Universidade de São Paulo
Instituto de Física de São Carlos

Projeto 1: Introdução à física computacional

Julia Martins Simão - 13694997

São Carlos
2025

Sumário

1 Tarefa 1

Nessa primeira tarefa, foi feito um programa simples que recebe um valor Q, uma quantidade N de parcelas mensais e o valor da taxa AJM de juros a ser acrescida às parcelas. O programa foi testado para $Q = 2000, N = 12$ e $AJM = 1.1$ (ou seja, um acréscimo de 10% ao valor que seria pago sem juros).

Saída:

```
digite Q, N e AJM:
2000 12 1.1
cada parcela sera:   183.333344
```

Código:

```
      write(*,*) 'digite Q, N e AJM:'
      read(*,*) Q, N, AJM
c valor da parcela sem juros
      SJ = Q/N
c valor da parcela com a taxa de juros
      V = AJM*SJ

      write(*,*) 'cada parcela sera:', V

      end
```

2 Tarefa 2

Na tarefa 2, foi criado um programa que lê os raios interno (r_1) e externo (r_2) de um tórus e calcula a sua área e volume pelas fórmulas:

$$A = \int_0^{2\pi} r_2\, d\phi \int_0^{2\pi} r_1\, d\theta = 4\pi r_1 r_2$$

$$V = \int_0^{2\pi} r_2\, d\phi \int_0^{2\pi} d\theta \int_0^{r_1} r\, dr = 2r_2(r_1\pi)^2$$

Com o objetivo de testar o código, deu-se como entrada os valores $r_1 = 5.1$ e $r_2 = 6.2$.

Exemplo

```
digite r1 e r2:
5.1 6.2
a área é:   1248.30762
o volume é:   3183.18408
```

Código:

```
      write(*,*) 'digite r1 e r2:'
      read(*,*) r1, r2

      pi = acos(-1e0)
c formulas da area e do volume de um torus
      a = 4 * pi * pi * r1 * r2
      v = 2 * pi * pi * r1 * r1 * r2

      write(*,*) 'a área é:', a
      write(*,*) 'o volume é:', v

      end
```

3 Tarefa 3

Nessa tarefa, foi criado um programa com o intuito de ler um arquivo de entrada, que foi disponibilizado pelo professor, o qual continha um número do tipo REAL*8 em cada linha. A quantidade de linhas era, inicialmente, desconhecida. O fato do número de linhas não ser conhecido unido à ausência de alocação dinâmica de memória na linguagem Fortran77 fez com que houvesse a necessidade de indicar um valor máximo de valores (linhas do arquivo) no vetor usado para armazenar os termos linhas. Após a leitura, contou-se quantas linhas N existiam nesse arquivo de entrada, sendo esse valor impresso na tela. Finalmente, pediu-se que o usuário inserisse um valor $M \leq N$ e foi implementado o algoritmo Selection Sort para ordenar os M primeiros números dentre os dados no arquivo de entrada. A ordenação foi guardada em um arquivo de saída, juntamente com o valor M. Para testar o funcionamento do código, foi dado o valor $M = 10$ como entrada.

3.1 Selection Sort

O algoritmo de ordenação Selection Sort lê uma lista de elementos e seleciona o menor (ou maior) entre eles, colocando-o na primeira posição de uma sublista, ou seja, uma lista que é iniciada sem nenhum termo. Esse processo é repetido até que todos os elementos estejam ordenados de forma crescente (ou decrescente).

Exemplo

```
N=         1003
digite M:
10
```

Arquivo de saída (*t3-saida.out*):

```
  7.6293945312500000E-006
 0.13153767585754395
 0.21895909309387207
 0.53276705741882324
 0.45865011215209961
 0.67886447906494141
  4.7044515609741211E-002
 0.75560522079467773
 0.67929625511169434
 0.93469285964965820
o valor de M é:          10
```

Código:

```
        parameter (max = 2000) !considerei que o arq tem menos de 2000 linhas
        double precision a(1:max) !vetor que armazena os termos do arquivo
        double precision aux

        n=0 !contador de linhas do arquivo

        open(unit=1, file="tarefa-3-entrada-1.in")
        open(unit=2, file="t3-saida.out")

        do j=1,max
                read(1, *, end=13) a(j) !le cada linha do arq e guarda em a
                n=n+1 !conta os termos do arq
        end do

13      write(*,*) "N=", n
        write(*,*) "digite M:"
        read(*,*) m

        do i=1,m-1 !implementacao de selection sort
                k=i
                do j=i+1,m
                        if(a(j) .lt. a(k)) then
                                k=j
                        end
                if(k .ne. i) then
                        aux=a(i)
                        a(i)=a(k)
                        a(k)=aux
                        end if
                end do
        end do

        do l=1,m !escreve os m termos ordenados no arq de saida
                write(2,*) a(l)
        end do

        write(2,*) "o valor de M é:", m
        close(1)
        close(2)

        end
```

4 Tarefa 4

Aqui, os itens A e B foram feitos em um só código. O objetivo da tarefa é escrever um programa que calcule a seguinte série:

$$ln(x) = -\sum_{n=1}^{\infty} \frac{(1-x)^n}{n}$$

e comparar com a função *log(x)* intrínseca do Fortran77. O programa se divide em duas partes:

1. Cálculo da série com precisão simples ($\epsilon = 10^{-5}$).
2. Cálculo da série com precisão dupla ($\epsilon = 10^{-15}$).

