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

$1^{\underline{a}}$ 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 m 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":

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

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

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

Algoritmo 1: código para resolução da tarefa 1

Figura 1: testes do código realizados

$2^{\underline{a}}$ 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 determinando 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.

```
c     tarefa 2

c     função produto escalar
      function esc(x1, y1, z1, x2, y2, z2)
      esc = (x1*x2 + y1*y2 + z1*z2)
      return
      end function esc

c     funções para o produto vetorial
      function vet1(y1, z1, y2, z2)
      vet1 = (y1*z2 - z1*y2)
      return
      end function vet1

      function vet2(x1, z1, x2, z2)
      vet2 = (z1*x2 - z2*x1)
      return
      end function vet2

      function vet3(x1, y1, x2, y2)
      vet3 = (x1*y2 - y1*x2)
      return
      end function vet3

c     função módulo de um vetor
      function  rmodulo(x1, y1, z1)
       rmodulo = sqrt(x1**2.e0 + y1**2.e0 + z1**2.e0)
      return
      end function rmodulo

c     função que calcula o volume
      function volume(x1, y1, z1, x2, y2, z2, x3, y3, z3)
      vx = vet1(y1, z1, y2, z2)
      vy = vet2(x1, z1, x2, z2)
      vz = vet3(x1, y1, x2, y2)
      volume = abs(esc(vx,vy,vz,x3-x2,y3-y2,z3-z2))
      return
      end function volume

c     função que calcula a área de duas faces igauis
      function areadp(x1,y1,z1,x2,y2,z2)
```

```
      vx = vet1(y1, z1, y2, z2)
      vy = vet2(x1, z1, x2, z2)
      vz = vet3(x1, y1, x2, y2)
      areadp = rmodulo(vx,vy,vz) * 2.e0
      return
      end function areadp

c     entrada de dados
      write(*,*) 'Insira as coordenadas de v1:'
      read(*,*) a1, a2, a3
      write(*,*) 'Insira as coordenadas de v2:'
      read(*,*) b1, b2, b3
      write(*,*) 'Insira as coordenadas de v3:'
      read(*,*) c1, c2, c3

      write(*,*) 'O volume é', volume(a1,a2,a3,b1,b2,b3,c1,c2,c3)
      write(*,*) 'A área total é',(areadp(a1,a2,a3,b1,b2,b3)
     +                             + areadp(a1,a2,a3,c1-b1,c2-b2,c3-b3)
     +                             + areadp(b1,b2,b3,c1-b1,c2-b2,c3-b3))

      end
```

