7600017 - Introdução à Física Computacional
Projeto 1: Introdução à programação

Prof. Dr. Eric de Castro e Andrade

Luis Rodrigo Torres Neves
$n_{\text{USP}} = 10260563$

São Carlos, 2018

1 Introdução

A computação é uma ferramenta indispensável para a ciência nos dias de hoje, em particular para a Física. Diversos prolemas, devido à sua complexidade numérica, exigem uma abordagem computacional. Uma das linguagens de programação utilizadas para este fim é o `FORTRAN`, que estudaremos ao longo deste semestre.

Neste primeiro projeto, abordaram-se alguns problemas básicos de programação científica, a fim de começar a desenvolver alguma familiaridade com o `FORTRAN 90`. Este relatório apresenta o planejamento de cada programa, suas principais linhas de código e os resultados numéricos obtidos.

É importante salientar que os programas mencionados neste texto estarão reproduzidos apenas parcialmente, com enfoque no que for relevante para cada explicação. Assim, comandos como `end program`, `implicit none` e declarações de variáveis serão quase invariavelmente omitidos.

2 Métodos

2.1 Aproximação de Stirling

(a) Calculando e imprimindo fatoriais

Para calcular os fatoriais dos inteiros entre 1 e 20, usamos a definição recursiva:

$$n! = \begin{cases} 1, & n = 0; \\ n(n-1)!, & n > 0. \end{cases} \tag{1}$$

Em linguagem FORTRAN 90, isso se escreve da seguinte forma:

```
integer i
real*8 fac

open(1, file = 'problema1a.dat')
fac = 1.0d0

do i = 1, 20
fac = fac*real(i, 8)
write(1,*) fac
end do

close(1)
```

Optou-se por declarar a variável `fac` como real pois assim obtém-se uma precisão maior no cálculo. Pelo mesmo motivo, no cálculo iterativo do fatorial, o inteiro `i` foi previamente convertido para real com precisão dupla.

(b) Calculando e imprimindo logaritmos de fatoriais

As únicas modificações entre este programa e o do subitem anterior são: i) a mudança no intervalo de iteração, para atuar nos inteiros entre 2 e 30; ii) o cálculo do logaritmo do fatorial antes de sua impressão, utilizando-se a função intrínseca `log()`. Assim, o laço anterior passa a ser escrito como:

```
open(1, file = 'problema1b.dat')
fac = 1.0d0

do i = 2, 30
fac = fac*real(i, 8)
write(1,*) log(fac)
end do

close(1)
```

Note que não foi necessário atualizar o valor inicial da variável `fac` pois $1! = 0! = 1$.

(c) Trabalhando com a aproximação de Stirling

A fórmula de Stirling dá uma aproximação para os fatoriais de inteiros grandes n:

$$\ln n! \approx n \ln n - n + \frac{1}{2} \ln (2\pi n) \tag{2}$$

O que, em FORTRAN 90, escreve-se

```
approx = i*log(real(i,8)) - i + 0.5d0*log(2*pi*real(i,8))
```

onde as variáveis `approx` e `pi` são reais (precisão dupla), tendo-se atribuído previamente `pi = 4*atan(1.0d0)`.

O laço completo, incluindo a saída conforme pedido no enunciado do exercício, fica como abaixo:

```
open(1, file = 'problema1c.dat')
fac = 1.0d0

do i = 2, 30
fac = fac*real(i, 8)
approx = i*log( real(i,8) ) - i + 0.5d0*log(2*pi*real(i,8))
write(1,*) i, log(fac), approx, (log(fac) - approx)/log(fac)
end do

close(1)
```

2.2 Série de Taylor para o cosseno

A série de Taylor da função cosseno, em torno de $x_0 = 0$, é dada por

$$\cos x = \sum_{i=0}^{\infty} f_i(x), \tag{3}$$

onde

$$f_i(x) = (-1)^i \frac{x^{2i}}{(2i)!} \tag{4}$$

Para realizar tal cálculo, com precisão de 10^{-6}, implementou-se um laço condicionado pelo valor absoluto do incremento (dado por $f_i(x)$ acima) ser maior que ou igual a 10^{-6}.

```
fac = 1.0d0
approx = 1.0d0
inc = 1.0d0
i = 1

do while (abs(inc) >= 1e-6)

fac = 2*i*(2*i-1)*fac
inc = (-1)**i) * (x**(2*i)) / fac
approx = approx + inc
write(2,*) i, approx, inc
i = i + 1

end do
```

Onde usou-se ainda o fato matemático de que $(2i)! = (2i)(2i-1)(2[i-1])!$. O valor inicial da variável `inc` é essencialmente irrelevante para os cálculos; apenas é necessário inicializá-la com um valor suficientemente grande para que se entre no laço, por isso optou-se por fazê-la igual a 1.