Na parte 2, foram feitos testes variando o valor de ϵ até que sua precisão fosse igual à da função *dlog(x)*, função intrínseca da linguagem em precisão dupla, que atingiu até $\epsilon = 10^{-17}$.

O código foi testado com as entradas $x = 0.5$, $x = 0.0$ e $x = 2.0$, sendo os dois últimos casos limite.

Os casos limite foram escolhidos de modo a respeitar a matemática contida por trás da função e da série *ln(x)*: a função *ln(x)* não está definida para $x \leq 0$ e sua série diverge para $x \geq 2$.

Alguns outros testes de solidez do código foram feitos para garantir seu funcionamento como, por exemplo, a checagem de que a série *ln(x)* retornasse reultados negativos para valores de x entre 0 e 1.

Exemplos:

```
digite x
2.0
a serie diverge

digite x
0.0
fora do dominio

digite x
0.5
O valor da serie eh: -0.693139076
serie-log(x)=   8.10623169E-06
valor em dupla prec (dserie): -0.69314718055994506
dserie-dlog(x)=   2.2204460492503131E-016
```

Código:

```
        double precision dx
        double precision dserie
        double precision dprec
        double precision dtermo
        double precision ddif

        write(*,*) "digite x"
        read(*,*) x

        prec=1e-5
        dprec=1e-16
        serie=0.0
        dserie=0.0
        n=1
        dn=1

        termo=2*prec
        dtermo=2*dprec

         if(x .ge. 2.0) then
                write(*,*) "a serie diverge"
         else if(x .le. 0.0) then
                write(*,*) "fora do dominio"
        go to 10
        else
c loop que calcula o ln em precisao simples
                do while(abs(termo) .ge. prec)
                        termo=((1.0-x)**n)/n
                        serie=serie-termo
                        n=n+1
                end do
c meu x agora tem dupla precisao
                dx=x
c loop que calcula o ln em dupla precisao
                do while(abs(dtermo) .ge. dprec)
                        dtermo=(1.0-dx)**dn/dn
                        dserie=dserie-dtermo
                        dn=dn+1
                end do

                dif=abs(serie-log(x))
                ddif=abs(dserie-dlog(dx))

                write(*,*) "O valor da serie eh:", serie5
                write(*,*) "serie-log(x)=", dif
                write(*,*) "valor em dupla prec (dserie):", dserie
                write(*,*) "dserie-dlog(x)=", ddif
        end if

10      end
```

5 Tarefa 5

5.1 Paridade de permutações

A paridade de uma permutação está diretamente ligada à quantidade de vezes em que foram feitas mudanças de posição entre seus elementos. Como exemplo, considere um conjunto $\lambda = \{1, 2, 3\}$

- $\lambda_1 = \{2, 3, 1\}$ é permutação par de λ, pois houve duas mudanças de posição, e 2 é um número par: $1, 2, 3 \rightarrow 2, 1, 3 \rightarrow 2, 3, 1$
- $\lambda_2 = \{3, 2, 1\}$ é permutação ímpar de λ, pois houve três mudanças de posição entre seus elementos, e três é um número ímpar: $1, 2, 3 \rightarrow 2, 1, 3 \rightarrow 2, 3, 1 \rightarrow 3, 2, 1$

Para caracterizar a paridade de uma permutação, é necessário conhecer o seu sinal ($sgn(\lambda)$), sendo

$$sgn(\lambda) = (-1)^n$$

com n definido como o número de mudanças de posições entre elementos do conjunto. Ou seja, para n par, $sgn(\lambda) = 1$; para n ímpar, $sgn(\lambda) = -1$.

Tomando, novamente $\lambda = \{1, 2, 3\}$, tem-se que o número de permutações possível para esse conjuto é de 3!. Por conta disso, o código deve conter um loop de cálculo do fatorial do número de elementos do conjunto que, aqui, é uma matriz. Deve ser feito, também, um loop que lê as linhas dessa matriz e faz a troca de um elemento a cada iteração, produzindo uma alteração no equivalente à $sgn(\lambda)$ no programa. Além disso, é necessário que haja um loop que "guarde"cada uma das permutações em um arquivo de saída, que pode ser entendido como uma matriz de $n!$ linhas e $n + 1$ colunas, descontando a coluna que guarda a paridade de cada uma dessas permutações.

