7600017 - Introdução à Física Computacional

Projeto 1: Introdução à Programação

Prof: José A. Hoyos

Victor Foscarini Almeida - nUSP: 10728101

1 - Introdução

Hodiernamente, o conhecimento de criação modelos matemáticos e técnicas de soluções numéricas utilizando computadores para analisar e resolver problemas científicos e de engenharia é algo essencial. Nesse projeto serão realizadas tarefas básicas para computação científica, visando introduzir os alunos aos métodos e precisão para programação visando resolver problemas que se tornam mais simples dessa forma, como o cálculo númerico.

Uma das principais linguagens utilizadas no meio científico é o FORTRAN, que apesar de não ser a única existente, é uma das mais efetivas para o cálculo númerico, porém em troca desse maior rendimento, é necessário, diferentemente de outras linguagens como Python, tomar alguns cuidados com declaração de variáveis e precisão que serão discutidos no projeto. Nesse projeto, será utilizado o FORTRAN 95.

2 - Métodos e Resultados

2.1 - Fatoriais e a aproximação de Stirling

(a) Escreva um programa que imprima em um arquivo uma tabela com os fatoriais de todos os inteiros entre 1 e 20. Verifique e discuta a precisão de seus resultados.

Sabendo que pela definição, o fatorial de um número é o produto dos números inteiros consecutivos de 1 até o número dado ‘n’, a possibilidade de utilizar um código para o cálculo torna-se evidente.

Para o código fatorial, utiliza-se um loop do, que vai multiplicando consecutivamente os inteiros de 1 a 20 e os salva em um arquivo chamado ‘fatoriais.txt’ .

É tentador escrever o código assim:

```
program fatorialimpreciso
    implicit none
    integer a,b,c

    c = 1

    open(10,file='fatoriais.txt')

    do a=1,20
        b = c
        c = b*a
        write(10,*)c
    enddo


end program fatorialimpreciso
```

Apesar de o programa anterior calcular o fatorial corretamente até o valor 12, o problema está na definição da precisão dos inteiros, pois a precisão de um inteiro(4) não é suficiente para o cálculo dos fatoriais, como nota-se na saída a seguir:

```
 fat(           1 )=           1
 fat(           2 )=           2
 fat(           3 )=           6
 fat(           4 )=          24
 fat(           5 )=         120
 fat(           6 )=         720
 fat(           7 )=        5040
 fat(           8 )=       40320
 fat(           9 )=      362880
 fat(          10 )=     3628800
 fat(          11 )=    39916800
 fat(          12 )=   479001600
 fat(          13 )=  1932053504
 fat(          14 )=  1278945280
 fat(          15 )=  2004310016
 fat(          16 )=  2004189184
 fat(          17 )=  -288522240
```

```
 fat(          18 )=  -898433024
 fat(          19 )=   109641728
 fat(          20 )= -2102132736
```

Note que até o valor 12 os fatoriais foram calculados corretamente, porém a partir do 13 aparecem números estranhos. Isso ocorre pois quando não definido, o inteiro é considerado um inteiro(4), cujo valor máximo de precisão é de 2^(32-1) -1, ou seja, 2.147.486.647. Assim, sendo o valor real do do fatorial de 13 maior do que esse limite de precisão, o compilador do fortran entrega como output um lixo guardado na memória ao invés do valor esperado. O mesmo ocorre com valores acima de 13.

Para corrigir isso, declaram-se os inteiros com precisão 8, tendo como valor máximo de precisão 2^(64-1) -1, tendo um valor máximo na ordem de 9,22 * 10^8 e, assim, o programa retorna os valores corretos. Então, o código final preciso fica assim:

```
program fatorial
    implicit none
    integer(8) a,b,c

    c = 1

    open(10,file='fatoriais.txt')

    do a=1,20
        b = c
        c = b*a
        write(10,*)c
    enddo


end program fatorial
```

Já o código correto traz o seguinte output:

1
2
6
24
120
720
5040
40320
362880
3628800
39916800
479001600
6227020800
87178291200
1307674368000
20922789888000
355687428096000
6402373705728000
121645100408832000
2432902008176640000

São obtidos então os resultados corretos dos fatoriais de 1 a 20.

