as seguintes matrizes e obtive os resultados a seguir:

$$\det \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 5 & 6 & 0 \end{bmatrix} = 1$$

$$\det \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix} = -1$$

$$\det \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 3 & 4 & 5 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} = 120$$

$$\det \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} = 48$$

Implementei o Método de Cramer da seguinte forma:

```

1      INTEGER FUNCTION FATORIAL(m)
2          FATORIAL = 1
3          DO i = 1, m
4              FATORIAL = FATORIAL * i
5          END DO
6      END FUNCTION
7
8      REAL FUNCTION DETERMINANTE(N, MATRIZ, PERMUTA, linhas,
9      colunas)
10         INTEGER N, linhas, colunas
11         INTEGER PERMUTA(1040, 1041), MATRIZ(10,10)
12         INTEGER i,j,termo
13
14         DETERMINANTE = 0
15
16         DO i=1,linhas
17             termo = PERMUTA(i, (N+1))
18             DO j=1,N
19                 termo = termo * (MATRIZ(j, PERMUTA(i,j)))

```



```

20         DETERMINANTE = DETERMINANTE + termo
21     END DO
22 END FUNCTION
23
24
25 PROGRAM CRAMER
26 !Tipos
27 INTEGER PERMUTA(1040, 1041), MATRIZ(10, 10), X(1000,1), Y
(1000,1)
28 INTEGER MATRIZ_A(10,10)
29 INTEGER FATORIAL
30 INTEGER linhas, colunas, p, m, z, termo
31 CHARACTER nome*5, nomee*20, nomeee*80
32
33 !Parametros
34 p = 3
35 m = 4
36 z = 7
37
38
39 !Leitura e Escrita dos valores do arquivo de permuta o
40 WRITE(*,*) 'Diga o n mero de linhas na sua matriz'
41 READ(*,*) N
42 linhas = FATORIAL(N)
43 colunas = N + 1
44
45 WRITE(*,*) 'Diga aonde est sua matriz de refer ncia'
46 READ(*,*) nome
47 OPEN(UNIT=p, FILE=nome, STATUS='OLD')
48
49 DO i=1, linhas
50     READ(p,*) (PERMUTA(i,j), j=1, colunas)
51 END DO
52
53 CLOSE(p)
54 !Fim da Leitura e Escrita
55
56 !Leitura e Escrita dos valores da matriz
57 WRITE(*,*) 'Diga aonde est a matriz A que ser calculada'
58 READ(*,*) nomee
59 OPEN(UNIT=m, FILE=nomee, STATUS='OLD')
60
61 DO i=1, N
62     READ(m,*) (MATRIZ(i,j), j=1, N)
63 END DO
64
65 CLOSE(m)
66 !Fim da Leitura e Escrita
67
68 !Leitura e Escrita dos valores de Y
69 WRITE(*,*) 'Diga aonde est Y'
70 READ(*,*) nomeee
71 OPEN(UNIT=z, FILE=nomeee, STATUS='OLD')
72
73 DO i=1, N
74     READ(z,*) (Y(i,j), j=1, 1)
75 END DO

```

```

76      CLOSE(z)
77      !Fim de Y
78
79      !Loop para escrever X
80      DO j = 1,N !Pra cada coluna
81      MATRIZ_A = MATRIZ
82      DO i=1, N
83      MATRIZ_A(i,j) = Y(i, 1)
84      END DO
85      X(j,1) = DETERMINANTE(N, MATRIZ_A, PERMUTA,linhas,colunas)/
86      DETERM
87      &INANTE(N, MATRIZ, PERMUTA,linhas,colunas)
88      END DO
89
90      DO i=1,N
91      WRITE(*,*)X(i,1)
92      END DO
93
94      END PROGRAM

```

Listing 8: Tarefa 05 item C

Para testar, computei $Ax = y$ com:

$$A = \begin{bmatrix} 2 & 1 & 0 & 3 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 1 & 0 & 2 & 2 \end{bmatrix} \text{ e } y = \begin{bmatrix} 16 \\ 8 \\ 15 \\ 15 \end{bmatrix}$$

Obtendo como resultado:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Sexta Tarefa

Para essa tarefa, usei o Método de Monte Carlo para descobrir a região ocupada pelas n -bolas. Gerei $d \cdot M$ números aleatórios, sendo d a quantidade de dimensões a ser calculada e M a quantidade de vezes para rodar a simulação. A fim de descobrir quantos desses pontos estavam dentro da n -bola, usei $\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots x_d^2} < 1$.

