

Projeto 1 - Primeiro Relatório

Instituto de Física de São Carlos
Universidade de São Paulo

Gabriel Lima Alves (12558547)

Introdução à Física Computacional
Prof. Francisco Castilho Alcaraz

Setembro, 2022



1ª Tarefa

Nessa tarefa o objetivo era ler o valor do raio interno e externo de um torus e calcular seu volume e área. Um torus, ou toroide, é uma figura tridimensional obtida por meio de uma rotação de um círculo em torno de um determinado eixo. A distância entre o centro do círculo e o eixo é o raio externo r_e e o raio do círculo, ou raio interno, é r_i .

Por meio da integração, é simples obter as seguintes expressões para o cálculo do volume V e da área total A de um torus.

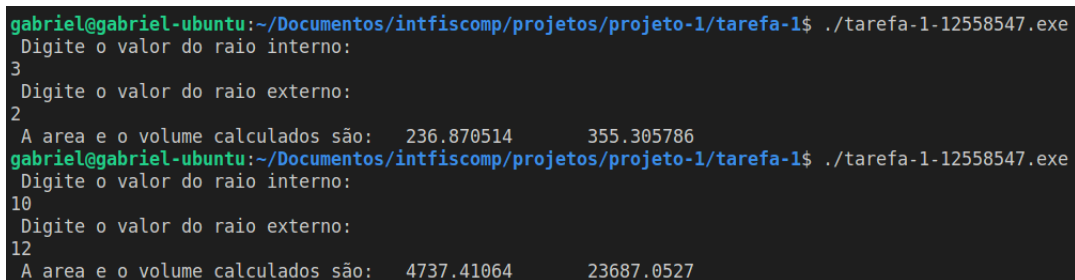
$$V = 2\pi^2 r_i^2 \cdot r_e$$

$$A = 4\pi^2 r_i \cdot r_e$$

Assim, tendo em mente essas equações foi feito o seguinte algoritmo para o cálculo do volume e área do torus. Logo em seguida, também pode se ver a entrada e saída do algoritmo

```
1  c      program tarefa-1
2
3      Parameter(rpi=acos(-1.e0))
4
5      write(*,*) 'Digite o valor do raio interno: '
6      Read(*,*) rr1
7
8      write(*,*) 'Digite o valor do raio externo: '
9      Read(*,*) rr2
10
11     rarea = 4*(rpi**2)*rr1*rr2
12     rvolume = 2*(rpi**2)*(rr1**2)*rr2
13
14     write(*,*) 'A area e o volume calculados são:', rarea, rvolume
15
16     end
17
```

Algoritmo 1: Código para resolução da tarefa 1



```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-1$ ./tarefa-1-12558547.exe
Digite o valor do raio interno:
3
Digite o valor do raio externo:
2
A area e o volume calculados são:   236.870514      355.305786
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-1$ ./tarefa-1-12558547.exe
Digite o valor do raio interno:
10
Digite o valor do raio externo:
12
A area e o volume calculados são:   4737.41064      23687.0527
```

Figura 1: testes do código realizados

2ª Tarefa

Na tarefa 2, dada as coordenadas cartesianas de três vetores pede-se para calcular o volume e a área lateral do paralelepípedo cujas arestas são definidas por $\vec{v}_1, \vec{v}_2, \vec{v}_3 - \vec{v}_2$.

Sabe-se que o produto vetorial tem como resultado um vetor perpendicular aos vetores do produto e tem módulo igual ao do plano contendo-os. Dessa forma, para o calculo da área do paralelepípedo basta fazer o produto vetorial entre cada vetor do paralelepípedo e multiplicar por 2 a soma dos módulos dos vetores resultados.

Para o cálculo do volume, é realizado o produto misto, pois seu volume é igual ao produto da área pela altura. Assim como anteriormente, a área da base definida como o produto vetorial dos vetores da base, já a altura é $\cos \theta \cdot (\vec{v}_3 - \vec{v}_2)$, em que θ é o angulo formado entre o vetor resultado do produto vetorial e o vetor $\vec{v}_3 - \vec{v}_2$. Dessa forma, o volme é igual ao resultado da equação abaixo.