Algoritmo 2: código para resolução da tarefa 2

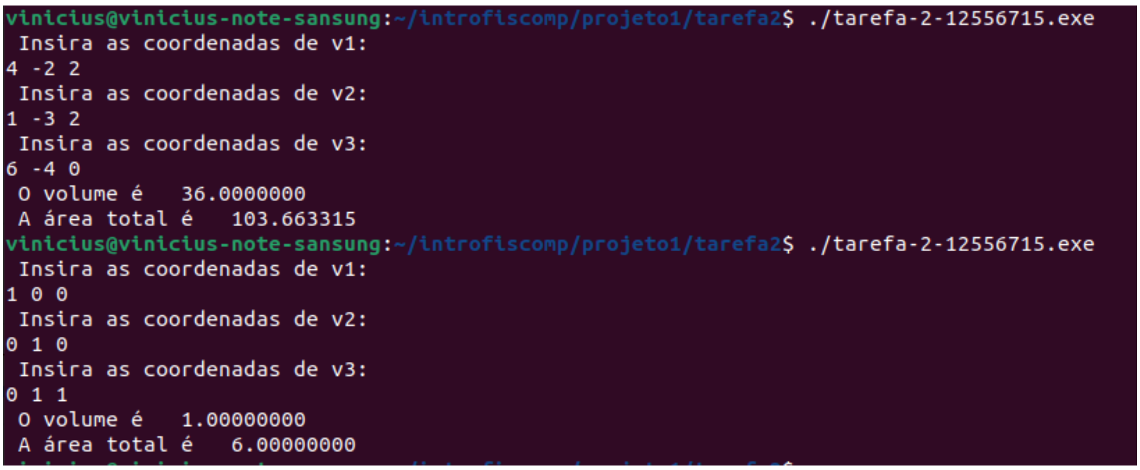


Figura 2: testes do código desta tarefa

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.

```
c     tarefa 3

      parameter(n = 50)
      dimension rLista(n)

      in = 10 !unidade arquivo entrada
      iout = 11 !unidade arquivo saída

      open(unit=in, FILE='entrada_t3.dat')
      open(unit=iout, FILE='saida_t3.dat')

c     escolha do usuário da qtde de números a serem ordenados
      write(*,*) 'Escolha a quantidade M (menor ou igual a 50) de núme
     +ros a serem ordenados:'
      read(*,*) M

c     faz a leitura do arquivo de entrada
      read(in,*) rLista

c     bubble sort
      do i = 1, M
         do j = M, 2, -1
                if (rLista(j)<rLista(j-1)) then
                        termo = rLista(j)
                        rLista(j) = rLista(j-1)
                        rLista(j-1) = termo
                 end if
         enddo
      enddo

c     salvando no arquivo de saída
      do i = 1, M
          write(iout,*) rLista(i)
      end do

c     escrevendo para o usuário
      do i = 1, M
          write(*,*) rLista(i)
      end do

c     fechando os arquivos
      close(in)
      close(iout)

      end
```

