

Projeto 1 - Primeira Série de Tarefas

Instituto de Física de São Carlos

Universidade de São Paulo

Vinícius Bastos Marcos (12556715)

Introdução à Física Computacional

Prof. Francisco Castilho Alcaraz

Setembro, 2022



1ª Tarefa

Essa tarefa tem como objetivo calcular o volume e a área total de um Torus, figura geométrica tridimensional obtida por meio de uma rotação de um círculo em torno de um eixo. A distância entre o centro do círculo e o eixo é o raio externo " r_e " e o raio do círculo " r_i ", raio interno.

Usando ferramentas de integração, facilmente obtemos as seguintes expressões para o volume V e área total " A ":

$$V = (2\pi r_e) \cdot (\pi r_i^2)$$

$$A = (2\pi r_e) \cdot (2\pi r_i)$$

```
1  c      tarefa 1
2      write(*,*) 'Insira os valores dos raios r_i e r_e, nesta ordem.'
3      read(*,*) r1, r2
4      pi = acos(-1.e0)
5      area = 2.e0*pi*r2*(2.e0*pi*r1)
6      write(*,*) 'A área do torus é', area
7      volume = 2.e0*pi*r2*pi*r1*r1
8      write(*,*) 'O volume do torus é', volume
9      end
```

Algoritmo 1: código para resolução da tarefa 1



```
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa1$ ./tarefa-1-#12556715.exe
Insira os valores dos raios r_i e r_e, nesta ordem.
10
12
A área do torus é 4737.41016
O volume do torus é 23687.0508
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa1$ ./tarefa-1-#12556715.exe
Insira os valores dos raios r_i e r_e, nesta ordem.
3
2
A área do torus é 236.870514
O volume do torus é 355.305786
```

Figura 1: testes do código realizados

2ª Tarefa

A segunda tarefa requisita o cálculo do volume e da área total determinada por três vetores dados: \vec{v}_1 , \vec{v}_2 e \vec{v}_3 . Porém o sólido pedido é o determinado por \vec{v}_1 , \vec{v}_2 e $\vec{v}_3 - \vec{v}_2$.

Sabemos que o produto vetorial fornece um outro vetor em que seu módulo é numericamente igual à área determinada pelos vetores multiplicados. Assim, basta fazer o módulo do produto vetorial entre cada vetor das arestas, e multiplicar por dois a soma dessas áreas.

O cálculo do volume usa esse mesmo recurso da área, mas escolhemos apenas dois vetores, no caso \vec{v}_1 e \vec{v}_2 . Assim, falta apenas a altura desse sólido. A projeção do outro

vetor $\vec{v}_3 - \vec{v}_2$ no vetor resultado do produto vetorial entre \vec{v}_1 e \vec{v}_2 , que é o produto vetorial entre esses vetores, nos fornece a altura. Assim conseguimos o volume do prisma.

No código, criei uma função que calcula o produto vetorial entre dois vetores, o módulo de um vetor, e cada componente do produto vetorial entre dois vetores. Fiz uso de funções para organizar melhor o código e deixá-lo mais claro e sucinto. Depois criei uma função que calcula o volume e outra que calcula a área dupla.

Abaixo estão o código implementado, dividido em duas imagens, e alguns testes.

```

1  c      tarefa 2
2
3  c      função produto escalar
4  function esc(x1, y1, z1, x2, y2, z2)
5  esc = (x1*x2 + y1*y2 + z1*z2)
6  return
7  end function esc
8
9  c      funções para o produto vetorial
10 function vet1(y1, z1, y2, z2)
11 vet1 = (y1*z2 - z1*y2)
12 return
13 end function vet1
14
15 function vet2(x1, z1, x2, z2)
16 vet2 = (z1*x2 - z2*x1)
17 return
18 end function vet2
19
20 function vet3(x1, y1, x2, y2)
21 vet3 = (x1*y2 - y1*x2)
22 return
23 end function vet3
24
25 c      função módulo de um vetor
26 function rmodulo(x1, y1, z1)
27 rmodulo = sqrt(x1**2.e0 + y1**2.e0 + z1**2.e0)
28 return
29 end function rmodulo
30
31 c      função que calcula o volume
32 function volume(x1, y1, z1, x2, y2, z2, x3, y3, z3)
33 vx = vet1(y1, z1, y2, z2)
34 vy = vet2(x1, z1, x2, z2)
35 vz = vet3(x1, y1, x2, y2)
36 volume = abs(esc(vx,vy,vz,x3-x2,y3-y2,z3-z2))
37 return
38 end function volume
39
40 c      função que calcula a área de duas faces iguais
41 function areadp(x1,y1,z1,x2,y2,z2)