$$volume = \vec{v}_1 \times \vec{v}_2 \cdot (\vec{v}_3 - \vec{v}_2)$$

Visto isso, o algoritmo para resolução dessa tarefa tem como entrada três vetores, $\vec{v}_1, \vec{v}_2, \vec{v}_3 - \vec{v}_2$. Logo em seguida o \vec{v}_3 é redefinido como $\vec{v}_3 - \vec{v}_2$. Depois, é feito o calculo do produto vetorial para o volume e a primeira parte da área e em seguida os outros produtos vetoriais necessários para o calculo da areá. Isso pode ser visto no código abaixo e logo em seguida a entrada e saída do algoritmo.

```
1  c      program tarefa-2
2
3      dimension vet(3,3)
4      dimension auxvet(3)
5
6      do i=1,3
7          write(*,3) i
8  3      format('Escreva o valor do',i2,'º vetor:')
9          Read(*,*) (vet(i,k),k=1,3)
10     end do
11
12     vet(3,1) = vet(3,1) - vet(2,1)
13     vet(3,2) = vet(3,2) - vet(2,2)
14     vet(3,3) = vet(3,3) - vet(2,3)
15
16     auxvet(1) = vet(1,2)*vet(2,3)-vet(1,3)*vet(2,2)
17     auxvet(2) = -(vet(1,1)*vet(2,3)-vet(1,3)*vet(2,1))
18     auxvet(3) = vet(1,1)*vet(2,2)-vet(1,2)*vet(2,1)
19
20     volume = auxvet(1)*vet(3,1) + auxvet(2)*vet(3,2)
21     + auxvet(3)*vet(3,3)
22
23     area = 2*sqrt(auxvet(1)**2 + auxvet(2)**2 + auxvet(3)**2)
24
25     auxvet(1) = vet(1,2)*vet(3,3)-vet(1,3)*vet(3,2)
26     auxvet(2) = -(vet(1,1)*vet(3,3)-vet(1,3)*vet(3,1))
27     auxvet(3) = vet(1,1)*vet(3,2)-vet(1,2)*vet(3,1)
```

```

28
29     area = area + 2*sqrt(auxvet(1)**2 + auxvet(2)**2 + auxvet(3)**2)
30
31     auxvet(1) = vet(3,2)*vet(2,3)-vet(3,3)*vet(2,2)
32     auxvet(2) = -(vet(3,1)*vet(2,3)-vet(3,3)*vet(2,1))
33     auxvet(3) = vet(3,1)*vet(2,2)-vet(3,2)*vet(2,1)
34
35     area = area + 2*sqrt(auxvet(1)**2 + auxvet(2)**2 + auxvet(3)**2)
36
37     write(*,*) "A area externa é:", area
38     write(*,*) "O volume é:", volume
39
40
41
42
43     end
44

```

Algoritmo 2: Código para resolução da tarefa 2

```

gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-2$ ./tarefa-2-12558547.exe
Escreva o valor do 1º vetor:
1 2 3
Escreva o valor do 2º vetor:
13 2 12
Escreva o valor do 3º vetor:
23 10 6
A area externa é: 666.683411
O volume é: 540.000000
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-2$ ./tarefa-2-12558547.exe
Escreva o valor do 1º vetor:
1 0 0
Escreva o valor do 2º vetor:
0 1 0
Escreva o valor do 3º vetor:
0 1 1
A area externa é: 6.00000000
O volume é: 1.00000000

```

Figura 2: testes do código realizados

3ª Tarefa

Na terceira tarefa desse projeto é pedido que leia uma lista de números de tamanho *ivet* e um valor inteiro *n*, depois que ordene os *n* primeiros números dessa lista em ordem crescente e imprima a lista ordenada em um arquivo.

O algoritmo ordena os *n* primeiros números da lista, desde que $ivet \leq n$ e $ivet = 20$. Para se ler uma lista com mais de 20 elementos deve se alterar o valor de *ivet* no algoritmo e recompilar o código. A seguir esta a implementação do código e a entrada e saída do algoritmo

```

1  c      program tarefa-3
2
3      Parameter(ivet = 20)!tamanho do vetor
4      dimension vet(ivet)

```

```

5      Parameter(ient = 10)
6
7      !leitura do vetor n do arquivo
8      open(unit=ient,file='entrada-1-12558547.dat')
9      Read(ient,*) vet
10     close(ient)
11
12     !leitura do numero de algarismos a ser ordenado
13     write(*,*) 'Digite o número de algarismos a ser ordenado: '
14     Read(*,*) n
15
16     !metodo de ordenação
17     do i = 1, n
18         do j = ivet, i, -1
19             if (vet(j)<vet(j-1)) then
20                 aux = vet(j-1)
21                 vet(j-1) = vet(j)
22                 vet(j) = aux
23             end if
24         end do
25     end do
26
27     open(unit=ient,file='saida-1-12558547.dat')
28     do i=1,ivet
29         write(ient,*) vet(i)
30     end do
31     write(ient,*) 'Número de alagarismos ordenados é: ', n
32     close(ient)
33
34     end
35

