Solução dos Exercícios de Álgebra Linear: Aspectos Teóricos e Computactionais

Juan Lopes

Agosto 2014

Conteúdo

Introdução

Este é o documento explicativo da entrega dos exercícios para a disciplina *Álgebra Linear: Aspectos Teóricos e Computacionais*, ministrada pelo professor Ricardo Carvalho de Barros.

O códigos-fonte das soluções dos exercícios podem ser encontrados nos devidos apêndices. Todos os programas foram implementados em C++, utilizando a entrada e saída padrões para testar seu funcionamento, todos também acompanham um exemplo de arquivo de entrada e sua respectiva saída esperada, para facilitar testes automáticos.

As soluções foram escritas e testadas em Linux, utilizando o compilador `g++ 4.7.2`. Este pacote também possui um script para facilitar o teste das implementações, que pode ser encontrado no apêndice A.1, na página 20.

Os arquivos com os códigos-fonte encontram-se na raiz do pacote, os respectivos exemplos de entrada e saída encontram-se em arquivos `.in` e `.out` dentro do diretório `data`.

É possível utilizar a linha de comando para compilar, executar e verificar o resultado. Por exemplo, para o exercício `1.1`, basta executar o comando:

```
g++ exercicio_1.1.cpp &&
    ./a.out < src/data/exercicio_1.1.in |
    diff - src/data/exercicio_1.1.out
```

ou utilizar o script

```
./test.sh 1.1
```

O script `test.sh` também permite testar todos os programas em sequência automaticamente. Para tanto, basta executá-lo sem argumentos.

```
./test.sh
```

1 Capítulo 1 - Matrizes

1.1 Igualdade de matrizes

Enunciado Faça um programa que leia duas matrizes e verifique se elas são iguais.

Solução O código-fonte do programa encontra-se no apêndice A.2, na página 20.

Entrada Cada entrada consiste em duas matrizes, como no exemplo:

```
2 3
1 2 3
4 5 6
2 3
1 2 3
4 5 6
```

Saída A saída é a string `T` caso as matrizes sejam iguais ou `F` caso sejam diferentes:

```
T
```

1.2 Soma, subtração e produto por escalar

Enunciado Faça um programa que faça a soma e subtração de matrizes e o produto de uma matriz por um escalar.

Solução O código-fonte do programa encontra-se no apêndice A.3, na página 21.

Entrada Cada entrada começa com uma string informando a operação (`add`, `sub` ou `mul`). Para `add` e `sub` são lidas duas matrizes; para `mul` é lida uma matriz e um número.

```
add
2 3
1 2 3
4 5 6
2 3
1 2 3
4 5 6

sub
2 3
1 2 3
4 5 6
2 3
2 3 4.5
5 6 7

mul
2 3
1 2 3
4 5 6
5
```

Saída A saída é a matriz resultante da operação escolhida, seguida por `---`.

```
2 4 6
8 10 12
---
-1 -1 -1.5
-1 -1 -1
---
5 10 15
20 25 30
---
```

1.3 Transposição

Enunciado Faça um programa que leia uma matriz e calcule sua transposta.

Solução O código-fonte do programa encontra-se no apêndice A.4, na página 23.

Entrada Cada entrada consiste na matriz a ser transposta.

```
2 3
1 2 3
4 5 6
```

Saída A saída é a matriz resultante da operação, seguida por `---`.

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

1.4 Produto de matrizes

Enunciado Faça um programa que calcule o produto de matrizes.

Solução O código-fonte do programa encontra-se no apêndice A.5, na página 25.

Entrada Cada entrada consiste em duas matrizes.

```
3 2
2 1
4 2
5 3

2 2
1 -1
0 4
```

Saída A saída é a matriz resultante da operação, seguida por `---`.

```
2 2
4 4
5 7
---
```

3 Capítulo 3 - Solução de Sistemas Lineares

3.1 Método de Gauss – solução manual

Enunciado Encontre a solução do sistema linear que segue, utilizando o algoritmo dado.

Sistema inicial:

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 1 & 1 & 2 & 2 \\ 4 & 3 & -2 & 3 \end{array}\right]$$

Passo 1 ($m = {}^1/_3$):

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 0 & {}^1/_3 & {}^2/_3 & {}^5/_3 \\ 4 & 3 & -2 & 3 \end{array}\right]$$

Passo 2 ($m = {}^4/_3$):

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 0 & {}^1/_3 & {}^2/_3 & {}^5/_3 \\ 0 & {}^1/_3 & {}^{-22}/_3 & {}^5/_3 \end{array}\right]$$

Passo 3 ($m = 1$):

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 0 & {}^1/_3 & {}^2/_3 & {}^5/_3 \\ 0 & 0 & -8 & 0 \end{array}\right]$$

Passo 4 (substituição retroativa):

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 0 & {}^1/_3 & {}^2/_3 & {}^5/_3 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Passo 5 (substituição retroativa):

$$\left[\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Passo 6 (substituição retroativa):

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

3.2 Substituição Retroativa

Enunciado Faça um programa que encontre a solução de um sistema linear através do método de substituição retroativa.

Solução O código-fonte do programa encontra-se no apêndice A.6, na página 26.

Entrada Cada entrada consiste em uma matriz estendida do sistema do sistema triangular superior a ser resolvido.

```
3 4
2 1 1 7
0 2 1 7
0 0 2 6
```

Saída A saída é a matriz resolvida do sistema, seguida por `---`.

```
1.000 0.000 0.000 1.000
0.000 1.000 0.000 2.000
0.000 0.000 1.000 3.000
---
```

3.3 Método de Gauss

Enunciado Faça um programa que calcule a solução de sistemas lineares utilizando o método de Gauss.

Solução O código-fonte do programa encontra-se no apêndice A.7, na página 27.

Entrada Cada entrada consiste em uma matriz estendida do sistema do sistema a ser resolvido.

```
4 5
-1 3 5 2 10
1 9 8 4 15
0 1 0 1 2
2 1 1 -1 -3
```

Saída A saída é a matriz resolvida do sistema, seguida por `---`.

```
1.000 0.000 0.000 0.000 -1.000
0.000 1.000 0.000 0.000 -0.000
0.000 0.000 1.000 0.000 1.000
0.000 0.000 0.000 1.000 2.000
---
```

3.4 Método de Gauss – Pivoteamento Parcial

Enunciado Faça um programa que calcule a solução de sistemas lineares utilizando o método de Gauss com a estratégia de pivoteamento parcial.

Solução O código-fonte do programa encontra-se no apêndice A.8, na página 29.

Entrada Cada entrada consiste em uma matriz estendida do sistema do sistema a ser resolvido.

```
4 5
-1 3 5 2 10
1 9 8 4 15
0 1 0 1 2
2 1 1 -1 -3
```

Saída A saída é a matriz resolvida do sistema, seguida por `---`.

```
1.000 0.000 0.000 0.000 -1.000
0.000 1.000 0.000 0.000 0.000
0.000 0.000 1.000 0.000 1.000
0.000 0.000 0.000 1.000 2.000
---
```

3.5 Decomposição LU – solução manual

Enunciado Fatore as matrizes que seguem na decomposição LU.

a) Sistema inicial

$$\begin{bmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{bmatrix}$$

Passo 1 ($m_{21} = 1.5$)

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 3 & 3 & 5 \end{bmatrix}$$

Passo 2 ($m_{31} = 1.5$)

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 4.5 & 3.5 \end{bmatrix}$$

Passo 3 ($m_{32} = 1$)

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix}$$

Resultado:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{bmatrix} U = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix}$$

b) Sistema inicial

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ -4.0231 & 6.0000 & 0 & 1.1973 \\ -1.0000 & -5.2107 & 1.1111 & 0 \\ 6.0235 & 7.0000 & 0 & -4.1561 \end{bmatrix}$$

Passo 1 ($m_{21} = -1.8492$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ -1.0000 & -5.2107 & 1.1111 & 0 \\ 6.0235 & 7.0000 & 0 & -4.1561 \end{bmatrix}$$

Passo 2 ($m_{31} = -0.4596$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & -3.3615 & 0.1122 & 2.3886 \\ 6.0235 & 7.0000 & 0 & -4.1561 \end{bmatrix}$$

Passo 3 ($m_{41} = 2.7687$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & -3.3615 & 0.1122 & 2.3886 \\ 0.0000 & -4.1386 & 6.0169 & -18.5440 \end{bmatrix}$$

Passo 4 ($m_{32} = -0.2501$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & 0.0000 & -0.8930 & 5.0917 \\ 0.0000 & -4.1386 & 6.0169 & -18.5440 \end{bmatrix}$$

Passo 5 ($m_{42} = -0.3079$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & 0.0000 & -0.8930 & 5.0917 \\ 0.0000 & 0.0000 & 4.7793 & -15.2161 \end{bmatrix}$$

Passo 6 ($m_{43} = -5.3523$)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & 0.0000 & -0.8930 & 5.0917 \\ 0.0000 & 0.0000 & 0.0000 & 12.0361 \end{bmatrix}$$

Resultado:

$$L = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ -1.8492 & 1.0000 & 0.0000 & 0.0000 \\ -0.4596 & -0.2501 & 1.0000 & 0.0000 \\ 2.7687 & -0.3079 & -5.3523 & 1.0000 \end{bmatrix} U = \begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ 0.0000 & 13.4395 & -4.0187 & 10.8070 \\ 0.0000 & 0.0000 & -0.8930 & 5.0917 \\ 0.0000 & 0.0000 & 0.0000 & 12.0361 \end{bmatrix}$$

3.6 Decomposição LU

Enunciado Faça um programa que decomponha uma matriz A dada no exemplos 3.6 e 3.7 e no exercício 3.5 em matrizes L e U.

Solução O código-fonte do programa encontra-se no apêndice A.9, na página 31.

