

# Universidade de São Paulo

## Instituto de Física de São Carlos

### **Projeto 1**

Pedro Calligaris Delbem 5255417

Professor: Francisco Alcaraz

Agosto de 2023

# Sumário

<b>1</b>	<b>Tarefas</b>	<b>2</b>
1.1	Tarefa 1 . . . . .	2
1.2	Tarefa 2 . . . . .	3
1.3	Tarefa 3 . . . . .	5
1.4	Tarefa 4 . . . . .	7
1.5	Tarefa 5 . . . . .	9
1.6	Tarefa 6 . . . . .	12
1.7	Tarefa 7 . . . . .	14
1.8	Tarefa 8 . . . . .	17
<b>2</b>	<b>Perguntas</b>	<b>20</b>
2.1	A . . . . .	20
2.2	B . . . . .	21



Caso  $\Delta$  seja maior que 0, calcula-se as duas raízes dada pela fórmula de Bhaskara, que é

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} \quad (3)$$

Por fim, escreve na tela que existem duas raízes e também quais são os seus valores.

Caso  $\Delta$  seja igual a 0, o " $\pm$ " da fórmula de Bhaskara não afeta o valor de modo que só haverá uma raiz. Assim o programa calcula a mesma por

$$x = \frac{-b}{2a} \quad (4)$$

E escreve na tela que há apenas uma raiz e qual é o valor da mesma.

No último caso em que  $\Delta$  é menor que 0, não existem raízes reais e o programa retorna este fato ao usuário, escrevendo-o na tela.

## 1.2 Tarefa 2

Tarefa: escrever um programa em FORTRAN77 que, dadas coordenadas cartesianas de dois vetores, calcule a área do triângulo formado pelos mesmos

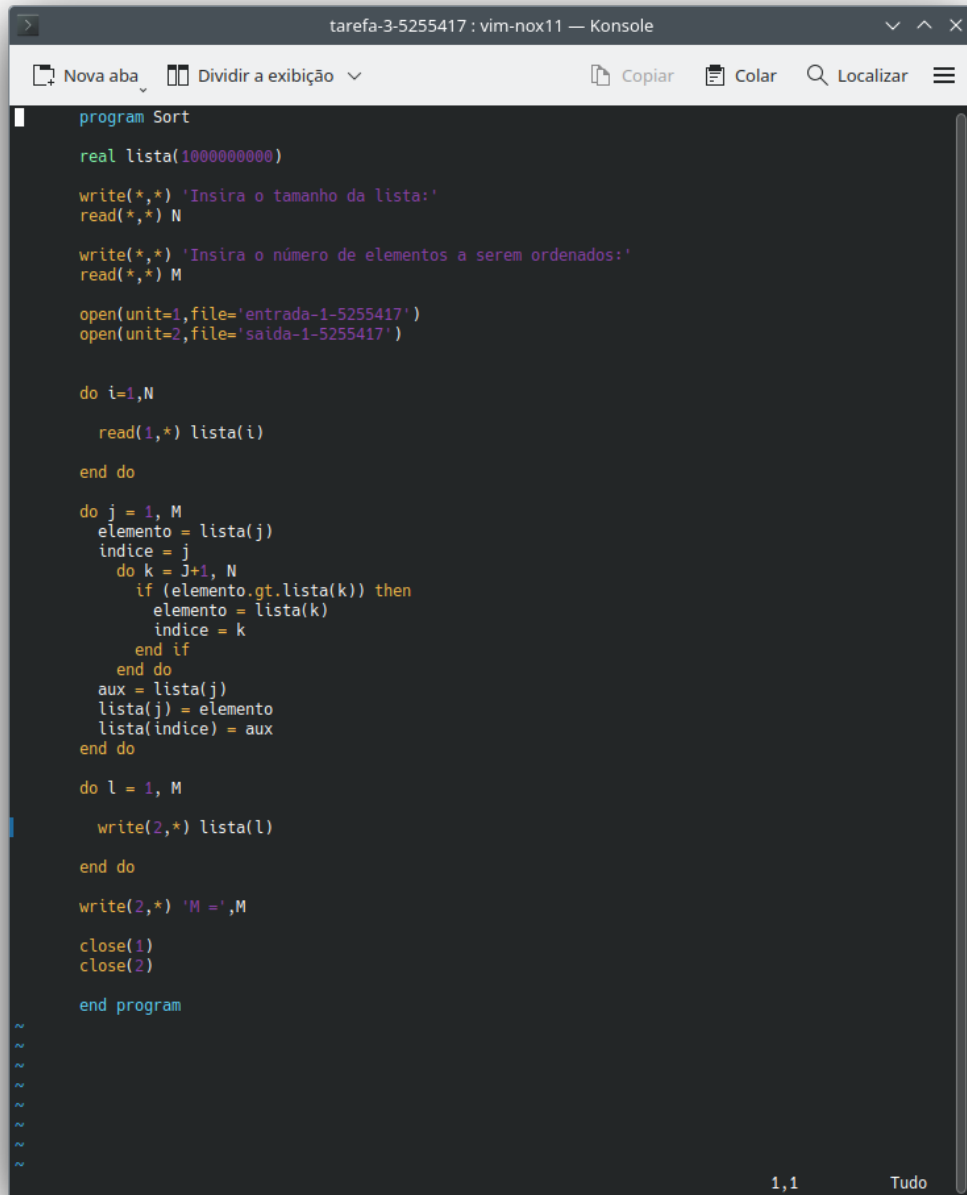
Código escrito:



## 1.3 Tarefa 3

Tarefa: escrever um programa em FORTRAN77 que lê N números reais, em um arquivo, e ordena os M primeiros menores numeros e os imprime em um arquivo de saída

Código escrito:



```
program Sort
  real lista(1000000000)

  write(*,*) 'Insira o tamanho da lista:'
  read(*,*) N

  write(*,*) 'Insira o número de elementos a serem ordenados:'
  read(*,*) M

  open(unit=1,file='entrada-1-5255417')
  open(unit=2,file='saida-1-5255417')

  do i=1,N
    read(1,*) lista(i)
  end do

  do j = 1, M
    elemento = lista(j)
    indice = j
    do k = j+1, N
      if (elemento.gt.lista(k)) then
        elemento = lista(k)
        indice = k
      end if
    end do
    aux = lista(j)
    lista(j) = elemento
    lista(indice) = aux
  end do

  do l = 1, M
    write(2,*) lista(l)
  end do

  write(2,*) 'M =',M

  close(1)
  close(2)

end program
```

Descrição: Primeiramente o programa declara uma lista tamanho grande o suficiente para armazenar a lista que será tralhadada. Em seguida pede o tamanho da mesma, lendo-o. Por fim pede e lê a quantidade de itens a serem ordenados.