```

Algoritmo 3: Código para resolução da tarefa 3

1	20.00000000
2	19.00000000
3	18.00000000
4	17.00000000
5	16.00000000
6	15.00000000
7	14.00000000
8	13.00000000
9	12.00000000
10	11.00000000
11	10.00000000
12	9.00000000
13	8.00000000
14	7.00000000
15	6.00000000
16	5.00000000
17	4.00000000
18	3.00000000
19	2.00000000
20	1.00000000

Figura 3: entrada do algoritmo

1	1.00000000
2	2.00000000
3	3.00000000
4	4.00000000
5	5.00000000
6	6.00000000
7	7.00000000
8	8.00000000
9	9.00000000
10	10.00000000
11	20.00000000
12	19.00000000
13	18.00000000
14	17.00000000
15	16.00000000
16	15.00000000
17	14.00000000
18	13.00000000
19	12.00000000
20	11.00000000
21	Número de algarismos ordenados é: 10
22	

Figura 4: saída do algoritmo

4ª Tarefa

A tarefa 4 tem como objetivo o cálculo do $\cos(x)$ usando a expansão em séries. Foi calculado o valor do cosseno com precisão simples e com precisão dupla e logo em seguida os valores calculado a partir da série, com precisão na quinta casa decimal, são comparado com as funções que já existem no Fortran, $\cos(x)$ e $\dcos(x)$ (que é uma função de dupla precisão).

O *loop* criado para o cálculo do cosseno depende da precisão solicitada, ele soma os termos no valor do cosseno somente enquanto os termos da serie forem maiores que essa precisão. O fatorial é feito iterativamente dentro do *loop*. A código do algoritmo e sua execução pode ser visto a seguir.

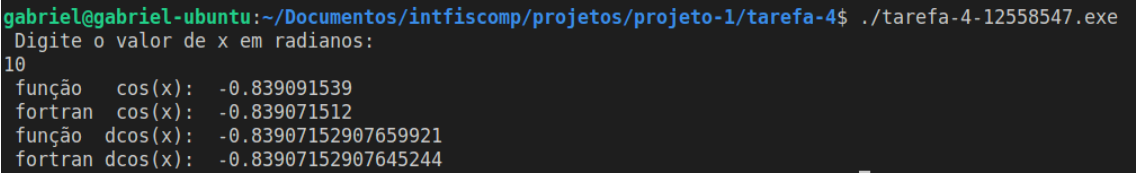
```
1  c      program tarefa-4
2
3      Parameter(eprec = 1e-15)
4      real*8 :: deprec = 1d-30
5      real*8 :: daux = 1d0
6      real*8 :: drespcos = 1d0
7      real*8 :: dx
8      real*8 :: dfat
9
10     !leitura de x
11     write(*,*) 'Digite o valor de x em radianos: '
12     Read(*,*) dx
13     rx = dx
14
15     aux = 1e0
16     respcos = 1e0
17     i=2
18     j = -1
19     fat = 1e0
20     do while (aux>eprec)
21         fat = fat*(i-1)*(i)
22         aux = ((rx**(i))/(fat))
23         respcos = respcos + aux*j
24         i = i + 2
25         j = j*(-1)
26     end do
27
28     i=2
29     j = -1
30     dfat = 1d0
31     do while (daux>deprec)
32         dfat = dfat*(i-1)*(i)
33         daux = ((dx**(i))/(dfat))
34         drespcos = drespcos + daux*j
35         i = i + 2
36         j = j*(-1)
37     end do
```

```

38
39     if (isnan(respcos)) then
40         write(*,*) 'função   cos(x): Memória insuficiente'
41     else
42         write(*,*) 'função   cos(x): ' ,respcos
43     end if
44
45     write(*,*) 'fortran   cos(x): ' ,cos(rx)
46
47     if (isnan(drespcos)) then
48         write(*,*) 'função   dcos(x): Memória insuficiente'
49     else
50         write(*,*) 'função   dcos(x): ' ,drespcos
51     end if
52
53     write(*,*) 'fortran   dcos(x): ' ,dcos(dx)
54     end

```

Algoritmo 4: Código para resolução da tarefa 4



```

gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-4$ ./tarefa-4-12558547.exe
Digite o valor de x em radianos:
10
função   cos(x): -0.839091539
fortran   cos(x): -0.839071512
função   dcos(x): -0.83907152907659921
fortran   dcos(x): -0.83907152907645244