Cabe observar que, pela forma como o código foi escrito, o valor final da variável `i` corresponde a um a mais que o real número de iterações realizadas. Consideramos, ainda, que tal número de iterações é equivalente à "ordem da expansão para que a precisão seja atingida", pois, efetivamente, o valor inicial `approx = 1.0d0` corresponde a $i = 0$ e nenhuma iteração.

Após gravar no arquivo de saída cada iteração, fez-se imprimir o valor final da aproximação, bem como o número de iterações para que tal valor fosse atingido:

```
write(2,*) (i - 1) , "iterações, cos(", x, ")  =", approx
```

Para a leitura das entradas, optou-se por armazenar os 5 valores de x em um vetor. Portanto, a entrada é um arquivo texto contendo cinco números reais separados por vírgulas. A declaração do vetor e a leitura são feitas como segue:

```
real*8 vec(5)

open(1, file = 'problema2.in')
read(1,*) vec
close(1)
```

Desta maneira, a variável `x` que aparece nas linhas de código (`...`) representadas acima é atribuída a partir da iteração sobre `vec`:

```
do k = 1, 5
x = vec(k)

...

end do
```

2.3 Desvio padrão e média

Inicialmente, vamos provar que o desvio padrão σ de uma lista $(x_i, ..., x_N)$ pode ser expresso como $\sqrt{\langle x^2 \rangle - \langle x \rangle^2}$. Da definição,

$$\sigma^2 = \sum_{i=1}^{N} \frac{(x_i - \langle x \rangle)^2}{N} \tag{5}$$

Expandindo o binômio e usando operações básicas do somatório, teremos

$$\sigma^2 = \sum_{i=1}^{N} \frac{x_i^2 - 2x_i \langle x \rangle + \langle x \rangle^2}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i^2 - 2\langle x \rangle \cdot \frac{1}{N}\sum_{i=1}^{N} x_i + \frac{1}{N}\sum_{i=1}^{N} \langle x \rangle^2 \tag{6}$$

É fácil perceber que

$$\frac{1}{N}\sum_{i=1}^{N} x_i^2 = \langle x^2 \rangle, \quad \frac{1}{N}\sum_{i=1}^{N} x_i = \langle x \rangle, \quad \sum_{i=1}^{N} \langle x \rangle^2 = N \cdot \langle x \rangle^2 \tag{7}$$

Assim,

$$\sigma^2 = \langle x^2 \rangle - 2\langle x \rangle \cdot \langle x \rangle + \frac{1}{N} \cdot N \cdot \langle x \rangle^2 \Rightarrow \sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$$

$$\therefore \boxed{\sigma = \sqrt{\langle x^2 \rangle - \langle x \rangle^2}} \tag{8}$$

Neste programa, deixou-se o número `n` de elementos da lista como parâmetro a ser definido em uma das primeiras linhas de código. Portanto, para fazê-lo operar com uma lista de comprimento diferente, deve-se alterar esta linha e recompilar o programa. Por exemplo, para $n = 6$, tem-se:

```
integer n
parameter (n = 6)
real*8 lista(n)
```

Assim como no problema anterior, a entrada é um arquivo de texto contendo números reais separados por vírgulas. A leitura se dá como segue:

```
open(1, file = 'problema3.in')
read(1,*) lista
close(1)
```

O valor médio foi armazenado na variável real de precisão dupla `xmed`, segundo o laço reproduzido abaixo:

```
xmed = 0
do i = 1, n
xmed = xmed + lista(i)/n
end do
```

Procedeu-se de maneira similar para o cálculo do desvio padrão. Inicialmente, através de um laço, calculou-se a variância `var` (cf. definição (5)), da qual em seguida extraiu-se a raiz quadrada:

```
var = 0
do i = 1, n
```

```
var = var + ((lista(i) - xmed)**2)/n
end do
devpad = sqrt(var)
```

2.4 Ordenação de um vetor

O algoritmo usado para a ordenação da lista de números foi o seguinte:

Seja x_i um elemento da lista. Seja x_j outro elemento da lista, com $j < i$. Se $x_j > x_i$, trocamos as posições destes elementos. (Para realizar a troca, é necessário armazenar um dos valores em uma variável auxiliar.) Procede-se desta maneira para todos os j entre 1 e i, e repete-se o processo para todos os i. Portanto, são necesários dois laços aninhados. O código FORTRAN fica como abaixo:

```
do i = 1, n

do j = 1, i

if (vec(i) < vec(j)) then
aux = vec(j)
vec(j) = vec(i)
vec(i) = aux
end if

end do

end do
```

Agora, com a lista já em ordem crescente, para imprimir somente os `m` menores valores, basta iterar sobre a lista entre 1 e m:

```
open(2, file = 'problema4.dat')
do i = 1, m
write(2,*) vec(i)
end do
```

2.5 Cálculo do autovalor dominante de uma matriz

Conforme elucidado no enunciado do problema, o método da potência para o cálculo do autovalor dominante de uma matriz $\mathbf{A}_{n\times n}$ é um processo iterativo descrito pelo seguinte algoritmo:

Toma-se um vetor $\vec{x}_0$, de dimensão n. (Este vetor, em princípio, precisa satisfazer uma certa condição; como ela não é muito restritiva, geralmente pode-se tomar 1 em todas as entradas.) Aplica-se, então, a seguinte iteração, com i inicialmente valendo 0:

$$\vec{x}_{i+1} = \mathbf{A} \cdot \vec{x}_i \tag{9}$$

$$a_{i+1} = \frac{\vec{x}_i \cdot \vec{x}_{i+1}}{\vec{x}_i \cdot \vec{x}_i} \tag{10}$$

Como demonstrado no enunciado, a sequência dos a_i converge para o autovalor dominante λ_1 de $\mathbf{A}$.[1]

Para a implementação desse algoritmo em FORTRAN, declararam-se duas variáveis do tipo vetor, `x1` e `x2`, para armazenar, a cada iteração, x_i e x_{i+1}, respectivamente. Da mesma forma, os reais `a1` e `a2` foram usados para armazenar a_i e a_{i+1}. Assim foi possível, a cada iteração, avaliar a diferença $|a_i - a_{i+1}|$, a qual serve de critério de parada quando se torna menor que ε (tolerância determinada no próprio código).

A implementação deste algoritmo iterativo está esquematizada no fluxograma em anexo. O código FORTRAN foi escrito da seguinte forma:

```
do i = 1, n
x0(i) = 1.0d0
end do

eps = 1e-12
k = 1

x1 = matmul(A, x0)
x2 = matmul(A, x1)
a2 = dot_product(x1,x2)/dot_product(x1, x1)

do while (k <= k_max)

x1 = x2
x2 = matmul(A, x1)

a1 = a2
a2 = dot_product(x1, x2)/dot_product(x1, x1)

open(2, file = 'problema5.dat')
write(2,*) x3
write(2,*) a2
write(2,*) ''

k = k + 1
if (k == k_max) then
fim = k
end if
```

[1] *Esclarecimento sobre a notação:* em (9), o ponto $(\cdot)$ denota o produto entre uma matriz e um vetor e, portanto, $\vec{x}_{i+1}$ e $\vec{x}_i$ devem ser pensados como vetores-coluna. Já em (10), o mesmo símbolo aparece entre dois vetores e representa simplesmente um produto interno.

```
if (abs(a2 - a1) < eps) then
fim = k - 1
exit
end if

end do
```

O inteiro `k` foi declarado para armazenar o número total de iterações. Essa informação é gravada no arquivo de saída:

```
write(2,*)'foram necessárias', fim, 'iterações'
close(2)
```

Salienta-se que, assim como no problema 2, o contador `k` acaba contando um a mais que o número real de iterações (a menos que se iguale a `k_max`, não entrando novamente no laço).

A entrada é um arquivo de texto contendo a matriz `A`, cada linha contendo os elementos separados por espaços. A leitura, então, é feita da seguinte forma:

```
open(1, file = 'problema5.in')
do i = 1, n
read(1,*) (A(i, j), j = 1, n)
end do
close(1)
```

O parâmetro `n` é declarado no início do programa e, para mudar a dimensão da matriz de entrada, deve-se atualizar o código e recompilá-lo. O mesmo vale para o inteiro `k_max`.

3 Resultados e discussão

3.1 Aproximação de Stirling

3.1.1 Imprimindo fatoriais

O resultado para os fatoriais de 1 a 20 está reproduzido abaixo:

```
1.0000000000000000
2.0000000000000000
6.0000000000000000
24.000000000000000
120.00000000000000
720.00000000000000
5040.0000000000000
40320.000000000000
362880.00000000000
3628800.0000000000
```

```
39916800.000000000
479001600.00000000
6227020800.0000000
87178291200.000000
1307674368000.0000
20922789888000.000
355687428096000.00
6402373705728000.0
1.2164510040883200E+017
2.4329020081766400E+018
```

Para os quatro maiores valores, fez-se o cálculo utilizando o Wolfram|Alpha e verificou-se concordância entre os resultados. De fato, a precisão dupla do FORTRAN comporta certeza até a quinta casa decimal e, portanto, considerando a quantidade de algarismos significativos nos resultados acima, era de se esperar que não houvesse erro.

3.1.2 Calculando e imprimindo logaritmos de fatoriais

O resultado para os logaritmos dos fatoriais dos inteiros entre 2 e 30 foi:

```
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
```

3.1.3 Trabalhando com a aproximação de Stirling

Na saída abaixo, as colunas representam, respectivamente: o inteiro `i`; o logaritmo do fatorial de `i`, calculado diretamente; a aproximação de Stirling para este inteiro, e o erro relativo cometido na aproximação.

```
2 0.69314718055994529 0.65180648460453594 5.9642017041767491E-002
3 1.7917594692280550 1.7640815435430568 1.5447344445693184E-002
4 3.1780538303479458 3.1572631582441804 6.5419508962468237E-003
5 4.7874917427820458 4.7708470515922246 3.476703895085761 4E-003
6 6.5792512120101012 6.5653750831870310 2.1090741751492960E-003
7 8.5251613610654147 8.5132646511195222 1.3954820843890348E-003
8 10.604602902745251 10.594191637483277 9.8176851669556312E-004
9 12.801827480081469 12.792572017898756 7.2297976184366628E-004
10 15.104412573075516 15.096082009642156 5.5153177212658908E-004
11 17.502307845873887 17.494734170385932 4.3272439010034666E-004
12 19.987214495661885 19.980271655554681 3.4736406659921081E-004
13 22.552163853123425 22.545754858935421 2.8418533271324276E-004
14 25.191221182738680 25.185269812625918 2.3624778130404078E-004
15 27.899271383840890 27.893716650288930 1.9909959208389360E-004
16 30.671860106080672 30.666652450161063 1.6978611344723750E-004
17 33.505073450136891 33.500172054188461 1.4628817202040308E-004
18 36.395445208033053 36.390816054283718 1.2719046910608648E-004
19 39.339884187199495 39.335498626950255 1.1147872800975242E-004
20 42.335616460753485 42.331450141061481 9.8411693044942686E-005
21 45.380138898476908 45.376170944258263 8.7438124143291159E-005
22 48.471181351835227 48.467393733766791 7.8141649590576936E-005
23 51.606675567764377 51.603052607539681 7.0203325148098907E-005
24 54.784729398112319 54.781257376729350 6.3375714749597394E-005
25 58.003605222980518 58.000272067343786 5.7464628688476174E-005
26 61.261701761002001 61.258496790773954 5.2316049602260062E-005
27 64.557538627006338 64.554452348323736 4.7806634952946398E-005
28 67.889743137181540 67.886767073197987 4.3836724754421305E-005
29 71.257038967168015 71.254165517805660 4.0325130036325496E-005
30 74.658236348830158 74.655458673900412 3.7205204215749244E-005
```

Ao menos dois fatos interessantes podem ser observados no resultado acima:

(a) a diferença relativa, cf. seç. 2.1 (c), foi calculada a partir da diferença `log(fac) - approx`, sem se tomar o valor absoluto. Assim mesmo, é fácil perceber que todos os valores da última coluna são positivos, revelando que, efetivamente, a fórmula de Stirling é uma aproximação *por falta*;

(b) conforme era de se esperar, os valores da última coluna (erro relativo) se tornam menores a cada iteração, pois a aproximação é tanto melhor quanto maior o inteiro `i`. É isso que significa dizer que a aproximação de Stirling vale para "inteiros grandes".

3.2 Série de Taylor para o cosseno

Deu-se como entrada um arquivo texto contendo uma única linha, reproduzida abaixo:

```
.5, 3, -2, .8, 5
```

A saída do programa foi:

```
1 :  x = 0.50000000000000000

1 0.87500000000000000 -0.12500000000000000
2 0.87760416666666663 2.6041666666666665E-003
3 0.87758246527777772 -2.1701388888888890E-005
4 0.87758256215897812 9.6881200396825397E-008

4 iterações, cos( 0.50000000000000000 ) = 0.87758256215897812
.............................

2 :  x = 3.0000000000000000

1 -3.5000000000000000 -4.5000000000000000
2 -0.12500000000000000 3.3750000000000000
3 -1.1375000000000000 -1.0125000000000000
4 -0.97477678571428572 0.16272321428571429
5 -0.99104910714285710 -1.6272321428571428E-002
6 -0.98993963068181812 1.1094764610389610E-003
7 -0.98999449490241898 -5.4864220600827741E-005
8 -0.98999243749414645 2.0574082725310405E-006
9 -0.98999249800615452 -6.0512008015618833E-008

9 iterações, cos( 3.0000000000000000 ) = -0.98999249800615452
.............................

3 :  x = -2.0000000000000000

1 -1.0000000000000000 -2.0000000000000000
2 -0.33333333333333337 0.66666666666666663
3 -0.42222222222222228 -8.8888888888888892E-002
4 -0.41587301587301595 6.3492063492063492E-003
5 -0.41615520282186957 -2.8218694885361550E-004
6 -0.41614665170220733 8.5511196622307738E-006
7 -0.41614683963890320 -1.8793669587320381E-007

7 iterações, cos( -2.0000000000000000 ) = -0.41614683963890320
.............................

4 :  x = 0.80000000000000004
```

```
1 0.67999999999999994 -0.32000000000000006
2 0.69706666666666661 1.7066666666666674E-002
3 0.69670257777777778 -3.6408888888888908E-004
4 0.69670673879365075 4.1610158730158767E-006
5 0.69670670920420452 -2.9589446208112906E-008

5 iterações, cos( 0.80000000000000004 ) = 0.69670670920420452
.............................

5 :  x = 5.0000000000000000

1 -11.500000000000000 -12.500000000000000
2 14.541666666666668 26.041666666666668
3 -7.1597222222222214 -21.701388888888889
4 2.5283978174603181 9.6881200396825395
5 -0.16274663800705413 -2.6911444554673722
6 0.34693981189206935 0.50968644989912348
7 0.27692793690592599 -7.0011874986143335E-002
8 0.28422084055031593 7.2929036443899311E-003
9 0.28362501508917298 -5.9582546114296823E-004
10 0.28366421413266923 3.9199043496247913E-005
11 0.28366209297230688 -2.1211603623510775E-006
12 0.28366218903935225 9.6067045396335028E-008

12 iterações, cos( 5.0000000000000000 ) = 0.28366218903935225
.............................
```

Para cada valor de `x`, imprimiu-se, em cada linha, o contador da iteração, a aproximação calculada na mesma e o último incremento utilizado (em módulo).

Observa-se que, em todos os casos, o último valor da última coluna é menor que 10^{-6}, e o penúltimo, maior. Isso mostra que o critério de parada funcionou: iterou-se até que o valor absoluto do incremento se tornasse menor que 10^{-6}, o que garante a precisão exigida, já que a série é sabidamente convergente. Cabe notar que o total de iterações necessárias para que se atingisse esse refinamento foi tanto maior quanto maior o módulo do argumento `x`, o que faz sentido, pois usou-se a expansão em torno de $x = 0$.

3.3 Desvio padrão e média

Abaixo, um exemplo de entrada:

```
3, 7, 4, 6, 3, 1
```

Neste caso, obteve-se como saída

```
média = 4.0000000000000000
desvio padrão = 2.0000000000000000
```

A seguir, outro exemplo de entrada; para este caso, foi preciso alterar o parâmetro `n` para `9` e recompilar o código:

```
3.7, 5.4, 9.1, 3.2, 5.4, 2.9, 1.8, 8.3, 6.6
```

Neste caso a saída foi:

```
média = 5.1555555555555559
desvio padrão = 2.3556603750263188
```

3.4 Ordenação de um vetor

Fez-se `n = 12`, `m = 5` e a entrada abaixo:

```
4 143 3.2 16 0 375.17 5 71.3 150 23 -48 -2.141
```

para a qual o resultado foi

```
-48.000000000000000
2.1410000000000000
0.0000000000000000
3.2000000000000002
4.0000000000000000
```

É interessante notar o algarismo 2 no final do penúltimo número listado. Embora a entrada tenha sido exatamente `3.2`, tal erro aparece na 16^a casa decimal, o que é condizente com a precisão dupla do FORTRAN.

3.5 Cálculo do autovalor dominante de uma matriz

Para a primeira matriz indicada no enunciado, fez-se a tolerância `eps = 1e-12` e deu-se como entrada o arquivo texto abaixo:

```
2 8 10
8 4 5
10 5 7
```

A saída, neste caso, foi:

```
11.579779678437067 9.8602866523845947 12.808828066828120
19.884219060289233

11.573499514619908 9.8632694559230885 12.812214442059794
19.884233816752324

11.575768209650068 9.8621919822052604 12.810995213707102
19.884235738609640

11.574949380552244 9.8625811079449335 12.811435618712121
```

```
19.884235989086633

11.575245015319114 9.8624406502448405 12.811276655979713
19.884236021731517

11.575138290569111 9.8624913603052029 12.811334047494309
19.884236025986148

11.575176820105447 9.8624730536631269 12.811313328870259
19.884236026540663

11.575162910475989 9.8624796626600535 12.811320808640396
19.884236026612925

11.575167932049386 9.8624772767285620 12.811318108349843
19.884236026622347

11.575166119193945 9.8624781380831923 12.811319083192629
19.884236026623576

11.575166773659593 9.8624778271225413 12.811318731261242
19.884236026623736


foram necessárias 11 iterações
```

Assim, após as 11 iterações, atingiu-se uma precisão de 10^{-12} no autovalor. Já nos autovetores, um cálculo rápido mostra que as diferenças das entradas correspondentes entre a penúltima e a última iterações é da ordem de 10^{-7}, um valor bem maior, sugerindo que a convergência da sequência de autovetores é bem mais lenta.

Em seguida, executou-se o mesmo programa, dando como entrada a segunda matriz sugerida:

```
10 -2 3 2
-2 10 -3 4
3 -3 6 3
2 4 3 6
```

Neste caso, foram necessárias 86 iterações para que a precisão no autovalor chegasse a 10^{-12}. Estão reproduzidas abaixo somente as saídas correspondentes às cinco primeiras e às cinco últimas iterações:

```
7.6219211278427386 3.5373799473069867 4.9453617194370585 7.4337255444392145
12.235509886799786

8.0622121938986826 2.8570071840382072 5.2384623813192839 7.2452747103720885
```

```
12.342029261393662

8.4952673628414281 2.0779798935379956 5.5589459796650571 7.0102444638985197
12.473599203949984

8.9114573797172518 1.2111760787522607 5.8855746234365913 6.7171809132485567
12.630862382706676

9.2929763434730521 0.27621297140788714 6.1987598823586412 6.3614174361831699
12.810231523714332

(...)

9.0677823907568289 -8.6687714184905769 6.6400382355585128 0.41228507399827852
14.199731058863815

9.0677810069396152 -8.6687735751345798 6.6400374536831723 0.41228275619782961
14.199731058866123

9.0677798445140567 -8.6687753867437483 6.6400367968969940 0.41228080921447385
14.199731058867748

9.0677788680606319 -8.6687769085192272 6.6400362451875150 0.41227917372243694
14.199731058868901

9.0677780478264047 -8.6687781868306502 6.6400357817439577 0.41227779988727198
14.199731058869704

foram necessárias 86 iterações
```

Dessa vez, a média dos incrementos entre as duas últimas iterações para cada entrada da aproximação do autovetor é da ordem de 10^{-6}, revelando, mais uma vez, que a precisão no autovetor tende a ser menor que no autovalor.