Em seguida são alocadas duas unidades de memória - uma para o arquivo de entrada e outra para o arquivo de saída. Deste modo, o arquivo de entrada com nome "entrada-1-5155417" é lido e salvo na lista "lista".

O programa então inicia um loop onde percorrerá os M primeiros valores da lista salvando seu valor e posição. A cada valor ele iniciará outro loop onde percorrerá a lista da posição do próprio valor até o final da mesma e caso algum valor seja menor que o inicial, tal valor (sobrescrevendo o antigo na mesma variável) e posição serão salvos, por fim teremos o menor valor deste intervalo e sua posição, que será trocado de posição com o valor do início do processo.

Assim, obtém-se a lista com os  $M$  primeiros menores valores nas  $M$  primeiras posições e em ordem crescente.

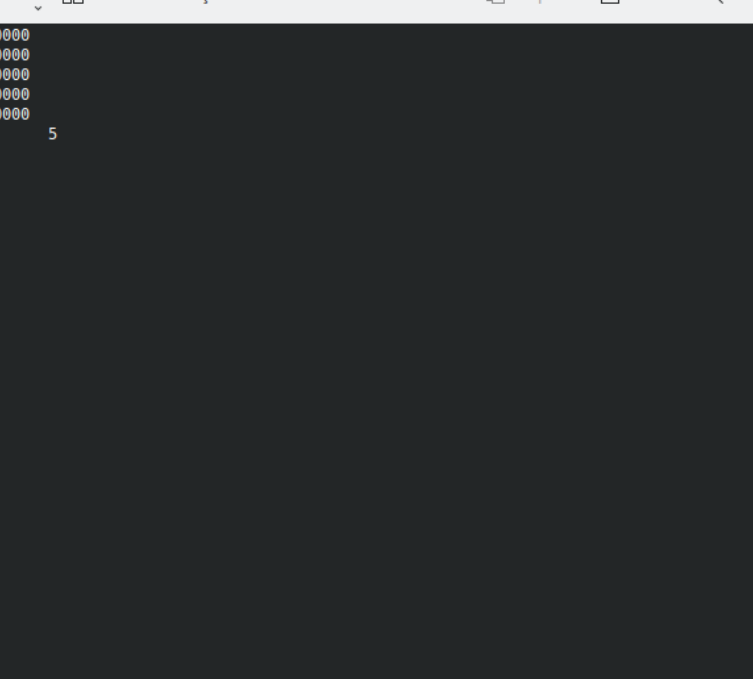
Por fim, inicia-se um loop escrevendo tais valores no arquivo de saída "saida-1-5155417" e após isso escreve-se também o valor de M. E ambas as unidades de memória são fechadas.

Os arquivos de entrada e saída estão ambos a seguir para o caso  $N=10$  e  $M=5$ .

Entrada:

The screenshot shows a terminal window titled "tarefa-3-5255417 : vim-nox11 — Konsole". The terminal interface includes a menu bar with options like "Nova aba", "Dividir a exibição", "Copiar", "Colar", "Localizar", and a hamburger menu. The main content area displays a list of numbers: 10, 7, 11, 100, 43, 54, 1, 12, 2, 3, followed by a series of tilde (~) characters. At the bottom, a status bar shows the file path "entrada-1-5255417" 10L, 27B, the cursor position 1,1, and the word "Tudo".

Saída:

