



**IFSC** **UNIVERSIDADE**  
**DE SÃO PAULO**  
Instituto de Física de São Carlos

November 23, 2022

## **Contents**

## section 0 Tarefa 1

- A área do tórus é dada por:  $A = (2\pi R)(2\pi r)$
- O volume do tórus é dado por:  $V = (\pi r^2)(2\pi R)$

Onde  $R$  é o raio externo e  $r$  é o raio interno.

```
[frame,linenos=true]fortran ! Tarefa 01 ! Calcula área e volume de um
tórus a partir de raios dados. write(*,*) "Digite os valores dos raios (interno,
externo):" read(*,*) ri, re pi = acos(-1e0)
aArea = 4.e0 * pi ** 2 * re * ri aVolume = (pi * ri ** 2) * (2 * pi * re)
write(*,*) "Area = ", aArea write(*,*) "Volume = ", aVolume end
```

## Resultados

```
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa1$ ./tarefa1.exe
Digite os valores dos raios (interno, externo):
3.1 3.4
Area =      416.102570
Volume =     644.958923
```

```
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa1$ ./tarefa1.exe
Digite os valores dos raios (interno, externo):
2.3 33
Area =      2996.41187
Volume =     3445.87402
```

## section 0 Tarefa 2

### Explicação

Sejam  $u = (x_1, y_1, z_1)$ ,  $v = (x_2, y_2, z_2)$  e  $w = (x_3, y_3, z_3)$ . Quero calcular a área lateral e volume do paralelepípedo formado pelos vetores  $u, v$  e  $w = (x_3 - x_2, y_3 - y_2, z_3 - z_2)$ . Para o cálculo da área temos

$$A_L = 2 \cdot [\langle u, v \rangle + \langle u, w \rangle + \langle v, w \rangle]$$

$$A_L = 2 \cdot [(x_1x_2 + y_1y_2 + z_1z_2) + (x_3 - x_2) \cdot (x_1 + x_2) + (y_3 - y_2) \cdot (y_1 + y_2) + (z_3 - z_2) \cdot (z_1 + z_2)]$$

Para o cálculo do volumes usamos o produto misto entre  $u, v$  e  $w$ , então  $V = [u, v, w] = \langle u \wedge v, w \rangle$ . O produto vetorial  $u \wedge v$  é  $(y_1z_2 - y_2z_1, x_2z_1 - x_1z_2, x_1y_2 - x_2y_1)$ . Temos então o volume dado por

$$V = (x_3 - x_2)(y_1z_2 - y_2z_1) + (y_3 - y_2)(x_2z_1 - x_1z_2) + (z_3 - z_2)(x_1y_2 - x_2y_1)$$

## Código

```
[frame,linenos=true]fortran ! Tarefa 02 ! Dados 3 vetores (u, v, w)
calcula o volume do paralelepipedo das arestas ! definidas por u, v e w -
v. dimension u(1:3), v(1:3), w(1:3) dimension aVec(1:3) write(*,*)"Digite as
coordenadas de cada vetor" read(*,*) u(1), u(2), u(3) read(*,*) v(1), v(2),
v(3) read(*,*) w(1), w(2), w(3) w(1) = w(1) - v(1) w(2) = w(2) - v(2) w(3)
= w(3) - v(3)
! A = 2 ( A1 + A2 + A3) aVec(1) = (u(2)*v(3)-v(2)*u(3)) aVec(2) =
(v(1)*u(3)-u(1)*v(3)) aVec(3) = (u(1)*v(2)-v(1)*u(2))
a1 = aVec(1)**2 + aVec(2)**2 + aVec(3)**2 a1 = sqrt(a1)
a2 = (u(2)*w(3) - u(1)*w(2))**2 + (u(3)*w(1) - u(1)*w(3))**2 + +
(u(1)*w(2) - u(2)*w(1))**2 a2 = sqrt(a2)
a3 = (v(2)*w(3)-v(3)*w(2))**2 + (v(3)*w(1)-v(1)*w(3))**2 + + (v(1)*w(2)
- v(2)*w(1))**2 a3 = sqrt(a3)
area = 2 * (a1 + a2 + a3) write(*,*) "Area do paralelepipedo: ", area
volume = aVec(1) * w(1) + aVec(2) * w(2) + aVec(3) * w(3)
write(*,*) "Volume do paralelepipedo: ", abs(volume)
```

## Resultados

```
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa2$ ./tarefa2.exe
Digite as coordenadas de cada vetor
1 0 0
0 1 0
0 1 1
Area do paralelepipedo:      6.00000000
Volume do paralelepipedo:    1.00000000
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa2$ ./tarefa2.exe
Digite as coordenadas de cada vetor
2 3 1
2 2 1
0 1 3
Area do paralelepipedo:      42.1373100
Volume do paralelepipedo:    6.00000000
```

## section 0 Tarefa 3

Foi definido como valor máximo de entrada  $N = 100$ , ou seja, a maior quantidade possível de valores em um arquivo deve ser 100. O input  $M$  deverá ser de tal modo que  $M \leq N$ .

Os dados utilizados para testar o programa estão no arquivo `input-tarefa3.dat` e foram gerados utilizando a rotina abaixo, que gera o arquivo que contém os  $N$  números aleatórios, dos quais  $M$  deles serão ordenados. Essa rotina está implementada no arquivo `./tarefa3/gerar_numeros.f`.