Algoritmo 3: código para resolução da tarefa 3

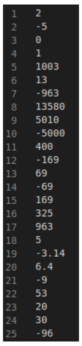


Figura 3: 25 primeiros números do arquivo de entrada

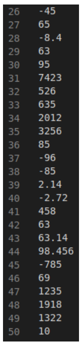


Figura 4: o restante dos número do arquivo de entrada

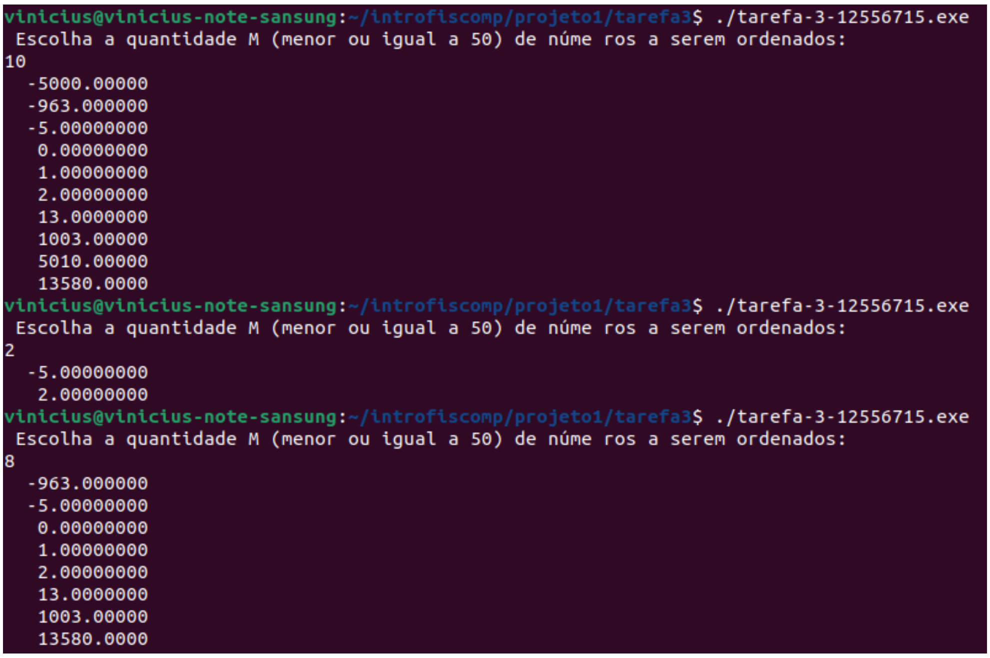


Figura 5: testando o programa com valores diferentes de M

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 $dcos(x)$ (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 que essa precisão.

```
c      tarefa 4
c      função para fatorial de um número

       function fat(i)
       fat = 1
       do 1 j = 1, i
       fat = fat * j
1      continue
       return
       end function fat

       real*8 dx !para usar no dcos

c      entrada de dados

       eprec = 1.e-5
       write(*,*) 'Insira o valor de x em radianos:'
       read(*,*) x
       dx = x

c      cálculo do cosseno com a precisão eprec

       ifator = 2
       ipot = 1
       termo = 1.e0 !iniciando com um número qualquer maior que eprec
       cosseno = 1.e0

       do while(abs(termo) > eprec)
           termo = ((-1)**ipot)*((x**ifator)/(fat(ifator)))
           ifator = ifator + 2
           ipot = ipot + 1
           cosseno = cosseno + termo
       enddo

       write(*,*) 'O cosseno implementado de x é', cosseno
       write(*,*) 'O cosseno do f77 é', cos(x)
       write(*,*) 'O cosseno com dupla precisão do f77 é', dcos(dx)

       end
```