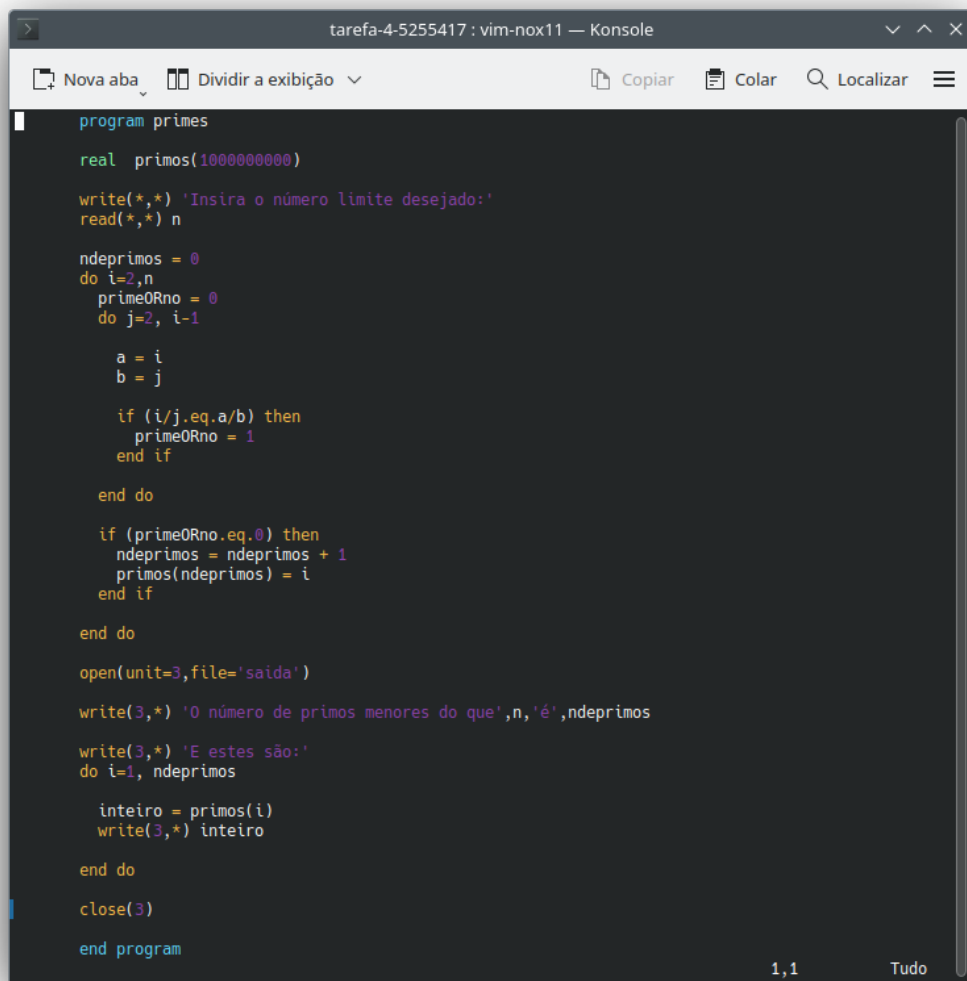


## 1.4 Tarefa 4

Tarefa: escrever um programa em FORTRAN77 que, dado um número N, calcula e retorna todos os números primos iguais ou menores que N e o número deles e escreva estes dados em um arquivo.

Código escrito:



A screenshot of a terminal window titled 'tarefa-4-5255417: vim-nox11 — Konsole'. The window displays a Fortran program named 'primes'. The program declares a large array 'primos' of size 100,000,000. It prompts the user to enter a limit 'n'. It then iterates from 2 to 'n', checking for divisibility by smaller numbers. If a number is prime, it is stored in the 'primos' array. Finally, it prints the count of primes and lists them. The status bar at the bottom right shows '1,1' and 'Tudo'.

```
program primes
real primos(1000000000)

write(*,*) 'Insira o número limite desejado:'
read(*,*) n

ndeprimos = 0
do i=2,n
  primeORno = 0
  do j=2, i-1

    a = i
    b = j

    if (i/j.eq.a/b) then
      primeORno = 1
    end if

  end do

  if (primeORno.eq.0) then
    ndeprimos = ndeprimos + 1
    primos(ndeprimos) = i
  end if

end do

open(unit=3,file='saida')

write(3,*) 'O número de primos menores do que',n,'é',ndeprimos

write(3,*) 'E estes são:'
do i=1, ndeprimos

  inteiro = primos(i)
  write(3,*) inteiro

end do

close(3)

end program
```

Descrição: Primeiramente o programa declara uma lista tamanho grande o suficiente para armazenar a lista que será tralanhada. Em seguinte pergunta ao usuário qual será o número limite desejado para os primos calculados e salva a resposta no inteiro N.

Em seguida, cria-se a variável `ndeprimos` para contabilizar os primos encontrados. Assim, inicia-se um loop que percorrerá os potenciais primos (números de 2 a N) e ao começar seta o valor de `primeORno` para zero, indicando que a princípio o número será considerado um primo.

Ademais, inicia-se um outro loop - dentro do anterior - que percorrerá valores que serão os divisores dos valores do primeiro loop. Fazendo esta divisão para o valor do primeiro loop real e também para o mesmo sendo inteiro, descobrimos que tal valor não é primo se ambas foram iguais, pois indica que tal valor é divisível por algum número sem ser 1 ou ele mesmo. Deste modo - caso as divisões sejam iguais - , seta-se o valor de `primeORno` para 1, indicando que o número não é primo.

Para encerrar um ciclo do loop, o programa verifica o valor de `primeORno` e caso seja zero adiciona tal valor a lista de primos e incrementa `ndeprimos` por 1.

O código aloca, então, uma unidade de memória para escrever o arquivo "saida" e escreve o número de primos e em seguida utilizando um loop que vai de 1 até o número de primos, escreve cada primo um por linha.

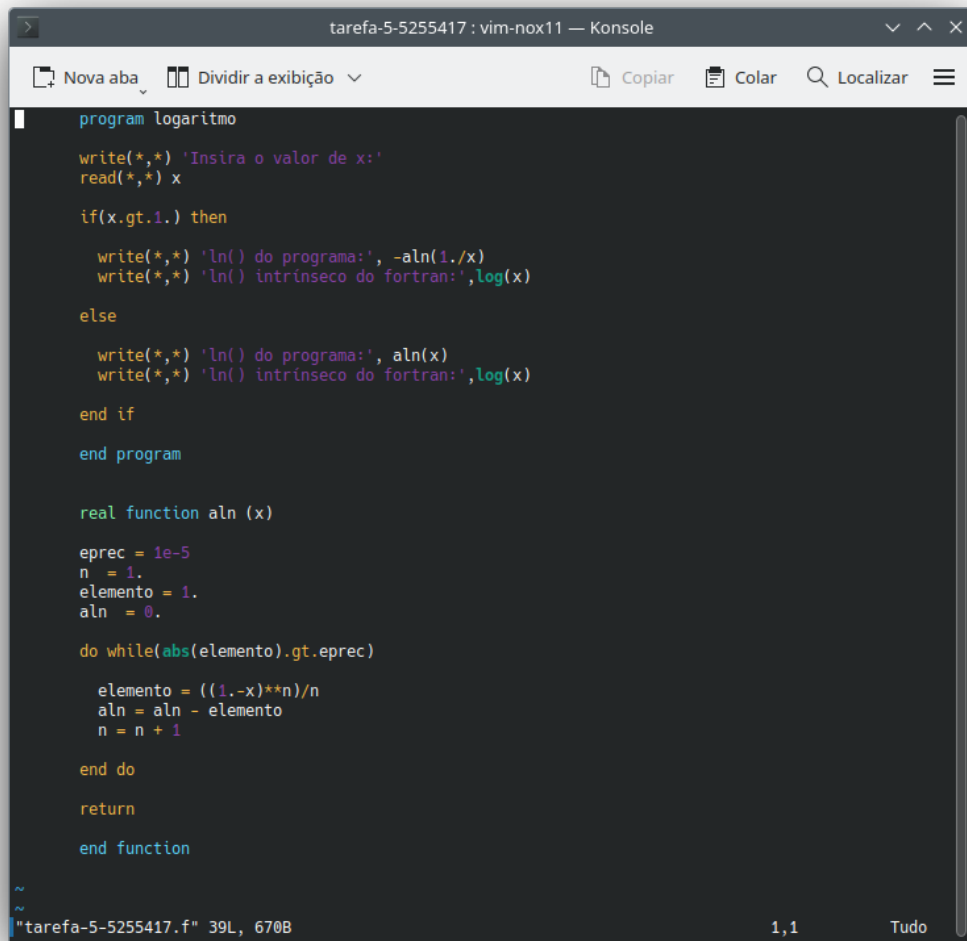
Por fim, é fechada a unidade de memória utilizada.