6 Tarefa 6

6.1 Métodos de Monte Carlo

Métodos de Monte Carlo são utilizados para gerar resultados determinísticos a partir de entradas aleatórias de distribuições de probabilidade em certo domínio.

No caso da presente tarefa, Monte Carlo é implementado com o objetivo de calcular o volume de uma esfera de d-dimensões. Para isso, são gerados M números aleatórios entre 0 e 1 (chamados de r no código) a partir da função *rand()*, com cada um desses números definindo um ponto no espaço de uma região qualquer. A distância D (chamada de *aux* no código) entre esses pontos é, então,

$$D = \sqrt{r_1^2 + r_2^2 + ... + r_d^2}$$

Considera-se agora que a região em análise é um cubo de lado $2R$, com $R = 1$, e que contenha inteiramente a esfera de interesse, também de raio unitário. Então, para que os pontos gerados estejam no interior dessa esfera, é necessário que a distância entre eles seja menor ou igual que $R = 1$.

Seguindo essa lógica, foi feito um loop que checa se os pontos pseudo-aleatórios gerados estão contidos no interior da esfera: se um ponto estiver, um contador n é incrementado.

Note que, como qualquer fenômeno probabilístico, o tamanho da amostragem é essencial. Ou seja, quanto mais pontos são gerados, mais precisa será a aproximação do volume V_e da esfera, já que mais amostras estarão contidas no seu interior. Então, para M suficientemente grande:

$$V_e \rightarrow V_c \cdot \frac{n}{M}$$

Sendo $V_c = (2R)^3$ o volume do cubo. Apesar do uso do cubo usual na explicação, esse "volume"é geral: para duas dimensões, por exemplo, tem-se um quadrado, com $V_c = A_q = (2R)^2$. Assim, generaliza-se esse raciocínio para d-dimensões, de modo que

$$V_e \to (2R)^d \cdot \frac{n}{M}$$

O volume aproximado obtido pelo processo descrito acima foi comparado com o calculado a partir da fórmula 1 para o volume V_d de uma esfera d-dimensional para valores $d = 2, 3$ e 4.

$$V_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(1 + \frac{d}{2})} \cdot R^d \tag{1}$$

sendo $\Gamma(x)$ a função Gama (vide seção 7.1). Note que, para $d \to \infty$, $V_d \to 0$.

Visto que o presente programa tem como objetivo calcular V_d para d's pré-definidos, $\Gamma(x)$ foi calculado para $x = 1 + \frac{2}{2}, 1 + \frac{3}{2}$ e $1 + \frac{4}{2}$ a partir de manipulações dos casos base fornecidos no enunciado:

- $d = 2$
$$\Gamma(1 + \frac{2}{2}) = \Gamma(1 + 1) = 1 \cdot \Gamma(1) = 1$$

- $d = 3$
$$\Gamma(1 + \frac{3}{2}) = \Gamma(1 + (1 + \frac{1}{2})) = (1 + \frac{1}{2}) \cdot \Gamma(1 + \frac{1}{2}) = \frac{3}{2} \cdot \frac{1}{2} \cdot \Gamma(\frac{1}{2}) = \frac{3\sqrt{\pi}}{4}$$

- $d = 4$
$$\Gamma(1 + \frac{4}{2}) = \Gamma(1 + (1 + 1)) = (1 + 1) \cdot \Gamma(1 + 1) = 2 \cdot \Gamma(1) = 2$$

O código foi testado para os valores $M = 100$ e $M = 10000$, com intenção de verificar que quanto maior for o valor de M, mais precisa será a aproximação do volume da esfera para $d = 2, 3$ e 4.

Para $M = 100$, o módulo da diferença entre o volume aproximado a partir de Monte Carlo e o volume calculado diretamente para cada d foi

- $d = 2$: $0,30159283 \to$ note que, pela fórmula, o volume é igual à π!
- $d = 3$: $0,05120945$
- $d = 4$: $0,50519753$

Para $M = 100$, o módulo da diferença entre o volume aproximado a partir de Monte Carlo e o volume calculado diretamente para cada d foi

- $d = 2$: $0,00800729$
- $d = 3$: $0,01519012$
- $d = 4$: $0,04680252$

  Exemplos:

```
digite m:
100
MONTE CARLO:
para d=           2
volume=   2.83999991
pontos dentro do raio unitario=          71


para d=           3
volume=   4.23999977
pontos dentro do raio unitario=          53


para d=           4
volume=   5.44000006
pontos dentro do raio unitario=          34

FORMULA:
para d=           2
funcao gamma=   1.00000000
volume pela formula=   3.14159274

para d=           3
funcao gamma=   1.32934046
volume pela formula=   4.18879032

para d=           4
funcao gamma=   2.00000000
volume pela formula=   4.93480253
digite m:
10000
MONTE CARLO:
para d=           2
volume=   3.14960003
pontos dentro do raio unitario=        7874


para d=           3
volume=   4.17360020
pontos dentro do raio unitario=        5217


para d=           4
volume=   4.88800001
pontos dentro do raio unitario=        3055

FORMULA:
para d=           2
funcao gamma=   1.00000000
volume pela formula=   3.14159274

para d=           3
funcao gamma=   1.32934046
volume pela formula=   4.18879032

para d=           4
funcao gamma=   2.00000000
volume pela formula=   4.93480253
```