Algoritmo 4: código para resolução da tarefa 4

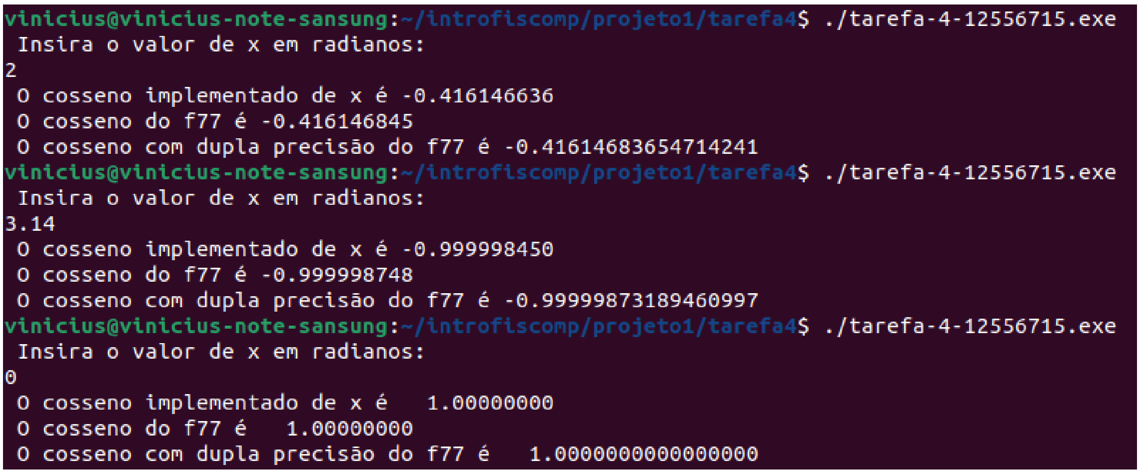


Figura 6: exemplos de testes

5ª Tarefa

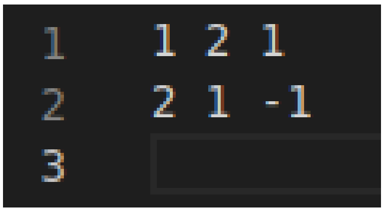
Nesta tarefa, o programa recebe um arquivo com números de 1 a N, e suas devidas paridades, retornando um arquivo de (N+1) com as paridades. A ideia utilizada neste código foi copiar as N linhas da entrada nas N primeiras linhas da matriz resultante e colocando na coluna (N+1) o valor (N+1). Assim, copiamos 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.

```fortran
c     tarefa 5

c     definindo os parâmetros da lista e os arquivo de entrada
      parameter(ncolunas = 3) !o fortran lê de maneira que tranpõe a
                              !matriz usada
      parameter(nlinhas = 2)
      dimension iMatriz_t(ncolunas,nlinhas)
      dimension iMatriz(nlinhas,ncolunas)
      dimension iMatriz_gerada(nlinhas*ncolunas, ncolunas+1)

      in = 10 !unidade arquivo entrada
      iout = 11 !unidade arquivo saída

c     abertura e leitura da entrada
      open(unit=in, FILE='entrada_t5.dat')
      open(unit=iout, FILE='saida_t5.dat')
      read(in,*) iMatriz_t

      iMatriz = transpose(iMatriz_t) !matriz desejada é a transposta da lida

      kcontaC = ncolunas + 1
```

```
      ncontaL = nlinhas * ncolunas

c     as primeiras nlinhas serão iguais a do input com o n+1
      do nl = 1,nlinhas
          nc = 1
          do while(nc < ncolunas)
          iMatriz_gerada(nl,ncolunas) = ncolunas
          iMatriz_gerada(nl,ncolunas+1) = iMatriz(nl,ncolunas)
          iMatriz_gerada(nl,nc) = iMatriz(nl,nc)
          nc = nc + 1
          enddo
      enddo

c     o restante como cópias das primeiras nlinhas
      kc = 1
      do nl = nlinhas + 1, ncontaL
          do kcol = kcontaC, 1, -1
              nl_ant = nlinhas * kc
              iMatriz_gerada(nl, kcol) = iMatriz_gerada(nl-nl_ant,kcol)
          enddo
          if (nl - ((nl/nlinhas)*nlinhas)==0) then
          kc = kc + 1
          end if
      enddo

c     permutando
      icontador = 1
      do nl = nlinhas+1, ncontaL
          k = 1
          do i = 1, icontador
          iaux = iMatriz_gerada(nl,kcontaC-k-1)
          iMatriz_gerada(nl,kcontaC-k-1) = iMatriz_gerada(nl,
     +                                            kcontaC-k)
          iMatriz_gerada(nl,kcontaC-k) = iaux
          iMatriz_gerada(nl, kcontaC) =((-1)**k)*
     +                                iMatriz_gerada(nl,kcontaC)
          k = k + 1
          enddo
          if (nl - ((nl/nlinhas)*nlinhas)==0) then
            icontador = icontador + 1
          end if
      enddo


c     imprimindo resultados e colocando no arquivo de saída
      do i = 1, ncontaL
          write(*,*) (iMatriz_gerada(i,j), j=1,kcontaC)
          write(iout,*) (iMatriz_gerada(i,j), j=1,kcontaC)
```

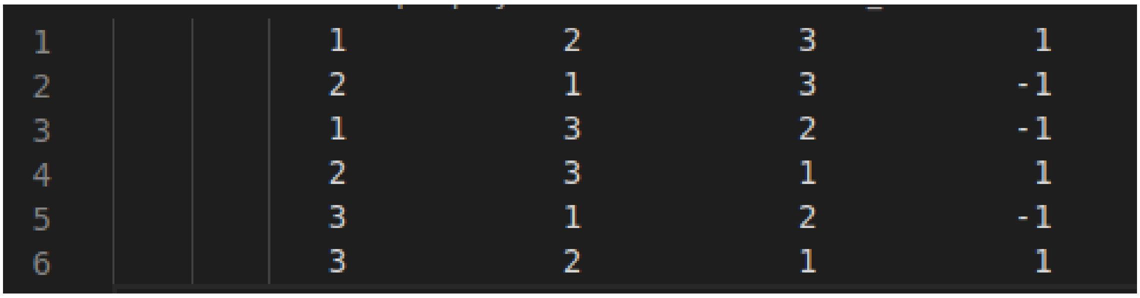
```
      enddo

c     fechando os arquivos
      close(in)
      close(iout)

      end
```

Algoritmo 5: código para resolução da tarefa 5



```
1 2 1
2 1 -1
```

Figura 7: exemplo de possível entrada



```
1    2    3    1
2    1    3   -1
1    3    2   -1
2    3    1    1
3    1    2   -1
3    2    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.

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