(**b**) **Escreva agora um programa que imprima em um arquivo uma tabela com os logaritmo dos fatoriais de todos os inteiros entre 2 e 30. Novamente, verifique e discuta seus resultados.**

Dessa vez, como será feito o cálculo do logaritmo, foram utilizados valores reais. Calcula-se o fatorial de forma semelhante à letra (a), porém utilizando o real de precisão 8, como fixado nas regras do curso. Nota-se que o valor da variável “a” foi convertido para um real na variável “aa” para realizar a multiplicação pelo real “b” e o cálculo do logaritmo. Os resultados são salvos em um arquivo chamado ‘logfatoriais.txt’.

```
program logfatorial
    implicit none
    integer a
```

```
    real(8) aa,b,c,d

    c = 1.0d0

    open(10,file='logfatoriais.txt')

    do a=2,30
        aa = real(a,8)
        b = c
        c = b*aa
        d = log(c)
        write(10,*)d
    enddo


end program logfatorial
```

É obtido o seguinte resultado:

```
 0.69314718055994529
  1.7917594692280550
  3.1780538303479458
  4.7874917427820458
  6.5792512120101012
  8.5251613610654147
  10.604602902745251
  12.801827480081469
  15.104412573075516
  17.502307845873887
  19.987214495661885
  22.552163853123425
  25.191221182738680
  27.899271383840890
  30.671860106080672
  33.505073450136891
  36.395445208033053
  39.339884187199495
  42.335616460753485
  45.380138898476908
  48.471181351835227
  51.606675567764377
  54.784729398112319
```

58.003605222980518
61.261701761002001
64.557538627006338
67.889743137181540
71.257038967168015
74.658236348830158

(**c) Compare os resultados do item 1b com a aproximação de Stirling:**

$$\ln n! \approx S = n \ln n - n + \frac{1}{2} \ln (2\pi n) . \qquad (1)$$

Especificamente, imprima novamente uma tabela com quatro colunas: n, ln n!, S e [ln n! − S] / ln n!. Discuta seus resultados.

Utiliza-se um inteiro “a” de forma semelhante ao item (b) e calcula-se a aproximação na variável “aprox” e o fatorial na variável “c”. Vale notar também que o valor de pi no fortran pode ser calculado por 4*atan(1.0d0). No final, imprime-se uma tabela como pedido em um arquivo chamado ‘comparacao.txt’.

```
program S
    implicit none
    real(8) n,aprox,pi,b,c
    integer a

    c = 1.0d0

    open(10,file='comparacao.txt')

    do a=2,30
        n = real(a,8)
        b = c
        c = n*b
        pi = 4*atan(1.0d0)
        aprox = n * log(n) - n + 0.5*log(2*pi*n)
        write(10,*)n,log(c),aprox,(log(c)-aprox)/log(c)
    enddo

end program S
```

Combinando os resultados em uma tabela, tem-se:

2.0000000000000000	0.69314718055994529	0.65180648460453594	5.9642017041767491E-002
3.0000000000000000	1.7917594692280550	1.7640815435430568	1.5447344445693184E-002
4.0000000000000000	3.1780538303479458	3.1572631582441804	6.5419508962468237E-003
5.0000000000000000	4.7874917427820458	4.7708470515922246	3.4767038950857614E-003
6.0000000000000000	6.5792512120101012	6.5653750831870310	2.1090741751492960E-003
7.0000000000000000	8.5251613610654147	8.5132646511195222	1.3954820843890348E-003
8.0000000000000000	10.604602902745251	10.594191637483277	9.8176851669556312E-004
9.0000000000000000	12.801827480081469	12.792572017898756	7.2297976184366628E-004
10.000000000000000	15.104412573075516	15.096082009642156	5.5153177212658908E-004
11.000000000000000	17.502307845873887	17.494734170385932	4.3272439010034666E-004
12.000000000000000	19.987214495661885	19.980271655554681	3.4736406659921081E-004
13.000000000000000	22.552163853123425	22.545754858935421	2.8418533271324276E-004
14.000000000000000	25.191221182738680	25.185269812625918	2.3624778130404078E-004
15.000000000000000	27.899271383840890	27.893716650288930	1.9909959208389360E-004
16.000000000000000	30.671860106080672	30.666652450161063	1.6978611344723750E-004
17.000000000000000	33.505073450136891	33.500172054188461	1.4628817202040308E-004
18.000000000000000	36.395445208033053	36.390816054283718	1.2719046910608648E-004
19.000000000000000	39.339884187199495	39.335498626950255	1.1147872800975242E-004
20.000000000000000	42.335616460753485	42.331450141061481	9.8411693044942686E-005
21.000000000000000	45.380138898476908	45.376170944258263	8.7438124143291159E-005
22.000000000000000	48.471181351835227	48.467393733766791	7.8141649590576936E-005
23.000000000000000	51.606675567764377	51.603052607539681	7.0203325148098907E-005
24.000000000000000	54.784729398112319	54.781257376729350	6.3375714749597394E-005
25.000000000000000	58.003605222980518	58.000272067343786	5.7464628688476174E-005
26.000000000000000	61.261701761002001	61.258496790773954	5.2316049602260062E-005
27.000000000000000	64.557538627006338	64.554452348323736	4.7806634952946398E-005
28.000000000000000	67.889743137181540	67.886767073197987	4.3836724754421305E-005
29.000000000000000	71.257038967168015	71.254165517805660	4.0325130036325496E-005
30.000000000000000	74.658236348830158	74.655458673900412	3.7205204215749244E-005