Segue a implementação em Fortran:

```

1      REAL FUNCTION GAMMA(d)
2      IF ((d - INT(d)) .NE. 0) THEN
3      GAMMA = GAMMA_HALF_INT(d)
4      ELSE
5      gamma = GAMMA_INT(d)
6      END IF
7      RETURN
8      END

```

```

9
10 REAL FUNCTION GAMMA_INT(d)
11     REAL d, pi
12     PARAMETER (pi = 3.14159265358979323846)
13
14     n = d - 1
15     GAMMA_INT = 1
16     DO i = 1, n
17         GAMMA_INT = GAMMA_INT * i
18     END DO
19 END
20
21 REAL FUNCTION GAMMA_HALF_INT(d)
22     REAL d, pi
23     PARAMETER (pi = 3.14159265358979323846)
24
25     n = INT(d) / 2
26
27     IF (n .EQ. 0) THEN
28         GAMMA_HALF_INT = SQRT(pi) / 2.0
29     ELSE
30         GAMMA_HALF_INT = 2.0
31         DO i = 1, n
32             GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33         END DO
34         GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
n)
35     ENDIF
36     RETURN
37 END
38
39 PROGRAM NBOLAS
40     REAL values(1000), regiao, formula, pi
41     INTEGER dimensions, pontos_dentro, M
42     PARAMETER (pi = 3.14159265358979323846)
43
44     WRITE(*,*) 'Insira a quantidade de dimens es desejada: '
45     READ(*,*) dimensions
46     WRITE(*,*) 'Insira M, a quantidade de vezes que vamos simular
: '
47     READ(*,*) M
48
49     pontos_dentro = 0
50     DO i = 1, M
51         soma = 0
52         DO j = 1, dimensions
53             values(j) = rand()
54             soma = soma + (values(j) * values(j))
55         END DO
56         IF (SQRT(soma) .LT. 1) THEN
57             pontos_dentro = pontos_dentro + 1
58         ELSE
59             ENDIF
60     END DO
61
62     regiao = (2.0**REAL(dimensions)) * (REAL(pontos_dentro)/REAL(
M))

```

```

63      formula = SQRT(pi**REAL(dimensions)) / GAMMA(1.0 + (REAL(
        dimensio
64      &ns)/2))
65      epsilon = ABS(regiao - formula)
66
67      WRITE(*,*) 'O valor da sua regi o pelo Metodo de Monte Carlo
        :'
68      WRITE(*,*) regiao
69      WRITE(*,*) 'O valor da sua regi o pela f rmula      :'
70      WRITE(*,*) formula
71      WRITE(*,*) 'O erro do seu valor de M      :'
72      WRITE(*,*) epsilon
73      END PROGRAM

```

Listing 9: Tarefa 06

Como uma prova de conceito, calculei o valor de π para $M = 100000000$.

```

Insira a quantidade de dimensões desejada:
2
Insira M, a quantidade de vezes que vamos simular:
100000000
O valor da sua região pelo Metodo de Monte Carlo é:
3.14143872
O valor da sua região pela fórmula é:
3.14159274
O erro do seu valor de M é:
1.54018402E-04

```

Sétima Tarefa

Para o primeiro item, usei a fórmula programada no Sexto Item. Segue a implementação:

```

1      REAL FUNCTION GAMMA(d)
2          IF ((d - INT(d)) .NE. 0) THEN
3              GAMMA = GAMMA_HALF_INT(d)
4          ELSE
5              gamma = GAMMA_INT(d)
6          END IF
7          RETURN
8      END
9
10     REAL FUNCTION GAMMA_INT(d)
11         REAL d, pi
12         PARAMETER (pi = 3.14159265358979323846)
13
14         n = d - 1
15         GAMMA_INT = 1
16         DO i = 1, n
17             GAMMA_INT = GAMMA_INT * i
18         END DO

