In [1]:
```
import numpy as np
import mylinalg

np.random.seed(584241)
np.set_printoptions(precision=2, linewidth=140)
```

Relatório Final - Algebra Linear Computacional (CKP8122)

Nome: Cornélio Albuquerque de Sousa

Matrícula: 584241

O arquivo `relatorio.pdf` *é a impressão deste notebook.*

Os códigos estão todos na pasta `src/mylinalg/` .

Resolução de Sistemas de Equações Algébricas Lineares

Eliminação de Gauss

$ Ax = b \Rightarrow RAx = b \Rightarrow Ux = b^* $

Matrizes pequenas

In [2]:
```
# Matriz 3x3 aleatória
A_01 = np.random.randint(1, 10, (3, 3))
x_01 = np.random.randint(0, 10, (3, 1))
b_01 = A_01.dot(x_01)

print("A_01:\n", A_01, sep="", end="\n\n")
print("x_01:\n", x_01, sep="", end="\n\n")
print("b_01:\n", b_01, sep="")
```

```
A_01:
[[4 4 7]
 [6 2 3]
 [1 5 6]]

x_01:
[[3]
 [0]
 [4]]

b_01:
[[40]
 [30]
 [27]]
```

Podemos utilizar o método `mylinalg.processing.ref` (Row Echelon Form) para executar a Eliminação de Gauss. Quando a matriz é quadrada, a sua forma REF é uma matrix triangular superior. Quando a matriz é quadrada e não-singular, todos os elementos na diagonal do REF são diferentes de zero.

$ REF([A|b]) = R[A|b] = [RA|Rb] = [U|b^*] $

In [3]:
```
A_01_aug = np.concatenate((A_01, b_01), axis=1)
A_01_ref = mylinalg.processing.ref(A_01_aug, pivoting="partial", column_lim=len(A_01))

print("REF de [A_01|b_01]:\n", A_01_ref, sep="")
```

```
REF de [A_01|b_01]:
[[ 6.00e+00  2.00e+00  3.00e+00  3.00e+01]
 [ 0.00e+00  4.67e+00  5.50e+00  2.20e+01]
 [ 0.00e+00 -4.44e-16  1.86e+00  7.43e+00]]
```

Agora, para resolver o sistema, basta usar o *backward substitution*.

In [4]:
```
U_01 = A_01_ref[:, :-1]
b_mod_01 = A_01_ref[:, [-1]]
x_01_s = mylinalg.solvers.backward_substitution(U_01, b_mod_01)

print("x_01 alvo:\n", x_01, sep="", end="\n\n")
print("x_01 solução:\n", x_01_s, sep="")
```

```
x_01 alvo:
[[3]
 [0]
 [4]]

x_01 solução:
[[3.]
 [0.]
 [4.]]
```

Matrizes maiores

In [5]:
```python
# Matriz 10x10 aleatória
A_02 = np.random.randint(0, 20, (10, 10))
x_02 = np.random.randint(0, 10, (10, 1))
b_02 = A_02.dot(x_02)

print("A_02:\n", A_02, sep="", end="\n\n")
print("x_02:\n", x_02.T, sep="", end="\n\n")
print("b_02:\n", b_02.T, sep="")
```

```
A_02:
[[ 5 13  6  7  1 16 10  3 14  5]
 [ 9 13  8 12 10 12  0 19  5  6]
 [ 1 19  5  0 18 11 13 10 17 13]
 [ 6 12  5  3  7  1  8  5  5  3]
 [19 16 14 19  5  4  9 17 11 17]
 [13 13 17  5  1 11  5  2 14 15]
 [14 13  7 13  2 19 11  3  9  8]
 [ 2 19  9  4 12  1 13 10 13 14]
 [10 10 15  5 12  3 13 16  4 12]
 [ 6  7  1  7 11  1 15 15 15 14]]

x_02:
[[2 7 0 7 7 7 9 5 0 4]]

b_02:
[[394 466 557 282 580 351 503 479 475 460]]
```

In [6]:
```python
A_02_aug = np.concatenate((A_02, b_02), axis=1)
A_02_ref = mylinalg.processing.ref(A_02_aug, pivoting="partial", column_lim=len(A_02))

print("REF de [A_02|b_02]:\n", A_02_ref.round(1), sep="")
```

```
REF de [A_02|b_02]:
[[ 1.90e+01  1.60e+01  1.40e+01  1.90e+01  5.00e+00  4.00e+00  9.00e+00  1.70e+01  1.10e+01  1.70e+01  5.80e+02]
 [ 0.00e+00  1.82e+01  4.30e+00 -1.00e+00  1.77e+01  1.08e+01  1.25e+01  9.10e+00  1.64e+01  1.21e+01  5.26e+02]
 [ 0.00e+00  0.00e+00  7.30e+00 -4.90e+00  7.80e+00 -0.00e+00  7.20e+00  6.30e+00 -3.20e+00  2.00e+00  1.24e+02]
 [ 0.00e+00  0.00e+00  0.00e+00  5.30e+00 -9.20e+00 -9.70e+00 -3.30e+00 -3.50e+00 -2.30e+00 -3.00e-01 -1.43e+02]
 [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00 -1.74e+01  1.20e+00 -1.14e+01 -1.87e+01  6.30e+00 -1.00e-01 -3.10e+02]
 [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  1.43e+01  6.00e+00  5.00e-01  2.80e+00 -5.20e+00  1.35e+02]
 [ 0.00e+00  0.00e+00  0.00e+00 -0.00e+00  0.00e+00  0.00e+00 -1.70e+01  1.20e+00 -3.30e+00 -5.00e-01 -1.49e+02]
 [ 0.00e+00  0.00e+00  0.00e+00 -0.00e+00  0.00e+00  0.00e+00  0.00e+00 -4.10e+00 -7.60e+00 -1.40e+00 -2.61e+01]
 [ 0.00e+00  0.00e+00  0.00e+00 -0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00 -1.25e+01 -1.19e+01 -4.77e+01]
 [ 0.00e+00  0.00e+00 -0.00e+00 -0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  4.00e-01  1.80e+00]]
```

In [7]:
```python
U_02 = A_02_ref[:, :-1]
b_mod_02 = A_02_ref[:, [-1]]
x_02_s = mylinalg.solvers.backward_substitution(U_02, b_mod_02)

print("x_02 alvo:\n", x_02, sep="", end="\n\n")
print("(PIVOTAÇÃO PARCIAL) x_02 solução:\n", x_02_s, sep="")
```

```
x_02 alvo:
[[2]
 [7]
 [0]
 [7]
 [7]
 [7]
 [9]
 [5]
 [0]
 [4]]

(PIVOTAÇÃO PARCIAL) x_02 solução:
[[ 2.00e+00]
 [ 7.00e+00]
 [-2.47e-13]
 [ 7.00e+00]
 [ 7.00e+00]
 [ 7.00e+00]
 [ 9.00e+00]
 [ 5.00e+00]
 [-1.72e-13]
 [ 4.00e+00]]
```

In [8]:
```
x_02_s = mylinalg.solvers.gauss_elimination_solver(A_02, b_02, pivoting="complete")

print("x_02 alvo:\n", x_02, sep="", end="\n\n")
print("(PIVOTAÇÃO COMPLETA) x_02 solução:\n", x_02_s, sep="")
```

```
x_02 alvo:
[[2]
 [7]
 [0]
 [7]
 [7]
 [7]
 [9]
 [5]
 [0]
 [4]]

(PIVOTAÇÃO COMPLETA) x_02 solução:
[[2.00e+00]
 [7.00e+00]
 [4.74e-13]
 [7.00e+00]
 [7.00e+00]
 [7.00e+00]
 [9.00e+00]
 [5.00e+00]
 [3.37e-13]
 [4.00e+00]]
```

Gauss-Jordan

$ [A|I|b] \rightarrow R[A|I|b] \rightarrow [RA|R|Rb] \rightarrow [I|R|x] \rightarrow R=A^{-1} $

In [9]:
```
# Matriz 6x6 aleatória
A_03 = np.random.randint(1, 20, (6, 6))
x_03 = np.random.randint(0, 10, (6, 1))
b_03 = A_03.dot(x_03)

print("A_03:\n", A_03, sep="", end="\n\n")
print("x_03:\n", x_03, sep="", end="\n\n")
print("b_03:\n", b_03, sep="")
```

```
A_03:
[[11  3 14  7 13  7]
 [17  1  9  7  8 19]
 [10 18 14  5 17  3]
 [ 4  1  8 10 12 16]
 [ 7  4 12  7  7 17]
 [ 2  5 18  6  3  6]]

x_03:
[[0]
 [5]
 [2]
 [2]
 [0]
 [3]]

b_03:
[[ 78]
 [ 94]
 [137]
 [ 89]
 [109]
 [ 91]]
```

Podemos utilizar o método `mylinalg.processing.rref` (Reduced Row Echelon Form) para executar o método de Gauss-Jordan. Quando a matriz é quadrada e não-singular, a sua forma RREF é a matriz identidade.