```

Figura 5: saída do algoritmo

5ª Tarefa

Nesta tarefa, o programa recebe um arquivo com uma matriz de $(N!)$ linhas e $(N+1)$ colunas, em cada linha esta um caso da permutação N (cada algarismo ocupando uma coluna) e na ultima coluna a paridade da permutação. A logica utilizada no algoritmo foi criar uma nova matriz com dimensão de $((N+1)! \times (N+2))$ e ir colocando os valores da matriz antiga na nova com algumas modificações. Nas linhas de $(1$ a $N!)$ a matriz antiga é totalmente copiada na nova, as diferenças são que na coluna $(N+1)$ é adicionado o valor (N) e a paridade é copiada para coluna $(N+2)$ ao invés da $(N+1)$.

Nas próximas $(N!+1$ a $2(N!))$ linhas o mesmo processo é repetido, porém o o valor (N) é colocado na coluna (N) , enquanto que os valores da matriz antiga agora pulam essa posição ficando nas colunas $(1$ a $N-1$ e $N+1)$ já a permutação permanece na mesma coluna, porém multiplicada por (-1) visto que houve uma mudança nas ordens do algarismos desses casos de permutação (quantidade impar).

Esse mesmo processo é repetido novamente nas linhas $(2(N!)+1$ a $3(N!))$ porém os valores da permutação da antiga matriz ocupam as colunas $(1$ a $N-2$ e N a $N+1)$, o valor (N) ocupa a coluna $(N-1)$ e permutação ainda continua na coluna $(N+1)$ mas agora multiplicado por (1) pois houve duas mudanças de posições nos algarismos da permutação (quantidade par).

Ou seja, esse processo acontece $\frac{(N+1)!}{N!} = N+1$ vezes e cada ciclo de (N!) linhas a matriz antiga é copiada na nova: sendo que a posição do valor (N) a cada ciclo vai mudando (indo da coluna (N+1 a 1)) e o valor da permutação é colocado na coluna (N+2) multiplicado por (-1) elevado ao número de vezes que esse ciclo se repetiu. Por ultimo, os valores da permutação antiga sempre são copiados nas colunas de (1 a N+1) sempre pulando a posição em que o valor (N) esta atualmente definido pelo número de ciclos ocorridos.