Aqui é interessante notar que a aproximação de stirling é razoável, apresentando erro apenas na segunda e terceira casa decimal para os valores calculados.

2.2- Série de Taylor para o cosseno

A expansão em série de Taylor de uma função f (x) ao redor de x0 é dada por :

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} f^{(n)}(x_0)(x - x_0)^n = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \dots \qquad (2)$$

Escreva um programa FORTRAN que, dado x ∈ R, calcule o valor de cos(x) com acurácia absoluta e relativa de 10^−6 por meio de sua série de Taylor ao redor de x0 = 0.

Resolvendo a expansão em série de Taylor para cosseno ao redor de x0=0, obtém-se a seguinte aproximação:

$$cos(x) = \sum_{i=0}^{\infty} (-1)^i \cdot \frac{x^{2i}}{(2i)!} \quad \textbf{(3)}$$

No Fortran, então, utiliza-se de um laço do while até que seja obtida a precisão de 10^-6.

Assim, vai-se adicionando um novo valor de i ao somatório do cosseno até que esse valor enquanto esse valor seja maior que 10^-6. Quando o valor do incremento for menor do que 10^-6, para-se o cálculo e, então, tem-se um valor de cosseno com precisão nessa casa decimal obtido pela série de Taylor. Vale notar também que o Fortran utiliza o valor de "x" em radianos.

```
program cosseno
    implicit none
    real(8) valor,incremento,i,fact,x

    read(*,*)x

    i = 1.0d0
    fact = 1.0d0
    valor = 1.0d0
    incremento = 1

    do while (abs(incremento) >= 0.000001)
        fact = (2*i)*(2*i - 1)*fact
        incremento = (-1)**i *  x**(2*i)/fact
        valor = valor + incremento
        i = i + 1.0d0
    enddo

    write(*,*)valor

end program cosseno
```

(a) Considere ao menos 5 valores de x ∈]0, π[(mais ou menos igualmente espaçados) e identifique a ordem da expansão para que as precisões exigidas sejam alcançadas. Apresente sua resposta em forma de uma tabela.

Serão utilizados os valores 0.5 , 1.0 , 1.5 , 2.0 e 2.5 para esse cálculo. Utiliza-se o código acima, modificando-o com um novo loop onde "x" toma esses

valores e imprime-se em um arquivo o valor e o número de loops que foram necessários (“x” e “i” respectivamente). Vale notar que dessa vez não será necessário computar o valor, visto que busca-se apenas a ordem da expansão(número de loops).

É interessante expor também que o valor de “i” obtido no final é o número de operações feitas no somatório, assim, por exemplo, a primeira operação começa com i=0 e a última com i=4, o valor final obtido será 5, visto que terão sido feitas 5 operações.

```
program cosseno
    implicit none
    real(8) valor,incremento,i,fact,x
    integer j

    open(10,file='ordemexpansao.txt')

    do j=1,5

        x = real(j,8) * 0.5

        i = 1.0d0
        fact = 1.0d0
        incremento = 1

        do while (abs(incremento) >= 0.000001)
            fact = (2*i)*(2*i - 1)*fact
            incremento = (-1)**i *  x**(2*i)/fact
            i = i + 1.0d0
        enddo

        write(10,*)x,i
    enddo

end program cosseno
```

São obtidos então os números de operações feitas para cada valor conforme dito anteriormente:

0.50000000000000000	**5.0000000000000000**
1.0000000000000000	**6.0000000000000000**
1.5000000000000000	**7.0000000000000000**

2.0000000000000000 8.0000000000000000
2.5000000000000000 9.0000000000000000

(b) Apresente os seus resultados agora em forma de gráfico. Considere um número razoavelmente maior de valores de x ∈ [0, π].