Entrada Cada entrada consiste em uma matriz a ser decomposta.

```
4 4
2.1756 4.0231 -2.1732 5.1967
-4.0231 6.0000 0 1.1973
-1.0000 -5.2107 1.1111 0
6.0235 7.0000 0 -4.1561
```

Saída A saída são duas matrizes L e U, seguidas por `---`.

```
L:
1.0000 0.0000 0.0000 0.0000
-1.8492 1.0000 0.0000 0.0000
-0.4596 -0.2501 1.0000 0.0000
2.7687 -0.3079 -5.3523 1.0000
U:
2.1756 4.0231 -2.1732 5.1967
0.0000 13.4395 -4.0187 10.8070
0.0000 0.0000 -0.8930 5.0917
0.0000 0.0000 0.0000 12.0361
---
```

3.7 Decomposição LU – Solução de Sistemas

Enunciado Faça um programa que resolva sistemas lineares utilizando a decomposição L e U .

Solução O código-fonte do programa encontra-se no apêndice A.10, na página 34.

Entrada Cada entrada consiste em uma matriz estendida do sistema a ser resolvido.

```
3 4
4 0 -3 -2
3 -4 1 9
1 2 2 3
```

Saída A saída são as diversas matrizes criadas nos passos da solução, seguidas por `---`. A última matriz contém a solução do sistema.

```
LY = B:
1.0000 0.0000 0.0000 -2.0000
0.7500 1.0000 0.0000 9.0000
0.2500 -0.5000 1.0000 3.0000
solved Y:
1.0000 0.0000 0.0000 -2.0000
0.0000 1.0000 0.0000 10.5000
0.0000 0.0000 1.0000 8.7500
UX = Y:
4.0000 0.0000 -3.0000 -2.0000
0.0000 -4.0000 3.2500 10.5000
0.0000 0.0000 4.3750 8.7500
solved X:
1.0000 0.0000 0.0000 1.0000
0.0000 1.0000 0.0000 -1.0000
0.0000 0.0000 1.0000 2.0000
---
```

3.8 Decomposição LU – Pivoteamento Parcial

Enunciado Refaça o programa do exercício anterior, utilizando também a estratégia de pivoteamento parcial.

Solução O código-fonte do programa encontra-se no apêndice A.11, na página 37.

Entrada Cada entrada consiste em uma matriz estendida do sistema a ser resolvido.

```
3 4
3 -4 1 9
1 2 2 3
4 0 -3 -2
```

Saída A saída são as diversas matrizes criadas nos passos da solução (inclusive P), seguidas por `---`. A última matriz contém a solução do sistema.

```
P:
0.0000 0.0000 1.0000
1.0000 0.0000 0.0000
0.0000 1.0000 0.0000
LY = PB:
```

```
1.0000 0.0000 0.0000 -2.0000
0.7500 1.0000 0.0000 9.0000
0.2500 -0.5000 1.0000 3.0000
solved Y:
1.0000 0.0000 0.0000 -2.0000
0.0000 1.0000 0.0000 10.5000
0.0000 0.0000 1.0000 8.7500
UX = Y:
4.0000 0.0000 -3.0000 -2.0000
0.0000 -4.0000 3.2500 10.5000
0.0000 0.0000 4.3750 8.7500
solved X:
1.0000 0.0000 0.0000 1.0000
0.0000 1.0000 0.0000 -1.0000
0.0000 0.0000 1.0000 2.0000
---
```

3.9 Fatoração de Cholesky – LDL^t

Enunciado Fatore a matriz dada em um produto LDL^t, onde L é triangular inferior, com todos os elementos diagonais iguais a 1, e D é uma matriz a diagonal.

Solução O código-fonte do programa encontra-se no apêndice A.12, na página 40.

Entrada Cada entrada consiste em uma matriz a ser fatorada.

```
4 4
16 -4 12 -4
-4 2 -1 1
12 -1 14 -2
-4 1 -2 83
```

Saída A saída são matrizes L, D e L^t, seguidas por `---`.

```
L:
1.0000 0.0000 0.0000 0.0000
-0.2500 1.0000 0.0000 0.0000
0.7500 2.0000 1.0000 0.0000
-0.2500 0.0000 1.0000 1.0000
D:
16.0000 0.0000 0.0000 0.0000
0.0000 1.0000 0.0000 0.0000
0.0000 0.0000 1.0000 0.0000
0.0000 0.0000 0.0000 81.0000
Lt:
1.0000 -0.2500 0.7500 -0.2500
0.0000 1.0000 2.0000 0.0000
0.0000 0.0000 1.0000 1.0000
0.0000 0.0000 0.0000 1.0000
---
```

3.10 Fatoração de Cholesky – GG^t

Enunciado Use o algoritmo de Cholesky para encontrar uma fatoração da forma $A = GG^t$ para as matrizes que seguem.

Solução O código-fonte do programa encontra-se no apêndice A.13, na página 43.

Entrada Cada entrada consiste em uma matriz a ser fatorada.

```
4 4
4 1 1 1
1 3 -1 1
1 -1 2 0
1 1 0 2
```

Saída A saída é a matriz G, seguida por `---`.

```
2.0000 0.0000 0.0000 0.0000
0.5000 1.6583 0.0000 0.0000
0.5000 -0.7538 1.0871 0.0000
0.5000 0.4523 0.0836 1.2403
---
```

3.12 Fatoração de Cholesky – Sistemas

Enunciado Faça um programa que resolva sistemas lineares utilizando a decomposição de Cholesky.

Solução O código-fonte do programa encontra-se no apêndice A.14, na página 44.

Entrada Cada entrada consiste em uma matriz estendida a ser resolvida.

```
4 5
4 1 1 1 0.65
1 3 -1 1 0.05
1 -1 2 0 0
1 1 0 2 0.5
```

Saída A saída são as diversas matrizes criadas nos passos da solução, seguidas por `---`. A última matriz contém a solução do sistema.

```
GY = B:
2.0000 0.0000 0.0000 0.0000 0.6500
0.5000 1.6583 0.0000 0.0000 0.0500
0.5000 -0.7538 1.0871 0.0000 0.0000
0.5000 0.4523 0.0836 1.2403 0.5000
solved Y:
1.0000 0.0000 0.0000 0.0000 0.3250
0.0000 1.0000 0.0000 0.0000 -0.0678
0.0000 0.0000 1.0000 0.0000 -0.1965
0.0000 0.0000 0.0000 1.0000 0.3101
GtX = Y:
2.0000 0.5000 0.5000 0.5000 0.3250
0.0000 1.6583 -0.7538 0.4523 -0.0678
0.0000 0.0000 1.0871 0.0836 -0.1965
0.0000 0.0000 0.0000 1.2403 0.3101
solved X:
1.0000 0.0000 0.0000 0.0000 0.2000
0.0000 1.0000 0.0000 0.0000 -0.2000
0.0000 0.0000 1.0000 0.0000 -0.2000
0.0000 0.0000 0.0000 1.0000 0.2500
---
```

3.13 Método de Jacobi

Enunciado Faça um programa para resolver os sistemas lineares do exercício 1, com $\epsilon = 10^{-3}$.

Solução O código-fonte do programa encontra-se no apêndice A.15, na página 48.

Entrada Cada entrada consiste em uma matriz estendida a ser resolvida.

```
5 6
4 1 1 0 1 6
-1 -3 1 1 0 6
2 1 5 -1 -1 6
-1 -1 -1 4 0 6
0 2 -1 1 4 6
```

Saída A saída é o resultado de cada uma das iterações, seguidas por `---`.

```
Iteration #1. Epsilon=1.000000
-0.500000 -0.250000 0.000000 0.333333

Iteration #2. Epsilon=0.416000
-0.520833 -0.041667 -0.216667 0.416667

Iteration #3. Epsilon=0.229368
-0.647917 -0.069792 -0.195833 0.565278

Iteration #4. Epsilon=0.110179
-0.672830 0.004340 -0.256597 0.591319

Iteration #5. Epsilon=0.074705
-0.713064 0.001888 -0.251962 0.644589

Iteration #6. Epsilon=0.033859
-0.724610 0.026423 -0.271153 0.655638

Iteration #7. Epsilon=0.024954
-0.738303 0.027274 -0.270765 0.674062

Iteration #8. Epsilon=0.010937
-0.743025 0.035400 -0.277018 0.678781

Iteration #9. Epsilon=0.008515
-0.747800 0.036197 -0.277281 0.685148

Iteration #10. Epsilon=0.003628
-0.749656 0.038917 -0.279350 0.687093

Iteration #11. Epsilon=0.002948
-0.751340 0.039350 -0.279566 0.689308

Iteration #12. Epsilon=0.001224
-0.752056 0.040270 -0.280260 0.690085

Iteration #13. Epsilon=0.001032
-0.752654 0.040470 -0.280374 0.690862

Iteration #14. Epsilon=0.000418
-0.752927 0.040785 -0.280609 0.691166

---
```

3.14 Método de Gauss–Seidel

Enunciado Faça um programa para resolver sistemas lineares pelo método de Gauss-Seidel. Resolva os dois sistemas do exercício 1, considerando $\epsilon = 10^{-3}$.

Solução O código-fonte do programa encontra-se no apêndice A.16, na página 51.