O programa principal faz a leitura do arquivo `./tarefa3/input-tarefa3.dat` e ordena os  $M$  primeiros menores números, após isso escreve os  $M$  números ordenados no arquivo `./tarefa3/output-tarefa3.dat`.

```
[frame,linenos=true]fortran ! Tarefa 03 parameter(N = 100) dimension
rList(N)
    in = 10 ! unidade para arquivo de entrada. iout = 20 ! unidade para
arquivo de saída open(unit=in, file="input-tarefa3.dat") open(unit=iout,
file="output-tarefa3.dat")
    write(*,*) "Quantidade de números a serem ordenados (M 100):" read(*,*)
M
    ! Lê o arquivo com os N números aleatórios do i = 1, N read(in, *,
end=1) rList(i) end do 1 continue
    Na sequência foi implementado a ordenação dos M menores números do
vetor em ordem decrescente utilizando o algoritmo bubble sort.
    [frame,linenos=true]fortran ! Implementacao do algoritmo bubble sort
do i = 1, M do j = N, 2, -1 if(rList(j) < rList(j-1)) then tmp = rList(j)
rList(j) = rList(j-1) rList(j-1) = tmp end if end do end do
    Após a ordenação os  $M$  primeiros menores números foi então escrito no
arquivo ./tarefa3/output-tarefa3.dat
    [frame,linenos=true]fortran ! Salva no arquivo "output-tarefa3.dat" do i
= 1, M write(iout, *) rList(i) end do write(iout, 999) M 999 format(i2 , '
numeros.')
    close(in) close(iout)
```

## Resultados

Pode ser testado o programa mudando os valores no arquivo `input-tarefa3.dat`, rodando o programa no terminal e depois analisando o arquivo gerado `output-tarefa3.dat`, que conterá os  $M$  primeiros menores valores do arquivo original, em ordem.

## Exemplo

```
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ ./tarefa3.exe
```

```
Quantidade de números a serem ordenados (M 100):
```

```

15
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ cat output-
tarefa3.dat
0.108032703
0.108789563
0.112783670
0.117679000
0.124440551
0.194475591
0.202756226
0.211118400
0.211621881
0.211978197
0.219443142
0.246241093
0.246837378
0.255061328
0.257259607
15 numeros.
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ ./tarefa3.exe
Quantidade de números a serem ordenados (M 100):
10
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ cat output-
tarefa3.dat
0.108032703
0.108789563
0.112783670
0.117679000
0.124440551
0.194475591
0.202756226
0.211118400
0.211621881
0.211978197
10 numeros.
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ ./tarefa3.exe
Quantidade de números a serem ordenados (M 100):
3
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa3$ cat output-
tarefa3.dat

```

```
0.108032703
0.108789563
0.112783670
3 numeros.
```

## section 0 Tarefa 4

Queremos calcular a série

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

que pode ser escrita como a série de potências

$$\cos x = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n}}{(2n)!}$$

Primeiramente, fazemos alguns ajustes que facilitam a programação do cálculo da série, manipulando a equação temos

$$\cos x = 1 + \sum_{i=1}^{\infty} (-1)^i \cdot \frac{x^{2i}}{(2i)!}$$

onde  $i$  será a iteração do loop. Escrevemos também o termo  $(2i)!$  como  $i(i+1)$ , que será multiplicado sempre pelo termo da iteração anterior, dessa forma todo o programa fica mais otimizado, com todo o cálculo da série sendo feito em um mesmo loop.

Teremos então, para o cálculo da série, o código abaixo: [frame,linenos=true]fortran ! Tarefa 4 ! Implementação da série de potências ! cos(x) = (-1)<sup>n</sup> \* (x<sup>2n</sup>)/(2n)!

```
double precision eprecd double precision cossd double precision accd double precision tmpd double precision xd double precision axd
eprecd = 1e-16 accd = eprecd * 2
eprec = 1e-5 acc = eprec * 2
ifat = 1
coss = 1e0 tmp = coss
write(*,*) "Informe x em radianos:" read(*,*) xd
x = xd ax = x**2
i = 1 do while (eprec <= acc) ifat = (i * (i + 1))
tmp = tmp * (-1) * ax / ifat coss = coss + tmp
acc = abs(tmp) - eprec
```

```

i = i+2 end do
cosd = 1d0 tmpd = cosd
axd = xd**2 ifat = 1 i = 1
do while (eprecd <= accd) ifat = (i * (i + 1))
tmpd = tmpd * (-1) * axd / ifat cosd = cosd + tmpd
accd = abs(tmpd) - eprecd
i = i+2 end do
write(*,*) "===== " write(*,*) "= Resultados
=" write(*,*) "===== "
write(*,*) "Precisão simples" write(*,*) "cos(x) = ", coss
errs = abs(cos(x)) - abs(coss) write(*,*) "Erro: ", errs
write(*,*) "_____"
write(*,*) "Precisão dupla" write(*,*) "cos(x) = ", cosd
errd = abs(dcos(xd)) - abs(cosd) write(*,*) "Erro: ", errs
end

```

## Resultados

```

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa4$ ./tarefa4.exe
Informe x em radianos:
- 1
=====
= Resultados =
=====
Precisão simples
cos(x) = 0.540302277
Erro: 0.00000000
-----
Precisão dupla
cos(x) = 0.54030230586813965
Erro: 0.00000000
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa4$ ./tarefa4.exe
Informe x em radianos:
3.14159265
=====
= Resultados =
=====
Precisão simples
cos(x) = -0.999999881
Erro: 1.19209290E-07

```

```
-----
Precisão dupla
cos(x) = -1.00000000000000002
Erro: 1.19209290E-07
```

## section 0 Tarefa 5

### Paridade de permutação

Seja  $\sigma$  uma permutação de  $n$  elementos, o par  $[i, j]$ , onde  $1 < i < j < n$  é chamado de inversão se  $i < j$  e  $\sigma(i) > \sigma(j)$ . Uma permutação é um mapa do tipo  $\sigma : \mathbb{N} \mapsto \mathbb{N}$ . A paridade de uma permutação  $\sigma$  é dada por  $(-1)^\lambda$ , onde  $\lambda$  é o número de inversões em  $\sigma$ . Permutações pares ocorrem se  $\lambda$  é par. O sinal da paridade é representado por  $P(\sigma)$ .