```

```

42     vx = vet1(y1, z1, y2, z2)
43     vy = vet2(x1, z1, x2, z2)
44     vz = vet3(x1, y1, x2, y2)
45     areadp = rmodulo(vx,vy,vz) * 2.e0
46     return
47 end function areadp
48
49 c     entrada de dados
50     write(*,*) 'Insira as coordenadas de v1:'
51     read(*,*) a1, a2, a3
52     write(*,*) 'Insira as coordenadas de v2:'
53     read(*,*) b1, b2, b3
54     write(*,*) 'Insira as coordenadas de v3:'
55     read(*,*) c1, c2, c3
56
57     write(*,*) 'O volume é', volume(a1,a2,a3,b1,b2,b3,c1,c2,c3)
58     write(*,*) 'A área total é', (areadp(a1,a2,a3,b1,b2,b3)
59 +                                     + areadp(a1,a2,a3,c1-b1,c2-b2,c3-b3)
60 +                                     + areadp(b1,b2,b3,c1-b1,c2-b2,c3-b3))
61
62     end

```

Algoritmo 2: código para resolução da tarefa 2

```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa2$ ./tarefa-2-12556715.exe
Insira as coordenadas de v1:
4 -2 2
Insira as coordenadas de v2:
1 -3 2
Insira as coordenadas de v3:
6 -4 0
O volume é 36.0000000
A área total é 103.663315
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa2$ ./tarefa-2-12556715.exe
Insira as coordenadas de v1:
1 0 0
Insira as coordenadas de v2:
0 1 0
Insira as coordenadas de v3:
0 1 1
O volume é 1.00000000
A área total é 6.00000000

```

Figura 2: testes do código desta tarefa

3ª Tarefa

A terceira tarefa desse projeto pede que sejam ordenados M números, fornecido pelo usuário, de um uma lista com N números.

O programa implementado ordena qualquer quantidade $M \leq N$ para qualquer arquivo com N números fornecidos. Porém, como exemplo de testagem, o arquivo que criei tem $N = 50$, mas como vai ficar claro, o código vale para qualquer valor de N .

O código começa definindo o tamanho da lista de leitura com o valor de N . Logo após ele abre o arquivo de entrada e o de saída.

```

1  c      tarefa 3
2
3      parameter(n = 50)
4      dimension rLista(n)
5
6      in = 10 !unidade arquivo entrada
7      iout = 11 !unidade arquivo saída
8
9      open(unit=in, FILE='entrada_t3.dat')
10     open(unit=iout, FILE='saida_t3.dat')
11
12  c      escolha do usuário da qtde de números a serem ordenados
13      write(*,*) 'Escolha a quantidade M (menor ou igual a 50) de núme
14 +ros a serem ordenados:'
15      read(*,*) M
16
17  c      faz a leitura do arquivo de entrada
18      read(in,*) rLista
19
20  c      bubble sort
21      do i = 1, M
22          do j = M, 2, -1
23              if (rLista(j)<rLista(j-1)) then
24                  termo = rLista(j)
25                  rLista(j) = rLista(j-1)
26                  rLista(j-1) = termo
27              end if
28          enddo
29      enddo
30
31  c      salvando no arquivo de saída
32      do i = 1, M
33          write(iout,*) rLista(i)
34      end do
35
36  c      escrevendo para o usuário
37      do i = 1, M
38          write(*,*) rLista(i)
39      end do
40
41  c      fechando os arquivos
42      close(in)
43      close(iout)
44
45      end