Abaixo está o código e a implementação com N = 2 e a saída e entrada do código. Para o calculo de outros valores, deve se mudar o valor do parâmetro *itmat* para o novo (N) e *ifat* para o valor (N!).

```

1  c      program tarefa-5
2
3      Parameter(itold_mat = 2)
4      Parameter(ifat = 2)
5      dimension iauxmat((itold_mat+1),ifat)
6      dimension ioldmat(ifat,(itold_mat+1))
7      dimension inewmat(ifat*(itold_mat+1),(itold_mat+2))
8      Parameter(ient = 10)
9
10     !leitura da matriz nxn do arquivo
11     open(unit=ient,file='entrada-2-12558547.dat')
12     Read(ient,*) iauxmat
13     ioldmat = transpose(iauxmat)
14     close(ient)
15
16     i = 1 !linha
17     j = 1 !coluna
18     k = 0
19     n = itold_mat+1
20     do while (i <= ifat*(itold_mat+1))
21         if (j == itold_mat+2) then
22             inewmat(i,j) = ioldmat(i-k,j-1)*((-1)**(k/ifat))
23             if ((i/ifat)>(k/ifat)) then
24                 k = k + ifat
25             endif
26             i = i + 1
27             j = 1
28         else
29             if (j == (n -(k/ifat))) then
30                 inewmat(i,j) = itold_mat+1
31             else
32                 if (j>(n -(k/ifat))) then
33                     inewmat(i,j) = ioldmat(i-k,j-1)
34                 else
35                     inewmat(i,j) = ioldmat(i-k,j)
36                 endif
37             endif
38             j = j + 1
39         endif

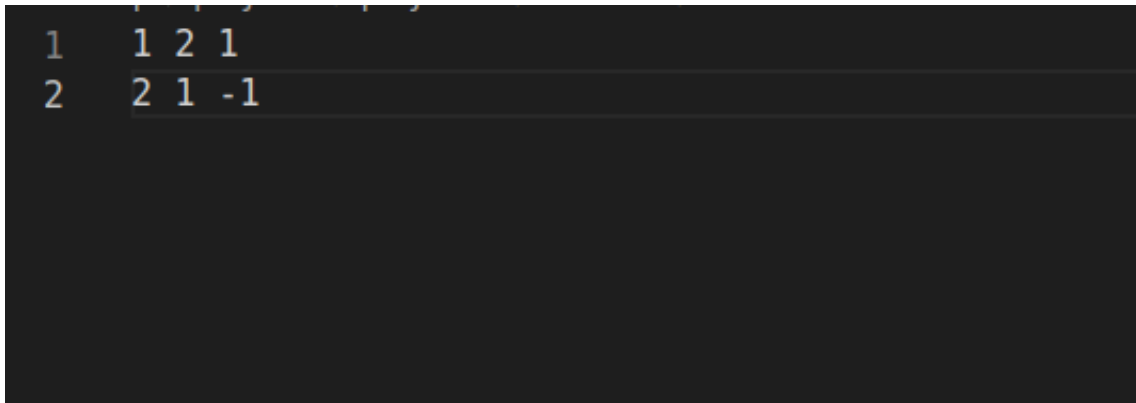
```

```

40     end do
41
42     !impressão da matriz (n+1)x(n+1) resultado no arquivo
43     open(unit=ient,file='saida-2-12558547.dat')
44     do i=1,ifat*(itold_mat+1)
45         write(ient,*) (inewmat(i,k),k=1,(itold_mat+2))
46     end do
47     close(ient)
48
49     end
50

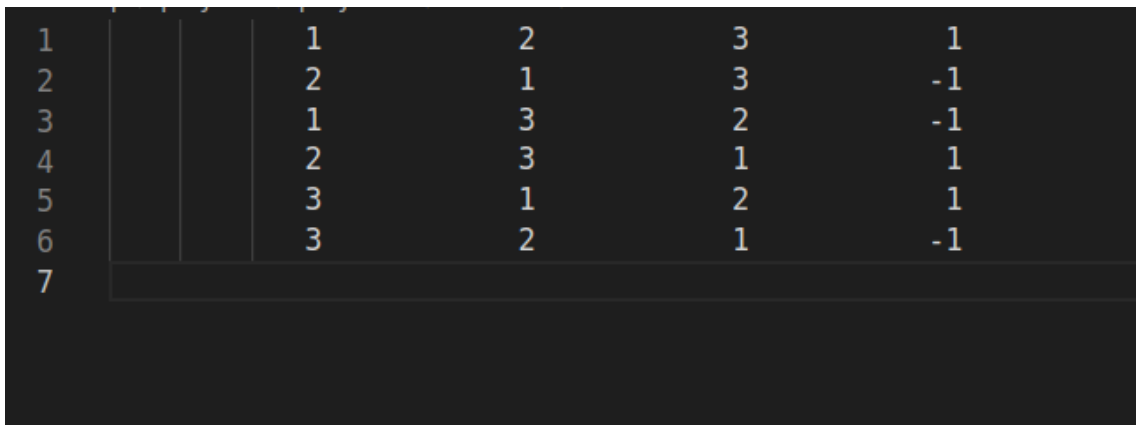
```

Algoritmo 5: Código para resolução da tarefa 5



1	1	2	1
2	2	1	-1

Figura 6: entrada do algoritmo



1			1	2	3	1
2			2	1	3	-1
3			1	3	2	-1
4			2	3	1	1
5			3	1	2	1
6			3	2	1	-1
7						

Figura 7: saída do algoritmo

6ª Tarefa

Na sexta tarefa é lido uma matriz de $N \times N$ no terminal e as permutações de N obtidas do programa anterior de um arquivo. O calculo do determinante é feito utilizando a definição matemática do determinante que pode ser vista na seguinte equação.

$$\sum_{i=1}^{N!} p(i, n+1) \cdot a(1, p(i, 1)) \cdot a(2, p(i, 2)) \cdot a(3, p(i, 3)) \dots a(n, p(i, n))$$

em que 'a' é a matriz NxN da qual vai ser calculado o determinante e 'p' a matriz permutação N gerada no programa anterior. Abaixo esta o código e a implementação para N=3 e suas entradas e saídas. Para o calculo de outros valores, deve se mudar o valor do parâmetro *itmat* para o novo (N) e *ifat* para o valor (N!), bem como mudar a matriz permutação.

```

1  c      program tarefa-6
2
3      Parameter(itmat = 3)
4      Parameter(ifat = 6)
5      dimension iauxmat((itmat+1),ifat)
6      dimension ipermat(ifat,(itmat+1))
7      dimension rmatn(itmat,itmat)
8      Parameter(ient = 10)
9      real*8 :: det = 0d0
10
11      !leitura da matriz permutação do arquivo
12      open(unit=ient,file='entrada-3-12558547.dat')
13      Read(ient,*) iauxmat
14      ipermat = transpose(iauxmat)
15      close(ient)
16
17      !leitura da matriz nxn
18      write(*,*) 'Escreva a matriz para o calculo do determinante: '
19      do i=1,itmat
20          Read(*,*) (rmatn(i,k),k=1,itmat)
21      end do
22
23      !calculo de determinante
24      do i=1,ifat
25          aux = 1
26          do j=1,itmat
27              aux = aux*rmatn(j,ipermat(i,j))
28          end do
29          aux = aux*ipermat(i,itmat+1)
30          det = det + aux
31      end do
32
33      !resultado do determinante
34      write(*,*) "O determiante é: ", det
35
36      end