O código foi testado para os casos  $N = 100, 1000, 10000$  e os arquivos resultantes estão nomeados como "saida-N-n°usp".

## 1.5 Tarefa 5

Tarefa: escrever dois códigos em FORTRAN77 - um para simples precisão e outro para dupla precisão - que calcula o  $\ln()$  de um número  $N$  dado.

Código escrito:



```
program logaritmo
  write(*,*) 'Insira o valor de x:'
  read(*,*) x

  if(x.gt.1.) then
    write(*,*) 'ln() do programa:', -aln(1./x)
    write(*,*) 'ln() intrínseco do fortran:', log(x)
  else
    write(*,*) 'ln() do programa:', aln(x)
    write(*,*) 'ln() intrínseco do fortran:', log(x)
  end if
end program

real function aln (x)
  eprec = 1e-5
  n = 1.
  elemento = 1.
  aln = 0.

  do while(abs(elemento).gt.eprec)
    elemento = ((1.-x)**n)/n
    aln = aln - elemento
    n = n + 1
  end do

  return
end function
```

~  
"tarefa-5-5255417.f" 39L, 670B 1,1 Tudo

Descrição: Primeiramente o programa pede o valor sobre o qual o usuário deseja aplicar a função  $\ln()$  e salva na variável  $x$ .

Como a série utilizada para aproximar  $\ln()$ , não converge para valores maiores do que 2, utiliza-se a seguinte propriedade das funções logarítmicas

$$\ln(x) = -\ln(1/x) \quad (5)$$

assim para  $x$  maior que 2 obtêm-se uma maneira de utilizar a série de modo que a mesma convirja. Contudo, nos teste, mostrou-se mais eficiente computacionalmente (tempo de execução do código menor) utilizar tal propriedade para todo  $x$  maior que 1, deste modo foi utilizada tal propriedade para todo  $x$  maior que 1.

O programa escreve na tela o resultado da expressão (6) e também escreve  $\ln(x)$  utilizando a função intrínseca do FORTRAN77 para fins de comparação.

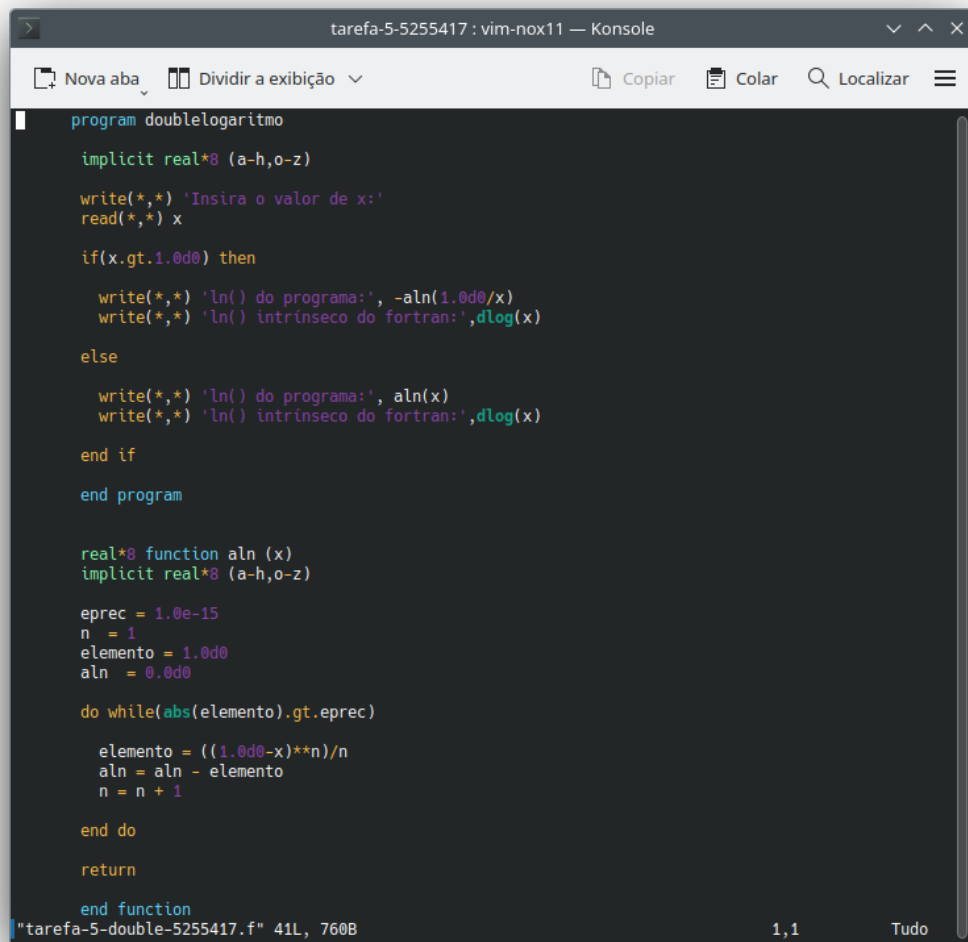
A função  $\text{aln}()$  é declarada após o fim do programa e consiste em definir uma precisão  $\text{eprec}$  que será o critério de para da série.

Inicia-se um loop que será encerrado caso o último elemento da série seja menor que eprec, tal elemento é dado por

$$\frac{(1-x)^n}{n} \quad (6)$$

de modo que n é incrementado por 1 a cada ciclo do loop e elemento é subtraído de aln, que é o valor que a função retornará.