### Exemplo: cálculo de paridades

Para  $n = 3 \Rightarrow n! = 6$ , ou seja, o conjunto de permutações  $S = \{123, 132, 213, 231, 312, 321\}$ .

- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{smallmatrix}\right) \Rightarrow N = 0 \Rightarrow P(\sigma) = (-1)^N = (-1)^0 = 1$
- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{smallmatrix}\right) \Rightarrow (2, 3) \Rightarrow N = 1 \Rightarrow P(\sigma) = -1$
- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{smallmatrix}\right) \Rightarrow (1, 2) \Rightarrow N = 1 \Rightarrow P(\sigma) = -1$
- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{smallmatrix}\right) \Rightarrow (1, 3), (2, 3) \Rightarrow N = 2 \Rightarrow P(\sigma) = 1$
- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{smallmatrix}\right) \Rightarrow (1, 2), (1, 3) \Rightarrow N = 2 \Rightarrow P(\sigma) = 1$
- $\sigma\left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{smallmatrix}\right) \Rightarrow (1, 2), (1, 3), (2, 3) \Rightarrow N = 3 \Rightarrow P(\sigma) = -1$

Podemos então colocar as permutações de  $n$  em com suas respectivas paridades

### Solução

Para o caso de  $n+1$ , com  $n = 3$  temos 24 permutações, podemos escrever como  $4 \cdot 3!$ , ou seja, 4 conjuntos de permutações a partir das permutações de 3. Então podemos escrever esses conjuntos levando em conta que o 4 vai ocupar uma posição diferente em cada grupo:



Table 1: Permutações para  $n = 3$  e suas paridades.

$\sigma$	$P(\sigma)$
1 2 3	1
1 3 2	-1
2 1 3	-1
2 3 1	1
3 1 2	1
3 2 1	-1

1 2 3 4	1 2 4 3	...	4 1 2 3
1 3 2 4	1 3 4 2	...	4 1 3 2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Note que, o número de linhas é  $3!$  e de colunas são  $n + 1$ . A análise das paridades pode ser feita apenas observando as colunas. O valor  $n + 1$  começa na coluna mais à direita, fazendo então as comparações que definem paridade vemos que para a primeira coluna todas paridades se mantêm, então as paridades são as mesmas que para  $3!$ . Para a segunda coluna vemos que em todas linhas temos que  $\sigma(3) = 4 > \sigma(4)$ , onde 3 e 4 são os índices (ou posição na coluna), então o número de inversões será ímpar, pois a paridade será  $(-1)^1$ . Para o caso seguinte, de 2 temos que  $\sigma(2) > \sigma(3)$  e  $\sigma(2) > \sigma(4)$ , pois 4 é maior que qualquer elemento (1, 2 ou 3) nas colunas seguintes. Então a paridade será  $(-1)^2 = 1$ .

Generalizando, seja  $j > 1$  o índice da coluna, a nova paridade será dada por  $(-1)$ . Segue a descrição do algoritmo em pseudo-código:

Gera  $(n + 1)$  permutações e suas respectivas paridades.  $n_{Perm} \leftarrow$  lista com  $P = n!$  permutações e suas paridades  $j \leftarrow n_{Cols} + 1$  to 1, step =  $-1$   $j < n_{Col} + 1$   $i \leftarrow 1$  to  $n_{Linhas}$ , step =  $-1$  aux =  $n_{Perm}(i, j)$   $n_{Perm}(i, j) = n_{Perm}(i, j-1)$   $n_{Perm}(i, j-1) =$  aux  $n_{Perm}(i, n_{Col}+1) = (-1) \cdot n_{Perm}(i, n_{Col}+1)$  Armazena(aux $_{Perm}$ )

### Prova de corretude do algoritmo

No algoritmo,  $n_{Perm}$  é uma matriz de dimensões  $n! \times (n + 1)$ , todas as linhas da coluna  $n$  possui inicialmente valores iguais à  $n$  e as linhas da coluna  $(n + 1)$  as paridades de cada permutação.

- **I)** O loop externo itera até que  $j \leftarrow 1$ , ou seja, essa é a sua condição de parada.

- **II)** A condição  $j < n_{\text{Col}} + 1$  garante que as colunas só vão começar a ser trocadas a partir da segunda iteração.
- **III)** O loop interno vai iterar até que  $i \leftarrow n_{\text{Linhas}}$ , então, por todas as linhas da matriz.
  - No loop interno é feita a troca de posições das colunas, ou seja, para  $j = n_{\text{Col}}$  teremos o seguinte loop

1	2	3	4		1	2	4	3		1	4	2	3		4	1	2	3
1	3	2	4	$\xrightarrow{j \leftarrow j-1}$	1	3	4	2	$\xrightarrow{j \leftarrow j-2}$	1	4	3	2	$\xrightarrow{j \leftarrow j-3}$	4	1	3	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$

Como esse procedimento é feito para todas colunas (exceto a das paridades), no fim do loop teremos permutações de  $(n + 1)$  com  $n!$  linhas. A paridade de cada permutação, ou seja, cada linha da matriz, é calculada por  $(-1) \cdot n_{\text{Perm}}(j, n_{\text{Col}} + 1)$ , a nova paridade é então um produto da antiga por  $(-1)$ , como descrito acima na descrição matemática.