```

      in = 10
      in2 = 11

c     abertura e leitura da entrada
      open(unit=in, FILE='permutacao-t6.dat')
      open(unit=in2, FILE='matriz.dat')
      read(in,*) iMatriz_t
      read(in2,*) matrizT
      iP = transpose(iMatriz_t) !matriz desejada é a transposta da lida
      matriz = transpose(matrizT)

c     cálculo do determinante a partir do arquivo 'permutacao-t6.dat'
c     obtido a partir do programa da tarefa 5

      det = 0.e0
      do i=1,nlinhas
         termo = 1.e0
        do j=1, N
         termo = termo * matriz(j,iP(i,j))
        end do
         termo = termo * iP(i, N+1)
         det = det + termo
      end do

      write(*,*) 'O determinante da matriz dada é', det

c     fechando os arquivos
      close(in)
      close(in2)

      end
```

Algoritmo 6: código para resolução da tarefa 6

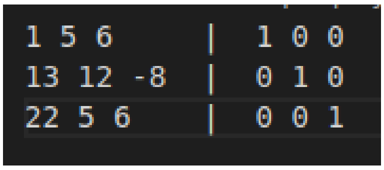


Figura 9: entradas para os testes do programa, respectivamente

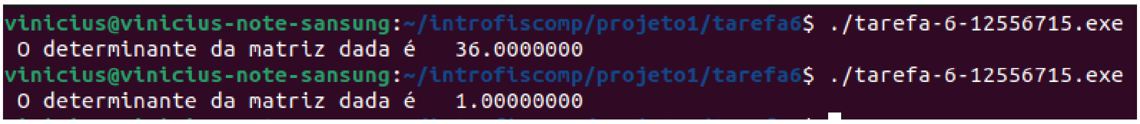


Figura 10: testes para o programa

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 circulo 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:

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

O código implementado e os testes estão dispostos logo abaixo.

```
c       tarefa 8
c       cáculo do volume de uma esfera em d dimensões
        write(*,*) 'Insira, respectivamente, os valores de d e M:'
        read(*,*) id, M

c       geração e testagem dos pontos aleatórios
        dentro = 0
        fora = 0
        do npassos = 1, M
            ponto2 = 0.e0
            do ndim = 1, id
                x = rand(iseed) !para ser "mais aleatório"
                ponto2 = ponto2 + x**2
            enddo
            if (sqrt(ponto2) <= 1.e0) then
                dentro = dentro + 1.e0
            else
                fora = fora + 1.e0
            endif
        enddo

c       cálculo do volume
        razao = dentro/(fora + dentro)
        volume = (2.e0**id) * (razao)

        write(*,*) 'O volume é', volume

        end
```

Algoritmo 7: código para resolução da tarefa 8

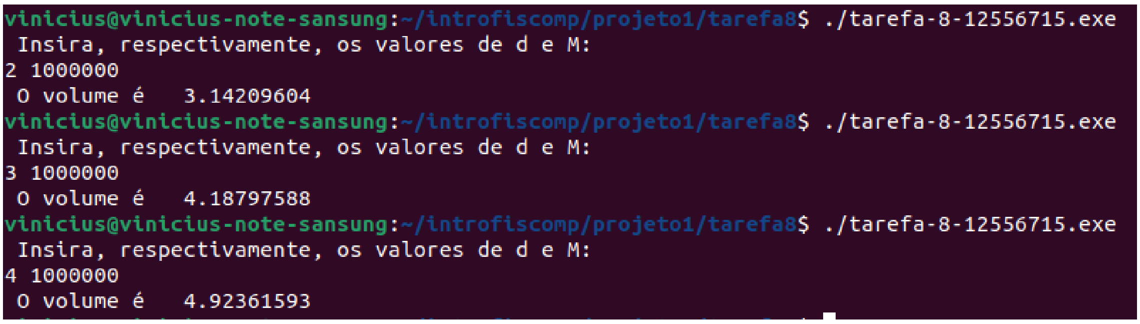


Figura 11: testes para o programa

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).

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

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