```

Algoritmo 6: Código para resolução da tarefa 6

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

Figura 8: matriz permutação 3

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-6$ ./tarefa-6-12558547.exe
Escreva a matriz para o calculo do determinante:
1      2      3
11     2      3
3      29     3
0 determiante é:      810.00000000000000
```

Figura 9: entrada e saída do algoritmo

7ª Tarefa

Nesta tarefa, é lido de arquivos a matriz $N \times N$ em que estão os coeficientes das variáveis das equações lineares, um vetor com as igualdades das das equações lineares e por ultimo uma matriz permutação N (calculado no programa 5). Depois disso, as variáveis das as equações lineares são encontradas pelo método de cramer.

Seguindo o método de cramer, sera calculado o determinante da matriz com os coeficientes das variáveis, depois uma matriz temporária é criada e nela é copiada a matriz com os coeficientes. Logo em seguida, coluna por coluna o coeficiente é trocado pela variável que esgava multiplicando e o determinante dessa matriz temporária é salvo. O determinante temporário de cada matriz é dividido pelo determinante da matriz coeficiente e assim é obtido o valor da variável procurado.

Abaixo esta o código e a implementação para $N=4$ e as saídas. Para o calculo de outros valores, deve se mudar o valor do parâmetro *itmat* para o novo (N) e *ifat* para o valor ($N!$), bem como mudar a matriz permutação.

```
1  c      program tarefa-7
2
3      Parameter(itmat = 4)
4      Parameter(ifat = 24)
5      dimension auxmatn(itmat,itmat)
6      dimension rmattemp(itmat,itmat)
7      dimension rmatA(itmat,itmat)
8      dimension rvetY(itmat)
9      Parameter(ients = 10)
10
11     !leitura da matriz nxn
12     open(unit=ients,file='entrada-5-12558547.dat')
```

```

13      Read(ient,*) auxmatn
14      rmatA = transpose(auxmatn)
15      close(ient)
16
17      !leitura do vetor
18      open(unit=ient,file='entrada-6-12558547.dat')
19      Read(ient,*) rvetY
20      close(ient)
21
22
23      !calculando os determinantes
24      detD = 0e0
25      call deter(itmat,ifat, rmatA, detD)
26      do i=1,itmat
27          rmattemp = rmatA
28          do j=1,itmat
29              rmattemp(j,i) = rvetY(j)
30          end do
31          detaux = 0e0
32          call deter(itmat,ifat, rmattemp, detaux)
33          write(*,*) (detaux/detD)
34      end do
35
36      end
37
38      subroutine deter(itmat,ifat, rmatn, det)
39          dimension iauxmat((itmat+1),ifat)
40          dimension ipermat(ifat,(itmat+1))
41          dimension rmatn(itmat,itmat)
42          Parameter(ientsub = 20)
43
44          !leitura da matriz permutação do arquivo
45          open(unit=ientsub,file='entrada-4-12558547.dat')
46          Read(ientsub,*) iauxmat
47          ipermat = transpose(iauxmat)
48          close(ientsub)
49
50          !calculo de determinante
51          do i=1,ifat
52              aux = 1
53              do j=1,itmat
54                  aux = aux*rmatn(j,ipermat(i,j))
55              end do
56              aux = aux*ipermat(i,itmat+1)
57              det = det + aux
58          end do
59
60          return

```

end

Algoritmo 7: Código para resolução da tarefa 7

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

Figura 10: Matriz Permutação 4x4

1	1	6	2	4
2	3	19	4	15
3	1	4	8	-12
4	5	33	9	3

Figura 11: Matriz coeficiente 4x4

1	8
2	25
3	18
4	72

Figura 12: vetor igualdade 4

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-7$ ./tarefa-7-12558547.exe
-138.000000
20.000000
11.000000
1.000000
```

Figura 13: Saída sistema 4x4

8ª Tarefa

Na oitava tarefa, é pedido para calcular o volume de uma esfera com d dimensões pelo Método de Monte Carlo. Para o calculo, foi lido um número n que será o número de iterações que o *loop* será executado.