Entrada Cada entrada consiste em uma matriz estendida a ser resolvida.

```
5 6
4 1 1 0 1 6
-1 -3 1 1 0 6
2 1 5 -1 -1 6
-1 -1 -1 4 0 6
0 2 -1 1 4 6
```

Saída A saída é o resultado de cada uma das iterações, seguidas por `---`.

```
Iteration #1. Epsilon=1.000000
1.500000 -2.500000 1.100000 1.525000 2.643750

Iteration #2. Epsilon=0.433865
1.189062 -1.521354 1.862396 1.882526 2.255645

Iteration #3. Epsilon=0.241892
0.850828 -1.035302 1.894363 1.927472 2.009374

Iteration #4. Epsilon=0.034274
0.782891 -0.987019 1.871616 1.916872 1.982195

Iteration #5. Epsilon=0.005662
0.783302 -0.998271 1.866147 1.912794 1.987474

Iteration #6. Epsilon=0.002079
0.786163 -1.002407 1.866070 1.912456 1.989607

Iteration #7. Epsilon=0.000261
0.786683 -1.002719 1.866283 1.912562 1.989790

---
```

4 Capítulo 4 – Autovalores e autovetores

4.1 Autovalores e autovetores

Enunciado Para cada matriz, encontre todos os autovalores e uma base para cada auto-espaço.

a)

$$A - \lambda I = \begin{bmatrix} 3-\lambda & 1 & 1 \\ 2 & 4-\lambda & 2 \\ 1 & 1 & 3-\lambda \end{bmatrix}$$

$$P(A - \lambda I) = (\lambda - 2)^2(\lambda - 6)$$

Para $\lambda = 2$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$x + y + z = 0$$

$$x = -y - z$$

$$v_1 = (1, -1, 0), v_2 = (1, 0, -1)$$

Para $\lambda = 6$:

$$\begin{bmatrix} -3 & 1 & 1 \\ 2 & -2 & 2 \\ 1 & 1 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$x = z$$

$$y = 2z$$

$$v_3 = (1, 2, 1)$$

Logo:

$$P_1 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 2 \\ 0 & -1 & 1 \end{bmatrix}$$

b)

$$B - \lambda I = \begin{bmatrix} 1-\lambda & 2 & 2 \\ 1 & 2-\lambda & -1 \\ -1 & 1 & 4-\lambda \end{bmatrix}$$

$$P(B - \lambda I) = (\lambda - 3)^2(\lambda - 1)$$

Para $\lambda = 3$:

$$\begin{bmatrix} -2 & 2 & 2 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$x = y + z$$

$$v_1 = (1, 1, 0), v_2 = (1, 0, 1)$$

Para $\lambda = 1$:

$$\begin{bmatrix} 0 & 2 & 2 \\ 1 & 1 & -1 \\ -1 & 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$x = 2z$$

$$y = -z$$

$$v_3 = (2, -1, 1)$$

Logo:

$$P_2 = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix}$$

c)

$$C - \lambda I = \begin{bmatrix} 1-\lambda & 1 & 0 \\ 0 & 1-\lambda & 0 \\ 0 & 0 & 1-\lambda \end{bmatrix}$$

$$P(C - \lambda I) = (\lambda - 1)^3$$

Para $\lambda = 1$:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

$$y = 0$$

$$v_1 = (1, 0, 0), v_2 = (0, 0, 1)$$

Logo, não existe P_3 tal que $P_3^{-1} C P_3$ seja diagonal.

4.2 Raio espectral

Enunciado Encontre o raio espectral para cada matriz que segue.

a)

$$A - \lambda I = \begin{bmatrix} 2-\lambda & 1 & 0 \\ 1 & 2-\lambda & 0 \\ 0 & 0 & 3-\lambda \end{bmatrix}$$

$$P(A - \lambda I) = (\lambda - 3)^2(\lambda - 1)$$

$$\rho(A) = max(3, 1) = 3$$

b)

$$B - \lambda I = \begin{bmatrix} 2-\lambda & 1 & 1 \\ 2 & 3-\lambda & 2 \\ 1 & 1 & 2-\lambda \end{bmatrix}$$

$$P(B - \lambda I) = (\lambda - 1)^2(\lambda - 5)$$

$$\rho(B) = max(1, 5) = 5$$

c)

$$C - \lambda I = \begin{bmatrix} -1-\lambda & 2 & 0 \\ 0 & 3-\lambda & 4 \\ 0 & 0 & 7-\lambda \end{bmatrix}$$

$$P(C - \lambda I) = (\lambda + 1)(\lambda - 3)(\lambda - 7)$$

$$\rho(C) = max(-1, 3, 7) = 7$$

d)

$$D - \lambda I = \begin{bmatrix} 3-\lambda & 2 & -1 \\ 1 & -2-\lambda & 3 \\ 2 & 0 & 4-\lambda \end{bmatrix}$$

$$P(D - \lambda I) = (\lambda + 2)(\lambda - 3)(\lambda - 4)$$

$$\rho(D) = max(-2, 3, 4) = 4$$

4.3 Método das Potências

4.3.1 Solução Normal

Enunciado Faça um programa que leia uma matriz e determine um dos seus autovalores e o autovetor correspondente pelo método das potências.

Solução O código-fonte do programa encontra-se no apêndice A.17, na página 54.

Entrada Cada entrada consiste em uma matriz a ser aplicada o método.

```
3 3
-4 14 0
-5 13 0
-1 0 2
```

Saída A saída é a descrição de cada uma das iterações do algoritmo.

```
Iteration #1. Epsilon=0.900000
Eigenvalue: 10.000000
Eigenvector:
1.000000 0.800000 0.100000

Iteration #2. Epsilon=0.211111
Eigenvalue: 7.200000
Eigenvector:
1.000000 0.750000 -0.111111

Iteration #3. Epsilon=0.076923
Eigenvalue: 6.500000
Eigenvector:
1.000000 0.730769 -0.188034

Iteration #4. Epsilon=0.032816
Eigenvalue: 6.230769
Eigenvector:
1.000000 0.722222 -0.220850

Iteration #5. Epsilon=0.015064
Eigenvalue: 6.111111
Eigenvector:
1.000000 0.718182 -0.235915

Iteration #6. Epsilon=0.007180
Eigenvalue: 6.054545
Eigenvector:
1.000000 0.716216 -0.243095

Iteration #7. Epsilon=0.003493
Eigenvalue: 6.027027
Eigenvector:
1.000000 0.715247 -0.246588

Iteration #8. Epsilon=0.001718
Eigenvalue: 6.013453
Eigenvector:
1.000000 0.714765 -0.248306

Iteration #9. Epsilon=0.000851
Eigenvalue: 6.006711
Eigenvector:
1.000000 0.714525 -0.249157

Iteration #10. Epsilon=0.000423
Eigenvalue: 6.003352
Eigenvector:
1.000000 0.714405 -0.249579

Iteration #11. Epsilon=0.000211
Eigenvalue: 6.001675
Eigenvector:
1.000000 0.714346 -0.249790

Iteration #12. Epsilon=0.000105
Eigenvalue: 6.000837
Eigenvector:
```

```
1.000000 0.714316 -0.249895

Iteration #13. Epsilon=0.000052
Eigenvalue: 6.000419
Eigenvector:
1.000000 0.714316 -0.249895

---
```

4.3.2 Método de Aitken

Enunciado Insira no programa do método das potências o método de Aitken.

Solução O código-fonte do programa encontra-se no apêndice A.18, na página 57.

Entrada Cada entrada consiste em uma matriz a ser aplicada o método.

```
3 3
-4 14 0
-5 13 0
-1 0 2
```

Saída A saída é a descrição de cada uma das iterações do algoritmo.

```
Iteration #1. Epsilon=0.900000
Eigenvalue: 0.000000
Eigenvector:
1.000000 0.800000 0.100000

Iteration #2. Epsilon=0.211111
Eigenvalue: 7.812500
Eigenvector:
1.000000 0.750000 -0.111111

Iteration #3. Epsilon=0.076923
Eigenvalue: 6.266667
Eigenvector:
1.000000 0.730769 -0.188034

Iteration #4. Epsilon=0.032816
Eigenvalue: 6.062500
Eigenvector:
1.000000 0.722222 -0.220850

Iteration #5. Epsilon=0.015064
Eigenvalue: 6.015385
Eigenvector:
1.000000 0.718182 -0.235915

Iteration #6. Epsilon=0.007180
Eigenvalue: 6.003831
Eigenvector:
1.000000 0.716216 -0.243095

Iteration #7. Epsilon=0.003493
Eigenvalue: 6.000957
Eigenvector:
```

```
1.000000 0.715247 -0.246588

Iteration #8. Epsilon=0.001718
Eigenvalue: 6.000239
Eigenvector:
1.000000 0.714765 -0.248306

Iteration #9. Epsilon=0.000851
Eigenvalue: 6.000060
Eigenvector:
1.000000 0.714525 -0.249157

Iteration #10. Epsilon=0.000423
Eigenvalue: 6.000015
Eigenvector:
1.000000 0.714405 -0.249579

Iteration #11. Epsilon=0.000211
Eigenvalue: 6.000004
Eigenvector:
1.000000 0.714346 -0.249790

Iteration #12. Epsilon=0.000105
Eigenvalue: 6.000001
Eigenvector:
1.000000 0.714316 -0.249895

Iteration #13. Epsilon=0.000052
Eigenvalue: 6.000000
Eigenvector:
1.000000 0.714316 -0.249895

---
```

A Apêndice: códigos-fonte

A.1 test.sh

```
#!/bin/bash

if [ $# -lt 1 ]; then
    success=0
    failed=0
    for f in $(ls *.cpp | sort -V)
    do

        BASE=$(basename $f .cpp)
        echo -n "$BASE..."
        if g++ $f && ./a.out < data/$BASE.in | diff - data/$BASE.out > /dev/null
        then
            echo OK
            ((++success))
        else
            echo FAILED
            ((++failed))
        fi
    done
    echo
    echo "PASSED: $success; FAILED: $failed."
    exit $failed
fi

g++ exercicio_$1.cpp && ./a.out < data/exercicio_$1.in | diff -y - data/exercicio_$1.out &&
    echo "OK" || echo "FAILED"
```

src/test.sh

A.2 exercicio_1.1.{cpp,in,out}

```
#include <iostream>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    bool equals_to(Matrix& b) {
        if (b.m!=m || b.n!=n)
            return false;
        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
```

```
                if (b.V[i][j] != V[i][j])
                    return false;
        return true;
    }

};

int main() {
    Matrix a, b;

    while(a.read() && b.read()) {
        cout << (a.equals_to(b) ? 'T' : 'F') << endl;
    }
}
```

src/exercicio_1.1.cpp

```
2 3
1 2 3
4 5 6
2 3
1 2 3
4 5 6

2 3
1 2 3
4 5 6
2 4
1 2 3 7
4 5 6 8

2 3
1 2 3
4 5 6
2 3
1 2 3
4 5 7
```

src/data/exercicio_1.1.in

```
T
F
F
```

src/data/exercicio_1.1.out

A.3 exercicio_1.2.{cpp,in,out}

```
#include <iostream>
#include <string>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
```

```
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << (j>0 ? " " : "") << V[i][j];
            cout << endl;
        }
    }

    void add(Matrix& a, Matrix& b) {
        m = a.m;
        n = a.n;
        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                V[i][j] = a.V[i][j] + b.V[i][j];
    }

    void sub(Matrix& a, Matrix& b) {
        m = a.m;
        n = a.n;
        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                V[i][j] = a.V[i][j] - b.V[i][j];
    }

    void mul(Matrix& a, double scalar) {
        m = a.m;
        n = a.n;
        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                V[i][j] = a.V[i][j] * scalar;
    }
};


int main() {
    string command;
    Matrix a, b, c;

    while(cin >> command) {
        if (command == "add") {
            a.read(); b.read();
            c.add(a, b);
        }

        if (command == "sub") {
            a.read(); b.read();
            c.sub(a, b);
        }

```

```
69          if (command == "mul") {
70              a.read();
71              double scalar; cin >> scalar;
72              c.mul(a, scalar);
73          }
74          c.write();
75          cout << "---" << endl;
76      }
77  }
```

src/exercicio_1.2.cpp

```
1  add
2  2 3
3  1 2 3
4  4 5 6
5  2 3
6  1 2 3
7  4 5 6
8
9  sub
10 2 3
11 1 2 3
12 4 5 6
13 2 3
14 2 3 4.5
15 5 6 7
16
17 mul
18 2 3
19 1 2 3
20 4 5 6
21 5
```

src/data/exercicio_1.2.in

```
1  2 4 6
2  8 10 12
3  ---
4  -1 -1 -1.5
5  -1 -1 -1
6  ---
7  5 10 15
8  20 25 30
9  ---
```

src/data/exercicio_1.2.out

A.4 exercicio_1.3.{cpp,in,out}

```
1  #include <iostream>
2  #include <string>
3  #define MAX 200
4  using namespace std;
5
6  struct Matrix {
7      int m, n;
8      double V[MAX][MAX];
```

```
9
10      bool read() {
11          if (not (cin >> m >> n))
12              return false;
13
14          for(int i=0; i<m; i++)
15              for(int j=0; j<n; j++)
16                  cin >> V[i][j];
17          return true;
18      }
19
20      void write() {
21          for(int i=0; i<m; i++) {
22              for(int j=0; j<n; j++)
23                  cout << (j>0 ? " " : "") << V[i][j];
24              cout << endl;
25          }
26      }
27
28      void transpose(Matrix& a) {
29          m = a.n;
30          n = a.m;
31
32          for(int i=0; i<m; i++)
33              for(int j=0; j<n; j++)
34                  V[i][j] = a.V[j][i];
35      }
36
37  };
38
39
40
41
42  int main() {
43      Matrix a, c;
44
45      while(a.read()) {
46          c.transpose(a);
47          c.write();
48          cout << "---" << endl;
49      }
50  }
```

src/exercicio_1.3.cpp

```
1  2 3
2  1 2 3
3  4 5 6
4
5  2 4
6  2 3 4.5 6.5
7  5 6 7 8.2
```

src/data/exercicio_1.3.in

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

```
2 5
3 6
4.5 7
6.5 8.2
---
```

src/data/exercicio_1.3.out

A.5 exercicio_1.4.{cpp,in,out}

```cpp
#include <iostream>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << (j>0 ? " " : "") << V[i][j];
            cout << endl;
        }
    }

    void multiply(Matrix& a, Matrix& b) {
        m = a.m;
        n = b.n;

        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++) {
                V[i][j] = 0;
                for(int k=0; k<a.m; k++)
                    V[i][j] += a.V[i][k] * b.V[k][j];

            }
        }
    }
};


int main() {
    Matrix a, b, c;

    while(a.read() && b.read()) {
        c.multiply(a, b);
        c.write();
```

```
        cout << "---" << endl;
    }
}
```

src/exercicio_1.4.cpp

```
3 2
2 1
4 2
5 3

2 2
1 -1
0 4

2 2
1 -1
0 4

3 2
2 1
4 2
5 3
```

src/data/exercicio_1.4.in

```
2 2
4 4
5 7
---
-2 -1
16 8
---
```

src/data/exercicio_1.4.out

A.6 exercicio_3.2.{cpp,in,out}

```
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

```

```
    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(3) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void solve_U() {
        for(int i=m-1; i>=0; i--) {
            double pivot = V[i][i];
            if (abs(pivot) < 1e-6) break;
            for(int j=i; j<n; j++) {
                V[i][j] /= pivot;
            }

            for(int j=i+1; j<n-1; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }
};


int main() {
    Matrix a;

    while(a.read()) {
        a.solve_U();
        a.write();
        cout << "---" << endl;
    }
}
```

src/exercicio_3.2.cpp

```
3 4
2 1 1 7
0 2 1 7
0 0 2 6
```

src/data/exercicio_3.2.in

```
1.000 0.000 0.000 1.000
0.000 1.000 0.000 2.000
0.000 0.000 1.000 3.000
---
```

src/data/exercicio_3.2.out

A.7 exercicio_3.3.{cpp,in,out}

```
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;
```

```cpp

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(3) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void solve() {
        for(int k=0; k<m-1; k++) {
            for(int i=k+1; i<m; i++) {
                double pivot = V[i][k] / V[k][k];
                for(int j=k; j<n; j++)
                    V[i][j] -= pivot*V[k][j];
            }
        }
        for(int i=m-1; i>=0; i--) {
            double pivot = V[i][i];
            if (abs(pivot) < 1e-6) break;
            for(int j=i; j<n; j++) {
                V[i][j] /= pivot;
            }

            for(int j=i+1; j<n-1; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }
};


int main() {
    Matrix a;

    while(a.read()) {
        a.solve();
        a.write();
        cout << "---" << endl;
    }
}
```

src/exercicio_3.3.cpp

```
3 4
3 2 4 1
1 1 2 2
4 3 -2 3

4 5
-1 3 5 2 10
1 9 8 4 15
0 1 0 1 2
2 1 1 -1 -3

2 3
1 2 2
1 2 3

2 3
1 2 2
1 2 2
```

src/data/exercicio_3.3.in

```
1.000 0.000 0.000 -3.000
0.000 1.000 0.000 5.000
0.000 0.000 1.000 0.000
---
1.000 0.000 0.000 0.000 -1.000
0.000 1.000 0.000 0.000 -0.000
0.000 0.000 1.000 0.000 1.000
0.000 0.000 0.000 1.000 2.000
---
1.000 2.000 2.000
0.000 0.000 1.000
---
1.000 2.000 2.000
0.000 0.000 0.000
---
```

src/data/exercicio_3.3.out

A.8 exercicio_3.4.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
```

```
17                cin >> V[i][j];
18        return true;
19    }
20
21    void write() {
22        for(int i=0; i<m; i++) {
23            for(int j=0; j<n; j++)
24                cout << setprecision(3) << fixed << (j>0 ? " " : "") << V[i][j];
25            cout <<  endl;
26        }
27    }
28
29    void swap_lines(int a, int b) {
30        for(int j=0; j<n; j++)
31            swap(V[a][j], V[b][j]);
32    }
33
34    void solve() {
35        for(int k=0; k<m-1; k++) {
36            int best = k;
37            for(int i=k+1; i<m; i++)
38                if (abs(V[i][k]) > abs(V[best][k]))
39                    best = i;
40            swap_lines(best, k);
41
42            for(int i=k+1; i<m; i++) {
43                double pivot = V[i][k] / V[k][k];
44                for(int j=k; j<n; j++)
45                    V[i][j] -= pivot*V[k][j];
46            }
47        }
48        for(int i=m-1; i>=0; i--) {
49            double pivot = V[i][i];
50            if (abs(pivot) < 1e-6) break;
51            for(int j=i; j<n; j++) {
52                V[i][j] /= pivot;
53            }
54
55            for(int j=i+1; j<n-1; j++) {
56                V[i][n-1] -= V[i][j]*V[j][n-1];
57                V[i][j] = 0;
58            }
59        }
60    }
61};
62
63
64int main() {
65    Matrix a;
66
67    while(a.read()) {
68        a.solve();
69        a.write();
70        cout << "---" << endl;
71    }
72}
```

src/exercicio_3.4.cpp

```
3 4
3 2 4 1
1 1 2 2
4 3 -2 3

4 5
-1 3 5 2 10
1 9 8 4 15
0 1 0 1 2
2 1 1 -1 -3

2 3
1 2 2
1 2 3

2 3
1 2 2
1 2 2

3 4
3 -4 1 9
1 2 2 3
4 0 -3 -2
```

src/data/exercicio_3.4.in

```
1.000 0.000 0.000 -3.000
0.000 1.000 0.000 5.000
0.000 0.000 1.000 0.000
---
1.000 0.000 0.000 0.000 -1.000
0.000 1.000 0.000 0.000 0.000
0.000 0.000 1.000 0.000 1.000
0.000 0.000 0.000 1.000 2.000
---
1.000 2.000 2.000
0.000 0.000 1.000
---
1.000 2.000 2.000
0.000 0.000 0.000
---
1.000 0.000 0.000 1.000
0.000 1.000 0.000 -1.000
0.000 0.000 1.000 2.000
---
```

src/data/exercicio_3.4.out

A.9 exercicio_3.6.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
```

```
8       int m, n;
9       double V[MAX][MAX];
10
11      bool read() {
12          if (not (cin >> m >> n))
13              return false;
14
15          for(int i=0; i<m; i++)
16              for(int j=0; j<n; j++)
17                  cin >> V[i][j];
18          return true;
19      }
20
21      void write() {
22          for(int i=0; i<m; i++) {
23              for(int j=0; j<n; j++)
24                  cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
25              cout <<  endl;
26          }
27      }
28
29      void decompose_LU(Matrix& b) {
30          b.m = m;
31          b.n = n;
32          for(int k=0; k<m; k++) {
33              b.V[k][k] = 1;
34              for(int i=k+1; i<m; i++) {
35                  double pivot = V[i][k] / V[k][k];
36                  b.V[i][k] = pivot;
37                  for(int j=k; j<n; j++)
38                      V[i][j] -= pivot*V[k][j];
39              }
40          }
41      }
42  };
43
44  int main() {
45      Matrix U, L;
46
47      while(U.read()) {
48          U.decompose_LU(L);
49          cout << "L:" << endl;
50          L.write();
51          cout << "U:" << endl;
52          U.write();
53          cout << "---" << endl;
54      }
55  }
```

src/exercicio_3.6.cpp

```
1  3 3
2  2 -1 1
3  3 3 9
4  3 3 5
5
6  4 4
7  2.1756 4.0231 -2.1732 5.1967
8  -4.0231 6.0000 0 1.1973
```

```
-1.0000 -5.2107 1.1111 0
6.0235 7.0000 0 -4.1561

3 3
3 1 4
1 5 9
2 6 5

3 3
1 5 9
2 6 5
3 1 4

3 3
3 -4 1
1 2 2
4 0 -3

3 3
4 0 -3
3 -4 1
1 2 2
```

src/data/exercicio_3.6.in

```
L:
1.0000 0.0000 0.0000
1.5000 1.0000 0.0000
1.5000 1.0000 1.0000
U:
2.0000 -1.0000 1.0000
0.0000 4.5000 7.5000
0.0000 0.0000 -4.0000
---
L:
1.0000 0.0000 0.0000 0.0000
-1.8492 1.0000 0.0000 0.0000
-0.4596 -0.2501 1.0000 0.0000
2.7687 -0.3079 -5.3523 1.0000
U:
2.1756 4.0231 -2.1732 5.1967
0.0000 13.4395 -4.0187 10.8070
0.0000 0.0000 -0.8930 5.0917
0.0000 0.0000 0.0000 12.0361
---
L:
1.0000 0.0000 0.0000
0.3333 1.0000 0.0000
0.6667 1.1429 1.0000
U:
3.0000 1.0000 4.0000
0.0000 4.6667 7.6667
0.0000 0.0000 -6.4286
---
L:
1.0000 0.0000 0.0000
2.0000 1.0000 0.0000
3.0000 3.5000 1.0000
U:
```

```
1.0000 5.0000 9.0000
0.0000 -4.0000 -13.0000
0.0000 0.0000 22.5000
---
L:
1.0000 0.0000 0.0000
0.3333 1.0000 0.0000
1.3333 1.6000 1.0000
U:
3.0000 -4.0000 1.0000
0.0000 3.3333 1.6667
0.0000 0.0000 -7.0000
---
L:
1.0000 0.0000 0.0000
0.7500 1.0000 0.0000
0.2500 -0.5000 1.0000
U:
4.0000 0.0000 -3.0000
0.0000 -4.0000 3.2500
0.0000 0.0000 4.3750
---
```

src/data/exercicio_3.6.out

A.10 exercicio_3.7.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void decompose_LU(Matrix& b) {
        b.m = m;
        b.n = n;
```

```
        for(int k=0; k<m; k++) {
            b.V[k][k] = 1;
            for(int i=k+1; i<m; i++) {
                double pivot = V[i][k] / V[k][k];
                b.V[i][k] = pivot;
                for(int j=k; j<m; j++)
                    V[i][j] -= pivot*V[k][j];
            }
        }
    }

    void copy_last_column_from(Matrix& b) {
        for(int i=0; i<m; i++)
            V[i][n-1] = b.V[i][n-1];
    }

    void solve_L() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<i; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }

    void solve_U() {
        for(int i=m-1; i>=0; i--) {
            double pivot = V[i][i];
            if (abs(pivot) < 1e-6) break;
            for(int j=i; j<n; j++) {
                V[i][j] /= pivot;
            }

            for(int j=i+1; j<n-1; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }
};

int main() {
    Matrix U, L;

    while(U.read()) {
        U.decompose_LU(L);
        L.copy_last_column_from(U);

        cout << "LY = B:" << endl;
        L.write();

        L.solve_L();
        cout << "solved Y:" << endl;
        L.write();

        U.copy_last_column_from(L);

        cout << "UX = Y:" << endl;
```

```
        U.write();

        U.solve_U();
        cout << "solved X:" << endl;
        U.write();

        cout << "---" << endl;
    }
}
```

src/exercicio_3.7.cpp

```
3 4
3 -4 1 9
1 2 2 3
4 0 -3 -2

3 4
4 0 -3 -2
3 -4 1 9
1 2 2 3
```

src/data/exercicio_3.7.in

```
LY = B:
1.0000 0.0000 0.0000 9.0000
0.3333 1.0000 0.0000 3.0000
1.3333 1.6000 1.0000 -2.0000
solved Y:
1.0000 0.0000 0.0000 9.0000
0.0000 1.0000 0.0000 0.0000
0.0000 0.0000 1.0000 -14.0000
UX = Y:
3.0000 -4.0000 1.0000 9.0000
0.0000 3.3333 1.6667 0.0000
0.0000 0.0000 -7.0000 -14.0000
solved X:
1.0000 0.0000 0.0000 1.0000
0.0000 1.0000 0.0000 -1.0000
0.0000 0.0000 1.0000 2.0000
---
LY = B:
1.0000 0.0000 0.0000 -2.0000
0.7500 1.0000 0.0000 9.0000
0.2500 -0.5000 1.0000 3.0000
solved Y:
1.0000 0.0000 0.0000 -2.0000
0.0000 1.0000 0.0000 10.5000
0.0000 0.0000 1.0000 8.7500
UX = Y:
4.0000 0.0000 -3.0000 -2.0000
0.0000 -4.0000 3.2500 10.5000
0.0000 0.0000 4.3750 8.7500
solved X:
1.0000 0.0000 0.0000 1.0000
0.0000 1.0000 0.0000 -1.0000
0.0000 0.0000 1.0000 2.0000
---
```

src/data/exercicio_3.7.out

A.11 exercicio_3.8.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void identity(int newM) {
        m = newM;
        n = newM;
        for(int i=0; i<newM; i++)
            for(int j=0; j<newM; j++)
                V[i][j] = i==j;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void swap_lines(int a, int b) {
        for(int j=0; j<n; j++)
            swap(V[a][j], V[b][j]);
    }

    void decompose_LU(Matrix& b, Matrix& p) {
        b.m = m;
        b.n = n;
        for(int k=0; k<m; k++) {
            int best = k;
            for(int i=k+1; i<m; i++)
                if (abs(V[i][k]) > abs(V[best][k]))
                    best = i;

            swap_lines(best, k);
            p.swap_lines(best, k);
            b.swap_lines(best, k);

            b.V[k][k] = 1;
            for(int i=k+1; i<m; i++) {
```

```
                double pivot = V[i][k] / V[k][k];
                b.V[i][k] = pivot;
                for(int j=k; j<m; j++)
                    V[i][j] -= pivot*V[k][j];
            }
        }
    }

    void copy_last_column_from(Matrix& b) {
        for(int i=0; i<m; i++)
            V[i][n-1] = b.V[i][n-1];
    }

    void solve_L() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<i; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }

    void solve_U() {
        for(int i=m-1; i>=0; i--) {
            double pivot = V[i][i];
            if (abs(pivot) < 1e-6) break;
            for(int j=i; j<n; j++) {
                V[i][j] /= pivot;
            }

            for(int j=i+1; j<n-1; j++) {
                V[i][n-1] -= V[i][j]*V[j][n-1];
                V[i][j] = 0;
            }
        }
    }
};

int main() {
    Matrix U, L, P;

    while(U.read()) {
        P.identity(U.m);

        U.decompose_LU(L, P);
        cout << "P:" << endl;
        P.write();

        L.copy_last_column_from(U);

        cout << "LY = PB:" << endl;
        L.write();

        L.solve_L();
        cout << "solved Y:" << endl;
        L.write();

        U.copy_last_column_from(L);
```

```
115
116         cout << "UX = Y:" << endl;
117         U.write();
118
119         U.solve_U();
120         cout << "solved X:" << endl;
121         U.write();
122
123         cout << "---" << endl;
124     }
125 }
```

src/exercicio_3.8.cpp

```
1 3 4
2 3 -4 1 9
3 1 2 2 3
4 4 0 -3 -2
5
6 3 4
7 4 0 -3 -2
8 3 -4 1 9
9 1 2 2 3
```

src/data/exercicio_3.8.in

```
1  P:
2  0.0000 0.0000 1.0000
3  1.0000 0.0000 0.0000
4  0.0000 1.0000 0.0000
5  LY = PB:
6  1.0000 0.0000 0.0000 -2.0000
7  0.7500 1.0000 0.0000 9.0000
8  0.2500 -0.5000 1.0000 3.0000
9  solved Y:
10 1.0000 0.0000 0.0000 -2.0000
11 0.0000 1.0000 0.0000 10.5000
12 0.0000 0.0000 1.0000 8.7500
13 UX = Y:
14 4.0000 0.0000 -3.0000 -2.0000
15 0.0000 -4.0000 3.2500 10.5000
16 0.0000 0.0000 4.3750 8.7500
17 solved X:
18 1.0000 0.0000 0.0000 1.0000
19 0.0000 1.0000 0.0000 -1.0000
20 0.0000 0.0000 1.0000 2.0000
21 ---
22 P:
23 1.0000 0.0000 0.0000
24 0.0000 1.0000 0.0000
25 0.0000 0.0000 1.0000
26 LY = PB:
27 1.0000 0.0000 0.0000 -2.0000
28 0.7500 1.0000 0.0000 9.0000
29 0.2500 -0.5000 1.0000 3.0000
30 solved Y:
31 1.0000 0.0000 0.0000 -2.0000
32 0.0000 1.0000 0.0000 10.5000
33 0.0000 0.0000 1.0000 8.7500
```

```
UX = Y:
4.0000 0.0000 -3.0000 -2.0000
0.0000 -4.0000 3.2500 10.5000
0.0000 0.0000 4.3750 8.7500
solved X:
1.0000 0.0000 0.0000 1.0000
0.0000 1.0000 0.0000 -1.0000
0.0000 0.0000 1.0000 2.0000
---
```

src/data/exercicio_3.8.out

A.12 exercicio_3.9.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void identity(int newM) {
        m = newM;
        n = newM;
        for(int i=0; i<newM; i++)
            for(int j=0; j<newM; j++)
                V[i][j] = i==j;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void decompose_LDLt(Matrix& b) {
        b.m = m;
        b.n = n;
        for(int k=0; k<m; k++) {
            b.V[k][k] = 1;
            for(int i=k+1; i<m; i++) {
                double pivot = V[i][k] / V[k][k];
                b.V[i][k] = pivot;
```

```
                for(int j=k; j<m; j++)
                    V[i][j] -= pivot*V[k][j];
            }
        }
        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                if (i!=j)
                    V[i][j] = 0;
    }

    void transpose(Matrix& a) {
        m = a.n;
        n = a.m;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                V[i][j] = a.V[j][i];
    }
};

int main() {
    Matrix L, D, Lt;

    while(D.read()) {
        D.decompose_LDLt(L);
        Lt.transpose(L);

        cout << "L:" << endl;
        L.write();

        cout << "D:" << endl;
        D.write();

        cout << "Lt:" << endl;
        Lt.write();

        cout << "---" << endl;
    }
}
```

src/exercicio_3.9.cpp

```
4 4
16 -4 12 -4
-4 2 -1 1
12 -1 14 -2
-4 1 -2 83

2 2
2 -1
-1 2

2 2
3 4
4 2

3 3
16 8 4
8 6 0
```

```
4 0 7

3 3
4 2 1
2 3 -2
1 -2 5
```

src/data/exercicio_3.9.in

```
L:
1.0000 0.0000 0.0000 0.0000
-0.2500 1.0000 0.0000 0.0000
0.7500 2.0000 1.0000 0.0000
-0.2500 0.0000 1.0000 1.0000
D:
16.0000 0.0000 0.0000 0.0000
0.0000 1.0000 0.0000 0.0000
0.0000 0.0000 1.0000 0.0000
0.0000 0.0000 0.0000 81.0000
Lt:
1.0000 -0.2500 0.7500 -0.2500
0.0000 1.0000 2.0000 0.0000
0.0000 0.0000 1.0000 1.0000
0.0000 0.0000 0.0000 1.0000
---
L:
1.0000 0.0000
-0.5000 1.0000
D:
2.0000 0.0000
0.0000 1.5000
Lt:
1.0000 -0.5000
0.0000 1.0000
---
L:
1.0000 0.0000
1.3333 1.0000
D:
3.0000 0.0000
0.0000 -3.3333
Lt:
1.0000 1.3333
0.0000 1.0000
---
L:
1.0000 0.0000 0.0000
0.5000 1.0000 0.0000
0.2500 -1.0000 1.0000
D:
16.0000 0.0000 0.0000
0.0000 2.0000 0.0000
0.0000 0.0000 4.0000
Lt:
1.0000 0.5000 0.2500
0.0000 1.0000 -1.0000
0.0000 0.0000 1.0000
---
L:
```

```
1.0000 0.0000 0.0000
0.5000 1.0000 0.0000
0.2500 -1.2500 1.0000
D:
4.0000 0.0000 0.0000
0.0000 2.0000 0.0000
0.0000 0.0000 1.6250
Lt:
1.0000 0.5000 0.2500
0.0000 1.0000 -1.2500
0.0000 0.0000 1.0000
---
```

src/data/exercicio_3.9.out

A.13 exercicio_3.10.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void decompose_cholesky(Matrix& a) {
        m = a.m;
        n = a.n;
        for(int k=0; k<m; k++) {
            double sum = 0;
            for(int j=0; j<k; j++)
                sum += V[k][j]*V[k][j];

            double r = a.V[k][k] - sum;
            V[k][k] = sqrt(r);

            for(int i=k+1; i<n; i++) {
                sum = 0;
```

```
                for(int j=0; j<k; j++) {
                    sum += V[i][j]*V[k][j];
                }
                V[i][k] = (a.V[i][k] - sum) / V[k][k];
            }
        }
    }

    void transpose(Matrix& a) {
        m = a.n;
        n = a.m;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                V[i][j] = a.V[j][i];
    }
};

int main() {
    Matrix A, G;

    while(A.read()) {
        G.decompose_cholesky(A);
        G.write();
        cout << "---" << endl;
    }
}
```

src/exercicio_3.10.cpp

```
4 4
4 1 1 1
1 3 -1 1
1 -1 2 0
1 1 0 2

4 4
6 2 1 -1
2 4 1 0
1 1 4 -1
-1 0 -1 3
```

src/data/exercicio_3.10.in

```
2.0000 0.0000 0.0000 0.0000
0.5000 1.6583 0.0000 0.0000
0.5000 -0.7538 1.0871 0.0000
0.5000 0.4523 0.0836 1.2403
---
2.4495 0.0000 0.0000 0.0000
0.8165 1.8257 0.0000 0.0000
0.4082 0.3651 1.9235 0.0000
-0.4082 0.1826 -0.4679 1.6066
---
```

src/data/exercicio_3.10.out

A.14 exercicio_3.12.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }

    void write() {
        for(int i=0; i<m; i++) {
            for(int j=0; j<n; j++)
                cout << setprecision(4) << fixed << (j>0 ? " " : "") << V[i][j];
            cout <<  endl;
        }
    }

    void decompose_cholesky(Matrix& a) {
        m = a.m;
        n = a.n;
        for(int k=0; k<m; k++) {
            double sum = 0;
            for(int j=0; j<k; j++)
                sum += V[k][j]*V[k][j];

            double r = a.V[k][k] - sum;
            V[k][k] = sqrt(r);

            for(int i=k+1; i<m; i++) {
                sum = 0;
                for(int j=0; j<k; j++) {
                    sum += V[i][j]*V[k][j];
                }
                V[i][k] = (a.V[i][k] - sum) / V[k][k];
            }
        }
    }

    void copy_last_column_from(Matrix& b) {
        for(int i=0; i<m; i++)
            V[i][n-1] = b.V[i][n-1];
    }

    void transpose2(Matrix& a) {
        //transposing only the square part
        m = a.m;
        n = a.n;
```

```
59
60          for(int i=0; i<m; i++)
61              for(int j=0; j<m; j++)
62                  V[i][j] = a.V[j][i];
63      }
64
65      void solve_L() {
66          for(int i=0; i<m; i++) {
67              double pivot = V[i][i];
68              for(int j=0; j<n; j++) {
69                  V[i][j] /= pivot;
70              }
71
72              for(int j=0; j<i; j++) {
73                  V[i][n-1] -= V[i][j]*V[j][n-1];
74                  V[i][j] = 0;
75              }
76          }
77      }
78
79      void solve_U() {
80          for(int i=m-1; i>=0; i--) {
81              double pivot = V[i][i];
82              for(int j=0; j<n; j++) {
83                  V[i][j] /= pivot;
84              }
85
86              for(int j=i+1; j<m; j++) {
87                  V[i][n-1] -= V[i][j]*V[j][n-1];
88                  V[i][j] = 0;
89              }
90          }
91      }
92  };
93
94  int main() {
95      Matrix A, G, Gt;
96
97      while(A.read()) {
98          G.decompose_cholesky(A);
99          Gt.transpose2(G);
100         G.copy_last_column_from(A);
101
102         cout << "GY = B:" << endl;
103         G.write();
104         G.solve_L();
105
106         cout << "solved Y:" << endl;
107         G.write();
108
109         Gt.copy_last_column_from(G);
110         cout << "GtX = Y:" << endl;
111         Gt.write();
112         Gt.solve_U();
113
114         cout << "solved X:" << endl;
115         Gt.write();
116
```

```
117
118         cout << "---" << endl;
119     }
120 }
```

src/exercicio_3.12.cpp

```
1  4 5
2  4 1 1 1 0.65
3  1 3 -1 1 0.05
4  1 -1 2 0 0
5  1 1 0 2 0.5
6
7  4 5
8  6 2 1 -1 0
9  2 4 1 0 7
10 1 1 4 -1 -1
11 -1 0 -1 3 -2
```

src/data/exercicio_3.12.in

```
1  GY = B:
2  2.0000 0.0000 0.0000 0.0000 0.6500
3  0.5000 1.6583 0.0000 0.0000 0.0500
4  0.5000 -0.7538 1.0871 0.0000 0.0000
5  0.5000 0.4523 0.0836 1.2403 0.5000
6  solved Y:
7  1.0000 0.0000 0.0000 0.0000 0.3250
8  0.0000 1.0000 0.0000 0.0000 -0.0678
9  0.0000 0.0000 1.0000 0.0000 -0.1965
10 0.0000 0.0000 0.0000 1.0000 0.3101
11 GtX = Y:
12 2.0000 0.5000 0.5000 0.5000 0.3250
13 0.0000 1.6583 -0.7538 0.4523 -0.0678
14 0.0000 0.0000 1.0871 0.0836 -0.1965
15 0.0000 0.0000 0.0000 1.2403 0.3101
16 solved X:
17 1.0000 0.0000 0.0000 0.0000 0.2000
18 0.0000 1.0000 0.0000 0.0000 -0.2000
19 0.0000 0.0000 1.0000 0.0000 -0.2000
20 0.0000 0.0000 0.0000 1.0000 0.2500
21 ---
22 GY = B:
23 2.4495 0.0000 0.0000 0.0000 0.0000
24 0.8165 1.8257 0.0000 0.0000 7.0000
25 0.4082 0.3651 1.9235 0.0000 -1.0000
26 -0.4082 0.1826 -0.4679 1.6066 -2.0000
27 solved Y:
28 1.0000 0.0000 0.0000 0.0000 0.0000
29 0.0000 1.0000 0.0000 0.0000 3.8341
30 0.0000 0.0000 1.0000 0.0000 -1.2477
31 0.0000 0.0000 0.0000 1.0000 -2.0440
32 GtX = Y:
33 2.4495 0.8165 0.4082 -0.4082 0.0000
34 0.0000 1.8257 0.3651 0.1826 3.8341
35 0.0000 0.0000 1.9235 -0.4679 -1.2477
36 0.0000 0.0000 0.0000 1.6066 -2.0440
37 solved X:
38 1.0000 0.0000 0.0000 0.0000 -0.8586
```

```
0.0000 1.0000 0.0000 0.0000 2.4188
0.0000 0.0000 1.0000 0.0000 -0.9581
0.0000 0.0000 0.0000 1.0000 -1.2723
---
```

src/data/exercicio_3.12.out

A.15 exercicio_3.13.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }
};

struct Jacobi {
    int n;
    double er;
    double X[MAX], X0[MAX];

    void zero(int newN) {
        n = newN;
        er = 1/0.0;
        for(int i=0; i<n; i++)
            X[i] = X0[i] = 0.0;
    }

    double iterate(Matrix& a) {
        double d = 0, dd = 0;
        for(int i=0; i<n; i++) {
            double sum = 0;
            for(int j=0; j<n; j++)
                if (j!=i)
                    sum+=a.V[i][j]*X0[j];

            X[i] = (a.V[i][n]-sum)/a.V[i][i];

            d = max(d, abs(X[i]-X0[i]));
            dd = max(dd, abs(X[i]));
        }
        for(int i=0; i<n; i++)
            X0[i] = X[i];
        er = d/dd;
```

```
50          return er;
51      }
52
53      void solve(Matrix& a, double err, int maxIter) {
54          zero(a.m);
55          for(int i=0; i<maxIter; i++) {
56              double currentErr = iterate(a);
57              cout << fixed << setprecision(6) <<"Iteration #" << (i+1) << ". Epsilon=" <<
    currentErr << endl;
58              for(int i=0; i<n; i++)
59                  cout << (i>0?" ":"") << X[i];
60              cout << endl << endl;
61
62              if (currentErr < err)
63                  break;
64          }
65      }
66  };
67
68  int main() {
69      Matrix A;
70      Jacobi J;
71
72      while(A.read()) {
73          J.solve(A, 1e-3, 20);
74
75          cout << "---" << endl;
76      }
77  }
```

src/exercicio_3.13.cpp

```
1  4 5
2  4 1 -1 1 -2
3  1 4 -1 -1 -1
4  -1 -1 5 1 0
5  1 -1 1 3 1
6
7  5 6
8  4 1 1 0 1 6
9  -1 -3 1 1 0 6
10 2 1 5 -1 -1 6
11 -1 -1 -1 4 0 6
12 0 2 -1 1 4 6
```

src/data/exercicio_3.13.in

```
1  Iteration #1. Epsilon=1.000000
2  -0.500000 -0.250000 0.000000 0.333333
3
4  Iteration #2. Epsilon=0.416000
5  -0.520833 -0.041667 -0.216667 0.416667
6
7  Iteration #3. Epsilon=0.229368
8  -0.647917 -0.069792 -0.195833 0.565278
9
10 Iteration #4. Epsilon=0.110179
11 -0.672830 0.004340 -0.256597 0.591319
12
```

```
Iteration #5. Epsilon=0.074705
-0.713064 0.001888 -0.251962 0.644589

Iteration #6. Epsilon=0.033859
-0.724610 0.026423 -0.271153 0.655638

Iteration #7. Epsilon=0.024954
-0.738303 0.027274 -0.270765 0.674062

Iteration #8. Epsilon=0.010937
-0.743025 0.035400 -0.277018 0.678781

Iteration #9. Epsilon=0.008515
-0.747800 0.036197 -0.277281 0.685148

Iteration #10. Epsilon=0.003628
-0.749656 0.038917 -0.279350 0.687093

Iteration #11. Epsilon=0.002948
-0.751340 0.039350 -0.279566 0.689308

Iteration #12. Epsilon=0.001224
-0.752056 0.040270 -0.280260 0.690085

Iteration #13. Epsilon=0.001032
-0.752654 0.040470 -0.280374 0.690862

Iteration #14. Epsilon=0.000418
-0.752927 0.040785 -0.280609 0.691166

---
Iteration #1. Epsilon=1.000000
1.500000 -2.000000 1.200000 1.500000 1.500000

Iteration #2. Epsilon=0.381443
1.325000 -1.600000 1.600000 1.675000 2.425000

Iteration #3. Epsilon=0.189041
0.893750 -1.350000 1.810000 1.831250 2.281250

Iteration #4. Epsilon=0.122521
0.814688 -1.084167 1.935000 1.838437 2.169688

Iteration #5. Epsilon=0.050074
0.744870 -1.013750 1.892583 1.916380 2.066224

Iteration #6. Epsilon=0.032634
0.763736 -0.978635 1.901323 1.905926 2.000926

Iteration #7. Epsilon=0.014948
0.769097 -0.985496 1.871603 1.921606 1.988167

Iteration #8. Epsilon=0.006229
0.781431 -0.991963 1.871415 1.913801 1.980247

Iteration #9. Epsilon=0.003418
0.785075 -0.998738 1.864630 1.915221 1.985385

```

```
Iteration #10. Epsilon=0.001512
0.787181 -1.001742 1.865839 1.912742 1.986721

Iteration #11. Epsilon=0.001218
0.787295 -1.002867 1.865368 1.912820 1.989145

Iteration #12. Epsilon=0.000342
0.787088 -1.003036 1.866048 1.912449 1.989571

---
```

src/data/exercicio_3.13.out

A.16 exercicio_3.14.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }
};

struct GaussSeidel {
    int n;
    double er;
    double X[MAX], X0[MAX];

    void zero(int newN) {
        n = newN;
        er = 1/0.0;
        for(int i=0; i<n; i++)
            X[i] = X0[i] = 0.0;
    }

    double iterate(Matrix& a) {
        double d = 0, dd = 0;
        for(int i=0; i<n; i++) {
            double sum = 0;
            for(int j=0; j<i; j++)
                sum+=a.V[i][j]*X[j];

            for(int j=i+1; j<n; j++)
                sum+=a.V[i][j]*X0[j];

```

```
            X[i] = -1/a.V[i][i]*(sum-a.V[i][n]);

            d = max(d, abs(X[i]-X0[i]));
            dd = max(dd, abs(X[i]));
        }
        for(int i=0; i<n; i++)
            X0[i] = X[i];
        er = d/dd;
        return er;
    }

    void solve(Matrix& a, double err, int maxIter) {
        zero(a.m);
        for(int i=0; i<maxIter; i++) {
            double currentErr = iterate(a);
            cout << fixed << setprecision(6) <<"Iteration #" << (i+1) << ". Epsilon=" <<
    currentErr << endl;
            for(int i=0; i<n; i++)
                cout << (i>0?" ":"") << X[i];
            cout << endl << endl;

            if (currentErr < err)
                break;
        }
    }
};

int main() {
    Matrix A;
    GaussSeidel G;

    while(A.read()) {
        G.solve(A, 1e-3, 20);

        cout << "---" << endl;
    }
}
```

src/exercicio_3.14.cpp

```
3 4
5 1 1 5
3 4 1 6
3 3 6 0

4 5
4 1 -1 1 -2
1 4 -1 -1 -1
-1 -1 5 1 0
1 -1 1 3 1

5 6
4 1 1 0 1 6
-1 -3 1 1 0 6
2 1 5 -1 -1 6
-1 -1 -1 4 0 6
0 2 -1 1 4 6
```

src/data/exercicio_3.14.in

```
Iteration #1. Epsilon=1.000000
1.000000 0.750000 -0.875000

Iteration #2. Epsilon=0.195122
1.025000 0.950000 -0.987500

Iteration #3. Epsilon=0.040943
1.007500 0.991250 -0.999375

Iteration #4. Epsilon=0.007363
1.001625 0.998625 -1.000125

Iteration #5. Epsilon=0.001325
1.000300 0.999806 -1.000053

Iteration #6. Epsilon=0.000251
1.000049 0.999976 -1.000013

---
Iteration #1. Epsilon=1.000000
-0.500000 -0.125000 -0.125000 0.500000

Iteration #2. Epsilon=0.200000
-0.625000 -0.000000 -0.225000 0.616667

Iteration #3. Epsilon=0.120235
-0.710417 0.025521 -0.260313 0.665417

Iteration #4. Epsilon=0.037131
-0.737812 0.035729 -0.273500 0.682347

Iteration #5. Epsilon=0.013480
-0.747894 0.039185 -0.278211 0.688430

Iteration #6. Epsilon=0.004741
-0.751457 0.040419 -0.279894 0.690590

Iteration #7. Epsilon=0.001686
-0.752726 0.040855 -0.280492 0.691358

Iteration #8. Epsilon=0.000598
-0.753176 0.041010 -0.280705 0.691630

---
Iteration #1. Epsilon=1.000000
1.500000 -2.500000 1.100000 1.525000 2.643750

Iteration #2. Epsilon=0.433865
1.189062 -1.521354 1.862396 1.882526 2.255645

Iteration #3. Epsilon=0.241892
0.850828 -1.035302 1.894363 1.927472 2.009374

Iteration #4. Epsilon=0.034274
0.782891 -0.987019 1.871616 1.916872 1.982195

Iteration #5. Epsilon=0.005662
0.783302 -0.998271 1.866147 1.912794 1.987474
```

```
59
60 Iteration #6. Epsilon=0.002079
61 0.786163 -1.002407 1.866070 1.912456 1.989607
62
63 Iteration #7. Epsilon=0.000261
64 0.786683 -1.002719 1.866283 1.912562 1.989790
65
66 ---
```

src/data/exercicio_3.14.out

A.17 exercicio_4.3.1.{cpp,in,out}

```
1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4  #define MAX 200
5  using namespace std;
6
7  struct Matrix {
8      int m, n;
9      double V[MAX][MAX];
10
11     bool read() {
12         if (not (cin >> m >> n))
13             return false;
14
15         for(int i=0; i<m; i++)
16             for(int j=0; j<n; j++)
17                 cin >> V[i][j];
18         return true;
19     }
20 };
21
22 struct PowerIteration {
23     int n, p;
24     double X[MAX], Y[MAX], err, mi;
25
26     void zero(int newN) {
27         n = newN;
28         p = 0;
29         err = 1/0.0;
30         mi = 0;
31         for(int i=0; i<n; i++)
32             X[i] = 1.0;
33     }
34
35     bool iterate(Matrix& a, double maxErr) {
36         int maxx = 0;
37         for(int i=0; i<n; i++) {
38             Y[i] = 0;
39             for(int j=0; j<n; j++)
40                 Y[i] += a.V[i][j] * X[j];
41             if (abs(Y[i]) > abs(Y[maxx]))
42                 maxx = i;
43         }
44         mi = Y[p];
45         p = maxx;
```

```
46
47          err = 0;
48          for(int i=0; i<n; i++)
49              err = max(err, abs(X[i] - Y[i]/Y[p]));
50
51          if (err < maxErr)
52              return true;
53
54          for(int i=0; i<n; i++)
55              X[i] = Y[i]/Y[p];
56      }
57
58      void solve(Matrix& a, double maxErr, int maxIter) {
59          zero(a.m);
60          for(int i=0; i<maxIter; i++) {
61              iterate(a, maxErr);
62              cout << fixed << setprecision(6) <<"Iteration #" << (i+1) << ". Epsilon=" << err
    << endl;
63              cout << "Eigenvalue: " << mi << endl;
64              cout << "Eigenvector:" << endl;
65              for(int i=0; i<n; i++)
66                  cout << (i>0?" ":"") << X[i];
67              cout << endl << endl;
68
69              if (err < maxErr)
70                  break;
71          }
72      }
73  };
74
75  int main() {
76      Matrix A;
77      PowerIteration P;
78
79      while(A.read()) {
80          P.solve(A, 1e-4, 20);
81
82          cout << "---" << endl;
83      }
84  }
```

src/exercicio_4.3.1.cpp

```
1  3 3
2  -4 14 0
3  -5 13 0
4  -1 0 2
```

src/data/exercicio_4.3.1.in

```
1  Iteration #1. Epsilon=0.900000
2  Eigenvalue: 10.000000
3  Eigenvector:
4  1.000000 0.800000 0.100000
5
6  Iteration #2. Epsilon=0.211111
7  Eigenvalue: 7.200000
8  Eigenvector:
9  1.000000 0.750000 -0.111111
```

```
10
11 Iteration #3. Epsilon=0.076923
12 Eigenvalue: 6.500000
13 Eigenvector:
14 1.000000 0.730769 -0.188034
15
16 Iteration #4. Epsilon=0.032816
17 Eigenvalue: 6.230769
18 Eigenvector:
19 1.000000 0.722222 -0.220850
20
21 Iteration #5. Epsilon=0.015064
22 Eigenvalue: 6.111111
23 Eigenvector:
24 1.000000 0.718182 -0.235915
25
26 Iteration #6. Epsilon=0.007180
27 Eigenvalue: 6.054545
28 Eigenvector:
29 1.000000 0.716216 -0.243095
30
31 Iteration #7. Epsilon=0.003493
32 Eigenvalue: 6.027027
33 Eigenvector:
34 1.000000 0.715247 -0.246588
35
36 Iteration #8. Epsilon=0.001718
37 Eigenvalue: 6.013453
38 Eigenvector:
39 1.000000 0.714765 -0.248306
40
41 Iteration #9. Epsilon=0.000851
42 Eigenvalue: 6.006711
43 Eigenvector:
44 1.000000 0.714525 -0.249157
45
46 Iteration #10. Epsilon=0.000423
47 Eigenvalue: 6.003352
48 Eigenvector:
49 1.000000 0.714405 -0.249579
50
51 Iteration #11. Epsilon=0.000211
52 Eigenvalue: 6.001675
53 Eigenvector:
54 1.000000 0.714346 -0.249790
55
56 Iteration #12. Epsilon=0.000105
57 Eigenvalue: 6.000837
58 Eigenvector:
59 1.000000 0.714316 -0.249895
60
61 Iteration #13. Epsilon=0.000052
62 Eigenvalue: 6.000419
63 Eigenvector:
64 1.000000 0.714316 -0.249895
65
66 ---
```

src/data/exercicio_4.3.1.out

A.18 exercicio_4.3.2.{cpp,in,out}

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
#define MAX 200
using namespace std;

struct Matrix {
    int m, n;
    double V[MAX][MAX];

    bool read() {
        if (not (cin >> m >> n))
            return false;

        for(int i=0; i<m; i++)
            for(int j=0; j<n; j++)
                cin >> V[i][j];
        return true;
    }
};

struct PowerIteration {
    int n, p;
    double X[MAX], Y[MAX], err, mi, mi0, mi1, mihat;

    void zero(int newN) {
        n = newN;
        p = 0;
        err = 1/0.0;
        mi, mi0, mi1, mihat = 0;
        for(int i=0; i<n; i++)
            X[i] = 1.0;
    }

    bool iterate(Matrix& a, double maxErr) {
        int maxx = 0;
        for(int i=0; i<n; i++) {
            Y[i] = 0;
            for(int j=0; j<n; j++)
                Y[i] += a.V[i][j] * X[j];
            if (abs(Y[i]) > abs(Y[maxx]))
                maxx = i;
        }
        mi = Y[p];
        mihat = mi0 - (mi1 - mi0)*(mi1 - mi0)/(mi-2*mi1+mi0);

        p = maxx;

        err = 0;
        for(int i=0; i<n; i++)
            err = max(err, abs(X[i] - Y[i]/Y[p]));

```

```
53          if (err < maxErr)
54              return true;
55
56          for(int i=0; i<n; i++)
57              X[i] = Y[i]/Y[p];
58          mi0 = mi1;
59          mi1 = mi;
60      }
61
62      void solve(Matrix& a, double maxErr, int maxIter) {
63          zero(a.m);
64          for(int i=0; i<maxIter; i++) {
65              iterate(a, maxErr);
66              cout << fixed << setprecision(6) <<"Iteration #" << (i+1) << ". Epsilon=" << err
    << endl;
67              cout << "Eigenvalue: " << mihat << endl;
68              cout << "Eigenvector:" << endl;
69              for(int i=0; i<n; i++)
70                  cout << (i>0?" ":"") << X[i];
71              cout << endl << endl;
72
73              if (err < maxErr)
74                  break;
75          }
76      }
77  };
78
79  int main() {
80      Matrix A;
81      PowerIteration P;
82
83      while(A.read()) {
84          P.solve(A, 1e-4, 20);
85
86          cout << "---" << endl;
87      }
88  }
```

src/exercicio_4.3.2.cpp

```
1  3 3
2  -4 14 0
3  -5 13 0
4  -1 0 2
```

src/data/exercicio_4.3.2.in

```
1  Iteration #1. Epsilon=0.900000
2  Eigenvalue: 0.000000
3  Eigenvector:
4  1.000000 0.800000 0.100000
5
6  Iteration #2. Epsilon=0.211111
7  Eigenvalue: 7.812500
8  Eigenvector:
9  1.000000 0.750000 -0.111111
10
11 Iteration #3. Epsilon=0.076923
12 Eigenvalue: 6.266667
```

```
Eigenvector:
1.000000 0.730769 -0.188034

Iteration #4. Epsilon=0.032816
Eigenvalue: 6.062500
Eigenvector:
1.000000 0.722222 -0.220850

Iteration #5. Epsilon=0.015064
Eigenvalue: 6.015385
Eigenvector:
1.000000 0.718182 -0.235915

Iteration #6. Epsilon=0.007180
Eigenvalue: 6.003831
Eigenvector:
1.000000 0.716216 -0.243095

Iteration #7. Epsilon=0.003493
Eigenvalue: 6.000957
Eigenvector:
1.000000 0.715247 -0.246588

Iteration #8. Epsilon=0.001718
Eigenvalue: 6.000239
Eigenvector:
1.000000 0.714765 -0.248306

Iteration #9. Epsilon=0.000851
Eigenvalue: 6.000060
Eigenvector:
1.000000 0.714525 -0.249157

Iteration #10. Epsilon=0.000423
Eigenvalue: 6.000015
Eigenvector:
1.000000 0.714405 -0.249579

Iteration #11. Epsilon=0.000211
Eigenvalue: 6.000004
Eigenvector:
1.000000 0.714346 -0.249790

Iteration #12. Epsilon=0.000105
Eigenvalue: 6.000001
Eigenvector:
1.000000 0.714316 -0.249895

Iteration #13. Epsilon=0.000052
Eigenvalue: 6.000000
Eigenvector:
1.000000 0.714316 -0.249895

---
```

src/data/exercicio_4.3.2.out