- **IV)** Quando o loop interno finaliza tem-se uma matriz com as permutações de  $n!$  com  $(n + 1)$  na posição  $j$ . É então armazenado esse valor na num arquivo de saída.
- **V)** Com o fim do loop externo temos um arquivo de saída com  $(n + 1)!$  permutações e suas devidas paridades.

Portanto, esse algoritmo computa as  $(n + 1)!$  permutações e suas respectivas paridades.  $\square$

## Código em Fortran

### Preparação dos dados

Primeiramente é feita a declaração das variáveis e realizada a leitura do arquivo `input-tarefa3.dat`, que contém as permutações para  $n = 3$  e suas respectivas paridades. [frame,linenos=true]fortran ! Lendo o arquivo 'input-tarefa3.dat' parameter(nRow = 6, nCol = 4) dimension iPerm(nRow,nCol), nAuxPerm(nCol, nRow) dimension nPerm(nRow, nCol + 1) dimension iauxPerm(nRow, nCol + 1)

open(unit=10, file="input-tarefa3.dat") open(unit=20, file="output-tarefa3.dat")

read(in, \*) nAuxPerm iPerm = transpose(nAuxPerm) Esse bloco de código gera a matriz com as  $n + 1$  colunas além da coluna com as paridades. [linenos,firstnumber=12,frame,linenos=true]fortran ! Gera a matrix  $n! \times (n + 1)$  i = 1 do while(i < nCol) do j = 1, nRow nPerm(j, nCol + 1) = iPerm(j, nCol) nPerm(j, nCol) = nCol nPerm(j, i) = iPerm(j, i) end do i = i + 1 end do

### Implementação do algoritmo (??) em Fortran

As linhas 31 à 33 não são equivalentes ao termo "Armazena( $n_{Perm}$ )", presente na descrição do algoritmo. Elas escrevem no arquivo **output-tarefa5.dat**. [linenos,firstnumber=21,frame,linenos=true]fortran ! Por fim escreve no arquivo 'output-tarefa5.dat' do j = nCol, 1, -1 iauxPerm = nPerm do i = 1, nRow iaux = iauxPerm(i, j) iauxPerm(i, j) = iauxPerm(i, nCol) iauxPerm(i, nCol) = iaux

iauxPerm(i, nCol + 1) = (-1)\*\*(j + 2)\*iauxPerm(i, nCol + 1) end do do k = 1, nRow write(20,\*) (iauxPerm(k,ih), ih=1, nCol + 1) end do end do close(10) close(20) end

### Resultados

A tabela abaixo é o resultado para o input de permutações da tabela ??.

$i$	$\sigma$	$P(\sigma)$
1	1 2 3 4	1
2	2 1 3 4	-1
3	1 3 2 4	-1
4	2 3 1 4	1
5	3 1 2 4	1
6	3 2 1 4	-1
7	1 2 4 3	-1
8	2 1 4 3	1
9	1 3 4 2	1
10	2 3 4 1	-1
11	3 1 4 2	-1
12	3 2 4 1	1
13	1 4 2 3	1
14	2 4 1 3	-1
15	1 4 3 2	-1
16	2 4 3 1	1
17	3 4 1 2	1
18	3 4 2 1	-1
19	4 1 2 3	-1
20	4 2 1 3	1
21	4 1 3 2	1
22	4 2 3 1	-1
23	4 3 1 2	-1
24	4 3 2 1	1

Foi gerado o código executável para os casos de  $n = 3, 4, 5$  e  $6$  e disponibilizados na pasta `./tarefa5/`, os arquivos das permutações  $n = 4, 5$  e  $6$  foram usados para o cálculo do determinante da tarefa ??.

## Replicabilidade e generalização do código em Fortran

Para inputs diferentes de  $n = 3$ , é necessário alterar o arquivo `tarefa5.f`. Esse código foi escrito pensando na implementação específica de  $n = 3$ , escrever um para um  $n$  qualquer envolveria um código com algumas especificidades maiores, por exemplo, como o `fortran77` magentanão possui alocação dinâmica de memória, seria necessário programar condições específicas para que o algoritmo não tentasse acessar índices além do número de elementos que a matriz possui.

Então, para casos gerais, é necessário alterar a linha um do programa, alterar o número de linhas,  $n!$ , e o número de colunas  $n + 1$ , além, é claro, de

substituir o arquivo `input-tarefa3.dat` pelo arquivo com as permutações de  $n!$ .

## section 0 Tarefa 6

### Definição do determinante através de permutações

Seja a matriz quadrada  $A = (a_{ij})$ , de dimensões  $n \times n$ , podemos calcular o seu determinante pela fórmula

$$\det A = \sum_{\sigma \in S} P(\sigma) \cdot A_{1,\sigma_1} \cdot A_{2,\sigma_2} \cdot \dots \cdot A_{n,\sigma_n}$$

Onde  $\sigma$  é uma permutação em um conjunto  $S$  de permutações e  $P(\sigma)$  é a paridade de cada permutação.