Os dois códigos diferem apenas na precisão das variáveis e de eprec, segue em seguida o código para dupla precisão com eprec ajustada para o valor de maior precisão que surtiu efeito na comparação com a função dlog() intrínseca do FORTRAN77



```
tarefa-5-5255417: vim-nox11 — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ≡
program doublelogaritmo
  implicit real*8 (a-h,o-z)
  write(*,*) 'Insira o valor de x:'
  read(*,*) x
  if(x.gt.1.0d0) then
    write(*,*) 'ln() do programa:', -aln(1.0d0/x)
    write(*,*) 'ln() intrínseco do fortran:', dlog(x)
  else
    write(*,*) 'ln() do programa:', aln(x)
    write(*,*) 'ln() intrínseco do fortran:', dlog(x)
  end if
end program

real*8 function aln(x)
  implicit real*8 (a-h,o-z)
  eprec = 1.0e-15
  n = 1
  elemento = 1.0d0
  aln = 0.0d0
  do while(abs(elemento).gt.eprec)
    elemento = ((1.0d0-x)**n)/n
    aln = aln - elemento
    n = n + 1
  end do
  return
end function
"tarefa-5-double-5255417.f" 41L, 760B  1,1  Tudo
```

Comparando o resultado do código de precisão simples com a função intrínseca log() do FORTRAN77 percebe-se que os valores concordam até a quinta casa decimal, como mostra a tela de terminal abaixo

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-5-5255417> ./tarefa-5-5255417.exe
Insira o valor de x:
0.5
ln() do programa: -0.693139076
ln() intrínseco do fortran: -0.693147182
```

Ao fazer a comparação para dupla precisão, notou-se que qualquer valor de ordem menor que 10 a -15 não aumenta a precisão, assim obteve-se uma precisão de 15 casas decimais, como mostra a tela de terminal abaixo

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-5-5255417> ./tarefa-5-double-5255417.exe
Insira o valor de x:
0.5
ln() do programa: -0.69314718055994451
ln() intrínseco do fortran: -0.69314718055994529
```

## 1.6 Tarefa 6

Tarefa: escrever um programa em FORTRAN77 que, dado um N, encontra as N raízes da equação

$$(z - 2)^N = 3 \quad (7)$$

Código escrito:



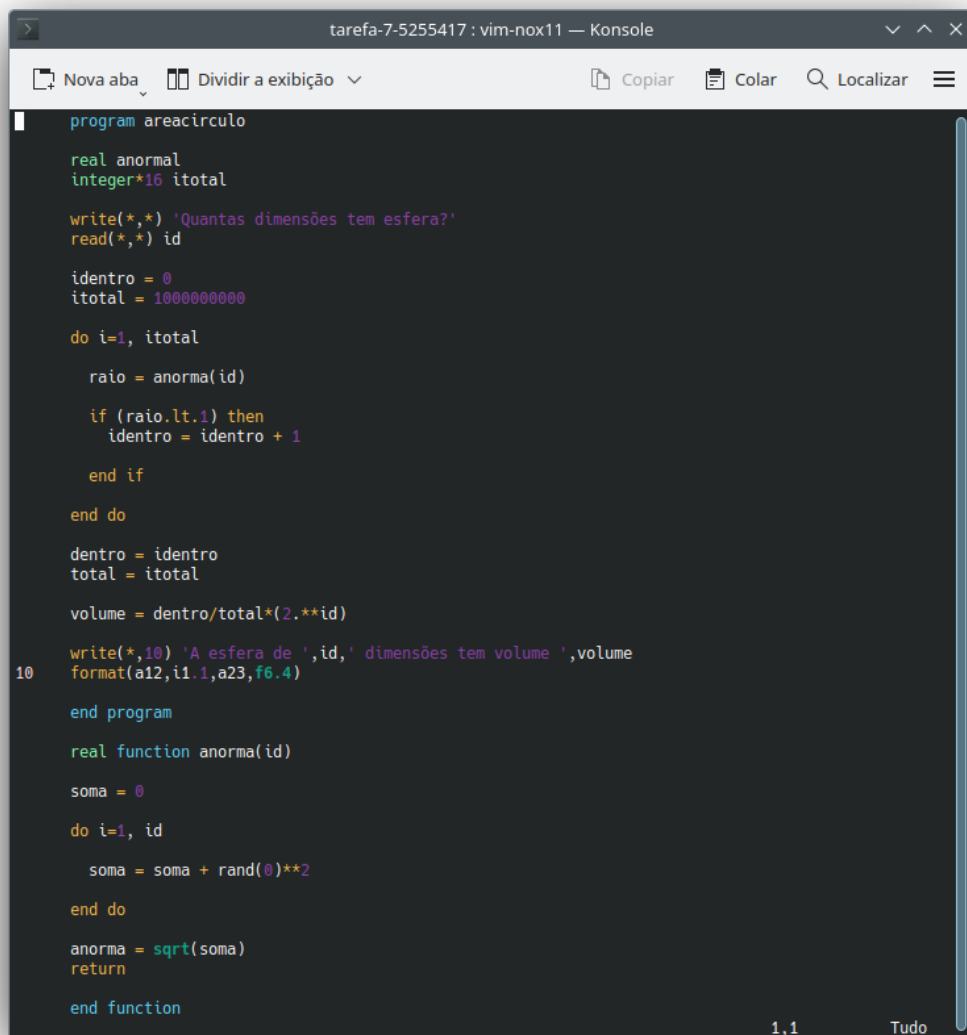
```
tarefa-6-5255417 : bash — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ≡
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
1
As raízes complexas são:
5.00, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
2
As raízes complexas são:
0.27,-0.00i
3.73, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
3
As raízes complexas são:
1.28, 1.25i
1.28,-1.25i
3.44, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
4
As raízes complexas são:
2.00, 1.32i
0.68,-0.00i
2.00,-1.32i
3.32, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
5
As raízes complexas são:
2.38, 1.18i
0.99, 0.73i
0.99,-0.73i
2.38,-1.18i
3.25, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> ./tarefa-6-5255417.exe
Insira o expoente N:
6
As raízes complexas são:
2.60, 1.04i
1.40, 1.04i
0.80,-0.00i
1.40,-1.04i
2.60,-1.04i
3.20, 0.00i
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-6-5255417> 
```

onde todas as respostas corresponderam com os valores calculados manualmente.