```
c      tarefa 9
c      função que calcula a função gamma
       function fgamma(x)
       pi = acos(-1.e0)
       fator = 1.e0
       do while((abs(x - 0.5) > 0.0001).and.(abs(x - 1.e0) > 0.0001))
         x = x - 1.e0
         fator = fator * x
       enddo
       if (abs(x - 0.5) <= 0.0001) then
         fgamma = sqrt(pi) * fator
       else if (abs(x - 1.e0) <= 0.0001) then
            fgamma = 1.e0 * fator
       end if
       return
       end function fgamma

c      arquivo de saída
       parameter(nlinhas = 20)
       parameter(ncolunas = 2)
       dimension saida(nlinhas, ncolunas)

       iout = 10
       open(unit=iout,FILE='dimensões-esferas.dat')
```

```
      write(*,*) '     Dimensão    Volume'
      do id = 1, 20
          d = id
          y = (d/2.e0) + 1.e0
          pi = acos(-1.e0)
          R = 1 !raio unitário

          volume = ((pi**(d/2.e0)*(R**d))/fgamma(y))
          saida(id, 1) = d
          saida(id, 2) = volume
          write(*,*)id, volume
      enddo

c     imprimindo resultados e colocando no arquivo de saída
      do i = 1, nlinhas
          write(iout,*) (saida(i,j), j=1,ncolunas)
      enddo

      close(iout) !fechando o arquivo de saída

      end
```

Algoritmo 8: código para resolução da tarefa 9

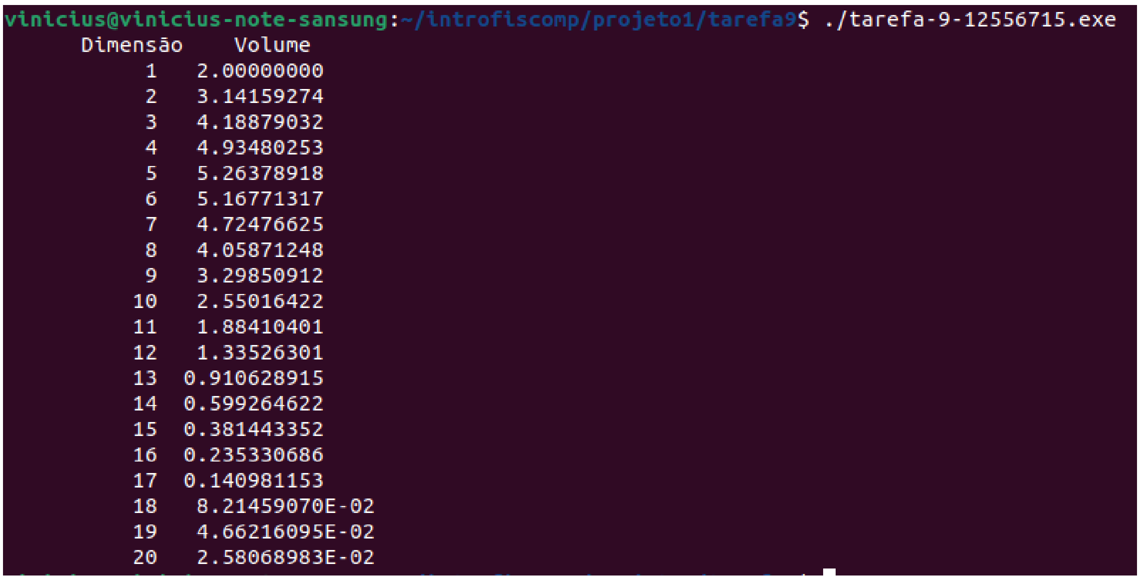


Figura 12: imagem do terminal quando rodamos o programa