### Exemplo de como construir o somatório

Seja  $n = 3$ , sabemos que uma matriz  $A$ ,  $n \times n$ , tem determinante igual à  $\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$ . Mostramos a então que o determinante calculado pela definição é equivalente:

- $\sigma(1) = \{1, 2, 3\}, P(1) = 1$
- $\sigma(2) = \{1, 3, 2\}, P(2) = -1$
- $\sigma(3) = \{2, 1, 3\}, P(3) = -1$
- $\sigma(4) = \{2, 3, 1\}, P(4) = 1$
- $\sigma(5) = \{3, 1, 2\}, P(5) = 1$
- $\sigma(6) = \{3, 2, 1\}, P(6) = -1$

Os elementos de cada permutação vão servir para selecionar quais os índices dos elementos em cada iteração do somatório. Cada permutação  $\sigma$  define as posições dos elementos em  $A$  que devem ser multiplicados, fazendo o somatório temos

$$\det A = P(1) \cdot A_{1,1}A_{2,2}A_{3,3} + P(2) \cdot A_{1,1}A_{2,3}A_{3,2} + P(3) \cdot A_{1,2}A_{2,1}A_{3,3} + P(4) \cdot A_{1,2}A_{2,3}A_{3,1} + P(5) \cdot A_{1,3}A_{2,1}A_{3,2} + P(6) \cdot A_{1,3}A_{2,2}A_{3,1}$$

$$\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

## Código implementado em Fortran

```
[frame,linenos=true]fortran ! Calcula determinante pela definição pa-
parameter(nRow = 24, nCol = 5) dimension nPerm(nRow, nCol), nAux-
Perm(nCol, nRow) parameter(nMatrix = 4) dimension rMatrix(nMatrix,
nMatrix)
open(10, file="permutacoes-n4.dat") open(20, file="matriz-4x4.dat")
read(10, *) nAuxPerm nPerm = transpose(nAuxPerm)
read(20, *) rMatrix rMatrix = transpose(rMatrix)
det = 0e0 do i = 1, nRow
signal = nPerm(i, nCol) aux = 1e0 do j = 1, nCol - 1 aux = aux *
rMatrix(j, nPerm(i, j)) end do det = det + signal * aux
end do
write(*, *) "Determinante = ", det end
```

## Resultados

Na pasta que contém o código tem também os arquivos de entrada para as matrizes às quais o cálculo do determinante é realizado e também as respectivas matrizes de permutações. Foram gerados 3 arquivos executáveis, `determinante-4x4.exe`, `=determinante-5x5.exe` e `determinante-6x6.exe`, cada um realiza o calculo do determinante para matrizes de dimensões  $4 \times 4$ ,  $5 \times 5$  e  $6 \times 6$ , respectivamente. Para se alterar a matriz basta modificar os arquivos `matriz-4x4.dat`, `matriz-5x5.dat` e `matriz-6x6.dat`, também presentes na pasta `./tarefa6/`.

Abaixo alguns exemplos de uso dos programas:

```
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ cat matriz-
4x4.dat
3 2 1 2
4 4 1 2
3 0 2 7
2 1 4 4
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ ./determinante-
4x4.exe
Determinante =    -47.0000000
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ cat matriz-
5x5.dat
2 1 1 4 2
9 3 9 0 3
1 1 9 6 3
```

```

1 3 5 5 4
1 9 2 2 2
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ ./determinante-
5x5.exe
Determinante = -4068.00000
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ cat matriz-
6x6.dat
1 2 3 4 50 6
11 2 3 14 5 2
3 29 3 4 15 6
4 1 13 4 5 6
9 4 32 4 59 6
2 0 -3 4 -1 46
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa6$ ./determinante-
6x6.exe
Determinante = -61599620.0

```

## section 0 Tarefa 7

O sistema linear do tipo  $AX = Y$  pode ser resolvido usando determinantes usando a regra de Cramer, ou seja  $X_i = \frac{\det A_i}{\det A}$ , onde a matriz  $A_i$  tem a coluna  $i$  trocada pelo vetor  $Y$ . Usando o a definição de determinante explicada na tarefa anterior podemos escrever um programa para resolver um sistema desse tipo.

Segue abaixo a implementação do código para o cálculo da solução de um sistema linear para matriz de ordem  $4 \times 4$ , com as matrizes  $A$  e  $Y$  dadas: [frame, linenos=true]fortran ! Calcula as soluções de um sistema linear !  $AX = Y$  com  $A$  e  $Y$  dados. parameter(nRow = 24, nCol = 5) parameter(nMatrix = 4)

```

dimension nPerm(nRow, nCol), nAuxPerm(nCol, nRow) dimension rMa-
trix(nMatrix, nMatrix), auxMatrix(nMatrix, nMatrix)
dimension Y(1:nMatrix)
open(30, file='y.dat') read(30, *) Y ! print *, Y close(30)
open(10, file="permutacoes-n4.dat") open(20, file="matriz-4x4.dat")
read(10, *) nAuxPerm nPerm = transpose(nAuxPerm)
read(20, *) rMatrix rMatrix = transpose(rMatrix)
detA = 0e0 call det(nRow, nCol, nPerm, nMatrix, rMatrix, detA)
write(*, *) "Matriz solução X: " do j = 1, nMatrix auxMatrix = rMatrix
do i = 1, nMatrix auxMatrix(i, j) = Y(i) end do

```

```

    detXj = 0e0 call det(nRow,nCol,nPerm,nMatrix,auxMatrix,detXj) write(*,
*) detXj / detA
    end do
    close(10) close(20) end