```

Algoritmo 3: código para resolução da tarefa 3

```

1 2
2 -5
3 0
4 1
5 1003
6 13
7 -963
8 13580
9 5010
10 -5000
11 400
12 -169
13 69
14 -69
15 169
16 325
17 963
18 5
19 -3.14
20 6.4
21 -9
22 53
23 20
24 30
25 -96

```

Figura 3: 25 primeiros números do arquivo de entrada

```

26 -45
27 65
28 -8.4
29 63
30 95
31 7423
32 526
33 635
34 2012
35 3256
36 85
37 -96
38 -85
39 2.14
40 -2.72
41 458
42 63
43 63.14
44 98.456
45 -785
46 69
47 1235
48 1918
49 1322
50 10

```

Figura 4: o restante dos número do arquivo de entrada

```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa3$ ./tarefa-3-12556715.exe
Escolha a quantidade M (menor ou igual a 50) de núme ros a serem ordenados:
10
-5000.000000
-963.000000
-5.00000000
0.00000000
1.00000000
2.00000000
13.00000000
1003.000000
5010.000000
13580.000000
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa3$ ./tarefa-3-12556715.exe
Escolha a quantidade M (menor ou igual a 50) de núme ros a serem ordenados:
2
-5.00000000
2.00000000
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa3$ ./tarefa-3-12556715.exe
Escolha a quantidade M (menor ou igual a 50) de núme ros a serem ordenados:
8
-963.000000
-5.00000000
0.00000000
1.00000000
2.00000000
13.00000000
1003.000000
13580.000000

```

Figura 5: testando o programa com valores diferentes de M

4ª Tarefa

Esta tarefa tem como objetivo o cálculo do $\cos(x)$ usando a expansão em séries. Para isso, fez necessário a criação de uma função para o fatorial de um número.

O valor calculado a partir da série, com precisão na quinta casa decimal, deve ser comparado com as funções que já existem no Fortran, $\cos(x)$ e $\operatorname{dcos}(x)$ (que é uma variável de dupla precisão). Por isso, mando o programa escrever no terminal esses valores. Portanto, fez necessário, pela sintaxe da linguagem, declarar uma variável em dupla precisão, o dx .

O *loop* criado para o cálculo do cosseno depende da precisão solicitada, só vai somar termos no cosseno somente se forem maiores que essa precisão.

```

1  c      tarefa 4
2  c      função para fatorial de um número
3
4      function fat(i)
5      fat = 1
6      do 1 j = 1, i
7      fat = fat * j
8  1      continue
9      return
10     end function fat
11
12     real*8 dx !para usar no dcos
13
14  c      entrada de dados
15
16     eprec = 1.e-5
17     write(*,*) 'Insira o valor de x em radianos:'
18     read(*,*) x
19     dx = x
20
21  c      cálculo do cosseno com a precisão eprec
22
23     ifator = 2
24     ipot = 1
25     termo = 1.e0 !iniciando com um número qualquer maior que eprec
26     cosseno = 1.e0
27
28     do while(abs(termo) > eprec)
29         termo = ((-1)**ipot)*((x**ifator)/(fat(ifator)))
30         ifator = ifator + 2
31         ipot = ipot + 1
32         cosseno = cosseno + termo
33     enddo
34
35     write(*,*) 'O cosseno implementado de x é', cosseno
36     write(*,*) 'O cosseno do f77 é', cos(x)
37     write(*,*) 'O cosseno com dupla precisão do f77 é', dcos(dx)
38
39     end

```

Algoritmo 4: código para resolução da tarefa 4

```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa4$ ./tarefa-4-12556715.exe
Insira o valor de x em radianos:
2
0 cosseno implementado de x é -0.416146636
0 cosseno do f77 é -0.416146845
0 cosseno com dupla precisão do f77 é -0.41614683654714241
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa4$ ./tarefa-4-12556715.exe
Insira o valor de x em radianos:
3.14
0 cosseno implementado de x é -0.999998450
0 cosseno do f77 é -0.999998748
0 cosseno com dupla precisão do f77 é -0.99999873189460997
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa4$ ./tarefa-4-12556715.exe
Insira o valor de x em radianos:
0
0 cosseno implementado de x é 1.000000000
0 cosseno do f77 é 1.000000000
0 cosseno com dupla precisão do f77 é 1.000000000000000000

```

Figura 6: exemplos de testes

5ª Tarefa

Nesta tarefa, o programa recebe um arquivo com números de 1 a N, e suas devidas paridades, retornando um arquivo de (N+1) com as paridades. A ideia utilizada neste código foi copiar as N linhas da entrada nas N primeiras linhas da matriz resultante e colocando na coluna (N+1) o valor (N+1). Assim, copiamos as restantes linhas como cópias nas N primeiras. Dessa forma, basta percorrer a matriz resultante, a partir da linha (N+1), e ir trocando as últimas colunas, como o algoritmo de *Bubble Sort*. Com o número de trocas consertamos a paridade copiada, sendo esse número a quantidade de vezes que multiplicamos por -1. Abaixo está o código e a implementação com N = 2.

```

1  c      tarefa 5
2
3  c      definindo os parâmetros da lista e os arquivo de entrada
4  parameter(ncolunas = 3) !o fortran lê de maneira que tranpõe a
5  !matriz usada
6  parameter(nlinhas = 2)
7  dimension iMatriz_t(ncolunas,nlinhas)
8  dimension iMatriz(nlinhas,ncolunas)
9  dimension iMatriz_gerada(nlinhas*ncolunas, ncolunas+1)
10
11  in = 10 !unidade arquivo entrada
12  iout = 11 !unidade arquivo saída
13
14  c      abertura e leitura da entrada
15  open(unit=in, FILE='entrada_t5.dat')
16  open(unit=iout, FILE='saida_t5.dat')
17  read(in,*) iMatriz_t
18
19  iMatriz = transpose(iMatriz_t) !matriz desejada é a transposta da
↪ lida
20
21  kcontaC = ncolunas + 1

```



```

22     ncontaL = nlinhas * ncolunas
23
24 c   as primeiras nlinhas serão iguais a do input com o n+1
25 do nl = 1,nlinhas
26     nc = 1
27     do while(nc < ncolunas)
28         iMatriz_gerada(nl,ncolunas) = ncolunas
29         iMatriz_gerada(nl,ncolunas+1) = iMatriz(nl,ncolunas)
30         iMatriz_gerada(nl,nc) = iMatriz(nl,nc)
31         nc = nc + 1
32     enddo
33 enddo
34
35 c   o restante como cópias das primeiras nlinhas
36 kc = 1
37 do nl = nlinhas + 1, ncontaL
38     do kcol = kcontaC, 1, -1
39         nl_ant = nlinhas * kc
40         iMatriz_gerada(nl, kcol) = iMatriz_gerada(nl-nl_ant,kcol)
41     enddo
42     if (nl - ((nl/nlinhas)*nlinhas)==0) then
43         kc = kc + 1
44     end if
45 enddo
46
47 c   permutando
48 icontador = 1
49 do nl = nlinhas+1, ncontaL
50     k = 1
51     do i = 1, icontador
52        iaux = iMatriz_gerada(nl,kcontaC-k-1)
53         iMatriz_gerada(nl,kcontaC-k-1) = iMatriz_gerada(nl,
54 +                                     kcontaC-k)
55         iMatriz_gerada(nl,kcontaC-k) = iaux
56         iMatriz_gerada(nl, kcontaC) =((-1)**k)*
57 +                                     iMatriz_gerada(nl,kcontaC)
58         k = k + 1
59     enddo
60     if (nl - ((nl/nlinhas)*nlinhas)==0) then
61         icontador = icontador + 1
62     end if
63 enddo
64
65
66 c   imprimindo resultados e colocando no arquivo de saída
67 do i = 1, ncontaL
68     write(*,*) (iMatriz_gerada(i,j), j=1,kcontaC)
69     write(iout,*) (iMatriz_gerada(i,j), j=1,kcontaC)

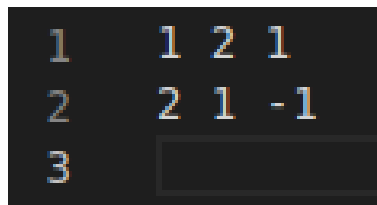
```

```

70     enddo
71
72 c     fechando os arquivos
73     close(in)
74     close(iout)
75
76     end

```

Algoritmo 5: código para resolução da tarefa 5

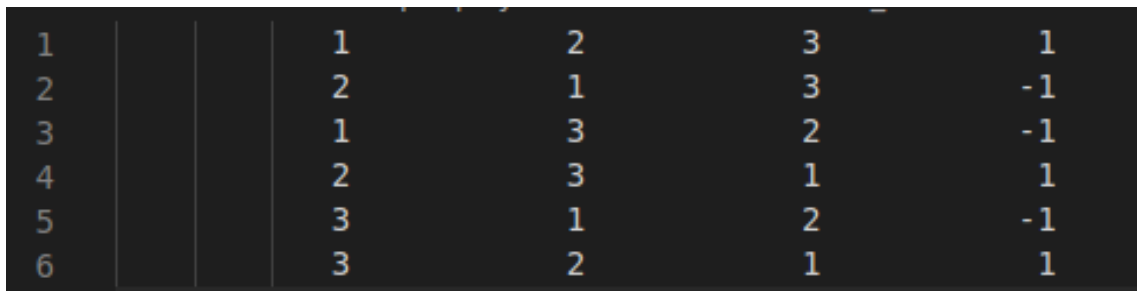


```

1    1 2 1
2    2 1 -1
3    [ ]

```

Figura 7: exemplo de possível entrada



```

1    1    2    3    1
2    2    1    3    -1
3    3    3    2    -1
4    4    3    1    1
5    5    1    2    -1
6    6    3    1    1

```

Figura 8: saída para o exemplo de entrada

6ª Tarefa

Usando as permutações obtidas do programa anterior e colocando em um arquivo de entrada. Em outro arquivo de entrada é colocado a matriz desejada no cálculo do determinante.

Para o cálculo do determinante, percorremos todas linhas da matriz permutação e da matriz desejada, como se explicita no código abaixo.

```

1 c     teste 6
2 c     definindo os parâmetros para a matriz permutação
3     parameter(ncolunas = 4) !o fortran lê de maneira que tranpõe a
4                               !matriz usada
5     parameter(nlinhas = 6) !N = 3 -> nlinhas = N+1 = 4
6     parameter(N = 3)
7     dimension iMatriz_t(ncolunas,nlinhas)
8     dimension iP(nlinhas,ncolunas)
9     dimension matrizT(N,N)
10    dimension matriz(N,N)

```

```

11
12     in = 10
13     in2 = 11
14
15 c    abertura e leitura da entrada
16     open(unit=in, FILE='permutacao-t6.dat')
17     open(unit=in2, FILE='matriz.dat')
18     read(in,*) iMatriz_t
19     read(in2,*) matrizT
20     iP = transpose(iMatriz_t) !matriz desejada é a transposta da lida
21     matriz = transpose(matrizT)
22
23 c    cálculo do determinante a partir do arquivo 'permutacao-t6.dat'
24 c    obtido a partir do programa da tarefa 5
25
26     det = 0.e0
27     do i=1,nlinhas
28         termo = 1.e0
29         do j=1, N
30             termo = termo * matriz(j,iP(i,j))
31         end do
32         termo = termo * iP(i, N+1)
33         det = det + termo
34     end do
35
36     write(*,*) 'O determinante da matriz dada é', det
37
38 c    fechando os arquivos
39     close(in)
40     close(in2)
41
42     end

```

Algoritmo 6: código para resolução da tarefa 6

1	5	6		1	0	0
13	12	-8		0	1	0
22	5	6		0	0	1

Figura 9: entradas para os testes do programa, respectivamente

```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa6$ ./tarefa-6-12556715.exe
O determinante da matriz dada é 36.0000000
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa6$ ./tarefa-6-12556715.exe
O determinante da matriz dada é 1.00000000

```

Figura 10: testes para o programa

8ª Tarefa

Usando M pontos aleatórios, a tarefa pede para que seja calculado o volume de uma esfera, de raio unitário, em d dimensões. Para isso, vamos imaginar, em $d = 2$, um quarto de círculo inscrito em um quadrado. Os pontos podem estar dentro da circunferência e fora dela. O volume é dado pelo quadruplo da razão dos número de pontos interiores em relação ao total. Quanto maior o M, mais próximo chegamos ao resultado real.

Extrapolando para d dimensões, temos a seguinte fórmula:

$$V(d) = 2^d \frac{N_{dentro}}{N_{total}} \quad (1)$$

O código implementado e os testes estão dispostos logo abaixo.

```
1  c      tarefa 8
2  c      cálculo do volume de uma esfera em d dimensões
3  write(*,*) 'Insira, respectivamente, os valores de d e M:'
4  read(*,*) id, M
5
6  c      geração e testagem dos pontos aleatórios
7  dentro = 0
8  fora = 0
9  do npassos = 1, M
10     ponto2 = 0.e0
11     do ndim = 1, id
12         x = rand(iseed) !para ser "mais aleatório"
13         ponto2 = ponto2 + x**2
14     enddo
15     if (sqrt(ponto2) <= 1.e0) then
16         dentro = dentro + 1.e0
17     else
18         fora = fora + 1.e0
19     endif
20 enddo
21
22 c      cálculo do volume
23 razao = dentro/(fora + dentro)
24 volume = (2.e0**id) * (razao)
25
26 write(*,*) 'O volume é', volume
27
28 end
```

Algoritmo 7: código para resolução da tarefa 8

```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa8$ ./tarefa-8-12556715.exe
Insira, respectivamente, os valores de d e M:
2 1000000
O volume é 3.14209604
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa8$ ./tarefa-8-12556715.exe
Insira, respectivamente, os valores de d e M:
3 1000000
O volume é 4.18797588
vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa8$ ./tarefa-8-12556715.exe
Insira, respectivamente, os valores de d e M:
4 1000000
O volume é 4.92361593

```

Figura 11: testes para o programa

9ª Tarefa

Para usar a fórmula apresentada, fez-se necessária a implementação de uma função, recursiva, para o cálculo da função Γ . Os condicionais foram feitos dessa forma para que qualquer errinho de precisão do Fortran não atrapalhasse. O volume calculado a partir da seguinte equação, escrevendo tanto na tela do terminal quanto no arquivo de saída o volume a respectiva dimensão (de um a vinte).

$$V(d) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{1}{2} + 1)} R^d \quad (2)$$

O arquivo de saída foi usado para plotar o gráfico do Volume versus dimensão no *Xmgrace*. O gráfico está presente logo abaixo.

```

1  c      tarefa 9
2  c      função que calcula a função gamma
3      function fgamma(x)
4      pi = acos(-1.e0)
5      fator = 1.e0
6      do while((abs(x - 0.5) > 0.0001).and.(abs(x - 1.e0) > 0.0001))
7          x = x - 1.e0
8          fator = fator * x
9      enddo
10     if (abs(x - 0.5) <= 0.0001) then
11         fgamma = sqrt(pi) * fator
12     else if (abs(x - 1.e0) <= 0.0001) then
13         fgamma = 1.e0 * fator
14     end if
15     return
16 end function fgamma
17
18 c      arquivo de saída
19     parameter(nlinhas = 20)
20     parameter(ncolunas = 2)
21     dimension saida(nlinhas, ncolunas)
22
23     iout = 10
24     open(unit=iout,FILE='dimensões-esferas.dat')

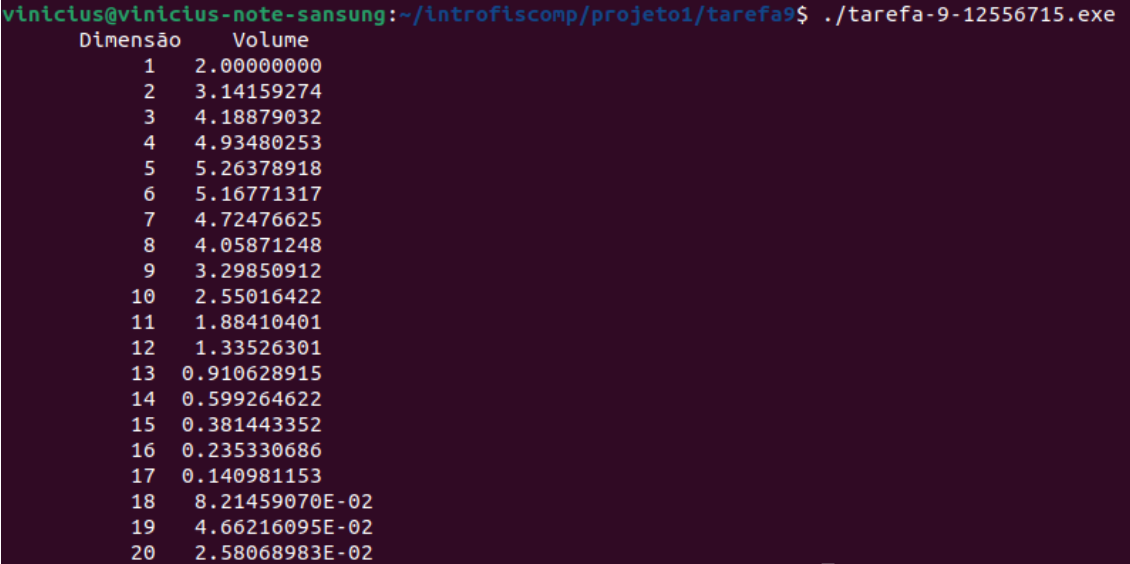
```

```

25
26     write(*,*) '          Dimensão      Volume'
27     do id = 1, 20
28         d = id
29         y = (d/2.e0) + 1.e0
30         pi = acos(-1.e0)
31         R = 1 !raio unitário
32
33         volume = ((pi**(d/2.e0)*(R**d))/fgamma(y))
34         saida(id, 1) = d
35         saida(id, 2) = volume
36         write(*,*)id, volume
37     enddo
38
39 c    imprimindo resultados e colocando no arquivo de saída
40     do i = 1, nlinhas
41         write(iout,*) (saida(i,j), j=1,ncolunas)
42     enddo
43
44     close(iout) !fechando o arquivo de saída
45
46     end

```

Algoritmo 8: código para resolução da tarefa 9



```

vinicius@vinicius-note-samsung:~/introfiscomp/projeto1/tarefa9$ ./tarefa-9-12556715.exe

```

Dimensão	Volume
1	2.00000000
2	3.14159274
3	4.18879032
4	4.93480253
5	5.26378918
6	5.16771317
7	4.72476625
8	4.05871248
9	3.29850912
10	2.55016422
11	1.88410401
12	1.33526301
13	0.910628915
14	0.599264622
15	0.381443352
16	0.235330686
17	0.140981153
18	8.21459070E-02
19	4.66216095E-02
20	2.58068983E-02

Figura 12: imagem do terminal quando rodamos o programa

1	1.00000000	2.00000000
2	2.00000000	3.14159274
3	3.00000000	4.18879032
4	4.00000000	4.93480253
5	5.00000000	5.26378918
6	6.00000000	5.16771317
7	7.00000000	4.72476625
8	8.00000000	4.05871248
9	9.00000000	3.29850912
10	10.00000000	2.55016422
11	11.00000000	1.88410401
12	12.00000000	1.33526301
13	13.00000000	0.910628915
14	14.00000000	0.599264622
15	15.00000000	0.381443352
16	16.00000000	0.235330686
17	17.00000000	0.140981153
18	18.00000000	8.21459070E-02
19	19.00000000	4.66216095E-02
20	20.00000000	2.58068983E-02

Figura 13: arquivo de saída do programa

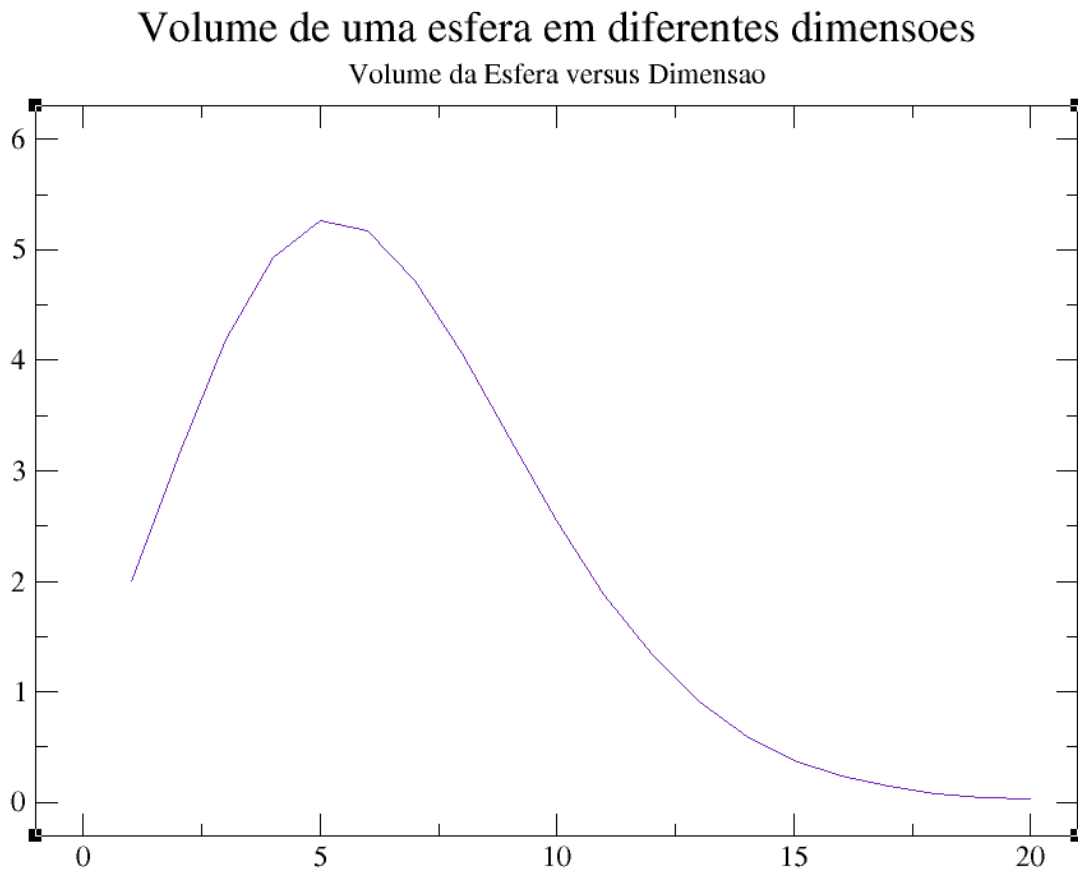


Figura 14: gráfico plotado no Xmgrace

Pergunta A

Tomando como base o limite do gráfico plotado com $d \rightarrow \infty$, temos que o volume da esfera tende a zero. Assim, a razão entre o volume do cubo com o volume da esfera vai para infinito.

Logo, o volume do cubo é infinitas vezes maior que a esfera quando a dimensão tende ao infinito.

Pergunta B

Considerando a célula um cubo e o átomo uma esfera, temos o seguinte. Como o volume da esfera tende à zero, quando $d \rightarrow \infty$, a quantidade de átomos presentes em uma célula tende ao infinito também. Logo, a constante de Avogadro vai para o infinito.