## 1.7 Tarefa 7

Tarefa: escrever um programa em FORTRAN77 que, dado um  $d$ , calcula o volume de uma esfera de  $d$  dimensões, utilizando a função `rand()` intrínseca do FORTRAN77

Código escrito:

A screenshot of a terminal window titled 'tarefa-7-5255417: vim-nox11 — Konsole'. The window displays Fortran code for a program named 'areacirculo'. The code includes variable declarations for 'anormal' (real) and 'itotal' (integer\*16), a loop to read dimensions and generate random points, a function 'anorma' to calculate the distance from the origin, and a final calculation of the sphere's volume based on the ratio of points inside to the total points. The code is color-coded with syntax highlighting. The terminal interface includes a menu bar with options like 'Nova aba', 'Dividir a exibição', 'Copiar', 'Colar', 'Localizar', and a status bar at the bottom showing '1,1' and 'Tudo'.

```
program areacirculo
  real anormal
  integer*16 itotal

  write(*,*) 'Quantas dimensões tem esfera?'
  read(*,*) id

  identro = 0
  itotal = 1000000000

  do i=1, itotal

    raio = anorma(id)

    if (raio.lt.1) then
      identro = identro + 1
    end if
  end do

  dentro = identro
  total = itotal

  volume = dentro/total*(2.**id)

  write(*,10) 'A esfera de ',id,' dimensões tem volume ',volume
10  format(a12,i1.1,a23,f6.4)

end program

real function anorma(id)
  soma = 0

  do i=1, id
    soma = soma + rand(0)**2
  end do

  anorma = sqrt(soma)
  return
end function
```

Descrição: o código pede o número  $d$  de dimensões da esfera que deseja-se calcular o volume, em seguida define quantos pontos aleatórios serão utilizados.

É iniciado um loop que chama a função `anormal`, passando o parâmetro "id" de quantas dimensões possui a esfera. Tal função, por sua vez, inicia uma soma do quadrado de id números aleatórios (entre 0 e 1) - obtidos com a função `rand()` - para ao fim retirar a raiz e assim retornar a norma do vetor formado por esses números, ou seja, sua distância para a origem. Caso tal distância seja maior do que 1 (raio da esfera unitária) o ponto é considerado dentro e incrementa-se a variável que faz esta contagem.

Ao fim da contagem, é feita a razão dos pontos dentro da esfera com o total de pontos, tal valor nos retorna o volume da esfera em um divisão de eixos dimensionais (no caso de duas dimensões, é o "volume" de um quadrante) que deve ser multiplicada por 2 elevado a  $d$  para que se obtenha o volume total da esfera.



Assim, o código calcula o volume, tal qual descrito anteriormente, e o retorna ao usuário.

Comparando os resultados - para  $d = 2, 3, 4$  variando o número de números aleatórios entre 1000, 1000000 e 1000000000 - com os resultados da expressão

$$V_d = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2} + 1)} R^d \quad (9)$$

obteve-se os seguintes resultados

$d=2$  (1000):

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
2
A esfera de 2 dimensões tem volume 3.1680
```

$d=2$  (1000000):

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
2
A esfera de 2 dimensões tem volume 3.1421
```

$d=2$  (1000000000):

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
2
A esfera de 2 dimensões tem volume 3.1416
```

Colocando  $d=2$  e  $R=1$  na equação (10), temos  $V_2 = \pi$  e nota-se que quanto mais pontos aleatórios, mais o valor converge para 3,14159...

$d=3$  (1000):

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
3
A esfera de 3 dimensões tem volume 4.2400
```

$d=3$  (1000000):

```
pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
3
A esfera de 3 dimensões tem volume 4.1880
```

$d=3$  (1000000000):

```

pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
3
A esfera de 3 dimensões tem volume 4.1888

```

Colocando  $d=3$  e  $R=1$  na equação (10), temos  $V_3 = \frac{3}{4}\pi$  e nota-se que quanto mais pontos aleatórios, mais o valor converge para 4,18879...

$d=$  (1000):

```

pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
4
A esfera de 4 dimensões tem volume 4.9440

```

$d=4$  (1000000):

```

pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
4
A esfera de 4 dimensões tem volume 4.9236

```

$d=4$  (1000000000):

```

pedro@Pedro-Lenovo:~/USP/projetos/projeto-1-5255417/tarefa-7-5255417> ./tarefa-7-5255417.exe
Quantas dimensões tem esfera?
4
A esfera de 4 dimensões tem volume 4.9349

```

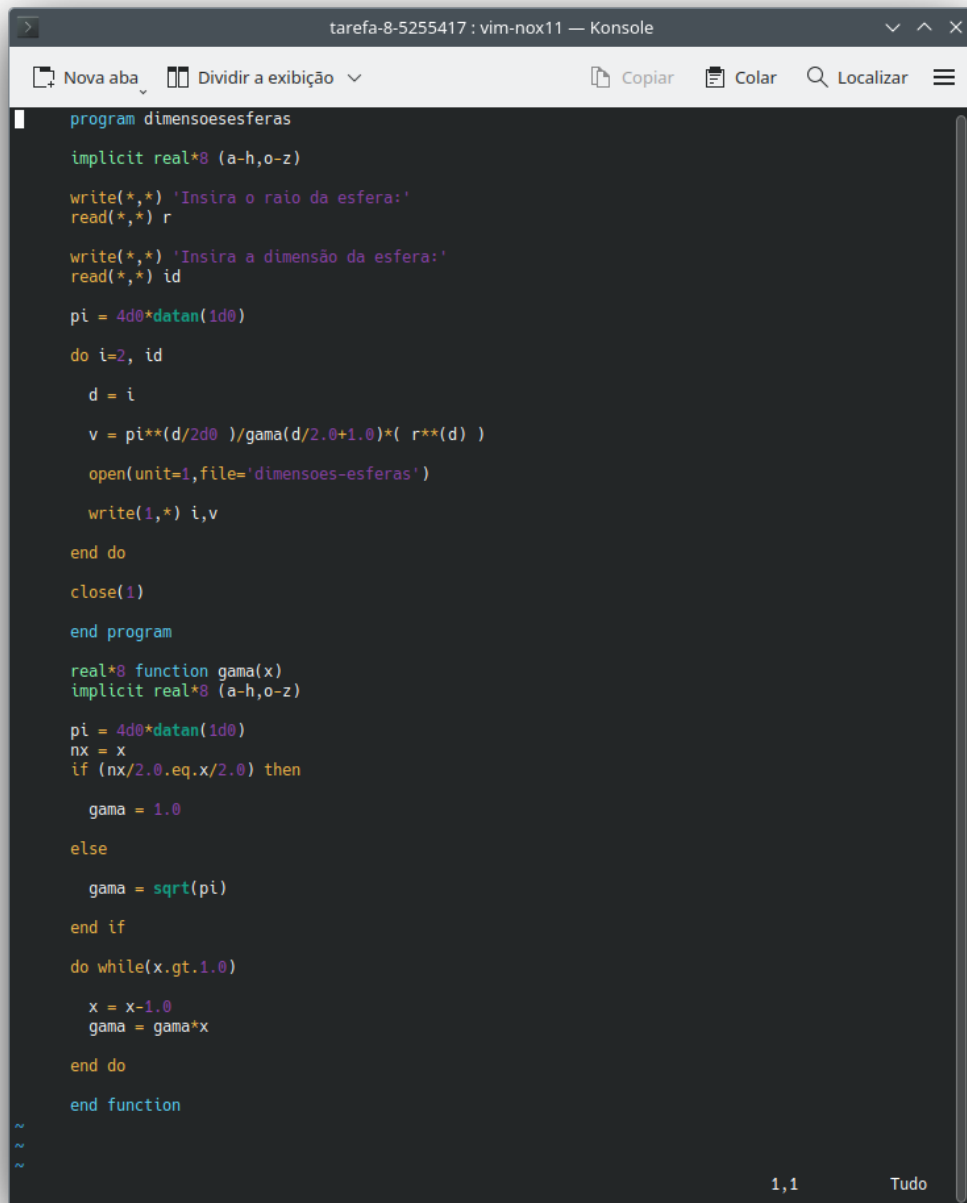
Colocando  $d=4$  e  $R=1$  na equação (10), temos  $V_4 = \frac{3}{4}\pi$  e nota-se que quanto mais pontos aleatórios, mais o valor converge para 4,9348...