$ RREF([A|I|b]) = R[A|I|b] = [RA|RI|Rb] = [I|R=A^{-1}|x] $

$ Ax = b \Rightarrow RAx = Rb \Rightarrow Ix= x = Rb$

In [10]:
```
A_03_aug = np.concatenate((A_03, b_03), axis=1)
A_03_rref = mylinalg.processing.rref(A_03_aug, pivoting="partial", column_lim=len(A_03))

print("REF de [A_03|b_03]:\n", A_03_rref.round(4), sep="", end="\n\n")
print("x_03 alvo:\n", x_03, sep="")
```

```
REF de [A_03|b_03]:
[[ 1.  0.  0.  0.  0.  0. -0.]
 [ 0.  1.  0.  0. -0. -0.  5.]
 [ 0.  0.  1.  0.  0. -0.  2.]
 [ 0.  0.  0.  1.  0.  0.  2.]
 [ 0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  3.]]

x_03 alvo:
[[0]
 [5]
 [2]
 [2]
 [0]
 [3]]
```

In [11]:
```
x_03_s = mylinalg.solvers.gauss_jordan_solver(A_03, b_03)

print("x_03 solução:\n", x_03_s, sep="")
```

```
x_03 solução:
[[-9.88e-16]
 [ 5.00e+00]
 [ 2.00e+00]
 [ 2.00e+00]
 [ 1.50e-15]
 [ 3.00e+00]]
```

Decomposição LU

A decomposição LU consiste em escrever a matriz A como o produto de duas matrizes: à esquerda, uma matriz triangular inferior L, e à direita, uma matriz triangular superior (no caso de matrizes quadradas) ou trapezoidal superior (no caso de matrizes retangulares) U.

$A = LU$

A decomposição LU é obtida a partir do processo de Eliminação de Gauss, com a adição de alguns passos que permitem registrar, na matriz L, os multiplicadores usados nas operações de eliminação que transformam A em U.

Decomposição e reconstrução

In [12]:
```
# Matriz 7x5 aleatória
A_04 = np.random.randint(0, 20, (7, 5))

print("A_04:\n", A_04, sep="")
```

```
A_04:
[[16  4 10 14 12]
 [15  4 19  8 18]
 [ 8 18 16 18 15]
 [16  6 12  5  1]
 [ 1  8  2 10 10]
 [ 4  8 16 15  4]
 [ 7 11  2 11 18]]
```

In [13]:
```
L_04, U_04, P_04, Q_04 = mylinalg.decompositions.lu(A_04, pivoting="none")

print(f"L_04 {L_04.shape}:\n", L_04.round(2), sep="", end="\n\n")
print(f"U_04 {U_04.shape}:\n", U_04.round(4), sep="", end="\n\n")
print("(P_04, Q_04):", (P_04, Q_04))
```

```
L_04 (7, 7):
[[ 1.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]
 [ 9.40e-01  1.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]
 [ 5.00e-01  6.40e+01  1.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]
 [ 1.00e+00  8.00e+00  1.20e-01  1.00e+00  0.00e+00  0.00e+00  0.00e+00]
 [ 6.00e-02  3.10e+01  4.90e-01 -1.60e-01  1.00e+00  0.00e+00  0.00e+00]
 [ 2.50e-01  2.80e+01  4.20e-01 -1.15e+00 -4.14e+00  1.00e+00  0.00e+00]
 [ 4.40e-01  3.70e+01  5.90e-01  6.40e-01  3.82e+00  0.00e+00  1.00e+00]]

U_04 (7, 5):
[[ 1.60e+01  4.00e+00  1.00e+01  1.40e+01  1.20e+01]
 [ 0.00e+00  2.50e-01  9.62e+00 -5.12e+00  6.75e+00]
 [ 0.00e+00  0.00e+00 -6.05e+02  3.39e+02 -4.23e+02]
 [ 0.00e+00  0.00e+00  0.00e+00 -1.00e+01 -1.26e+01]
 [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  5.67e+00]
 [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]
 [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]]

(P_04, Q_04): (None, None)
```

In [14]:
```
print("A_04:\n", A_04, sep="", end="\n\n")
print("A_04 = L.U:\n", L_04.dot(U_04), sep="")
```

```
A_04:
[[16  4 10 14 12]
 [15  4 19  8 18]
 [ 8 18 16 18 15]
 [16  6 12  5  1]
 [ 1  8  2 10 10]
 [ 4  8 16 15  4]
 [ 7 11  2 11 18]]

A_04 = L.U:
[[16.  4. 10. 14. 12.]
 [15.  4. 19.  8. 18.]
 [ 8. 18. 16. 18. 15.]
 [16.  6. 12.  5.  1.]
 [ 1.  8.  2. 10. 10.]
 [ 4.  8. 16. 15.  4.]
 [ 7. 11.  2. 11. 18.]]
```

In [15]:
```
L_04, U_04, P_04, Q_04 = mylinalg.decompositions.lu(A_04, pivoting="complete")

print("P.A.Q:\n", P_04.dot(A_04).dot(Q_04), sep="", end="\n\n")
print("P.A.Q = L.U:\n", L_04.dot(U_04), sep="")
```

```
P.A.Q:
[[19. 18.  8. 15.  4.]
 [ 2. 18. 11.  7. 11.]
 [16.  4. 15.  4.  8.]
 [12.  1.  5. 16.  6.]
 [10. 12. 14. 16.  4.]
 [16. 15. 18.  8. 18.]
 [ 2. 10. 10.  1.  8.]]

P.A.Q = L.U:
[[19. 18.  8. 15.  4.]
 [ 2. 18. 11.  7. 11.]
 [16.  4. 15.  4.  8.]
 [12.  1.  5. 16.  6.]
 [10. 12. 14. 16.  4.]
 [16. 15. 18.  8. 18.]
 [ 2. 10. 10.  1.  8.]]
```

Solução de sistemas

Para o caso sem pivotação:

$ Ax = b \Rightarrow LUx = b $

Chamando $Ux=y$, tem-se $Ly=b$. y pode ser obtido usando *forward substitution*. Depois de obtido y, solucionar $Ux=y$ usando *backward substitution*.

Para o caso com pivotação parcial:

$ PA = LU \Rightarrow PAx = LUx \Rightarrow Pb = LUx $

Ou seja, aplicar a pivotação P em b e solucionar usando $b^* = Pb$.

In [16]:
```
# Matriz 6x6 aleatória
A_05 = np.random.randint(0, 20, (6, 6))
x_05 = np.random.randint(0, 10, (6, 1))
b_05 = A_05.dot(x_05)

print("A_05:\n", A_05, sep="", end="\n\n")
print("x_05:\n", x_05, sep="", end="\n\n")
print("b_05:\n", b_05, sep="")
```

```
A_05:
[[13 12  3  6 11 12]
 [16  8 18 10 18  1]
 [14 12 13  5 19 12]
 [ 7  6 14  3  9  7]
 [15  4  9  7  1  6]
 [ 6 13 13  4 16  7]]

x_05:
[[7]
 [4]
 [1]
 [9]
 [4]
 [3]]

b_05:
[[276]
 [327]
 [316]
 [171]
 [215]
 [228]]
```

In [17]:
```
x_05_s = mylinalg.solvers.lu_solver(A_05, b_05)

print("x_05 alvo:\n", x_05, sep="", end="\n\n")
print("x_05 solução:\n", x_05_s, sep="")
```

```
x_05 alvo:
[[7]
 [4]
 [1]
 [9]
 [4]
 [3]]

x_05 solução:
[[7.]
 [4.]
 [1.]
 [9.]
 [4.]
 [3.]]
```

RREF

Extensão do REF, onde elementos abaixo e acima dos pivôs são zerados e os pivôs são escalonados para serem igual a 1.

In [18]:
```
# Matriz 4x6 aleatória
A_06 = np.random.randint(0, 10, (4, 6))

print("A_06:\n", A_06, sep="")
```

```
A_06:
[[6 3 0 4 2 7]
 [9 5 3 7 5 9]
 [3 9 7 4 3 4]
 [0 7 4 6 1 3]]
```

In [19]:
```
A_06_rref, A_06_pivots = mylinalg.processing.rref(
    A_06, pivoting="complete", return_pivots_loc=True
)

print("Pivôs A_06:", A_06_pivots, end="\n\n")
print("RREF A_06:\n", A_06_rref.round(5), sep="")
```

```
Pivôs A_06: [(0, 0), (1, 1), (2, 2), (3, 3)]

RREF A_06:
[[ 1.   -0.   -0.   -0.    0.41  0.71]
 [ 0.    1.   -0.    0.   -0.41  0.64]
 [ 0.    0.    1.    0.    0.2   0.2 ]
 [ 0.    0.    0.    1.    0.67 -0.67]]
```

In [20]:
```
# Matriz 6x4 aleatória
A_07 = np.random.randint(0, 10, (6, 4))

print("A_07:\n", A_07, sep="")
```

```
A_07:
[[2 9 4 2]
 [2 7 5 9]
 [2 0 7 8]
 [4 0 4 5]
 [9 3 9 5]
 [8 6 6 6]]
```

In [21]:
```
A_07_rref, A_07_pivots = mylinalg.processing.rref(
    A_07, pivoting="complete", return_pivots_loc=True
)

print("Pivôs A_07:", A_07_pivots, end="\n\n")
print("RREF A_07:\n", A_07_rref.round(5), sep="")
```

```
Pivôs A_07: [(0, 0), (1, 1), (2, 2), (3, 3)]

RREF A_07:
[[ 1. -0.  0.  0.]
 [ 0.  1. -0. -0.]
 [ 0.  0.  1. -0.]
 [ 0.  0.  0.  1.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
```

Decomposição de Cholesky

$ A = LL^T$

A decomposição de Cholesky é definida apenas para matrizes quadradas, simétricas e definidas positivas. Ela pode ser vista como um caso particular da decomposição LU, no qual a matriz superior satisfaz $U = L^T$.