Código:

```
                write(*,*) "digite m:"
                read(*,*) m
        c m eh a quantidade total de ptos a serem considerados
                write(*,*) "MONTE CARLO:"
                v2=montecarlo(m,2)
                v3=montecarlo(m,3)
                v4=montecarlo(m,4)

                write(*,*) "FORMULA:"
                v2_gamma=vol_gamma(2)
                v3_gamma=vol_gamma(3)
                v4_gamma=vol_gamma(4)

                end

                c casos da funcao gamma para d=2,3 e 4
                function vol_gamma(id)
                pi=acos(-1e0)
                gamma=0.0

                if(id .eq. 2) then
                        gamma=1.0
                end if
        if(id .eq. 3) then
                gamma=0.75*sqrt(pi)
        end if

        if(id .eq. 4) then
                gamma=2.0
        end if
c formula do volume dada no enunciado
        vgamma=pi**(id/2.0)/gamma

        write(*,*) "para d=", id
        write(*,*) "funcao gamma=", gamma
        write(*,*) "volume pela formula=", vgamma
        write(*,*) "                    "

        end

c funcao que calcula o metodo monte carlo
        function montecarlo(m,id)
        n=0

        do i=1,m
                r=0.0
                aux=0.0
                do j=1,id
c gero numeros aletorios que compoem um raio para a esfera que eu quero
                        r=rand()
                        aux=aux+r**2
                end do
c o meu raio eh menor ou igual que 1? se sim, adiciono 1 no meu contador
c de ptos no interior de um raio 1
                if(sqrt(aux) .le. 1.0) then
                        n=n+1
```

```
                end if
            end do

c o volume contido em um raio unitario vai ser a razao entre os
c pts interno/pts totais*a area do meu cubo em d dimensoes
            v=2.0**id*(float(n)/float(m))
            write(*,*) "para d=", id
            write(*,*) "volume=", v
            write(*,*) "pontos dentro do raio unitario=", n
            write(*,*) "                  "
            write(*,*) "                  "

            end
```

7 Tarefa 7

Nessa tarefa pede-se, novamente, que seja calculado o volume V_d de uma esfera d-dimensional mas, agora, d não é pré-definido: ou seja, o código deve ser válido para qualquer dimensão. Para isso, foi preciso implementar um loop que calculasse a função Gama para uma dimensão dada pelo usuário. Os resultados para todos os d's até o d fornecido na entrada e seus respectivos V_d foram guardados em um arquivo de saída.

7.1 Função Gama

A função Gama é a extensão da função Fatorial para os conjuntos dos números reais e complexos e, por conta disso, é frequentemente usada na descrição de problemas que envolvem distribuições probabilísticas.

Para um número n inteiro positivo, a função é definida como

$$\Gamma(n) = (n-1)! \tag{2}$$

No caso dessa tarefa, n é a dimensão da d-esfera (definida como id no código), sendo, então um valor inteiro. Todavia, o argumento da função Gama na equação 1 é $1 + \frac{d}{2} = x$, podendo, então, ser um valor não inteiro ou, mais especificamente, um número real positivo. A função Gama para um número real positivo x é definida como

$$\Gamma(x) = g(x-1)$$

com $g(x)$ sendo

$$g(x) = \int_0^\infty e^{-y} y^x dy = x!$$

A partir disso, foi escrito um programa que calculasse a função Gamma para um valor x. Como x está diretamente ligado à dimensão d, a função criada em Fortran77 tem como argumento a dimensão. Além disso, a fórmula 1 depende de R que, como agora o raio é fornecido pelo usuário, ele também é um argumento da função, diferentemente do caso anterior, em que R poderia ser omitido pois era igual à 1. Finalmente, foram criados if para cada um dos casos base de Gama ($x = \frac{1}{2}$ e $x = 1$), de modo que qualquer valor de x, inteiro ou não, pudesse ser adaptado para que Gama fosse uma função fatorial simples, como na equação 2, por recursão.

7.2 A

Na pergunta A), pede-se que o volume da esfera seja calculado para a dimensões $1, 2, ..., d$, sendo d uma dimensão fornecida pelo usuário. Os dados são armazenados em diferentes arquivos de saída (pois eles seriam usados no item b). O código foi testado para $R = 1.0, 2.0$ e $d = 4, 20$, de modo que, com $R = 1$ e $d = 4$, o resultado final pudesse ser comparado com a tarefa 6.