```
1.00000000     2.00000000
2.00000000     3.14159274
3.00000000     4.18879032
4.00000000     4.93480253
5.00000000     5.26378918
6.00000000     5.16771317
7.00000000     4.72476625
8.00000000     4.05871248
9.00000000     3.29850912
10.0000000     2.55016422
11.0000000     1.88410401
12.0000000     1.33526301
13.0000000    0.910628915
14.0000000    0.599264622
15.0000000    0.381443352
16.0000000    0.235330686
17.0000000    0.140981153
18.0000000     8.21459070E-02
19.0000000     4.66216095E-02
20.0000000     2.58068983E-02
```

Figura 13: arquivo de saída do programa

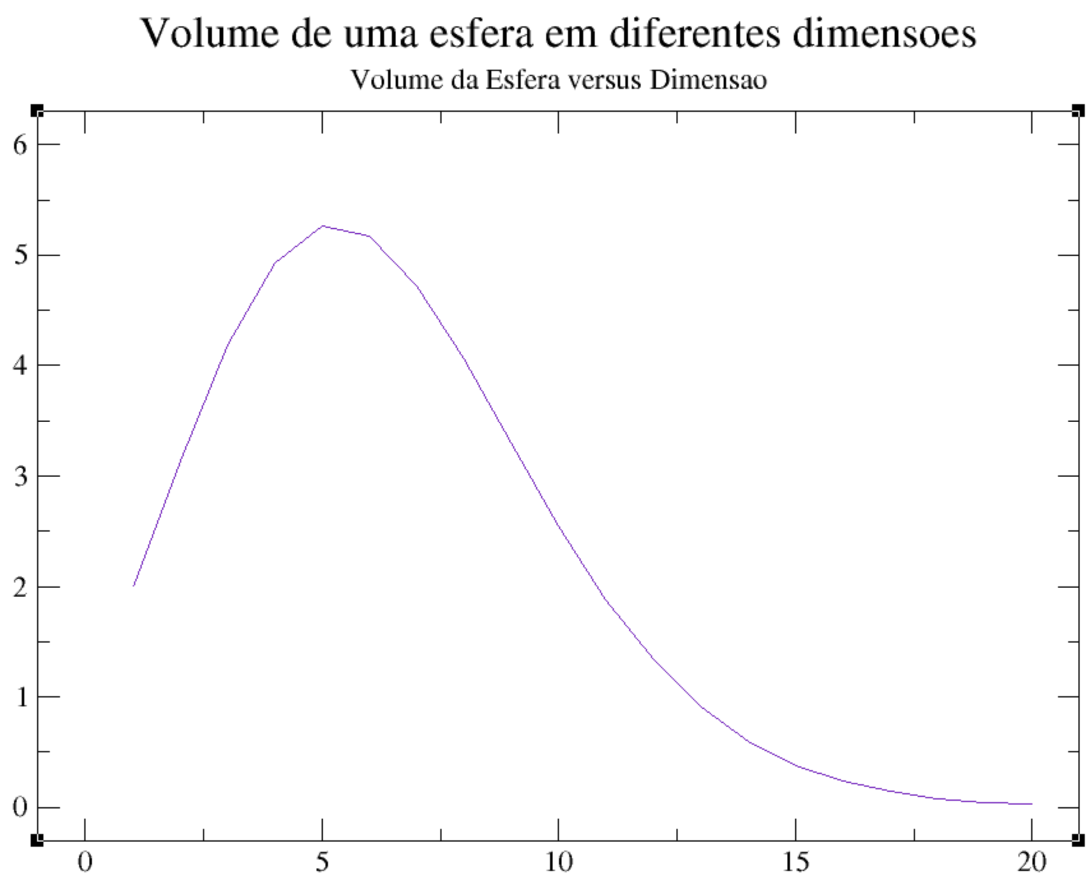


Figura 14: gráfico plotado no Xmgrace

Pergunta A

Tomando como base o limite do gráfico plotado com $d \to \infty$, 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 $d \to \infty$, a quantidade de átomos presentes em uma célula tende ao infinito também. Logo, a constante de Avogadro vai para o infinito.