In [22]:
```
# Matriz simétrica 6x6 aleatória com diagonal destacada
A_08 = np.random.randint(0, 10, (6, 6))
for i in range(6):
    for j in range(i + 1, 6):
        A_08[i, j] = A_08[j, i]
np.fill_diagonal(A_08, np.random.randint(20, 30, (6,)))

print("A_08:\n", A_08, sep="")
```

```
A_08:
[[26  1  7  5  8  7]
 [ 1 29  7  9  4  7]
 [ 7  7 29  1  2  8]
 [ 5  9  1 28  9  1]
 [ 8  4  2  9 24  4]
 [ 7  7  8  1  4 22]]
```

In [23]:
```
L = mylinalg.decompositions.cholesky(A_08)

print("A_08 = L.L^T:\n", L.dot(L.T), sep="")
```

```
A_08 = L.L^T:
[[26.  1.  7.  5.  8.  7.]
 [ 1. 29.  7.  9.  4.  7.]
 [ 7.  7. 29.  1.  2.  8.]
 [ 5.  9.  1. 28.  9.  1.]
 [ 8.  4.  2.  9. 24.  4.]
 [ 7.  7.  8.  1.  4. 22.]]
```

In [24]:
```
# Matriz simétrica 6x6 aleatória com diagonal reduzida
A_09 = np.random.randint(20, 30, (6, 6))
for i in range(6):
    for j in range(i + 1, 6):
        A_09[i, j] = A_09[j, i]
np.fill_diagonal(A_09, np.random.randint(1, 10, (6,)))

print("A_09:\n", A_09, sep="")
```

```
A_09:
[[ 7 24 20 27 26 21]
 [24  1 21 22 24 21]
 [20 21  2 28 25 25]
 [27 22 28  9 25 26]
 [26 24 25 25  2 24]
 [21 21 25 26 24  1]]
```

A linha de código abaixo irá gerar uma exceção, pois `A_09` não é uma matriz definida positivamente (positive definite). A linha está comentada para não interromper a execução contínua dos código.

In [25]:
```
# L = mylinalg.decompositions.cholesky(A_09)
```

Ortogonalização de Gram-Schmidt

In [26]:
```
# Espaço R^6, porém com apenas 3 vetores
A_10 = np.random.randint(0, 10, (6, 3))

print("A_10:\n", A_10, sep="")
```

```
A_10:
[[2 5 5]
 [7 2 2]
 [0 3 3]
 [4 6 3]
 [9 0 3]
 [9 8 2]]
```

In [27]:
```python
# Estender com vetores unitários
A_10_ext = np.concatenate((A_10, np.identity(6)), axis=1)

_, A_10_base, _ = mylinalg.decompositions.rank_revealing_qr(A_10_ext)

print("A_10 Base estendida:\n", A_10_base.round(5), sep="", end="\n\n")
print("A_10 Normas das colunas:", np.linalg.norm(A_10_base, axis=0))
```

```
A_10 Base estendida:
[[ 0.13  0.46  0.61 -0.   -0.29 -0.56]
 [ 0.46 -0.19  0.04  0.87  0.   -0.  ]
 [ 0.    0.34  0.41  0.05 -0.18  0.82]
 [ 0.26  0.45  0.01 -0.04  0.85  0.  ]
 [ 0.59 -0.54  0.38 -0.45  0.07  0.08]
 [ 0.59  0.38 -0.56 -0.2  -0.39  0.04]]

A_10 Normas das colunas: [1. 1. 1. 1. 1. 1.]
```

Mínimos Quadrados

Alguns sistemas $Ax=b$ não possuem solução pois $ b \notin \mathbb{Col}(A) $. Porém, é possível obter uma solução $Ax=b^*$ de modo a minimizar a distância L2 entre b e b^*. Para tal, basta projetar b no espaço de colunas de A e resolver o sistema com esse novo vetor b projetado.

In [28]:
```python
# Colunas de `A_11` não conseguem spannar vetores que tenham
# pelo menos uma das duas últimas componentes diferente de zero
A_11 = np.random.randint(0, 10, (6, 3))
A_11[-2:, :] = 0
x_11 = np.random.randint(0, 10, (3, 1))
# `b_11_proj` pertence ao espaço de colunas e é a projeção de `b_11_desl`
b_11_proj = A_11.dot(x_11)
# `b_11_desl` não pertence ao espaço de colunas
b_11_desl = b_11_proj.copy()
b_11_desl[-2:, :] = np.random.randint(1, 10, (2, 1))

print("A_11:\n", A_11, sep="", end="\n\n")
print("x_11:\n", x_11, sep="", end="\n\n")
print("b_11_proj:\n", b_11_proj.T, sep="", end="\n\n")
print("b_11_desl:\n", b_11_desl.T, sep="")
```

```
A_11:
[[6 2 5]
 [9 2 9]
 [4 1 3]
 [2 0 8]
 [0 0 0]
 [0 0 0]]

x_11:
[[2]
 [0]
 [8]]

b_11_proj:
[[52 90 32 68  0  0]]

b_11_desl:
[[52 90 32 68  2  4]]
```

In [29]:
```python
x_11_ls = mylinalg.solvers.least_squares_solver(A_11, b_11_desl)
b_11_ls = A_11.dot(x_11_ls)

print("b_11 alvo:\n", b_11_proj.T, sep="", end="\n\n")
print("b_11 mínimos quadrados:\n", b_11_ls.T, sep="", end="\n\n")
print("x_11 alvo:\n", x_11, sep="", end="\n\n")
print("x_11 mínimos quadrados:\n", x_11_ls, sep="")
```

```
b_11 alvo:
[[52 90 32 68  0  0]]

b_11 mínimos quadrados:
[[52. 90. 32. 68.  0.  0.]]

x_11 alvo:
[[2]
 [0]
 [8]]

x_11 mínimos quadrados:
[[ 2.00e+00]
 [-2.27e-13]
 [ 8.00e+00]]
```

Autovalores e Autovetores

Método QR

Foi implementado apenas para matrizes simétricas.

In [30]:
```
A_12 = np.random.randint(0, 10, (8, 8))
for i in range(8):
    for j in range(i + 1, 8):
        A_12[i, j] = A_12[j, i]

print("A_12:\n", A_12, sep="")
```

```
A_12:
[[9 5 3 6 8 4 6 6]
 [5 9 9 7 9 0 1 4]
 [3 9 9 1 4 8 2 9]
 [6 7 1 7 6 3 9 5]
 [8 9 4 6 0 6 8 5]
 [4 0 8 3 6 3 4 9]
 [6 1 2 9 8 4 7 2]
 [6 4 9 5 5 9 2 6]]
```