Para tal, altera-se o loop j indo de 1 até 30. Assim, dessa vez, “x” será o valor real de “j” multiplicado por 0.1. Dessa forma serão obtidos o número de operações para os valores de 0.1 , 0.2 , 0.2 , 0.4 , … , 3.0 . Muda-se o nome do arquivo onde será salvo o resultado para ‘ordemgraf.txt’

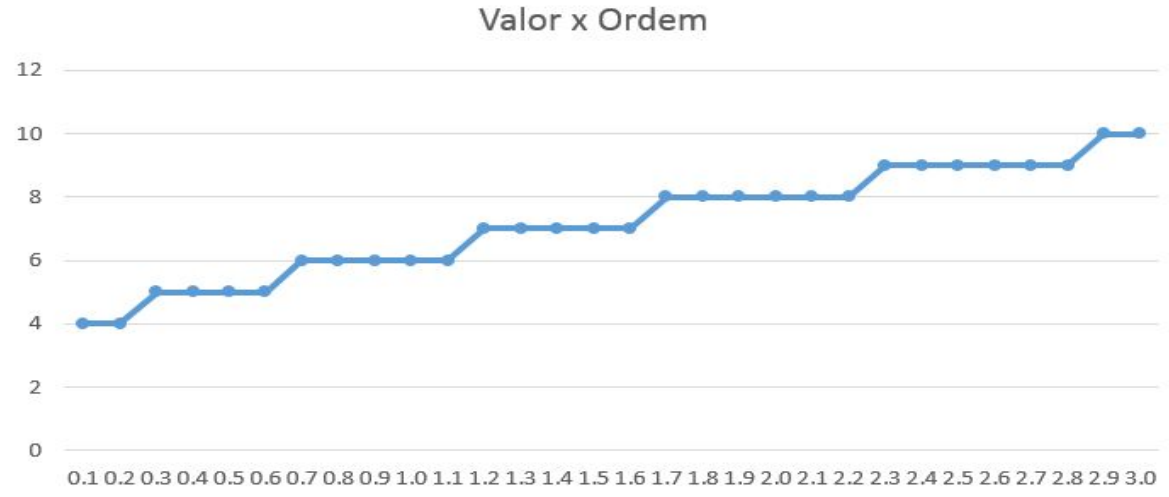
```
program cosseno
    implicit none
    real(8) valor,incremento,i,fact,x
    integer j

    open(10,file='ordemegraf.txt')

    do j=1,30

        x = real(j,8) * 0.1

        i = 1.0d0
        fact = 1.0d0
        incremento = 1

        do while (abs(incremento) >= 0.000001)
            fact = (2*i)*(2*i - 1)*fact
            incremento = (-1)**i *  x**(2*i)/fact
            i = i + 1.0d0
        enddo

        write(10,*)x,i
    enddo

end program cosseno
```

Então, são obtidos os seguintes resultados:

0.10000000149011612 4.0000000000000000
0.20000000298023224 4.0000000000000000
0.30000000447034836 5.0000000000000000

0.40000000596046448	5.0000000000000000
0.50000000745058060	5.0000000000000000
0.60000000894069672	5.0000000000000000
0.70000001043081284	6.0000000000000000
0.80000001192092896	6.0000000000000000
0.90000001341104507	6.0000000000000000
1.0000000149011612	6.0000000000000000
1.1000000163912773	6.0000000000000000
1.2000000178813934	7.0000000000000000
1.3000000193715096	7.0000000000000000
1.4000000208616257	7.0000000000000000
1.5000000223517418	7.0000000000000000
1.6000000238418579	7.0000000000000000
1.7000000253319740	8.0000000000000000
1.8000000268220901	8.0000000000000000
1.9000000283122063	8.0000000000000000
2.0000000298023224	8.0000000000000000
2.1000000312924385	8.0000000000000000
2.2000000327825546	8.0000000000000000
2.3000000342726707	9.0000000000000000
2.4000000357627869	9.0000000000000000
2.5000000372529030	9.0000000000000000
2.6000000387430191	9.0000000000000000
2.7000000402331352	9.0000000000000000
2.8000000417232513	9.0000000000000000
2.9000000432133675	10.000000000000000
3.0000000447034836	10.000000000000000

Ao plotar esses valores em um gráfico de valor por ordem de expansão, obtém-se:



2.3 - Valores médios e desvio padrão

No arquivo votes.dat (que deve ser baixado em http://www.ifsc.usp.br/~hoyos/courses/2019/7600017/criterio-avaliacao.html) há uma uma população de N = 10^6 números inteiros xi, i = 1, · · · , N que designam os votos válidos de uma eleição. Há apenas dois candidatos: 0 e 1.