Dentro do *loop*, para cada dimensão é gerado um número aleatório entre 0 e 1 que é elevado ao quadrado, quando todos os componentes de cada dimensão são somados é feito sua a raiz quadrada obtendo se assim a distancia daquele ponto em d dimensões da origem, se essa distancia for menor que o raio estipulado (raio unitário) é somado em um contador. No final esse contador é dividido pelo número total de iterações e essa razão sera o volume da esfera.

Em seguida esta o código do algoritmo e suas entradas e saída.

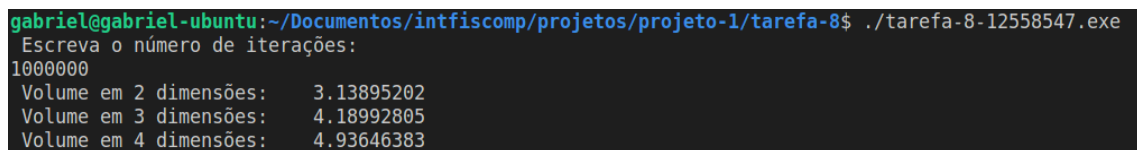
```
1  c      program tarefa-8
2          parameter(iseed = 10)
3
4          write(*,*) 'Escreva o número de iterações:'
5          read(*,*) n
6
7          print *, "Volume em 2 dimensões: ", volume(n,2)
8          print *, "Volume em 3 dimensões: ", volume(n,3)
9          print *, "Volume em 4 dimensões: ", volume(n,4)
10
11         end
12
13         function volume(n, idim)
14             count = 0
15             raio = 1e0
16             r = rand(iseed)
```

```

17
18     do i = 1, n
19         aux = 0e0
20         do id = 1, idim
21             r = rand()
22             aux = aux + (r**2)
23         end do
24         aux = sqrt(aux)
25         if (aux <= raio) then
26             count = count + 1
27         end if
28     end do
29
30     volume = (2**idim)*(count/n)
31     return
32 end

```

Algoritmo 8: Código para resolução da tarefa 8



```

gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-1/tarefa-8$ ./tarefa-8-12558547.exe
Escreva o número de iterações:
1000000
Volume em 2 dimensões: 3.13895202
Volume em 3 dimensões: 4.18992805
Volume em 4 dimensões: 4.93646383

```

Figura 14: Entrada e saída

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).

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

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

```

1  c      program tarefa-9
2
3      Parameter(pi=acos(-1.e0))
4      Parameter(raio = 1e0 )
5      Parameter(ient = 10)
6      real*8:: volume
7      real*8:: gama
8      n = 20

```



```

9      dim = 0e0
10
11      open(unit=ient,file='saida-3-12558547.dat')
12      do i=1,n
13          dim = dim +1
14          volume = ((raio**(dim))*(pi**(dim/2)))/gama((dim/2)+1)
15          write(ient,*)'Volume da esfera de ', i, 'dimensões: ', volume
16      end do
17      close(ient)
18
19  end
20
21  function gama(rent)
22      real*8 :: dpi=acos(-1.d0)
23      real*8 :: gama
24
25      gama = 1
26      do while (1>0)
27          if (rent==1) then
28              gama = gama*1
29              return
30          end if
31
32          if (rent == 0.5) then
33              gama = gama*sqrt(dpi)
34              return
35          end if
36
37          rent = rent - 1
38          gama = gama*rent
39      end do
40  end
41

```

Algoritmo 9: Código para resolução da tarefa 9

1	Volume da esfera de	1 dimensões:	2.0000000600821290
2	Volume da esfera de	2 dimensões:	3.1415927410125732
3	Volume da esfera de	3 dimensões:	4.1887904934656577
4	Volume da esfera de	4 dimensões:	4.9348025321960449
5	Volume da esfera de	5 dimensões:	5.2637894109810048
6	Volume da esfera de	6 dimensões:	5.1677131652832031
7	Volume da esfera de	7 dimensões:	4.7247663651515186
8	Volume da esfera de	8 dimensões:	4.0587126413981123
9	Volume da esfera de	9 dimensões:	3.2985092598196348
10	Volume da esfera de	10 dimensões:	2.5501642862955731
11	Volume da esfera de	11 dimensões:	1.8841040669762279
12	Volume da esfera de	12 dimensões:	1.3352629767523871
13	Volume da esfera de	13 dimensões:	0.91062893229810682
14	Volume da esfera de	14 dimensões:	0.59926462324838792
15	Volume da esfera de	15 dimensões:	0.38144336494028214
16	Volume da esfera de	16 dimensões:	0.23533068460131448
17	Volume da esfera de	17 dimensões:	0.14098114834791514
18	Volume da esfera de	18 dimensões:	8.2145908900669640E-002
19	Volume da esfera de	19 dimensões:	4.6621612253639921E-002
20	Volume da esfera de	20 dimensões:	2.5806897683118387E-002
21			