```

A subrotina **det** é a implementação do cálculo do determinante pela definição feito na tarefa ?? adaptada para este programa. [frame,linenos=true]fortran subroutine det(nRow,nCol,nPerm,nMatrix,rMatrix, detx) dimension nPerm(nRow, nCol), nAuxPerm(nCol, nRow) dimension rMatrix(nMatrix, nMatrix)

```

    do i = 1, nRow signal = nPerm(i, nCol) aux = 1e0 do j = 1, nCol - 1
aux = aux * rMatrix(j, nPerm(i, j)) end do detx = detx + signal * aux end
do return end

```

Na pasta **./tarefa3/** consta 3 arquivos **.exe**, executáveis que podem ser usados para calcular as soluções com matrizes  $4 \times 4$ ,  $5 \times 5$ , e  $6 \times 6$ . Para testar basta rodar os arquivos e alterar os arquivos **matriz-4x4.dat**, **matriz-5x5.dat** e **matriz-6x6.dat**. Além de também alterar os inputs do arquivo **y-4x4.dat**, **y-5x5.dat** e **y-6x6.dat**.

## Resultados

```

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat y-
4x4.dat

```

```

4
3
13
1

```

```

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat matriz-
4x4.dat

```

```

3 2 1 2
4 4 1 2
3 0 2 7
2 1 4 4

```

```

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ ./sistema4x4.exe

```

Matriz solução X:

```

    4.17021275
    -3.08510637
    -0.787234068
    -0.276595742

```

```

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat y-
5x5.dat

```



```

5
3
9
1
3
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat matriz-
5x5.dat
2 1 1 4 2
9 3 9 0 3
1 1 9 6 3
1 3 5 5 4
1 9 2 2 2
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ ./sistema5x5.exe
Matriz solução X:
  1.07964599
  0.469026536
  0.519174039
  2.59587026
 -4.26548672
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat y-
6x6.dat
5
1
3
1
9
-1
jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ cat matriz-
6x6.dat
1 2 3 4 50 6
11 2 3 14 5 2
3 29 3 4 15 6
4 1 13 4 5 6
9 4 32 4 59 6
2 0 -3 4 -1 46

jefter66@muaddib~/Workspace/intro_fiscomp/projeto1/tarefa7$ ./sistema6x6.exe
Matriz solução X:
0.341461062
4.57333997E-02

```

5.84237371E-03  
-0.242816135  
0.111934975  
-1.26564093E-02

## section 0 Tarefa 8

### Breve explicação sobre Método de Monte Carlo

Podemos escrever a média de uma função  $f$  pelas expressões

$$\langle f \rangle = \frac{1}{b-a} \int_a^b f(x) dx = \frac{1}{N} \sum_i f(x_i)$$

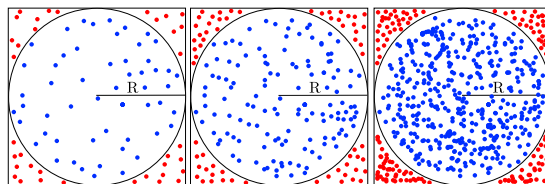
onde o termo  $N$  é o número total de pontos calculados e  $x_i$  são pontos que residem estritamente no intervalo  $[a, b]$ . Podemos então escrever uma aproximação para integral como

$$\int_a^b f(x) dx \approx \frac{(b-a)}{N} \sum_i f(x_i)$$

### Implementação do Método de Monte Carlo para cálculo do volume da N-esfera

Vamos chegar em uma fórmula que foi implementada no código em Fortran.

Primeiramente para o caso básico de  $d = 2$ , ou seja, o volume em  $2D$ , que é equivalente à área do círculo,  $A_c = \pi R^2$ . Pelo método de Monte Carlo essa área pode ser calculada gerando  $N$  valores  $X_i$  aleatórios e contando  $N_i$  desses números que obedecem a regra  $\sum_i^N X_i^2 \leq R^2$ , ou seja, quantos pontos  $X_i$  estão dentro da área do círculo.



Para  $N$  muito grande a figura tende a ficar completamente preenchida e então a área do quadrado  $A_q \approx N$  e a área do círculo  $A_c \approx N_i$ , fazendo a

razão abaixo temos uma aproximação melhor para a área do círculo

$$\frac{N_i}{N} \approx \frac{A_c}{A_q} \Rightarrow A_c \approx A_q \left( \frac{N_i}{N} \right)$$

Já sabemos que a área do quadrado é dada por  $A_q = (2R)^2$ , queremos chegar a uma fórmula para o volume podemos então considerar  $V_{cubo}(d) = (2R)^d, d \in \mathbb{N}$ , volume do cubo. Generalizando a área do círculo para o volume temos  $V(d) = V_{cubo}(d) \left( \frac{N_i}{N} \right)$ , ou seja

$$V(d) = (2R)^d \left( \frac{N_i}{N} \right)$$

Essa fórmula foi implementada no código abaixo e usada para calcular o volume da esfera em 2, 3 e 4 dimensões.