Finalmente, deu-se como entrada a matriz 5x5 abaixo:

```
-10 2 -3 -2 -1
2 -10 3 -4 -2
-3 3 -6 -3 -3
-2 -4 -3 -6 -4
-1 -2 -3 -4 -13
```

A precisão de 10^{-12} no autovalor foi atingida após 45 iterações. Novamente, reproduziram-se as cinco primeiras e cinco últimas iterações, conforme gravadas no arquivo de saída. [2]

[2] No arquivo de saída, cada iteração começava com uma única linha contendo um vetor de dimensão 5; pela formatação deste documento, a última entrada de cada vetor acabou ficando na linha abaixo.

```
-5.4439676665028660 -4.3101954829662290 -5.6859217850141039 -8.3675799318469970
-12.778589630859575
-17.698325487186771

5.2151352088011969 4.1874115233923899 5.6985741891936863 8.2697656805278097
13.022998669019195
-17.749861038555927

-5.0938598748402866 -4.1384751359775906 -5.6975727288051381 -8.2228692853229912
-13.127625697468718
-17.761099657072407

5.0291624881729753 4.1240428767748636 5.6921158707868100 8.2019181337871139
13.175025493416383
-17.763615599269627

-4.9932261360352399 -4.1243098447293560 -5.6856526221082335 -8.1931977829205174
-13.197423606969899
-17.764218351167994

(...)

-4.9214878987561272 -4.1757275663657696 -5.6514687093524509 -8.1934804913659462
-13.223013217865411
-17.764516417316301

4.9214851746967527 4.1757310132557350 5.6514667791976283 8.1934810152670785
13.223013643550221
-17.764516417320063

-4.9214830101927056 -4.1757337521112028 -5.6514652455201801 -8.1934814315517031
-13.223013981792951
-17.764516417322433

4.9214812903045013 4.1757359283714104 5.6514640268788909 8.1934817623258418
13.223014250555707
-17.764516417323929

-4.9214799237026012 -4.1757376576005143 -5.6514630585615171 -8.1934820251545801
-13.223014464110801
-17.764516417324877

foram necessárias 45 iterações
```

Aqui, nota-se um fenômeno curioso na série dos $\vec{x}_k$: se, em uma dada iteração, todas as entradas são positivas, na iteração seguinte são todas negativas. Ora, pelo menos quando a aproximação já é boa, isto é uma mera consequência do fato de o autovalor λ ser negativo, uma vez que $\vec{x}_{k+1} = A \cdot \vec{x}_k \approx \lambda \vec{x}_k$. Evidentemente, tal efeito também não representa

nenhuma dificuldade na identificação do autovetor, já que a alternância de sinais não descaracteriza a "direção"dos vetores da sequência, que é o que de fato importa.

Novamente, um rápido cálculo mostra que, quando a precisão no autovalor chega a 10^{-12}, a precisão do autovetor ainda é da ordem de 10^{-6}.

Podemos concluir que, ao menos para os três exemplos sugeridos e testados, a precisão para os autovalores é muito maior que para os autovetores, numa mesma iteração.

Em todos os casos, utilizou-se como palpite inicial para o autovetor um vetor com o número 1 em todas as entradas. Um questionamento natural: qual teria sido o impacto desta escolha no desempenho do algoritmo? (Entenda-se que, nesta situação em particular, ter um bom desempenho significaria atingir uma dada precisão em poucas iterações.) Ora, é imediato pensar que tal desempenho seria tanto melhor quanto mais "próximo"o palpite inicial estivesse do autovetor dominante. Podemos quantificar esta proximidade calculando a distância euclidiana entre os vetores $\vec{x}_0 = (1, ..., 1)$ e $\vec{x}_N$ (aproximação final dada pelo programa), ambos normalizados.

Para o primeiro exemplo, faremos

```
x0 = (0.57735026918962584, 0.57735026918962584, 0.57735026918962584)
xN = (0.58212781009847026, 0.49599480784261957, 0.64429524544506933)
```

Procedendo ao cálculo conforme descrito acima, obteve-se uma distância de `0.10546642040935023`.

No segundo exemplo,

```
x0 =(0.5, 0.5, 0.5, 0.5)
xN = (0.63858801340904403, -0.61048889946515572, 0.4676170100838819, 0.029034197773044636)
```

A distância calculada foi de `1.2145985666874408`.

Finalmente,

```
x0 =(0.44721359549995793, 0.44721359549995793, 0.44721359549995793, 0.44721359549995793,
0.44721359549995793)
xN = (0.27703990404731049, 0.23506058704351107, 0.31813210817547871, 0.46122741721040017,
0.74434981248433585)
```

A distância, no último exemplo, foi de `0.42322137619969186`.

Faça-se lembrar que as quantidades de iterações necessárias para atingir a mesma precisão de 10^{-12} foram, respectivamente, 11, 86 e 45. Portanto, para estes três exemplos, verificou-se o que seria de se esperar: quanto mais *próximo* (no sentido definido acima) o autovetor dominante estiver do palpite inicial, melhor o desempenho do programa.