```

```

19      END
20
21      REAL FUNCTION GAMMA_HALF_INT(d)
22          REAL d, pi
23          PARAMETER (pi = 3.14159265358979323846)
24
25          n = INT(d) / 2
26
27          IF (n .EQ. 0) THEN
28              GAMMA_HALF_INT = SQRT(pi) / 2.0
29          ELSE
30              GAMMA_HALF_INT = 2.0
31              DO i = 1, n
32                  GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33              END DO
34              GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
n)
35          ENDIF
36          RETURN
37      END
38
39      PROGRAM VOLUMES
40      REAL raio, pi
41      INTEGER ARQUIVO, dimensions
42      PARAMETER (pi = 3.14159265358979323846)
43      arquivo = 10
44      OPEN(UNIT=arquivo, FILE='saida.txt', STATUS='NEW')
45
46      WRITE(*,*) 'Insira o raio que voc deseja calcular:'
47      READ(*,*) raio
48      WRITE(*,*) 'Insira a quantidade de dimens es:'
49      READ(*,*) dimensions
50
51      DO i = 0, dimensions
52          volume = (raio ** REAL(i)) * (SQRT(pi**REAL(i))/GAMMA(1.0
53      & + (REAL(i)/2)))
54          WRITE(arquivo, *) volume
55      END DO
56
57      CLOSE(UNIT=arquivo)
58      END

```

Listing 10: Tarefa 07 item a

Para o segundo item, adaptei o código do primeiro item para escrever em 3 arquivos diferentes e depois plotar o resultado usando *XMGRACE*. Segue o gráfico gerado e o código:

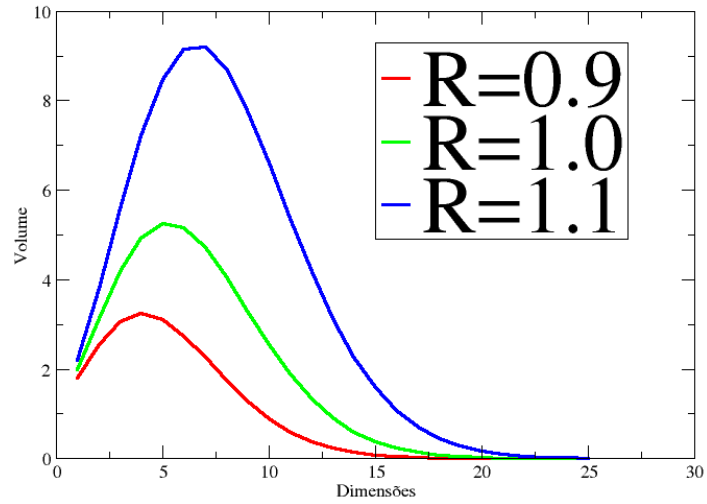


Figura 1: Gráfico para 30 dimensões de n-bolas de raios 0.9, 1.0 e 1.1.

```

1  REAL FUNCTION GAMMA(d)
2      IF ((d - INT(d)) .NE. 0) THEN
3          GAMMA = GAMMA_HALF_INT(d)
4      ELSE
5          gamma = GAMMA_INT(d)
6      END IF
7      RETURN
8  END
9
10 REAL FUNCTION GAMMA_INT(d)
11     REAL d, pi
12     PARAMETER (pi = 3.14159265358979323846)
13
14     n = d - 1
15     GAMMA_INT = 1
16     DO i = 1, n
17         GAMMA_INT = GAMMA_INT * i
18     END DO
19 END
20
21 REAL FUNCTION GAMMA_HALF_INT(d)
22     REAL d, pi
23     PARAMETER (pi = 3.14159265358979323846)
24
25     n = INT(d) / 2
26
27     IF (n .EQ. 0) THEN
28         GAMMA_HALF_INT = SQRT(pi) / 2.0