(a) Calcule a média aritmética da população. Qual o candidato eleito?

```
program media
    implicit none
    real(8) resul,voto
    integer i
    open(10,file='votes.dat')

    resul = 0

!calcula-se a media dos votos somando-os e depois dividindo
pelo número de votos

    do i=1,10**6
        read(10,*) voto
        resul = resul + voto
    enddo

    resul = resul/10**6

!imprime-se entao, no compilador o valor da media dos votos e
o resultado da eleicao

    write(*,*) resul

    if (resul>0.5) then
        write(*,*) "O Candidato 1 ganhou a eleição"
    else if (resul < 0.5) then
        write(*,*) "O Candidato 0 ganhou a eleição"
    else
        write(*,*) "Eleição empatada"
    endif

end program media
```

Ao rodar o código, obtém-se o resultado de que o candidato 1 foi eleito e tem-se uma média aritmética de 0,52.

(b) Sorteie aleatoriamente k = 100 amostras de tamanho M ≤ N da população e calcule a média de cada amostra.
(c) Calcule o valor médio dessas amostragens e o desvio padrão não enviesado correspondente .

```
program sorteio
    implicit none
    real(8) inicio,final,transitorio
    real(8) resul,voto,media,desvio
    real(8), dimension(100) :: resultados
    integer i,j,k
    open(20,file='mediassorteadas.txt')


    do i=1,100

!serao chamados dois numeros aleatorios que serao ordenados(inicio eh menor e final eh maior)
!eu optei aqui por sortear tambem o tamanho de cada amostra M<=N para evitar que haja algum viés

        CALL RANDOM_NUMBER(inicio)
        CALL RANDOM_NUMBER(final)

        resul = 0


        if (final < inicio) then
            transitorio = inicio
            inicio = final
            final = transitorio
        else if (final == inicio) then
            final = inicio + 1
        endif


        open(10,file='votes.dat') !chama-se o arquivo com os dados
```

```
!multiplica-se o numero aleatorio entre 0 e 1 pelo valor de
votos para ter o numero inicial e final do conjunto de votos
sorteados

        inicio = NINT(inicio * 10**6)
        final = NINT(final * 10**6)

!realiza-se a soma dos votos e entao a divisao pelo numero
total para o calculo da media, salvando a lista das medias de
votos em um vetor chamado resultados

        do j=1,10**6
            read(10,*) voto
            if (j>=inicio .AND. j<final) then
                resul = resul + voto
            endif
        enddo

        resul = resul/(final-inicio)

        resultados(i) = resul

        close(10)

    enddo

    media = 0
    desvio = 0

!calcula-se entao a media das medias de votos no real chamado
'media'

    do k=1,100
        media = media + resultados(k)
    enddo

    media = media/100

!calcula-se entao o desvio no real chamado 'desvio',
naturalmente por um loop que vai de 1 a 100 e finaliza-se a
operacao
```

```
    do k=1,100
        desvio = desvio + ( resultados(k) - media )**2
    enddo

    desvio = ( desvio/(100-1) ) ** 0.5


!imprime-se entao, em um arquivo chamado 'mediassorteadas.txt'
os resultados das medias de votos, a media geral e o desvio
padrao

    write(20,*)'Medias de votos:',resultados
    write(20,*)'Media geral:',Media
    write(20,*)'Desvio padrao:',desvio

end program sorteio
```

São obtidos os seguintes resultados:

Medias de votos:

0.69454123112659694	0.66972477064220182	0.35840707964601770
0.29197080291970801	0.59999999999999998	0.35294117647058826
0.61927945472249268	0.63048498845265588	0.43228346456692912
0.51468315301391032	0.62004405286343611	0.56073539067629674
0.73154362416107388	0.33464566929133860	0.49312977099236643
0.73127753303964760	0.67849686847599167	0.43398268398268397
0.70731707317073167	0.41666666666666669	0.74761904761904763
0.67105263157894735	0.69014084507042250	0.63022813688212931
0.47103513770180439	0.57558139534883723	0.43430290872617855
0.69565217391304346	0.70449678800856530	0.37634408602150538
0.69325842696629214	0.67418032786885251	0.53341080766995930
0.77777777777777779	0.68257261410788383	0.70254403131115462
0.35202863961813841	0.42608695652173911	0.65925925925925921
0.58247422680412375	0.55555555555555558	0.37190082644628097
0.26530612244897961	0.43965517241379309	0.47750642673521854
0.70491803278688525	0.53782013103037518	0.47396768402154399
0.33333333333333331	0.36363636363636365	0.33717579250720459
0.35319148936170214	0.38709677419354838	0.37777777777777777
0.68133802816901412	0.31200000000000000	0.50000000000000000
0.73575129533678751	0.48663967611336034	0.90909090909090906
0.66839378238341973	0.71794871794871795	0.60468750000000004

0.56842105263157894 0.44444444444444442 0.71052631578947367
0.57763061074319355 0.66099290780141839 0.68224299065420557
0.54602675059008654 0.55242390078917702 0.71028037383177567
0.73333333333333328 0.59862778730703259 0.60219594594594594
0.54320987654320985 0.35323383084577115 0.35439560439560441
0.72340425531914898 0.35737704918032787 0.32608695652173914
0.63741134751773054 0.39056831922611851 0.68674698795180722
0.71666666666666667 0.68287526427061307 0.53660012099213550
0.68376068376068377 0.31515151515151513 0.33038348082595870
0.57142857142857140 0.69762845849802368 0.32936507936507936
0.40769230769230769 0.52941176470588236 0.33858267716535434
0.43320235756385067 0.62828282828282833 0.54545454545454541
0.54086088583905179

Media geral: 0.54593820913973956
Desvio padrao: 0.14708768035106276

(d) Variando M de 10 até 10^4, mostre que sM ≈ σ0/√M para M >> 1. Isso é consequência do teorema central do limite.

obs: Não entendi muito bem o que o exercício pede, então não consegui fazer a (d) e a (e).

(e) Para qual valor de M poderemos dizer que a média de apenas 1 amostra tem 90% de chance de acertar o resultado?

(f) (Opcional) Qual a importância de se fazer uma amostragem aleatória? Refaça sua análise considerando amostras enviesadas obtidas sorteando apenas a primeira metade da população.

Fazendo a análise com apenas a primeira metade, obtém-se um resultado diferente onde o candidato 0 é eleito, além de obter-se um desvio padrão menor. Dessa forma, mostra-se que é possível manipular os dados facilmente.

Foi rodado o código dos itens (b) e (c) mudando o alcance de 10**6 para 10**3 e o seguinte resultado foi obtido:

Medias de votos:
0.36522974893415444 0.35580524344569286 0.34414414414414413
0.32835820895522388 0.29166666666666669 0.34855769230769229

0.35941223193010324 0.35566037735849054 0.34855305466237940
0.35101010101010099 0.35563538392456218 0.35075026795284031
0.36885245901639346 0.34083601286173631 0.35295950155763239
0.34594594594594597 0.36348122866894200 0.34719150818222028
0.34653465346534651 0.35183900069396251 0.35797665369649806
0.34677419354838712 0.35446685878962536 0.35492629945694337
0.34947694691979853 0.35863377609108160 0.34766584766584768
0.35483870967741937 0.36069868995633186 0.37280701754385964
0.36176066024759285 0.36204013377926419 0.35341555977229600
0.42528735632183906 0.36525423728813561 0.36690647482014388
0.34259259259259262 0.34545454545454546 0.35454545454545455
0.35659186535764376 0.34782608695652173 0.34569983136593591
0.33884297520661155 0.35135135135135137 0.35213854631330360
0.35914552736982641 0.35302698760029178 0.34970674486803521
0.34259259259259262 0.36283185840707965 0.33490011750881316
0.34027777777777779 0.34298608499435879 0.31818181818181818
0.36017253774263119 0.34747145187601958 0.35280494081317548
0.36363636363636365 0.34909090909090912 0.19230769230769232
0.36286919831223630 0.36119402985074628 0.35309114085404714
0.35161290322580646 0.42857142857142855 0.36428571428571427
0.35664965475833083 0.36421540243196293 0.36450381679389315
0.35292228644829798 0.35572940635066730 0.36398467432950193
0.42342342342342343 0.35749299719887956 0.35849706997587039
0.37279596977329976 0.33604060913705586 0.34157303370786518
0.35930735930735930 0.33823529411764708 0.30473372781065089
0.35685848715164675 0.34336457819437594 0.33168316831683170
0.35260770975056688 0.36606373815676141 0.35299925944211308
0.35888501742160278 0.30940594059405940 0.34903381642512077
0.35699630693475587 0.36914378029079159 0.32414910858995138
0.35736677115987459 0.32558139534883723 0.34991974317817015
0.34895833333333331 0.35737840065952187 0.35245263431092655
0.35243188184364654

Media geral: 0.35156538712938207
Desvio padrao: 2.4502462606049252E-002

2.4- Organize uma lista

(a) Escreva um programa que leia os primeiros N ≤ 10^4 números reais do arquivo Rnumbers.dat (que deve ser baixado em

http://www.ifsc.usp.br/~hoyos/courses/2019/7600017/criterio-avaliacao.html) e os armazene como um vetor. O seu programa deve perguntar e ler qual o valor de N no terminal.
(b) Em seguida, o seu programa deve ordenar, em ordem crescente, os M menores números, com M ≤ N. Novamente, o seu programa deve perguntar e ler o valor de M no terminal.
(c) Finalmente, o seu programa deve imprimir o resultado em um arquivo chamado menores.dat. Submeta o resultado para o caso N = 10*3 e M = 20.

Para esse exercício utiliza-se um único código contendo uma subrotina recursiva:

```
program vetor
    implicit none
    integer N,M,i
    real(8), dimension(10**4) :: numeros

!sera lida uma entrada de um inteiro 'N'
    write(*,*)'Insira o valor de N'
    read(*,*) N

    open(10,file='Rnumbers.dat')
    open(20,file='menores.dat')

!utiliza-se de um loop de 1 a 'N' que eh salvo num vetor
chamado 'numeros' para ler os numeros do arquivo
'Rnumbers.dat'

    do i=1,N

        read(10,*) numeros(i)

    enddo

!chama-se uma subrotina chamada quicksort muito usada no
fortran para ordenar o vetor numeros

    call quicksort(numeros,1,N)

!basta entao receber o valor de M e imprimir os M primeiros
numeros no arquivo
```

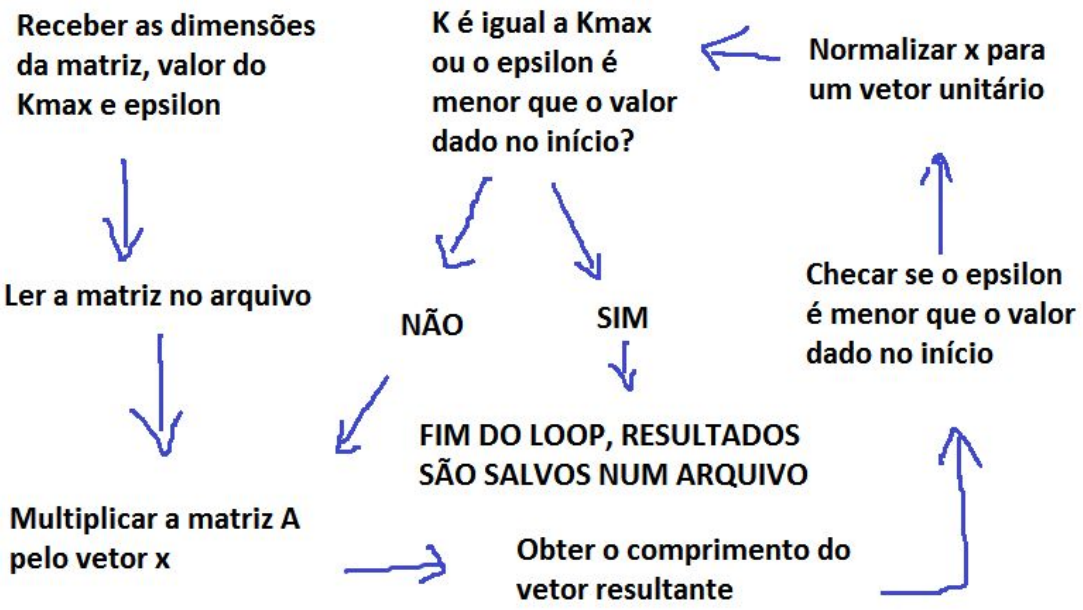
```
    write(*,*)'Insira o valor de M'
    read(*,*)M

    do i=1,M
        write(20,*) numeros(i)
    enddo


end program vetor



recursive subroutine quicksort(a, first, last)
  implicit none
  real*8  a(*), x, t
  integer first, last
  integer i, j

  x = a( (first+last) / 2 )
  i = first
  j = last
  do
     do while (a(i) < x)
        i=i+1
     end do
     do while (x < a(j))
        j=j-1
     end do
     if (i >= j) exit
     t = a(i);  a(i) = a(j);  a(j) = t
     i=i+1
     j=j-1
  end do
  if (first < i-1) call quicksort(a, first, i-1)
  if (j+1 < last)  call quicksort(a, j+1, last)
end subroutine quicksort
```

Utilizando-se os valores pedidos no exercícios obtém-se os seguintes valores salvos no arquivo ‘menores.dat’:

4.3118330650031567E-003
4.3614865280687809E-003
5.0199292600154877E-003
5.1138941198587418E-003
5.4667252115905285E-003
5.5907065980136395E-003
6.3367416150867939E-003
8.2659986801445484E-003
8.5781202651560307E-003
8.7963747791945934E-003
9.0342182666063309E-003
1.0173494927585125E-002
1.0968453716486692E-002
1.1085866950452328E-002
1.1366684921085835E-002
1.1606859043240547E-002
1.1813488788902760E-002
1.3223906978964806E-002
1.4952653087675571E-002
1.6753977164626122E-002

2.5- Método da potência para o cálculo do autovalor/autovetor dominante

(a) Explique em detalhes a implementação desse algoritmo por meio de um fluxograma.



(b) Escreva agora um código FORTRAN que implemente esse algoritmo. Seu código deve possuir uma precisão relativa ε para a convergência do autovalor dominante $\lambda 1$ bem como um número máximo de iterações kmax, acima do qual ele termina, mesmo que a tolerância ε ainda não tenha sido alcançada.

```
program potencia

    implicit none
    integer k,kmax,n
    integer i,j,l
    real(8) eps,norm,anterior
    real(8),dimension(10,10) :: A
    real(8),dimension(10) :: resul,x,resulsort,xsort
```

```
    write(*,*) 'Insira o valor de Kmax'
    read(*,*) kmax
    write(*,*) 'Insira o valor de epsilon(eps)'
    read(*,*) eps
    write(*,*) 'Insira o numero de linhas da matirz '
    read(*,*) n

      !aqui sera lida uma matriz 3x3, eh preciso mudar as
entradas para uma matriz de tamanho diferente
    open(10,file='matriz.txt')

    read(10,*) A(0,0) , A(0,1) , A(0,2)
    read(10,*) A(1,0) , A(1,1) , A(1,2)
    read(10,*) A(2,0) , A(2,1) , A(2,2)


    k=0

    do i=0,n-1
        x(i) = 1
    enddo

    do while (k<kmax)
        !multiplica-se a matriz A pelo vetor x
        do i=0,n-1
            resul(i) = 0
            do j=1,n-1
                  resul(i) = resul(i) + A(i,j) * x(j) !produto
da coluna pela linha
            enddo
        enddo

         !obter o comprimento do vetor resultante(resul) e do
vetor anterior a multiplicacao(x) e checar a precisao relativa
eps
        norm = 0
        anterior = 0
        do l=0,n-1
            norm = norm + resul(l)*resul(l)
            anterior = anterior + x(l)*x(l)
        enddo
```

```
        norm = norm**0.5
        anterior = anterior**0.5

        if ((norm-anterior)**2 < eps**2) then
            k = kmax
        endif


           !normalizar x para um vetor unitario para proxima
iteracao
        do l=1,n-1
            x(l) = resul(l)/norm
        enddo

        k = k+1

    enddo

    open(20,file='resultado.txt')

    write(20,*)'Autovalor:',norm
    write(20,*)'Autovetor:'
    do i=0,n-1
        write(20,*),resul(i)
    enddo
```

end program potencia

(c) Discuta também seu palpite para ~x0 (o valor inicial do vetor ~x).

Utilizou-se um vetor unitário para x0, de forma que seja mais fácil realizar os loops e chegar enfim no valor do autovalor/autovetor respectivo.

(d) Aplique seu algoritmo para as seguintes matrizes simétricas usando $\varepsilon = 10{-}5$:

Para a matriz 3x3, obtém-se:

Autovalor: 9.1284500549479652

Autovetor:
7.2060375857767189
2.7022640946662695
4.9091131053103902

Para a matriz 4x4, obtém-se:

Autovalor: 7.1809172855632912
Autovetor:
-3.4648871730421207
3.5933421339389824
-3.1950917361827091
4.05455435849929

Para a matriz 5x5, obtém-se:

Autovalor: 13.589056427387719
Autovetor:
-8.8127418933165447E-003
-10.424271680942564
-0.50258901557601310
-7.7648693480856039
-3.9307944658541620

3-Conclusão

Por fim, terminado o projeto, foi possível utilizar o Fortran 95 para resolver problemas, desde os mais simples como o fatorial, até outros mais complexos como o método das potências para encontrar o autovalor dominante. Pode-se concluir então que o Fortran é uma ferramenta boa para realizar computação científica, sendo excelente na alocação de memória, definição de variáveis e uma ferramenta poderoso para realizar contas grandes, o que mostra que é razoável esperar que a linguagem tenha um bom rendimento para o cálculo numérico.

Referênias:

ONLINE GDB . <https://www.onlinegdb.com/online_fortran_compiler>

AZEREDO, Paulo A. (1996). *Métodos de Classificação de Dados e Análise de suas Complexidades*. <Método quicksort>