In [31]:
```
A_12_eigenvals, A_12_eigenvecs = mylinalg.eigen.qr_method(A_12)
```

```
>>> PASSO 1
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.2  -0.4  -0.54 -0.27 -0.4  -0.4 ]
 [ 0.   -0.2   0.97 -0.06 -0.08 -0.04 -0.06 -0.06]
 [ 0.   -0.4  -0.06  0.88 -0.16 -0.08 -0.12 -0.12]
 [ 0.   -0.54 -0.08 -0.16  0.78 -0.11 -0.16 -0.16]
 [ 0.   -0.27 -0.04 -0.08 -0.11  0.95 -0.08 -0.08]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08  0.88 -0.12]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08 -0.12  0.88]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8  -6.5  -0.2   6.7  -2.2   1.5   1.4]
 [ -0.   -6.5   3.7  -5.9  -4.1   3.8  -3.8   2.7]
 [ -0.   -0.2  -5.9   0.3  -0.7  -0.7   4.6  -0.3]
 [ -0.    6.7  -4.1  -0.7  -6.    2.6   4.4   0.1]
 [ -0.   -2.2   3.8  -0.7   2.6   1.1   1.9   6.3]
 [  0.    1.5  -3.8   4.6   4.4   1.9   5.   -1. ]
 [  0.    1.4   2.7  -0.3   0.1   6.3  -1.    2.1]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.2  -0.4  -0.54 -0.27 -0.4  -0.4 ]
 [ 0.   -0.2   0.97 -0.06 -0.08 -0.04 -0.06 -0.06]
 [ 0.   -0.4  -0.06  0.88 -0.16 -0.08 -0.12 -0.12]
 [ 0.   -0.54 -0.08 -0.16  0.78 -0.11 -0.16 -0.16]
 [ 0.   -0.27 -0.04 -0.08 -0.11  0.95 -0.08 -0.08]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08  0.88 -0.12]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08 -0.12  0.88]]

>>> PASSO 2
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.   -0.66 -0.02  0.68 -0.22  0.15  0.14]
 [ 0.    0.   -0.02  1.    0.01 -0.    0.    0.  ]
 [ 0.    0.    0.68  0.01  0.72  0.09 -0.06 -0.06]
 [ 0.    0.   -0.22 -0.    0.09  0.97  0.02  0.02]
 [ 0.    0.    0.15  0.   -0.06  0.02  0.99 -0.01]
 [ 0.    0.    0.14  0.   -0.06  0.02 -0.01  0.99]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8   9.8  -0.    0.    0.    0.    0. ]
 [ -0.    9.8   3.6   4.4  -5.3   1.5   4.7  -3.8]
 [ -0.    0.    4.4   0.5  -4.9   0.7   3.8  -1.2]
 [ -0.    0.   -5.3  -4.9  -5.    3.4   1.    2.9]
 [ -0.   -0.    1.5   0.7   3.4   0.5   3.2   5.6]
 [  0.   -0.    4.7   3.8   1.    3.2   3.4  -1.1]
 [  0.   -0.   -3.8  -1.2   2.9   5.6  -1.1   3.2]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.4  -0.5  -0.28 -0.39 -0.4 ]
 [ 0.   -0.2  -0.71 -0.08  0.61 -0.26  0.09  0.08]
 [ 0.   -0.4  -0.1   0.88 -0.14 -0.09 -0.12 -0.12]
 [ 0.   -0.54  0.57 -0.16  0.52 -0.02 -0.22 -0.22]
 [ 0.   -0.27 -0.28 -0.08 -0.01  0.91 -0.06 -0.06]
 [ 0.   -0.4   0.07 -0.12 -0.21 -0.06  0.87 -0.13]
 [ 0.   -0.4   0.05 -0.12 -0.21 -0.07 -0.13  0.87]]

>>> PASSO 3
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.   -0.47  0.57 -0.16 -0.51  0.41]
 [ 0.    0.    0.    0.57  0.78  0.06  0.2  -0.16]
 [ 0.    0.    0.   -0.16  0.06  0.98 -0.06  0.05]
 [ 0.    0.    0.   -0.51  0.2  -0.06  0.82  0.14]
 [ 0.    0.    0.    0.41 -0.16  0.05  0.14  0.89]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8   9.8   0.   -0.    0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3   0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4  -4.6   3.4  -0.1   0.1]
 [ -0.    0.    0.   -4.6  -6.    2.6   3.3   1.8]
 [ -0.   -0.   -0.    3.4   2.6   1.    3.5   5.1]
 [  0.   -0.   -0.   -0.1   3.3   3.5   0.1   0.9]
 [  0.   -0.    0.    0.1   1.8   5.1   0.9   2. ]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01 -0.65 -0.24 -0.26 -0.5 ]
 [ 0.   -0.2  -0.71  0.41  0.42 -0.21  0.26 -0.06]
```

```
 [ 0.   -0.4  -0.1  -0.47  0.38 -0.24 -0.58  0.26]
 [ 0.   -0.54  0.57  0.4   0.3   0.04 -0.03 -0.37]
 [ 0.   -0.27 -0.28 -0.11  0.    0.91 -0.07 -0.05]
 [ 0.   -0.4   0.07 -0.55 -0.05 -0.11  0.72 -0.01]
 [ 0.   -0.4   0.05  0.37 -0.4  -0.01  0.04  0.73]]

>>> PASSO 4
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.   -0.81  0.59 -0.02  0.02]
 [ 0.    0.    0.    0.    0.59  0.81  0.01 -0.01]
 [ 0.    0.    0.    0.   -0.02  0.01  1.    0.  ]
 [ 0.    0.    0.    0.    0.02 -0.01  0.    1.  ]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.   -0.    0.    0. ]
 [-14.9  34.8   9.8   0.    0.    0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.    0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9   2.6  -0.7   1.7]
 [ -0.    0.    0.   -0.    2.6   1.    4.8   5.1]
 [  0.   -0.   -0.   -0.   -0.7   4.8   0.    1. ]
 [  0.   -0.    0.    0.    1.7   5.1   1.    2. ]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38 -0.57 -0.25 -0.52]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.08  0.25 -0.04]
 [ 0.   -0.4  -0.1  -0.47 -0.43  0.02 -0.59  0.27]
 [ 0.   -0.54  0.57  0.4  -0.23  0.21 -0.04 -0.36]
 [ 0.   -0.27 -0.28 -0.11  0.53  0.74 -0.06 -0.06]
 [ 0.   -0.4   0.07 -0.55 -0.04 -0.11  0.72 -0.01]
 [ 0.   -0.4   0.05  0.37  0.33 -0.25  0.05  0.72]]

>>> PASSO 5
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -0.81  0.21 -0.55]
 [ 0.    0.    0.    0.    0.    0.21  0.97  0.07]
 [ 0.    0.    0.    0.    0.   -0.55  0.07  0.83]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.    0.    0.    0. ]
 [-14.9  34.8   9.8   0.    0.   -0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.    0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9  -3.2   0.    0. ]
 [  0.    0.   -0.    0.   -3.2   4.   -5.2  -3. ]
 [  0.   -0.    0.   -0.   -0.   -5.2   2.3  -1. ]
 [  0.   -0.    0.    0.    0.   -3.   -1.   -3.2]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38  0.7  -0.4  -0.13]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.01  0.26 -0.06]
 [ 0.   -0.4  -0.1  -0.47 -0.43 -0.29 -0.56  0.17]
 [ 0.   -0.54  0.57  0.4  -0.23  0.02 -0.01 -0.42]
 [ 0.   -0.27 -0.28 -0.11  0.53 -0.58  0.09 -0.46]
 [ 0.   -0.4   0.07 -0.55 -0.04  0.25  0.67  0.1 ]
 [ 0.   -0.4   0.05  0.37  0.33 -0.19  0.04  0.74]]

>>> PASSO 6
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.   -0.86 -0.51]
 [ 0.    0.    0.    0.    0.    0.   -0.51  0.86]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.    0.   -0.    0. ]
 [-14.9  34.8   9.8   0.    0.   -0.   -0.   -0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.   -0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9  -3.2  -0.    0. ]
 [  0.    0.   -0.    0.   -3.2   4.    6.   -0. ]
 [ -0.    0.   -0.   -0.   -0.    6.    0.    2.9]]
```

```
 [  0.   -0.   -0.    0.    0.    0.    2.9  -0.9]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38  0.7   0.41  0.09]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.01 -0.19 -0.19]
 [ 0.   -0.4  -0.1  -0.47 -0.43 -0.29  0.39  0.43]
 [ 0.   -0.54  0.57  0.4  -0.23  0.02  0.23 -0.36]
 [ 0.   -0.27 -0.28 -0.11  0.53 -0.58  0.15 -0.44]
 [ 0.   -0.4   0.07 -0.55 -0.04  0.25 -0.63 -0.26]
 [ 0.   -0.4   0.05  0.37  0.33 -0.19 -0.41  0.62]]
```

In [32]:
```
print("A_12 Autovalores:\n", A_12_eigenvals.round(4), sep="", end="\n\n")
print("A_12 Autovetores:\n", A_12_eigenvecs.round(5), sep="")
```

```
A_12 Autovalores:
[[ 43.76   0.     0.     0.     0.    -0.    -0.    -0.  ]
 [ -0.    14.22   0.     0.     0.     0.    -0.    -0.  ]
 [  0.     0.     9.02  -0.    -0.     0.     0.     0.  ]
 [ -0.     0.    -0.   -10.68   0.    -0.    -0.    -0.  ]
 [  0.    -0.     0.     0.     3.19   0.     0.     0.  ]
 [ -0.     0.    -0.    -0.    -0.    -5.86   0.    -0.  ]
 [  0.     0.     0.    -0.     0.     0.    -3.87  -0.  ]
 [ -0.     0.     0.     0.    -0.    -0.     0.     0.22]]

A_12 Autovetores:
[[ 0.38 -0.25  0.07  0.09  0.82 -0.15  0.27 -0.12]
 [ 0.37  0.13 -0.77  0.45 -0.02  0.07 -0.22 -0.  ]
 [ 0.36  0.59 -0.04 -0.26 -0.21 -0.16  0.51 -0.35]
 [ 0.35 -0.41 -0.11 -0.18 -0.33 -0.38  0.27  0.58]
 [ 0.37 -0.14 -0.08 -0.68  0.04  0.1  -0.56 -0.21]
 [ 0.3   0.22  0.48  0.36 -0.1  -0.52 -0.46 -0.  ]
 [ 0.31 -0.49  0.25  0.29 -0.4   0.36  0.14 -0.46]
 [ 0.37  0.31  0.3   0.03  0.08  0.63  0.    0.52]]
```

Método da Potência

O Método da Potência é um procedimento iterativo que busca um autovalor (e seu autovetor correspondente) próximo de um ponto de busca determinado pelo *shift*.