Assim, é notório que quanto mais pontos são utilizados, mais o valor calculado se aproxima do valor da expressão (10).

## 1.8 Tarefa 8

Tarefa: escrever um código em FORTRAN77 que, dado  $d$  dimensões e um raio  $r$  -utilizando a expressão (10) -, calcula o volume das esferas de 2 até  $d$  dimensões e escreve em um arquivo chamado "dimensoes-esferas"

Código escrito:

A screenshot of a terminal window titled 'tarefa-8-5255417: vim-nox11 — Konsole'. The window contains Fortran code for calculating the volume of spheres. The code defines a program 'dimensoes-esferas' that takes a radius 'r' and a dimension 'id' as input. It calculates the volume 'v' for dimensions 2 and 3 using a gamma function. The gamma function is defined as a separate function 'gama(x)' which uses a loop to calculate the gamma value for non-integer values of x. The program also writes the results to a file named 'dimensoes-esferas'.

```
program dimensoes-esferas
  implicit real*8 (a-h,o-z)

  write(*,*) 'Insira o raio da esfera:'
  read(*,*) r

  write(*,*) 'Insira a dimensão da esfera:'
  read(*,*) id

  pi = 4d0*datan(1d0)

  do i=2, id
    d = i

    v = pi**(d/2d0 )/gama(d/2.0+1.0)*( r**(d) )

    open(unit=1,file='dimensoes-esferas')

    write(1,*) i,v
  end do

  close(1)

end program

real*8 function gama(x)
  implicit real*8 (a-h,o-z)

  pi = 4d0*datan(1d0)
  nx = x
  if (nx/2.0.eq.x/2.0) then

    gama = 1.0
  else

    gama = sqrt(pi)

  end if

  do while(x.gt.1.0)

    x = x-1.0
    gama = gama*x

  end do

end function
```

Descrição: o código pede o raio e até quantas dimensões deseja-se calcular os volumes das esferas e salva ambos. Após isso, declara pi como 4 vezes o arcotangente de 1 e inicia um loop calculando o volume da esfera (através da expressão (10)) de 2 até d dimensão salvando os resultados em um arquivo chamado "dimensoes-esferas"

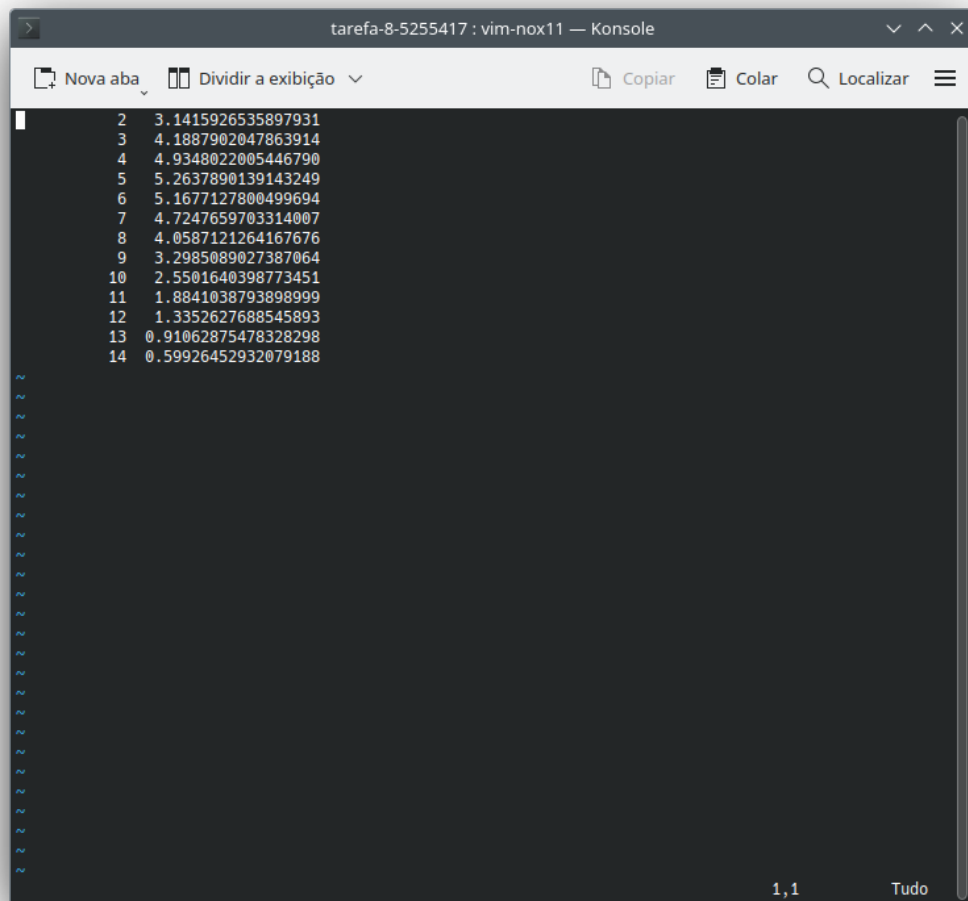
A função  $\Gamma$  declarada após o fim do código utiliza a propriedade