## Código

```
[frame,linenos=true]fortran ! Tarefa 8 - Cálculo do volume de uma N-
Esfera
write(*, *) "Informe a quantidade N de números aleatórios." read(*, *)
n
write(*, *) "Resultados" v2 = volume(n, 2) v3 = volume(n, 3) v4 =
volume(n, 4)
print *, "Para d = 2, Volume = ", v2 print *, "Para d = 3, Volume = ",
v3 print *, "Para d = 4, Volume = ", v4 end
! m - número de iteracoes function volume(n, id) ni = 1 ! Para uma
esfera de raio unitário r = 1 do j = 1, n rtmp = 0e0 raux = 0e0 do i
= 1, id call random_number(rtmp)raux = raux + rtmp ** 2enddo raux =
sqrt(raux)if (raux <= r) then ni = ni + 1endif enddo ni = ni - 1!V(d) =
(2R)^d * (Ni)/Nvolume = (2 * r) ** id * ((ni * 1e0)/(n * 1e0))end
```

## Resultados

Para fazer a comparação com o esperado, ou seja, o cálculo da expressão (??), foi feito o desvio padrão cada uma das 3 dimensões calculadas. Na tabela abaixo consta os valores médios do volume nas três dimensões e seus respectivos desvios padrão. Foi utilizado o método um número  $N = 1000$  de vezes com intuito de calcular o desvio padrão do método de Monte Carlo para o volume da n-esfera.

Table 2: Volume da esfera calculada em 2, 3 e 4 dimensões para diferentes ordens de grandezas.

Ordem	2 dimensões	3 dimensões	4 dimensões
$10^1$	2.79999995	4.00000000	6.40000010
$10^2$	3.31999993	4.15999985	4.63999987
$10^3$	3.16799998	4.21600008	4.96799994
$10^4$	3.15280008	4.15999985	4.86880016
$10^5$	3.13836002	4.18616009	4.93279982
$10^6$	3.14278412	4.18742418	4.93838406
$10^7$	3.14097643	4.18836260	4.93273115

Table 3: Volume médio da esfera calculado para um número de amostras  $N = 1000$  e seu desvio padrão.

Ordem	2 dimensões	3 dimensões	4 dimensões
$10^1$	$3.1 \pm 0.5$	$4 \pm 1$	$5 \pm 2$
$10^2$	$3.1 \pm 0.2$	$4.2 \pm 0.4$	$4.9 \pm 0.7$
$10^3$	$3.14 \pm 0.05$	$4.19 \pm 0.12$	$4.9 \pm 0.2$
$10^4$	$3.14 \pm 0.01$	$4.18 \pm 0.04$	$4.93 \pm 0.07$
$10^5$	$3.141 \pm 0.005$	$4.18 \pm 0.01$	$4.93 \pm 0.02$
$10^6$	$3.141 \pm 0.001$	$4.188 \pm 0.004$	$4.934 \pm 0.007$
$10^7$	$3.1415 \pm 0.0005$	$4.188 \pm 0.001$	$4.934 \pm 0.002$

Comparando os valores médios encontrados com os calculados pela expressão (??), que estão na tabela (??) é a aproximação é razoável, principalmente para ordens cada vez maiores de amostras para o método de Monte Carlo.

## section 0 Tarefa 9

O volume da n-esfera é definido como

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

Foi feito um programa para executar o cálculo do volume da n-esfera em  $d$  dimensões,  $d = 2, 3, \dots, 20$ . Os valores calculados estão na tabela (??) abaixo.

## Código

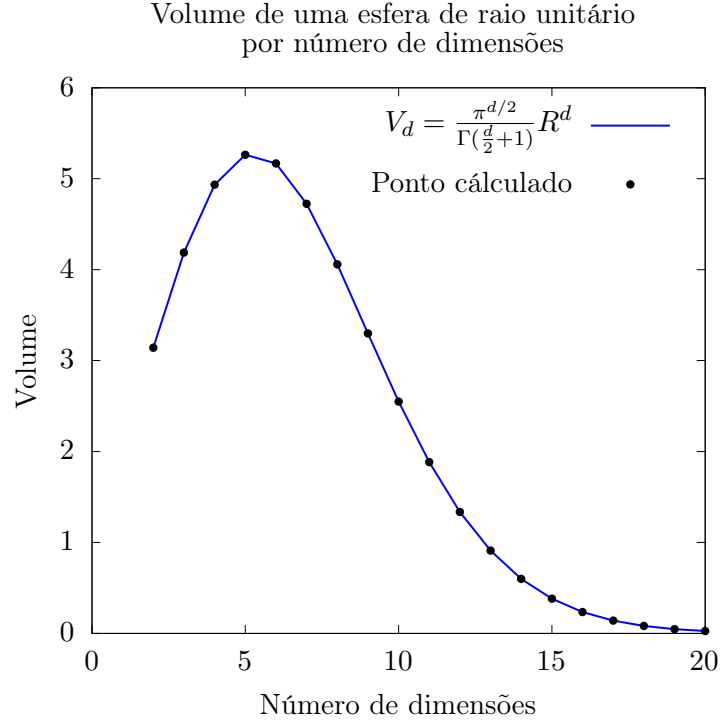
```
[frame,linenos=true]fortran external gamma open(unit=10, file="dimensoes-
esferas.dat")
R = 1.0e0 PI = acos(-1.0e0) do n = 2, 20 ! V(d) = [(d/2)/(d/2 + 1)] *
R(d)Volume = (PI**((n*1.0e0)/2)/gamma(n))*R**(n)write(10,*)Volumeenddo
close(10) end
function gamma(n) aux = 1e0 gamma = 1e0
! (d/2 + 1) x = (1.0e0 * n) / 2.0e0 + 1.0e0
do while(x >= 0e0) call baseCases(x, aux) if(x > 1) then gamma =
gamma * (x - 1) end if x = x - 1 gamma = gamma * aux end do end
subroutine baseCases(x, aux) if(x == 0e0) then aux = 1 else if (x ==
0.5e0) then aux = sqrt(acos(-1e0)) end if return end
```

Com base nos valores dessa tabela foi construído o gráfico da variação de volume em função do aumento das dimensões.