Abaixo, esse método será utilizado para validar alguns dos autovalores obtidos pelo Método QR acima.

`shift = 0.0`

In [33]:
```
shift = 0.0
A_12_eval, A_12_evec = mylinalg.eigen.inverse_power_iteration(A_12, shift=shift)

print("Autovalor achado com shift=0.0:\n", A_12_eval, sep="", end="\n\n")
print("Autovetor correspondente:\n", A_12_evec, sep="")
```

```
Autovalor achado com shift=0.0:
0.22253899023942217

Autovetor correspondente:
[[ 0.12]
 [ 0.  ]
 [ 0.35]
 [-0.58]
 [ 0.21]
 [ 0.  ]
 [ 0.46]
 [-0.52]]
```

`shift = 10.0`

In [34]:
```
shift = 10.0
A_12_eval, A_12_evec = mylinalg.eigen.inverse_power_iteration(A_12, shift=shift)

print(f"Autovalor achado com shift={shift}:\n", A_12_eval, sep="", end="\n\n")
print("Autovetor correspondente:\n", A_12_evec, sep="")
```

```
Autovalor achado com shift=10.0:
9.015300298296026

Autovetor correspondente:
[[ 0.07]
 [-0.77]
 [-0.04]
 [-0.11]
 [-0.08]
 [ 0.48]
 [ 0.25]
 [ 0.3 ]]
```

shift = -10.0

In [35]:
```
shift = -10.0
A_12_eval, A_12_evec = mylinalg.eigen.inverse_power_iteration(A_12, shift=shift)

print(f"Autovalor achado com shift={shift}:\n", A_12_eval, sep="", end="\n\n")
print("Autovetor correspondente:\n", A_12_evec, sep="")
```

```
Autovalor achado com shift=-10.0:
-10.68022368126153

Autovetor correspondente:
[[-0.09]
 [-0.45]
 [ 0.26]
 [ 0.18]
 [ 0.68]
 [-0.36]
 [-0.29]
 [-0.03]]
```

shift = 100.0

In [36]:
```
shift = 100.0
A_12_eval, A_12_evec = mylinalg.eigen.inverse_power_iteration(A_12, shift=shift)

print(f"Autovalor achado com shift={shift}:\n", A_12_eval, sep="", end="\n\n")
print("Autovetor correspondente:\n", A_12_evec, sep="")
```

```
Autovalor achado com shift=100.0:
43.757985457954575

Autovetor correspondente:
[[0.38]
 [0.37]
 [0.36]
 [0.35]
 [0.37]
 [0.3 ]
 [0.31]
 [0.37]]
```

Método de Householder

Gera matrizes similares/semelhantes a partir de transformações de reflexão aplicadas em ambos os lados das matrizes.

Matrizes simétricas

Gera matrizes tridiagonais.

In [37]:
```
A_12_H, A_12_S = mylinalg.eigen.hessenberg_reduction(A_12)
```

```
>>> PASSO 1
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.2  -0.4  -0.54 -0.27 -0.4  -0.4 ]
 [ 0.   -0.2   0.97 -0.06 -0.08 -0.04 -0.06 -0.06]
 [ 0.   -0.4  -0.06  0.88 -0.16 -0.08 -0.12 -0.12]
 [ 0.   -0.54 -0.08 -0.16  0.78 -0.11 -0.16 -0.16]
 [ 0.   -0.27 -0.04 -0.08 -0.11  0.95 -0.08 -0.08]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08  0.88 -0.12]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08 -0.12  0.88]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8  -6.5  -0.2   6.7  -2.2   1.5   1.4]
 [ -0.   -6.5   3.7  -5.9  -4.1   3.8  -3.8   2.7]
 [ -0.   -0.2  -5.9   0.3  -0.7  -0.7   4.6  -0.3]
 [ -0.    6.7  -4.1  -0.7  -6.    2.6   4.4   0.1]
 [ -0.   -2.2   3.8  -0.7   2.6   1.1   1.9   6.3]
 [  0.    1.5  -3.8   4.6   4.4   1.9   5.   -1. ]
 [  0.    1.4   2.7  -0.3   0.1   6.3  -1.    2.1]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.2  -0.4  -0.54 -0.27 -0.4  -0.4 ]
 [ 0.   -0.2   0.97 -0.06 -0.08 -0.04 -0.06 -0.06]
 [ 0.   -0.4  -0.06  0.88 -0.16 -0.08 -0.12 -0.12]
 [ 0.   -0.54 -0.08 -0.16  0.78 -0.11 -0.16 -0.16]
 [ 0.   -0.27 -0.04 -0.08 -0.11  0.95 -0.08 -0.08]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08  0.88 -0.12]
 [ 0.   -0.4  -0.06 -0.12 -0.16 -0.08 -0.12  0.88]]

>>> PASSO 2
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.   -0.66 -0.02  0.68 -0.22  0.15  0.14]
 [ 0.    0.   -0.02  1.    0.01 -0.    0.    0.  ]
 [ 0.    0.    0.68  0.01  0.72  0.09 -0.06 -0.06]
 [ 0.    0.   -0.22 -0.    0.09  0.97  0.02  0.02]
 [ 0.    0.    0.15  0.   -0.06  0.02  0.99 -0.01]
 [ 0.    0.    0.14  0.   -0.06  0.02 -0.01  0.99]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8   9.8  -0.    0.    0.    0.    0. ]
 [ -0.    9.8   3.6   4.4  -5.3   1.5   4.7  -3.8]
 [ -0.    0.    4.4   0.5  -4.9   0.7   3.8  -1.2]
 [ -0.    0.   -5.3  -4.9  -5.    3.4   1.    2.9]
 [ -0.   -0.    1.5   0.7   3.4   0.5   3.2   5.6]
 [  0.   -0.    4.7   3.8   1.    3.2   3.4  -1.1]
 [  0.   -0.   -3.8  -1.2   2.9   5.6  -1.1   3.2]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.4  -0.5  -0.28 -0.39 -0.4 ]
 [ 0.   -0.2  -0.71 -0.08  0.61 -0.26  0.09  0.08]
 [ 0.   -0.4  -0.1   0.88 -0.14 -0.09 -0.12 -0.12]
 [ 0.   -0.54  0.57 -0.16  0.52 -0.02 -0.22 -0.22]
 [ 0.   -0.27 -0.28 -0.08 -0.01  0.91 -0.06 -0.06]
 [ 0.   -0.4   0.07 -0.12 -0.21 -0.06  0.87 -0.13]
 [ 0.   -0.4   0.05 -0.12 -0.21 -0.07 -0.13  0.87]]

>>> PASSO 3
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.   -0.47  0.57 -0.16 -0.51  0.41]
 [ 0.    0.    0.    0.57  0.78  0.06  0.2  -0.16]
 [ 0.    0.    0.   -0.16  0.06  0.98 -0.06  0.05]
 [ 0.    0.    0.   -0.51  0.2  -0.06  0.82  0.14]
 [ 0.    0.    0.    0.41 -0.16  0.05  0.14  0.89]]

Matriz similar:
[[  9.  -14.9  -0.   -0.   -0.   -0.    0.    0. ]
 [-14.9  34.8   9.8   0.   -0.    0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3   0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4  -4.6   3.4  -0.1   0.1]
 [ -0.    0.    0.   -4.6  -6.    2.6   3.3   1.8]
 [ -0.   -0.   -0.    3.4   2.6   1.    3.5   5.1]
 [  0.   -0.   -0.   -0.1   3.3   3.5   0.1   0.9]
 [  0.   -0.    0.    0.1   1.8   5.1   0.9   2. ]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01 -0.65 -0.24 -0.26 -0.5 ]
 [ 0.   -0.2  -0.71  0.41  0.42 -0.21  0.26 -0.06]
```

```
 [ 0.   -0.4  -0.1  -0.47  0.38 -0.24 -0.58  0.26]
 [ 0.   -0.54  0.57  0.4   0.3   0.04 -0.03 -0.37]
 [ 0.   -0.27 -0.28 -0.11  0.    0.91 -0.07 -0.05]
 [ 0.   -0.4   0.07 -0.55 -0.05 -0.11  0.72 -0.01]
 [ 0.   -0.4   0.05  0.37 -0.4  -0.01  0.04  0.73]]

>>> PASSO 4
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.   -0.81  0.59 -0.02  0.02]
 [ 0.    0.    0.    0.    0.59  0.81  0.01 -0.01]
 [ 0.    0.    0.    0.   -0.02  0.01  1.    0.  ]
 [ 0.    0.    0.    0.    0.02 -0.01  0.    1.  ]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.   -0.    0.    0. ]
 [-14.9  34.8   9.8   0.    0.    0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.    0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9   2.6  -0.7   1.7]
 [ -0.    0.    0.   -0.    2.6   1.    4.8   5.1]
 [  0.   -0.   -0.   -0.   -0.7   4.8   0.    1. ]
 [  0.   -0.    0.    0.    1.7   5.1   1.    2. ]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38 -0.57 -0.25 -0.52]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.08  0.25 -0.04]
 [ 0.   -0.4  -0.1  -0.47 -0.43  0.02 -0.59  0.27]
 [ 0.   -0.54  0.57  0.4  -0.23  0.21 -0.04 -0.36]
 [ 0.   -0.27 -0.28 -0.11  0.53  0.74 -0.06 -0.06]
 [ 0.   -0.4   0.07 -0.55 -0.04 -0.11  0.72 -0.01]
 [ 0.   -0.4   0.05  0.37  0.33 -0.25  0.05  0.72]]

>>> PASSO 5
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -0.81  0.21 -0.55]
 [ 0.    0.    0.    0.    0.    0.21  0.97  0.07]
 [ 0.    0.    0.    0.    0.   -0.55  0.07  0.83]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.    0.    0.    0. ]
 [-14.9  34.8   9.8   0.    0.   -0.    0.    0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.    0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9  -3.2   0.    0. ]
 [  0.    0.   -0.    0.   -3.2   4.   -5.2  -3. ]
 [  0.   -0.    0.   -0.   -0.   -5.2   2.3  -1. ]
 [  0.   -0.    0.    0.    0.   -3.   -1.   -3.2]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38  0.7  -0.4  -0.13]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.01  0.26 -0.06]
 [ 0.   -0.4  -0.1  -0.47 -0.43 -0.29 -0.56  0.17]
 [ 0.   -0.54  0.57  0.4  -0.23  0.02 -0.01 -0.42]
 [ 0.   -0.27 -0.28 -0.11  0.53 -0.58  0.09 -0.46]
 [ 0.   -0.4   0.07 -0.55 -0.04  0.25  0.67  0.1 ]
 [ 0.   -0.4   0.05  0.37  0.33 -0.19  0.04  0.74]]

>>> PASSO 6
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.   -0.86 -0.51]
 [ 0.    0.    0.    0.    0.    0.   -0.51  0.86]]

Matriz similar:
[[  9.  -14.9  -0.   -0.    0.    0.   -0.    0. ]
 [-14.9  34.8   9.8   0.    0.   -0.   -0.   -0. ]
 [ -0.    9.8   3.6  -9.3  -0.   -0.   -0.    0. ]
 [ -0.    0.   -9.3   5.4   5.7   0.   -0.   -0. ]
 [  0.   -0.   -0.    5.7  -5.9  -3.2  -0.    0. ]
 [  0.    0.   -0.    0.   -3.2   4.    6.   -0. ]
 [ -0.    0.   -0.   -0.   -0.    6.    0.    2.9]]
```

```
 [  0.   -0.   -0.    0.    0.    0.    2.9  -0.9]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.34 -0.29 -0.01  0.38  0.7   0.41  0.09]
 [ 0.   -0.2  -0.71  0.41 -0.47  0.01 -0.19 -0.19]
 [ 0.   -0.4  -0.1  -0.47 -0.43 -0.29  0.39  0.43]
 [ 0.   -0.54  0.57  0.4  -0.23  0.02  0.23 -0.36]
 [ 0.   -0.27 -0.28 -0.11  0.53 -0.58  0.15 -0.44]
 [ 0.   -0.4   0.07 -0.55 -0.04  0.25 -0.63 -0.26]
 [ 0.   -0.4   0.05  0.37  0.33 -0.19 -0.41  0.62]]
```