Exemplos:

```
digite a dimensao d:
4
digite o raio:
1.0
```

Arquivo de saída (*dim-esferas.dat*):

```
0   1.00000000
1   2.00000000
2   3.14159274
3   4.18879032
4   4.93480253
```

O valor para $d = 4$ e $R = 1.0$ condiz com o obtido na tarefa 6.

```
digite a dimensao d:
20
digite o raio:
2.0
```

Arquivo de saída (*dim-a.dat*):

```
0   1.00000000
1   4.00000000
2   12.5663710
3   33.5103226
4   78.9568405
5   168.441254
6   330.733643
7   604.770081
8   1039.03040
9   1688.83667
10  2611.36816
11  3858.64502
12  5469.23730
13  7459.87207
14  9818.35156
15  12499.1357
16  15422.6318
17  18478.6816
18  21534.0566
19  24443.1504
20  27060.4941
```

O primeiro e o último valor foram comparados com os resultados obtidos "à mão"pela fórmula.

Código:

```
write(*,*) "digite a dimensao d:"
read(*,*) id

write(*,*) "digite o raio:"
read(*,*) r
```

```
c loop para escrever as dimensoes e seus respectivos volumes num arq
c aqui, foram abertos 4 arq diferentes para guardar separadamente
c os casos em que r=0.9,1.0 e 1.1
c pois esses valores serao usados no plot de um grafico
        if(r .eq. 0.9) then
                open(1, file='dim-esferas09.dat')
                do i=0,id
                        write(1,*) i, fgamma(i,r)
                end do
        else if(r .eq. 1.0) then
                open(2, file='dim-esferas.dat')
                do i=0,id
                        write(2,*) i, fgamma(i,r)
                end do
        else if(r .eq. 1.1) then
                open(3, file='dim-esferas11.dat')
                do i=0,id
                        write(3,*) i, fgamma(i,r)
                end do
        else
                open(4, file='dim-a.dat')
                do i=0,id
                        write(4,*) i, fgamma(i,r)
                end do
        end if

        close(1)
        close(2)
        close(3)
        close(4)

        end
c calculo gamma(n)=(n-1)!
                else if(x .gt. 1.0) then
                        xgamma=xgamma*(x-1)
                end if

                x=x-1
                xgamma=xgamma*aux
        go to 10
        end if
c formula do volume da esfera
        v=(pi**(id/2.0)/xgamma)*r**id
        fgamma=v
        end

                function fgamma(id,r)
        pi=acos(-1.0e0)
        xgamma=1.0
        aux=1.0
```

```
c argumento da minha funcao gama
        x=id/2.0+1.0

10      if(x .gt. 0.0) then
c casos base gamma(1)=1 e gamma(1/2)=sqrt(pi)
                if(x .eq. 1.0) then
                        aux=1.0
                else if(x .eq. 0.5) then
                        aux=sqrt(pi)
```

7.3 B

O código é o mesmo do exercício anterior mas, nesse item, testou-se o código para $d = 25$ e $R = 0.9, 1.0$ e 1.1. Com os arquivos dos resultados, foi plotado um gráfico da relação dimensão X volume para cada R pelo graficador XMGRACE, mostrado na figura 1.

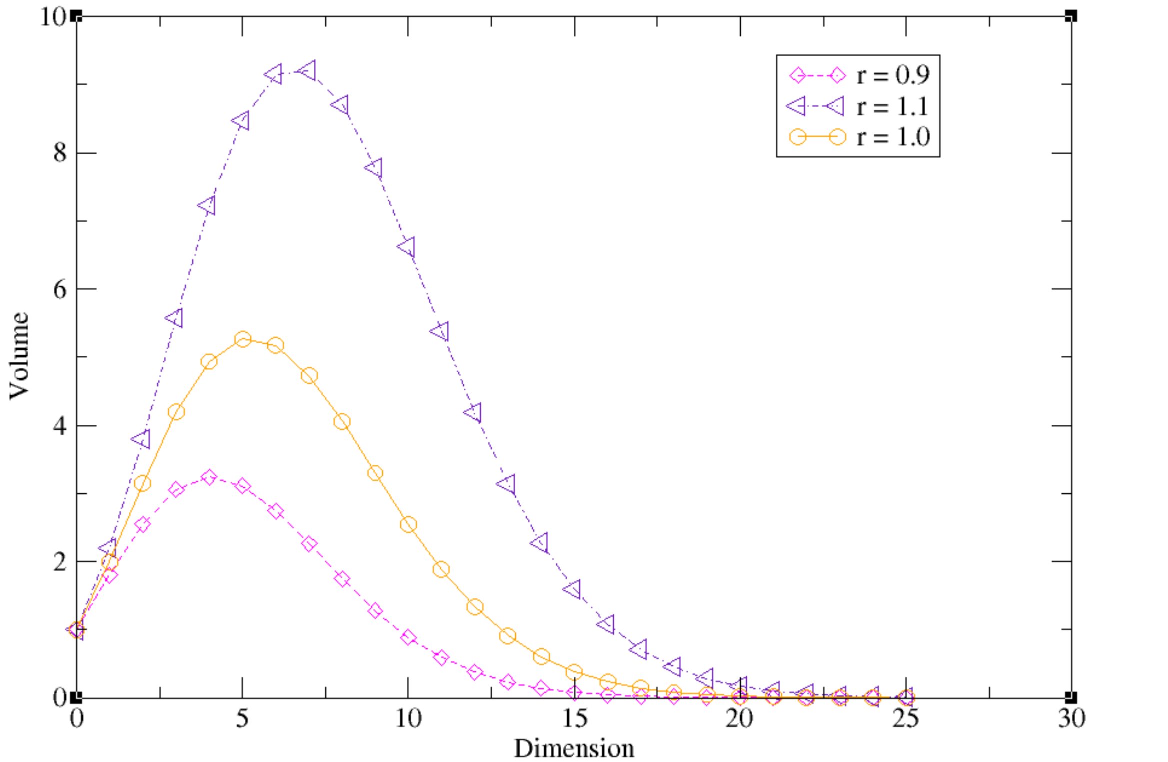

Figura 1: Gráfico do volume da esfera em dimensões de 0 a 25 para três valores de raio r.

8 Tarefa 8

Nessa tarefa, o código do exercício anterior foi adaptado de modo que o raio da esfera fosse mantido fixo $R = r = 1$ mas d continuasse a ser fornecido pelo usuário. Além disso, pede-se que o volume da esfera de $r = 1$ seja comparado com o volume de um cubo de mesmo raio e dimensão, cujo volume será

$$V_c = (2R)^d$$

Para esse novo cálculo, foi criada uma função *vcubo* com argumentos iguais ao da função *fgamma* (que calcula o valor da função Gama e o volume V_e da esfera). Finalmente, os resultados para V_c e V_e nas dimensões $d = 0, 1, ..., d$ foram guardados em arquivos, assim como a razão $\frac{V_e}{V_c}$.

Um gráfico da relação entre dimensões $d = 0, ..., 100$ e volume para uma esfera de raio unitário é mostrado na figura 2, assim como para a relação entre o volume do cubo em cada dimensão é representado em 3.

A razão $\frac{V_e}{V_c}$ para cada d foi também plotada, sendo mostrada na figura 4.

Nota-se que, para $d = 0$, $\frac{V_e}{V_c} = 1$ mas, para d suficientemente grande, o volume da esfera tende à 0, enquanto que o volume do cubo tende ao infinito. Além disso, é fácil perceber que o comportamento da razão entre os volumes seria o mesmo para qualquer raio (desde que seja o mesmo para a esfera e o cubo) visto que R cancelaria-se:

$$\frac{V_e}{V_c} = \frac{\frac{\pi^{\frac{d}{2}} \cdot R^d}{\Gamma(1+\frac{d}{2})}}{2^d \cdot R^d} = \frac{\frac{\pi^{\frac{d}{2}}}{\Gamma(1+\frac{d}{2})}}{2^d}$$

Assim, para $d \to \infty$:

$$\lim_{d\to\infty} \frac{V_e}{V_c} = \lim_{d\to\infty} \frac{\pi^{\frac{d}{2}}}{\Gamma(1+\frac{d}{2}) \cdot 2^d} = \lim_{d\to\infty} \frac{(\frac{\sqrt{\pi}}{2})^d}{\Gamma(1+\frac{d}{2})} \to 0$$

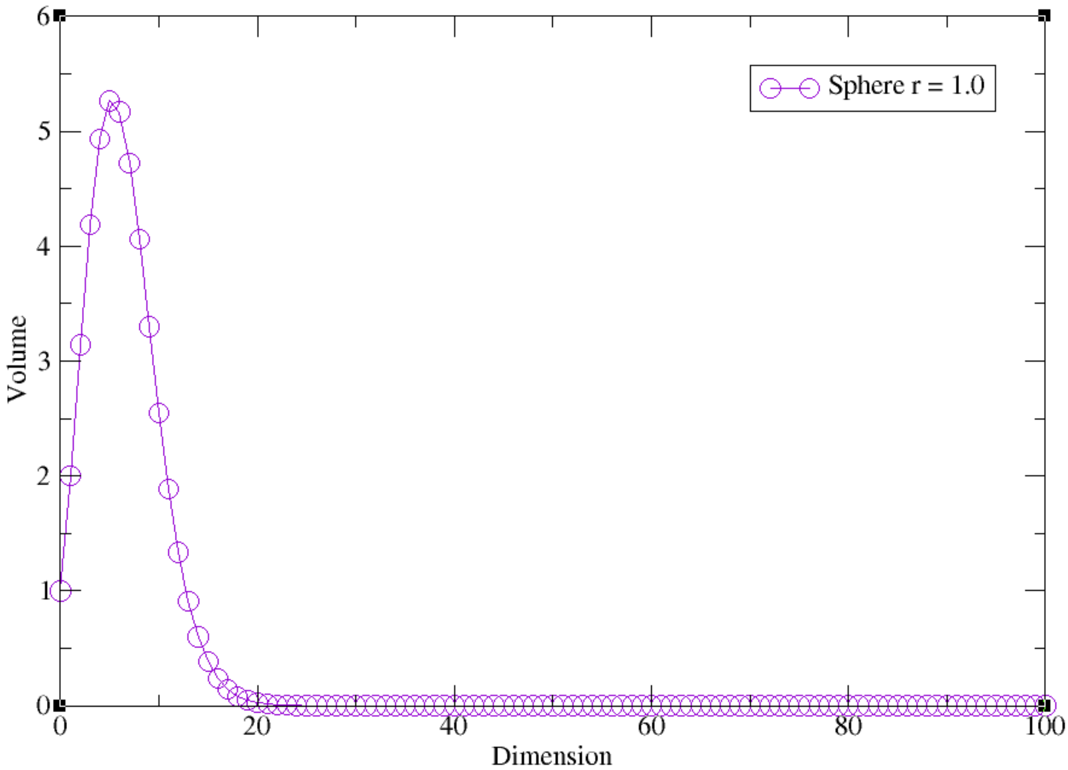

Figura 2: Gráfico do volume da esfera de raio unitário em dimensões de 0 a 100 (arquivo de saída: *dim-esferas-new.dat*).

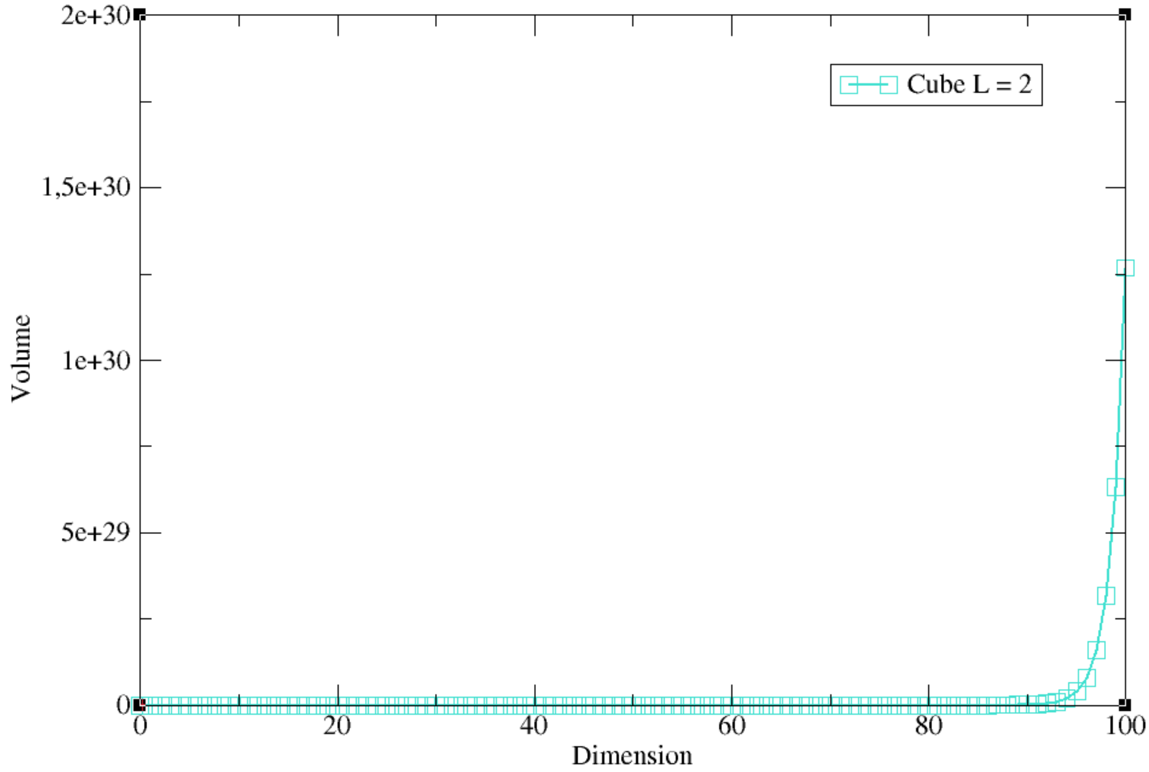

Figura 3: Gráfico do volume da cubo de raio unitário ($L = 2R$) em dimensões de 0 a 100 (arquivo de saída: *dim-cubo.dat*).

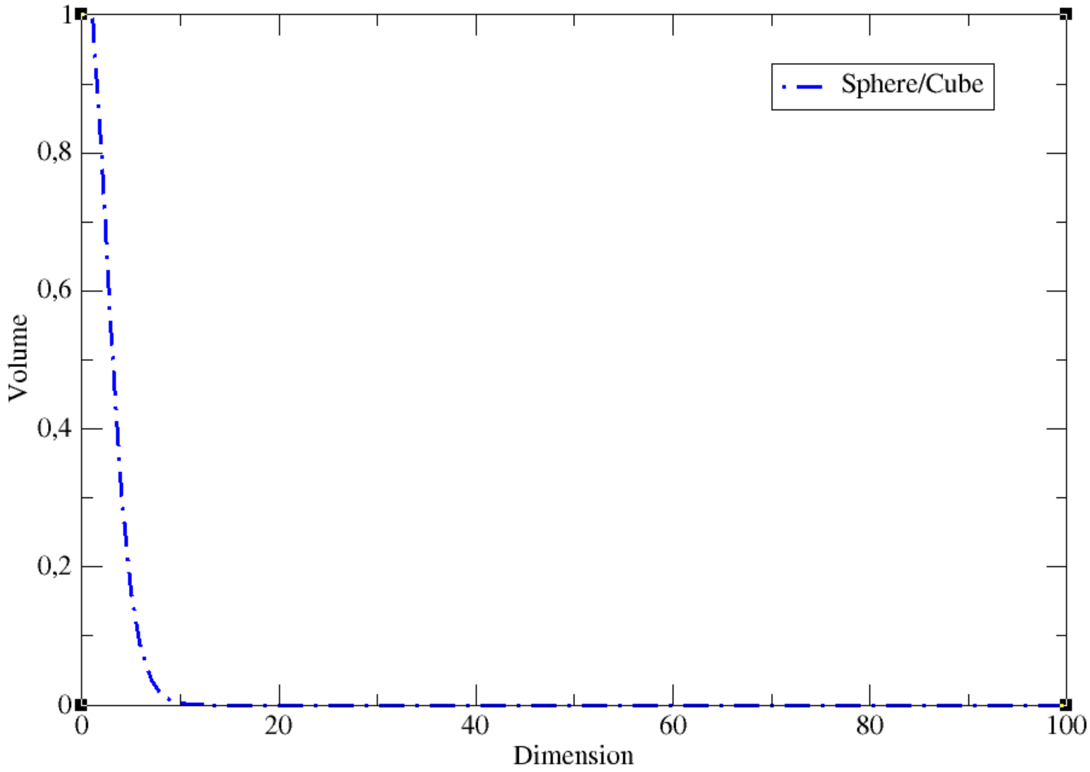

Figura 4: Gráfico da razão entre os volumes da esfera de raio unitário e os volumes do cubo em dimensões de 0 a 100 (arquivo de saída: *razao.dat*).

Código:

```
        write(*,*) "digite a dimensao d:"
        read(*,*) id

        r=1.0

        open(1, file='dim-esferas-new.dat')
        open(2, file='dim-cubo.dat')
        open(3, file='razao.dat')

        do i=0,id
                write(1,*) i, fgamma(i,r)
                write(2,*) i, vcubo(i,r)
                write(3,*) i, fgamma(i,r)/vcubo(i,r)
        end do

        close(1)
        close(2)
        close(3)

        end

        function fgamma(id,r)
        pi=acos(-1.0e0)
        xgamma=1.0
        aux=1.0

c argumento da minha funcao gama
        x=id/2.0+1.0

10      if(x .gt. 0.0) then
c casos base gamma(1)=1 e gamma(1/2)=sqrt(pi)
                if(x .eq. 1.0) then
                        aux=1.0
                else if(x .eq. 0.5) then
                        aux=sqrt(pi)
c calculo gamma(n)=(n-1)!
                else if(x .gt. 1.0) then
                        xgamma=xgamma*(x-1)
                end if

                x=x-1
                xgamma=xgamma*aux
        go to 10
        end if
c formula do volume da esfera de raio 1
        v=(pi**(id/2.0)/xgamma)*r**id
        fgamma=v
        end

        function vcubo(id,r)
        vc=(2*r)**id
        vcubo=vc
        end
```

Referências

[1] takeUforward. Selection sort algorithm. https://takeuforward.org/sorting/selection-sort-algorithm/. Acesso em: 19 ago. 2025.

[2] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function. Acesso em: 20 ago. 2025.

[3] Wikipedia. Monte carlo method. https://en.wikipedia.org/wiki/Monte_Carlo_method. Acesso em: 19 ago. 2025.

[4] Wikipedia. N-sphere. https://en.wikipedia.org/wiki/N-sphere. Acesso em: 20 ago. 2025.