```

```

29         ELSE
30             GAMMA_HALF_INT = 2.0
31             DO i = 1, n
32                 GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33             END DO
34             GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
n)
35         ENDIF
36         RETURN
37     END
38
39     REAL FUNCTION VOLUME(raio, d)
40         REAL raio
41         INTEGER d
42         PARAMETER (pi = 3.14159265358979323846)
43
44         VOLUME = (raio ** REAL(d)) * (SQRT(pi**REAL(d))/GAMMA
(1.0
& + (REAL(d)/2)))
45
46     END
47
48     PROGRAM VOLUMES
49     REAL raios(3), pi
50     INTEGER arquivo_1, arquivo_2, arquivo_3, dimensions
51     PARAMETER (pi = 3.14159265358979323846)
52     raios(1) = 0.9
53     raios(2) = 1.0
54     raios(3) = 1.1
55     arquivo_1 = 1
56     arquivo_2 = 2
57     arquivo_3 = 3
58     OPEN(UNIT=arquivo_1, FILE='dados1.dat', STATUS='UNKNOWN')
59     OPEN(UNIT=arquivo_2, FILE='dados2.dat', STATUS='UNKNOWN')
60     OPEN(UNIT=arquivo_3, FILE='dados3.dat', STATUS='UNKNOWN')
61
62
63     DO i = 1,3
64         DO j = 1, 25
65             x = REAL(j)
66             WRITE(i, *) x, VOLUME(raios(i), j)
67         END DO
68     END DO
69
70     CLOSE(UNIT=1)
71     CLOSE(UNIT=2)
72     CLOSE(UNIT=3)
73     END

```

Listing 11: Tarefa 07 item b

Oitava Tarefa

Para calcular, reaproveitei o código da Sétima Questão e o re-adapte para calcular a razão entre a n-bola e um n-cubo.

```

1  REAL FUNCTION GAMMA(d)
2      IF ((d - INT(d)) .NE. 0) THEN
3          GAMMA = GAMMA_HALF_INT(d)
4      ELSE
5          gamma = GAMMA_INT(d)
6      END IF
7      RETURN
8  END
9
10 REAL FUNCTION GAMMA_INT(d)
11     REAL d, pi
12     PARAMETER (pi = 3.14159265358979323846)
13
14     n = d - 1
15     GAMMA_INT = 1
16     DO i = 1, n
17         GAMMA_INT = GAMMA_INT * i
18     END DO
19 END
20
21 REAL FUNCTION GAMMA_HALF_INT(d)
22     REAL d, pi
23     PARAMETER (pi = 3.14159265358979323846)
24
25     n = INT(d) / 2
26
27     IF (n .EQ. 0) THEN
28         GAMMA_HALF_INT = SQRT(pi) / 2.0
29     ELSE
30         GAMMA_HALF_INT = 2.0
31         DO i = 1, n
32             GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33         END DO
34         GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
n)
35     ENDIF
36     RETURN
37 END
38
39 PROGRAM VOLUMES
40     REAL raio, pi
41     INTEGER ARQUIVO, dimensions
42     PARAMETER (pi = 3.14159265358979323846)
43
44     WRITE(*,*) 'Insira o raio que voc  deseja calcular:'
45     READ(*,*) raio
46     WRITE(*,*) 'Insira a quantidade de dimens es:'
47     READ(*,*) dimensions
48
49     DO i = 1, dimensions
50         volume = (raio ** REAL(i)) * (SQRT(pi**REAL(i))/GAMMA(1.0
& + (REAL(i)/2)))
51         WRITE(*,*) (2.0**REAL(i))/volume
52     END DO
53 END
54

```

Listing 12: Tarefa 08

Para aproximar o limite $d \rightarrow \infty$, simulei para 75 dimensões. Segue o output:

Insira o raio que você deseja calcular:

1

Insira a quantidade de dimensões:

75

1.00000000
1.27323949
1.90985906
3.24227762
6.07927036
12.3845882
27.0912876
63.0741882
155.221634
401.542755
1086.98877
3067.56055
8995.97949
27340.1738
85905.2812
278484.719
929712.938
3191199.75
11245600.0
40631612.0
150342592.
569071616.
2.20135424E+09
8.69477274E+09
3.50356439E+10
1.43916876E+11
6.02218299E+11
2.56536897E+12
1.11181373E+13
4.89949359E+13
2.19418806E+14
9.98116711E+14
4.60964935E+15
2.16043040E+16
1.02710793E+17
4.95134172E+17
2.41934559E+18
1.19780654E+19
6.00679255E+19
3.05018881E+20

1.56785756E+21
8.15560403E+21
4.29195476E+22
2.28448838E+23
1.22955461E+24
6.69001150E+24
3.67896630E+25
2.04431688E+26
1.14763024E+27
6.50726281E+27
3.72608033E+28
2.15417906E+29
1.25721115E+30
7.40552173E+30
4.40200984E+31
2.64012107E+32
1.59737175E+33
9.74836681E+33
5.99981901E+34
3.72360202E+35
2.32995765E+36
1.46972159E+37
9.34477238E+37
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity
Infinity