In [38]:
```
print("A_12 Matriz similar:\n", A_12_S.round(4), sep="")
```

```
A_12 Matriz similar:
[[  9.   -14.9   -0.    -0.     0.     0.    -0.     0.  ]
 [-14.9   34.81   9.84   0.     0.    -0.    -0.    -0.  ]
 [ -0.     9.84   3.64  -9.28  -0.    -0.    -0.     0.  ]
 [ -0.     0.    -9.28   5.37   5.74   0.    -0.    -0.  ]
 [  0.    -0.    -0.     5.74  -5.91  -3.17  -0.     0.  ]
 [  0.     0.    -0.     0.    -3.17   3.99   6.01  -0.  ]
 [ -0.     0.    -0.    -0.    -0.     6.01   0.04   2.93]
 [  0.    -0.    -0.     0.     0.     0.     2.93  -0.94]]
```

Matrizes não simétricas

Gera matrizes Upper-Hessenberg.

In [39]:
```
A_13 = np.random.randint(0, 10, (8, 8))

print("A_13:\n", A_13, sep="")
```

```
A_13:
[[0 4 0 9 8 9 7 9]
 [5 0 6 5 4 9 6 0]
 [2 2 1 5 3 1 2 1]
 [2 7 0 8 6 0 1 4]
 [8 0 8 9 3 4 9 3]
 [3 2 2 9 3 6 5 4]
 [8 1 8 2 0 1 0 1]
 [1 0 7 4 9 9 1 4]]
```