$$\Gamma(x + 1) = x\Gamma(x) \quad (10)$$

de modo que o resultado da função equivale a calcular o fatorial de  $x-1$  - onde  $x$  é o valor recebido pela função (e multiplicá-lo por  $\sqrt{\pi}$  caso a  $x$  não seja inteiro - uma

vez que sabe-se que  $x$ , para o caso trabalhado,  $x$  só podera assumir valores inteiros ou múltiplos de  $1/2$  - já que  $\Gamma(1/2) = \sqrt{\pi}$ .

Utilizando o valor de raio 1 e indo até a 14<sup>a</sup> dimensão, o arquivo resultante concordou com a expressão (10), veja

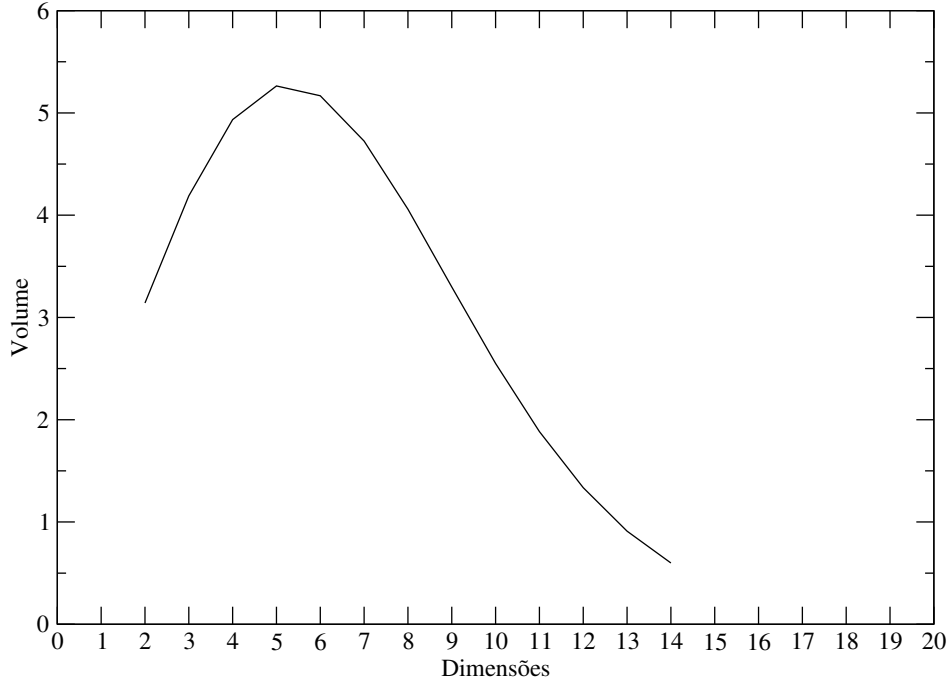


The image shows a terminal window titled "tarefa-8-5255417 : vim-nox11 — Konsole". The window contains a list of numerical values corresponding to dimensions 2 through 14. The values are displayed in a monospaced font. The terminal interface includes a menu bar with options like "Nova aba", "Dividir a exibição", "Copiar", "Colar", "Localizar", and a status bar at the bottom showing "1,1" and "Tudo".

Dimensão	Valor
2	3.1415926535897931
3	4.1887902047863914
4	4.9348022005446790
5	5.2637890139143249
6	5.1677127800499694
7	4.7247659703314007
8	4.0587121264167676
9	3.2985089027387064
10	2.5501640398773451
11	1.8841038793898999
12	1.3352627688545893
13	0.91062875478328298
14	0.59926452932079188

Plotando tais valores utilizando xmgrace obteve-se

Figura 1: Gráfico: Volume X Dimensões



É perceptível que o volume da esfera tem um máximo na 5ª dimensão e depois passa a ser cada vez menor. O motivo disso? Uma boa pergunta para fazer a Deus...

## 2 Perguntas

### 2.1 A

Se um cubo de dimensão  $d$  tem lado  $1m$ , ele terá volume  $1m^d$ . Quantas vezes este volume será maior que o da esfera nesta dimensão? O que acontece quando  $d \rightarrow \infty$ ?

Fazendo a razão da expressão que dá o volume do cubo, pela expressão (10)

$$\frac{1m^d}{\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} 1m^d} \quad (11)$$

simplificando, obtemos

$$\frac{\Gamma(\frac{d}{2} + 1)}{\pi^{d/2}} \quad (12)$$

Fazendo,  $d \rightarrow \infty$

$$\lim_{x \rightarrow \infty} \frac{\Gamma(\frac{d}{2} + 1)}{\pi^{d/2}} = \infty \quad (13)$$

O que significa que caberão infinitas esferas em um cubo de infinitas dimensões.

## 2.2 B

Se o volume de uma proteína em  $d$  dimensões é  $1\mu^d$  e o volume de um átomo neste mundo for  $1\text{\AA}^d$  e se tipicamente um volume macroscópico for  $1\text{mm}^d$  qual deverá ser a ordem típica da número de Avogrado neste mundo  $d$ -dimensional?

Dividindo o tamanho macroscópico pelo tamanho de um átomo, teremos

$$\frac{1\text{\AA}^d}{1\text{mm}^d} = 10^{7d} \quad (14)$$

Comparando com a ordem conhecida no nosso mundo, em que  $d = 3$  e o número de Avogrado é da ordem de  $10^{-23}$ , utilizando a relação (14) teremos uma ordem de  $10^{-21}$  que é próxima da verdadeira ordem.