Table 4: Volume da esfera para diferentes dimensões.

Dimensão	Volume
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	0.00821459070
19	0.00466216095
20	0.00258068983

A partir desses cálculos foi feito o gráfico abaixo e na sequência respondido algumas perguntas acerca da n-esfera.



### Questão A

Saber quantas vezes maior será o volume é equivalente a saber a razão entre a cubo e a esfera em  $n$  dimensões, como o volume do cubo em  $n$  dimensões é dado por  $V_c = (2R)^d$  e vamos usar o volume da esfera (??), então

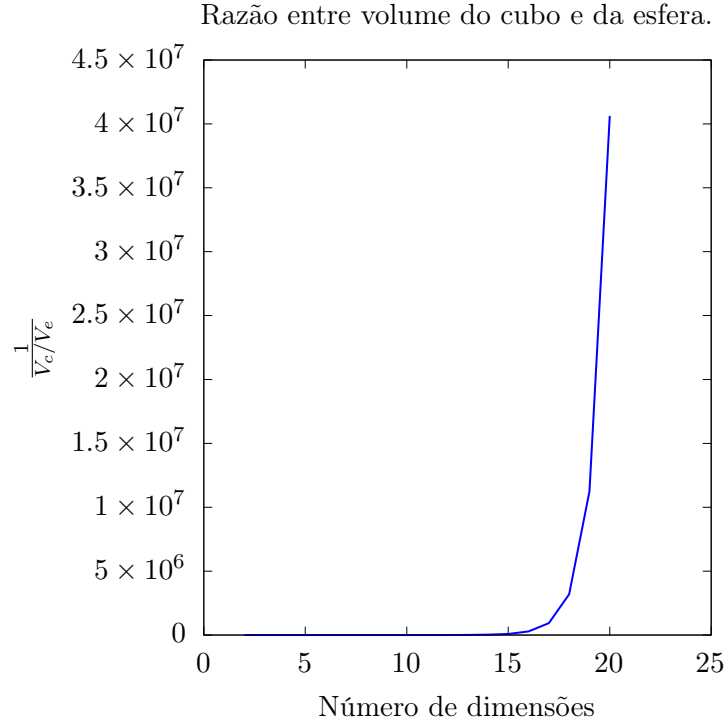
$$\frac{V_e}{V_c} = \frac{\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} R^d}{(2R)^d} = \frac{\pi^{d/2} R^d}{(2R)^d \Gamma(\frac{d}{2} + 1)} = \left( \frac{\sqrt{\pi}}{2} \right)^d \frac{1}{\Gamma(\frac{d}{2} + 1)}$$

ou seja, a diferença dos volumes não depende do raio, apenas da dimensão. Assim, para qualquer  $d$ , a razão do cubo e da esfera será de  $\frac{V_e}{V_d} = \left( \frac{\sqrt{\pi}}{2} \right)^d \frac{1}{\Gamma(\frac{d}{2}+1)}$ . No caso de  $d \mapsto \infty$  temos que, como  $\lim_{d \rightarrow +\infty} \Gamma(\frac{d}{2}+1) = +\infty$  e  $\left( \frac{\sqrt{\pi}}{2} \right)^d < 1$ , então

$$\lim_{d \rightarrow \infty} \left( \frac{\sqrt{\pi}}{2} \right)^d \frac{1}{\Gamma(\frac{d}{2} + 1)} = 0$$

Portanto, a razão  $\frac{V_c}{V_e} \rightarrow \infty$ , podemos dizer que para dimensões cada

vez maiores, o tamanho do cubo será muito maior que o da esfera. Esse resultado pode ser melhor exemplificado através do gráfico abaixo:



### Questão B

Sabemos que o número de partículas pode ser calculado pela relação  $n = \frac{V_e}{V_c}$ . Nessa hipótese temos que  $[V_c]$  em  $\mu^d$  e  $[V_e]$  em  $\text{\AA}^d$ . Considerando o volume da célula como equivalente ao de um cubo em  $d$  dimensões, ou seja,  $v_{\text{cel}} = (2R)^d$  e o do átomo aproximado por uma esfera,  $v_{\text{atom}} = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} R^d$ , sabemos que  $n_p = \frac{V_{\text{cel}}}{V_{\text{atom}}} =$  número de partículas, ajustando para as unidades dadas, temos que

$$n_p = \frac{V_{\text{cel}}}{V_{\text{atom}}} \cdot \frac{1 \cdot \mu^d}{1 \cdot \text{\AA}^d} = \frac{V_{\text{cel}}}{V_{\text{atom}}} \cdot \frac{(1 \cdot 10^{-4})^d}{(1 \cdot 10^{-10})^d}$$

$$np = \frac{V_{\text{cel}}}{V_{\text{atom}}} 10^{4d} \quad (2)$$

Sabemos que o número de Avogadro na dimensão  $d = 3$  é tal que  $n_p = 6 \cdot 10^{23}$ , temos

$$n_p = 6 \cdot 10^{23} = \frac{V_{\text{cel}}}{V_{\text{atom}}} 10^{4 \cdot 3} = \frac{V_{\text{cel}}}{V_{\text{atom}}} 10^{12}$$

como os números não batem, precisamos fazemos um ajuste para que possamos generalizar a ordem de grandeza para o número de Avogadro em  $d$  dimensões. Para isso fazemos  $\frac{10^{23}}{10^{12}} = 10^{11}$ . Então, em  $d$  dimensões, o número de Avogadro é da ordem de  $10^{4d} \cdot 10^{11} = 10^{4d+11}$ .