In [40]:
```
A_13_H, A_13_S = mylinalg.eigen.hessenberg_reduction(A_13)
```

```
>>> PASSO 1
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.15 -0.15 -0.61 -0.23 -0.61 -0.08]
 [ 0.   -0.15  0.98 -0.02 -0.07 -0.03 -0.07 -0.01]
 [ 0.   -0.15 -0.02  0.98 -0.07 -0.03 -0.07 -0.01]
 [ 0.   -0.61 -0.07 -0.07  0.73 -0.1  -0.27 -0.03]
 [ 0.   -0.23 -0.03 -0.03 -0.1   0.96 -0.1  -0.01]
 [ 0.   -0.61 -0.07 -0.07 -0.27 -0.1   0.73 -0.03]
 [ 0.   -0.08 -0.01 -0.01 -0.03 -0.01 -0.03  1.  ]]

Matriz similar:
[[  0.  -14.8  -2.1   6.9  -0.3   5.9  -1.3   8. ]
 [-13.1  16.8 -11.1 -10.9   2.4  -5.5  -1.   -3.4]
 [ -0.   -2.1  -1.6   2.6   0.2  -1.6  -1.4   0.3]
 [ -0.   -5.5  -3.5   4.7  -0.5  -4.   -6.1   2.8]
 [ -0.    0.7  -0.3   1.2  -0.7  -3.5   2.9   1.1]
 [ -0.   -4.6  -1.9   5.3  -1.4   2.   -0.3   2.9]
 [ -0.    9.6   0.6  -4.9  -0.2  -5.2  -2.6  -0.4]
 [ -0.   -8.7   5.    2.1   4.7   6.6  -3.6   3.3]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.15 -0.15 -0.61 -0.23 -0.61 -0.08]
 [ 0.   -0.15  0.98 -0.02 -0.07 -0.03 -0.07 -0.01]
 [ 0.   -0.15 -0.02  0.98 -0.07 -0.03 -0.07 -0.01]
 [ 0.   -0.61 -0.07 -0.07  0.73 -0.1  -0.27 -0.03]
 [ 0.   -0.23 -0.03 -0.03 -0.1   0.96 -0.1  -0.01]
 [ 0.   -0.61 -0.07 -0.07 -0.27 -0.1   0.73 -0.03]
 [ 0.   -0.08 -0.01 -0.01 -0.03 -0.01 -0.03  1.  ]]

>>> PASSO 2
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.   -0.14 -0.37  0.05 -0.31  0.64 -0.58]
 [ 0.    0.   -0.37  0.88  0.02 -0.1   0.21 -0.19]
 [ 0.    0.    0.05  0.02  1.    0.01 -0.03  0.02]
 [ 0.    0.   -0.31 -0.1   0.01  0.92  0.17 -0.16]
 [ 0.    0.    0.64  0.21 -0.03  0.17  0.64  0.33]
 [ 0.    0.   -0.58 -0.19  0.02 -0.16  0.33  0.7 ]]

Matriz similar:
[[  0.  -14.8  -9.6   4.5  -0.    3.9   2.9   4.2]
 [-13.1  16.8   8.7  -4.4   1.6  -0.2 -12.1   6.7]
 [ -0.   15.    9.3  -4.8  -2.7  -3.7  -2.3   0.8]
 [ -0.   -0.   -2.2   1.5  -1.4  -5.3  -5.2   1.9]
 [ -0.    0.    1.4   2.3  -0.6  -2.9   1.8   2.2]
 [ -0.   -0.   -1.4   2.5  -2.1   0.8   0.8   1.8]
 [ -0.   -0.   -4.   -0.2   1.3  -3.7  -2.9   0. ]
 [ -0.   -0.   -2.1  -5.8   3.7   2.2   3.   -2.9]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.23 -0.18 -0.61 -0.25 -0.57 -0.12]
 [ 0.   -0.15 -0.17 -0.39 -0.02 -0.33  0.58 -0.59]
 [ 0.   -0.15 -0.39  0.86 -0.05 -0.13  0.14 -0.2 ]
 [ 0.   -0.61 -0.05 -0.06  0.73 -0.1  -0.28 -0.03]
 [ 0.   -0.23 -0.34 -0.13 -0.09  0.88  0.08 -0.18]
 [ 0.   -0.61  0.54  0.13 -0.3   0.06  0.39  0.28]
 [ 0.   -0.08 -0.59 -0.2  -0.01 -0.17  0.29  0.7 ]]

>>> PASSO 3
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.   -0.4   0.26 -0.27 -0.74 -0.39]
 [ 0.    0.    0.    0.26  0.95  0.05  0.14  0.07]
 [ 0.    0.    0.   -0.27  0.05  0.95 -0.14 -0.07]
 [ 0.    0.    0.   -0.74  0.14 -0.14  0.61 -0.21]
 [ 0.    0.    0.   -0.39  0.07 -0.07 -0.21  0.89]]

Matriz similar:
[[  0.  -14.8  -9.6  -6.6   2.    1.8  -3.    1.1]
 [-13.1  16.8   8.7   8.6  -0.9   2.3  -5.2  10.3]
 [ -0.   15.    9.3   3.6  -4.2  -2.1   2.1   3.2]
 [  0.    0.    5.4  -4.5  -0.3   1.8   0.   -1.3]
 [ -0.   -0.   -0.   -1.   -0.   -5.   -1.5   1.6]
 [ -0.   -0.    0.   -4.6  -0.8   1.   -1.3  -0.5]
 [ -0.   -0.    0.   -0.7   1.4   0.6   1.3  -0.9]
 [  0.    0.   -0.   -0.6   2.7   5.4   8.1  -1.8]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.23  0.44 -0.72 -0.13 -0.24  0.06]
 [ 0.   -0.15 -0.17  0.05 -0.1  -0.25  0.81 -0.47]
```

```
 [ 0.   -0.15 -0.39 -0.35  0.18 -0.36 -0.5  -0.54]
 [ 0.   -0.61 -0.05  0.46  0.63  0.   -0.    0.12]
 [ 0.   -0.23 -0.34 -0.19 -0.08  0.86  0.04 -0.19]
 [ 0.   -0.61  0.54 -0.54 -0.17 -0.07  0.04  0.09]
 [ 0.   -0.08 -0.59 -0.37  0.02 -0.2   0.21  0.65]]

>>> PASSO 4
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.   -0.2  -0.96 -0.14 -0.12]
 [ 0.    0.    0.    0.   -0.96  0.23 -0.11 -0.1 ]
 [ 0.    0.    0.    0.   -0.14 -0.11  0.98 -0.01]
 [ 0.    0.    0.    0.   -0.12 -0.1  -0.01  0.99]]

Matriz similar:
[[  0.  -14.8  -9.6  -6.6  -1.8  -1.3  -3.4   0.7]
 [-13.1  16.8   8.7   8.6  -2.6   0.9  -5.4  10.2]
 [ -0.   15.    9.3   3.6   2.2   3.    2.9   3.8]
 [  0.    0.    5.4  -4.5  -1.5   0.9  -0.1  -1.4]
 [  0.    0.   -0.    4.8   0.5  -0.5   0.5   0.5]
 [  0.    0.    0.    0.   -4.    1.7  -0.2  -1.7]
 [ -0.   -0.    0.    0.   -1.4  -1.2   1.2  -1.3]
 [  0.    0.   -0.    0.   -7.1  -2.    7.1  -2.9]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.23  0.44  0.3   0.69 -0.12  0.16]
 [ 0.   -0.15 -0.17  0.05  0.21 -0.01  0.84 -0.44]
 [ 0.   -0.15 -0.39 -0.35  0.44 -0.14 -0.46 -0.51]
 [ 0.   -0.61 -0.05  0.46 -0.14 -0.62 -0.09  0.04]
 [ 0.   -0.23 -0.34 -0.19 -0.8   0.29 -0.04 -0.27]
 [ 0.   -0.61  0.54 -0.54  0.08  0.14  0.06  0.12]
 [ 0.   -0.08 -0.59 -0.37  0.08 -0.15  0.21  0.66]]

>>> PASSO 5
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -0.49 -0.17 -0.86]
 [ 0.    0.    0.    0.    0.   -0.17  0.98 -0.1 ]
 [ 0.    0.    0.    0.    0.   -0.86 -0.1   0.51]]

Matriz similar:
[[  0.  -14.8  -9.6  -6.6  -1.8   0.6  -3.2   1.8]
 [-13.1  16.8   8.7   8.6  -2.6  -8.2  -6.5   4.9]
 [ -0.   15.    9.3   3.6   2.2  -5.3   1.9  -1. ]
 [  0.    0.    5.4  -4.5  -1.5   0.8  -0.1  -1.5]
 [  0.    0.   -0.    4.8   0.5  -0.3   0.5   0.6]
 [ -0.   -0.    0.   -0.    8.3  -2.6  -6.6   1.5]
 [ -0.   -0.    0.   -0.   -0.    1.1   0.8   0.7]
 [ -0.   -0.   -0.   -0.    0.    0.4   4.    1.7]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.23  0.44  0.3  -0.45 -0.25 -0.5 ]
 [ 0.   -0.15 -0.17  0.05  0.21  0.24  0.87 -0.3 ]
 [ 0.   -0.15 -0.39 -0.35  0.44  0.59 -0.38 -0.09]
 [ 0.   -0.61 -0.05  0.46 -0.14  0.28  0.01  0.56]
 [ 0.   -0.23 -0.34 -0.19 -0.8   0.1  -0.06 -0.38]
 [ 0.   -0.61  0.54 -0.54  0.08 -0.18  0.03 -0.07]
 [ 0.   -0.08 -0.59 -0.37  0.08 -0.52  0.17  0.44]]

>>> PASSO 6
Transformação de Householder:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.   -0.94 -0.35]
 [ 0.    0.    0.    0.    0.    0.   -0.35  0.94]]

Matriz similar:
[[  0.  -14.8  -9.6  -6.6  -1.8   0.6   2.3   2.8]
 [-13.1  16.8   8.7   8.6  -2.6  -8.2   4.3   6.9]
 [ -0.   15.    9.3   3.6   2.2  -5.3  -1.5  -1.6]
 [  0.    0.    5.4  -4.5  -1.5   0.8   0.6  -1.3]
 [  0.    0.   -0.    4.8   0.5  -0.3  -0.7   0.4]
 [ -0.   -0.    0.   -0.    8.3  -2.6   5.7   3.7]
 [  0.    0.   -0.    0.    0.   -1.2   2.4  -0.4]]
```

```
 [ -0.   -0.   -0.   -0.    0.   -0.   -3.7   0.1]]

Householder acumulada:
[[ 1.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.38 -0.23  0.44  0.3  -0.45  0.41 -0.37]
 [ 0.   -0.15 -0.17  0.05  0.21  0.24 -0.71 -0.59]
 [ 0.   -0.15 -0.39 -0.35  0.44  0.59  0.39  0.05]
 [ 0.   -0.61 -0.05  0.46 -0.14  0.28 -0.21  0.52]
 [ 0.   -0.23 -0.34 -0.19 -0.8   0.1   0.19 -0.33]
 [ 0.   -0.61  0.54 -0.54  0.08 -0.18 -0.   -0.07]
 [ 0.   -0.08 -0.59 -0.37  0.08 -0.52 -0.32  0.36]]
```

In [41]:
```
print("A_13 Matriz similar:\n", A_13_S.round(4), sep="")
```

```
A_13 Matriz similar:
[[  0.   -14.84  -9.56  -6.58  -1.82   0.65   2.34   2.84]
 [-13.08  16.83   8.72   8.56  -2.56  -8.24   4.35   6.86]
 [ -0.    14.98   9.25   3.59   2.15  -5.26  -1.47  -1.57]
 [  0.     0.     5.43  -4.52  -1.53   0.82   0.64  -1.32]
 [  0.     0.    -0.     4.83   0.51  -0.29  -0.66   0.39]
 [ -0.    -0.     0.    -0.     8.31  -2.55   5.7    3.73]
 [  0.     0.    -0.     0.     0.    -1.18   2.4   -0.4 ]
 [ -0.    -0.    -0.    -0.     0.    -0.    -3.71   0.08]]
```

Decomposição QR

Decompõem uma matriz A em uma matriz quadrada ortogonal, Q, e uma matriz triangular superior (ou trapezoidal, caso A tenha mais colunas do que linhas), R. É uma transformação muito importante para determinação de autovalores e autovetores e também para a decomposição SVD.