Figura 15: saída

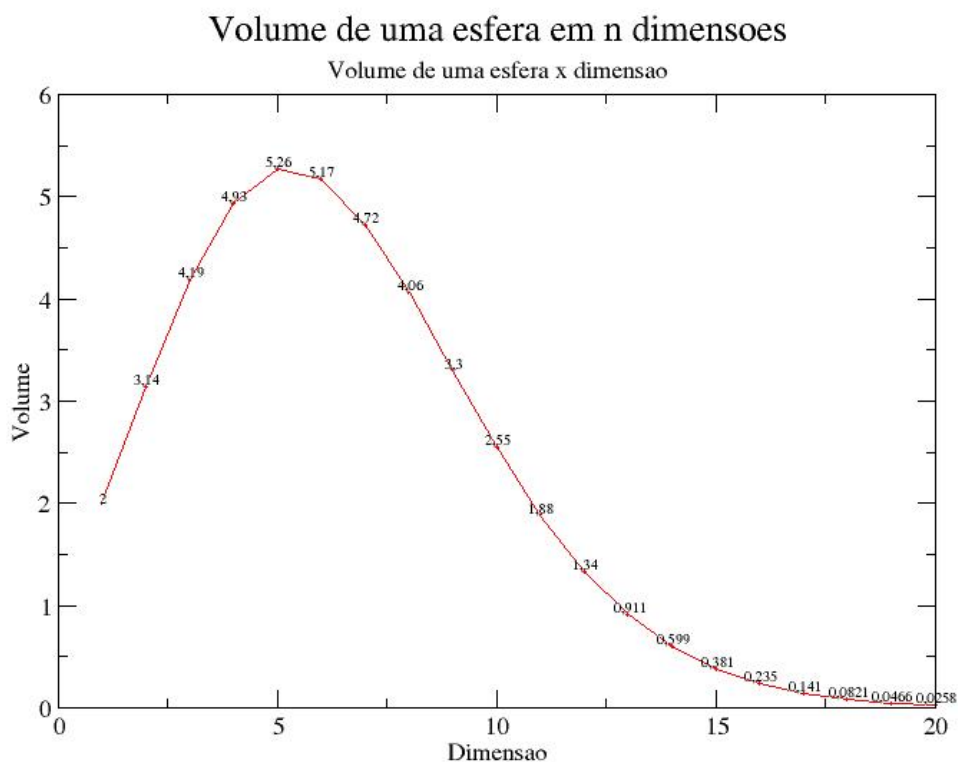


Figura 16: gráfico plotado no Xmgrace

Pergunta A

Observando o gráfico de Volume x dimensão é possível perceber que para dimensões infinitas o volume tende a 0. Dessa forma, a razão entre o volume do cubo com o volume da esfera vai para infinito. Assim, pode se concluir que o volume do cubo é infinitas vezes maior que a esfera quando a dimensão tende ao infinito. Ou seja, para dimensões cada vez maiores, o tamanho do cubo será muito maior que o da esfera.

Pergunta B

Se considerarmos o volume da célula como equivalente ao de um cubo em d dimensões, ou seja, $v_{cel} = (2R)^d$ e o volume aproximado de um átomo como o volume de uma esfera $v_{atom} = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} R^d$ e sabendo que o número de partículas é igual $n = \frac{v_{cel}}{v_{atom}}$, temos que:

$$n_p = \frac{v_{cel}}{v_{atom}} = \frac{1 \cdot u^d}{1 \cdot A^d} = \frac{v_{cel}}{v_{atom}} \cdot \frac{(1 \cdot 10^4)^d}{(1 \cdot 10^{10})^d}$$
$$n_p = \frac{v_{cel}}{v_{atom}} \cdot 10^{4d}$$

Assim a ordem de grandeza do número de Avogadro (n_a) é proporcional a sua dimensão ($n_a \sim 10^{4d}$). Para uma relação direta entre a dimensão e número de Avogadro provavelmente seria necessário uma correção que viria na forma de uma constante (α), por isso a ordem de grandeza do número de Avogadro seria algo similar a isso: $n_a = \alpha \cdot 10^{4d}$.