In [42]:
```
A_12_Q, A_12_R = mylinalg.decompositions.qr(A_12)

print(f"A_12_Q {A_12_Q.shape}:\n", A_12_Q.round(3), sep="", end="\n\n")
print(f"A_12_R {A_12_R.shape}:\n", A_12_R.round(3), sep="")
```

```
A_12_Q (8, 8):
[[-0.52  0.26  0.23  0.45  0.54 -0.15 -0.29 -0.11]
 [-0.29 -0.45 -0.24 -0.5   0.17 -0.55 -0.27 -0.01]
 [-0.17 -0.61 -0.3   0.24  0.22  0.51  0.16 -0.34]
 [-0.34 -0.17  0.41 -0.26  0.2   0.2   0.44  0.59]
 [-0.46 -0.2   0.29  0.25 -0.7  -0.22  0.14 -0.21]
 [-0.23  0.33 -0.59  0.11  0.04 -0.28  0.63  0.03]
 [-0.34  0.4   0.05 -0.59 -0.04  0.37 -0.   -0.48]
 [-0.34  0.11 -0.44  0.07 -0.31  0.34 -0.46  0.5 ]]

A_12_R (8, 8):
[[-17.41 -14.99 -13.5  -15.97 -15.34 -12.41 -14.53 -14.65]
 [  0.   -10.45  -5.32   0.52   0.29  -2.     1.05  -3.19]
 [ -0.     0.   -11.24   0.43  -4.49  -4.07   3.62  -6.69]
 [ -0.     0.     0.    -5.52  -5.22   2.97  -1.23   2.97]
 [ -0.     0.    -0.     0.     6.27  -2.48  -0.71   1.87]
 [  0.     0.     0.     0.    -0.     6.44   1.76   1.61]
 [  0.     0.     0.     0.    -0.    -0.     4.95   4.39]
 [  0.    -0.     0.     0.     0.    -0.    -0.     0.43]]
```

In [43]:
```
print("I = Q.Q^T:\n", A_12_Q.dot(A_12_Q.T).round(5), sep="")
```

```
I = Q.Q^T:
[[ 1.  0.  0. -0.  0. -0. -0. -0.]
 [ 0.  1. -0. -0. -0.  0.  0.  0.]
 [ 0. -0.  1.  0. -0.  0.  0.  0.]
 [-0. -0.  0.  1. -0.  0.  0.  0.]
 [ 0. -0. -0. -0.  1.  0. -0.  0.]
 [-0.  0.  0.  0.  0.  1. -0. -0.]
 [-0.  0.  0.  0. -0. -0.  1.  0.]
 [-0.  0.  0.  0.  0. -0.  0.  1.]]
```

In [44]:
```
print("A_12:\n", A_12, sep="", end="\n\n")
print("A_12 = QR:\n", A_12_Q.dot(A_12_R).round(5), sep="")
```

```
A_12:
[[9 5 3 6 8 4 6 6]
 [5 9 9 7 9 0 1 4]
 [3 9 9 1 4 8 2 9]
 [6 7 1 7 6 3 9 5]
 [8 9 4 6 0 6 8 5]
 [4 0 8 3 6 3 4 9]
 [6 1 2 9 8 4 7 2]
 [6 4 9 5 5 9 2 6]]

A_12 = QR:
[[ 9.  5.  3.  6.  8.  4.  6.  6.]
 [ 5.  9.  9.  7.  9. -0.  1.  4.]
 [ 3.  9.  9.  1.  4.  8.  2.  9.]
 [ 6.  7.  1.  7.  6.  3.  9.  5.]
 [ 8.  9.  4.  6. -0.  6.  8.  5.]
 [ 4.  0.  8.  3.  6.  3.  4.  9.]
 [ 6.  1.  2.  9.  8.  4.  7.  2.]
 [ 6.  4.  9.  5.  5.  9.  2.  6.]]
```

Decomposição SVD

Decomposição existente para qualquer que seja a matriz $A^{m, n}$. Condensa muitas informações importantes em seus fatores, como o rank, bases para espaços do domínio e codomínio, entre outros. Fatora a transformação de A em uma rotação com ou sem reflexão, um escalonamento e outra rotação com ou sem reflexão.

$ A = U \Sigma V^T$, onde U e V^T são matrizes ortogonais e Σ é uma matriz diagonal.

In [45]:
```
A_14 = np.random.randint(0, 20, (6, 9))

print("A_14:\n", A_14, sep="")
```

```
A_14:
[[12 10 16  6  8 11 17  9  4]
 [ 0  0 10 19 19 12 17  0 18]
 [ 6  2 11 12  2  0 10 10  7]
 [ 1  3 19 13 13 18  1 17 14]
 [17 10  6 18  6  8  2 10  8]
 [15  7 15  0 19  0 14 11  4]]
```

In [46]:
```
A_14_U, A_14_S, A_14_VT = mylinalg.decompositions.svd(A_14)

print(f"A_14_S {A_14_S.shape}:\n", A_14_S, sep="")
```

```
A_14_S (6, 6):
[[72.81  0.    0.    0.    0.    0.  ]
 [ 0.   26.07  0.    0.    0.    0.  ]
 [ 0.    0.   21.21  0.    0.    0.  ]
 [ 0.    0.    0.   17.38  0.    0.  ]
 [ 0.    0.    0.    0.   11.53  0.  ]
 [ 0.    0.    0.    0.    0.   10.32]]
```

In [47]:
```
print(f"A_14_U {A_14_U.shape}:\n", A_14_U, sep="", end="\n\n")
print("A_14 UU^T:\n", A_14_U.dot(A_14_U.T).round(5), sep="")
```

```
A_14_U (6, 6):
[[-0.42 -0.37  0.07 -0.03 -0.52 -0.64]
 [-0.47  0.57  0.59  0.31  0.11 -0.05]
 [-0.29 -0.07 -0.09  0.21 -0.61  0.7 ]
 [-0.48  0.39 -0.4  -0.67  0.06  0.04]
 [-0.37 -0.09 -0.6   0.59  0.36 -0.1 ]
 [-0.4  -0.62  0.34 -0.23  0.46  0.29]]

A_14 UU^T:
[[ 1. -0. -0.  0.  0.  0.]
 [-0.  1. -0.  0. -0.  0.]
 [-0. -0.  1.  0. -0. -0.]
 [ 0.  0.  0.  1. -0. -0.]
 [ 0. -0. -0. -0.  1. -0.]
 [ 0.  0. -0. -0. -0.  1.]]
```

In [48]:
```
print(f"A_14_VT {A_14_VT.shape}:\n", A_14_VT.round(4), sep="", end="\n\n")
print("A_14 V^T.V:\n", A_14_VT.dot(A_14_VT.T).round(5), sep="")
```

```
A_14_VT (6, 9):
[[-0.27 -0.17 -0.44 -0.38 -0.4  -0.3  -0.34 -0.31 -0.32]
 [-0.58 -0.3  -0.13  0.43  0.02  0.35 -0.22 -0.2   0.4 ]
 [-0.25 -0.2  -0.01 -0.26  0.43 -0.2   0.63 -0.44  0.05]
 [ 0.4   0.14 -0.45  0.58 -0.2  -0.23  0.24 -0.36  0.08]
 [ 0.27  0.05 -0.33 -0.09  0.72 -0.04 -0.51 -0.1   0.1 ]
 [-0.08 -0.38  0.13  0.21  0.06 -0.76 -0.09  0.39  0.21]]

A_14 V^T.V:
[[ 1.  0. -0. -0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.]
 [-0.  0.  1. -0.  0. -0.]
 [-0.  0. -0.  1. -0.  0.]
 [ 0.  0.  0. -0.  1.  0.]
 [ 0.  0. -0.  0.  0.  1.]]
```

In [49]:
```
print("A_14:\n", A_14, sep="", end="\n\n")
print("A_14 = U.S.V^T:\n", A_14_U.dot(A_14_S).dot(A_14_VT).round(5), sep="")
```

```
A_14:
[[12 10 16  6  8 11 17  9  4]
 [ 0  0 10 19 19 12 17  0 18]
 [ 6  2 11 12  2  0 10 10  7]
 [ 1  3 19 13 13 18  1 17 14]
 [17 10  6 18  6  8  2 10  8]
 [15  7 15  0 19  0 14 11  4]]

A_14 = U.S.V^T:
[[12. 10. 16.  6.  8. 11. 17.  9.  4.]
 [-0. -0. 10. 19. 19. 12. 17. -0. 18.]
 [ 6.  2. 11. 12.  2. -0. 10. 10.  7.]
 [ 1.  3. 19. 13. 13. 18.  1. 17. 14.]
 [17. 10.  6. 18.  6.  8.  2. 10.  8.]
 [15.  7. 15. -0. 19. -0. 14. 11.  4.]]
```

In [ ]: