Descomposición LU

Mateo Cumbal

2025-02-05

Tabla de Contenidos


```
import numpy as np
```

```
%load_ext autoreload

%autoreload 2
from src import descomposicion_LU, resolver_LU, matriz_aumentada
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

1 CONJUNTO DE EJERCICIOS

1.1 Ejercicio 1

Realice las siguientes multiplicaciones matriz-matriz:

```python
def multiplicacion_matrices(A: np.ndarray, B: np.ndarray) -> np.ndarray:
    """
    Realiza la multiplicación de dos matrices y muestra el paso a paso.

    Parámetros:
    A: np.ndarray -> Matriz de tamaño (m, n)
    B: np.ndarray -> Matriz de tamaño (n, p)

    Retorna:
    np.ndarray -> Matriz resultado de tamaño (m, p)
    """
    assert (
        A.shape[1] == B.shape[0]
    ), "El número de columnas de A debe coincidir con el número de filas de B."

    m, n = A.shape
    n, p = B.shape
    C = np.zeros((m, p))

    print("Multiplicación de matrices - Paso a Paso:")
    for i in range(m):
        for j in range(p):
            suma = 0
            for k in range(n):
                producto = A[i, k] * B[k, j]
                suma += producto
                print(
                    f"C[{i},{j}] += A[{i},{k}] * B[{k},{j}] -> {A[i, k]} * {B[k, j]} = {produ
                )

            C[i, j] = suma
            print(f"C[{i},{j}] = {suma}\n")

    return C
```

a.

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

```python
A = np.array([[2, -3], [3, -1]])
B = np.array([[1, 5], [2, 0]])
```

```python
C = multiplicacion_matrices(A, B)
print(f'La matriz resultante C es:\n{C}')
```

```
Multiplicación de matrices - Paso a Paso:
C[0,0] += A[0,0] * B[0,0] -> 2 * 1 = 2
C[0,0] += A[0,1] * B[1,0] -> -3 * 2 = -6
C[0,0] = -4

C[0,1] += A[0,0] * B[0,1] -> 2 * 5 = 10
C[0,1] += A[0,1] * B[1,1] -> -3 * 0 = 0
C[0,1] = 10

C[1,0] += A[1,0] * B[0,0] -> 3 * 1 = 3
C[1,0] += A[1,1] * B[1,0] -> -1 * 2 = -2
C[1,0] = 1

C[1,1] += A[1,0] * B[0,1] -> 3 * 5 = 15
C[1,1] += A[1,1] * B[1,1] -> -1 * 0 = 0
C[1,1] = 15

La matriz resultante C es:
[[-4. 10.]
 [ 1. 15.]]
```

b.

$$\begin{bmatrix} 2 & -3 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & 5 & -4 \\ -3 & 2 & 0 \end{bmatrix}$$

```python
A = np.array([[2, -3], [3, -1]])
B = np.array([[1, 5, -4], [-3, 2, 0]])

C = multiplicacion_matrices(A, B)
print(f"La matriz resultante C es:\n{C}")
```

```
Multiplicación de matrices - Paso a Paso:
C[0,0] += A[0,0] * B[0,0] -> 2 * 1 = 2
C[0,0] += A[0,1] * B[1,0] -> -3 * -3 = 9
C[0,0] = 11

C[0,1] += A[0,0] * B[0,1] -> 2 * 5 = 10
C[0,1] += A[0,1] * B[1,1] -> -3 * 2 = -6
```

```
C[0,1] = 4

C[0,2] += A[0,0] * B[0,2] -> 2 * -4 = -8
C[0,2] += A[0,1] * B[1,2] -> -3 * 0 = 0
C[0,2] = -8

C[1,0] += A[1,0] * B[0,0] -> 3 * 1 = 3
C[1,0] += A[1,1] * B[1,0] -> -1 * -3 = 3
C[1,0] = 6

C[1,1] += A[1,0] * B[0,1] -> 3 * 5 = 15
C[1,1] += A[1,1] * B[1,1] -> -1 * 2 = -2
C[1,1] = 13

C[1,2] += A[1,0] * B[0,2] -> 3 * -4 = -12
C[1,2] += A[1,1] * B[1,2] -> -1 * 0 = 0
C[1,2] = -12

La matriz resultante C es:
[[ 11.   4.  -8.]
 [  6.  13. -12.]]
```

c.

$$\begin{bmatrix} 2 & -3 & 1 \\ 4 & 3 & 0 \\ 5 & 2 & -4 \end{bmatrix} \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & -1 \\ 2 & 3 & -2 \end{bmatrix}$$

```python
A = np.array([[2, -3, 1], [4, 3, 0], [5, 2, -4]])
B = np.array([[0, 1, -2], [1, 0, -1], [2, 3, -2]])

C = multiplicacion_matrices(A, B)
print(f"La matriz resultante C es:\n{C}")
```

```
Multiplicación de matrices - Paso a Paso:
C[0,0] += A[0,0] * B[0,0] -> 2 * 0 = 0
C[0,0] += A[0,1] * B[1,0] -> -3 * 1 = -3
C[0,0] += A[0,2] * B[2,0] -> 1 * 2 = 2
C[0,0] = -1

C[0,1] += A[0,0] * B[0,1] -> 2 * 1 = 2
C[0,1] += A[0,1] * B[1,1] -> -3 * 0 = 0
C[0,1] += A[0,2] * B[2,1] -> 1 * 3 = 3
```

```
C[0,1] = 5

C[0,2] += A[0,0] * B[0,2] -> 2 * -2 = -4
C[0,2] += A[0,1] * B[1,2] -> -3 * -1 = 3
C[0,2] += A[0,2] * B[2,2] -> 1 * -2 = -2
C[0,2] = -3

C[1,0] += A[1,0] * B[0,0] -> 4 * 0 = 0
C[1,0] += A[1,1] * B[1,0] -> 3 * 1 = 3
C[1,0] += A[1,2] * B[2,0] -> 0 * 2 = 0
C[1,0] = 3

C[1,1] += A[1,0] * B[0,1] -> 4 * 1 = 4
C[1,1] += A[1,1] * B[1,1] -> 3 * 0 = 0
C[1,1] += A[1,2] * B[2,1] -> 0 * 3 = 0
C[1,1] = 4

C[1,2] += A[1,0] * B[0,2] -> 4 * -2 = -8
C[1,2] += A[1,1] * B[1,2] -> 3 * -1 = -3
C[1,2] += A[1,2] * B[2,2] -> 0 * -2 = 0
C[1,2] = -11

C[2,0] += A[2,0] * B[0,0] -> 5 * 0 = 0
C[2,0] += A[2,1] * B[1,0] -> 2 * 1 = 2
C[2,0] += A[2,2] * B[2,0] -> -4 * 2 = -8
C[2,0] = -6

C[2,1] += A[2,0] * B[0,1] -> 5 * 1 = 5
C[2,1] += A[2,1] * B[1,1] -> 2 * 0 = 0
C[2,1] += A[2,2] * B[2,1] -> -4 * 3 = -12
C[2,1] = -7

C[2,2] += A[2,0] * B[0,2] -> 5 * -2 = -10
C[2,2] += A[2,1] * B[1,2] -> 2 * -1 = -2
C[2,2] += A[2,2] * B[2,2] -> -4 * -2 = 8
C[2,2] = -4

La matriz resultante C es:
[[ -1.   5.  -3.]
 [  3.   4. -11.]
 [ -6.  -7.  -4.]]
```

d.

$$\begin{bmatrix} 2 & 1 & 2 \\ -2 & 3 & 0 \\ 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ -4 & 1 \\ 0 & 2 \end{bmatrix}$$

```python
A = np.array([[2, 1, 2], [-2, 3, 0], [2, -1, 3]])
B = np.array([[1, -2], [-4, 1], [0, 2]])

C = multiplicacion_matrices(A, B)
print(f"La matriz resultante C es:\n{C}")
```

```
Multiplicación de matrices - Paso a Paso:
C[0,0] += A[0,0] * B[0,0] -> 2 * 1 = 2
C[0,0] += A[0,1] * B[1,0] -> 1 * -4 = -4
C[0,0] += A[0,2] * B[2,0] -> 2 * 0 = 0
C[0,0] = -2

C[0,1] += A[0,0] * B[0,1] -> 2 * -2 = -4
C[0,1] += A[0,1] * B[1,1] -> 1 * 1 = 1
C[0,1] += A[0,2] * B[2,1] -> 2 * 2 = 4
C[0,1] = 1

C[1,0] += A[1,0] * B[0,0] -> -2 * 1 = -2
C[1,0] += A[1,1] * B[1,0] -> 3 * -4 = -12
C[1,0] += A[1,2] * B[2,0] -> 0 * 0 = 0
C[1,0] = -14

C[1,1] += A[1,0] * B[0,1] -> -2 * -2 = 4
C[1,1] += A[1,1] * B[1,1] -> 3 * 1 = 3
C[1,1] += A[1,2] * B[2,1] -> 0 * 2 = 0
C[1,1] = 7

C[2,0] += A[2,0] * B[0,0] -> 2 * 1 = 2
C[2,0] += A[2,1] * B[1,0] -> -1 * -4 = 4
C[2,0] += A[2,2] * B[2,0] -> 3 * 0 = 0
C[2,0] = 6

C[2,1] += A[2,0] * B[0,1] -> 2 * -2 = -4
C[2,1] += A[2,1] * B[1,1] -> -1 * 1 = -1
C[2,1] += A[2,2] * B[2,1] -> 3 * 2 = 6
C[2,1] = 1
```

```
La matriz resultante C es:
[[ -2.   1.]
 [-14.   7.]
 [  6.   1.]]
```

1.2 Ejercicio 2

Determine cuáles de las siguientes matrices son no singulares y calcule la inversa de esas matrices:

```python
def gauss_jordan_inversa(A: np.ndarray) -> np.ndarray:
    """
    Calcula la inversa de una matriz usando el método de Gauss-Jordan.

    :param A: Matriz cuadrada de tamaño n x n
    :return: Matriz inversa de A
    """
    n = A.shape[0]
    assert A.shape[0] == A.shape[1], "La matriz debe ser cuadrada."

    # Construimos la matriz aumentada [A | I]
    A = A.astype(float)
    I = np.eye(n)
    Augmented = np.hstack((A, I))

    # Aplicamos el método de Gauss-Jordan
    for i in range(n):
        # Pivoteo parcial
        if Augmented[i, i] == 0:
            for j in range(i + 1, n):
                if Augmented[j, i] != 0:
                    Augmented[[i, j]] = Augmented[[j, i]]  # Intercambio de filas
                    break
            else:
                raise ValueError("La matriz no es invertible.")

        # Hacer el pivote igual a 1
        pivot = Augmented[i, i]
        Augmented[i] = Augmented[i] / pivot

        # Hacer ceros en la columna i
        for j in range(n):
```

```
            if i != j:
                factor = Augmented[j, i]
                Augmented[j] -= factor * Augmented[i]

    # La parte derecha de la matriz aumentada ahora es la inversa de A
    return Augmented, Augmented[:, n:]
```

a.

$$\begin{bmatrix} 4 & 2 & 6 \\ 3 & 0 & 7 \\ -2 & -1 & -3 \end{bmatrix}$$

```
A = np.array(
    [[4, 2, 6],
    [3, 0, 7],
    [-2, -1, -3]]
)

try:
    aumentada, inversa = gauss_jordan_inversa(A)
    print(f"La matriz aumentada es:\n{aumentada}")
    print(f"\nPor lo tanto, la matriz inversa es:\n{inversa}")

except ValueError as e:
    print("ERROR:", e)
```

```
ERROR: La matriz no es invertible.
```

b.

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

```
B = np.array(
    [[1, 2, 0],
    [2, 1, -1],
    [3, 1 , 1]]
)

try:
    aumentada, inversa = gauss_jordan_inversa(B)
```

```
    print(f"La matriz aumentada es:\n{aumentada}")
    print(f"\nPor lo tanto, la matriz inversa es:\n{inversa}")

except ValueError as e:
    print("ERROR:", e)
```

```
La matriz aumentada es:
[[ 1.     0.     0.    -0.25   0.25   0.25 ]
 [-0.     1.     0.     0.625 -0.125 -0.125]
 [ 0.     0.     1.     0.125 -0.625  0.375]]

Por lo tanto, la matriz inversa es:
[[-0.25   0.25   0.25 ]
 [ 0.625 -0.125 -0.125]
 [ 0.125 -0.625  0.375]]
```

c.

$$\begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 2 & -4 & -2 \\ 2 & 1 & 1 & 5 \\ -1 & 0 & -2 & -4 \end{bmatrix}$$

```
C = np.array(
    [[1, 1, -1, 1],
     [1, 2, -4, -2],
     [2, 1, 1, 5],
     [-1, 0, -2, -4]]
)



try:
    aumentada, inversa = gauss_jordan_inversa(C)
    print(f"La matriz aumentada es:\n{aumentada}")
    print(f"\nPor lo tanto, la matriz inversa es:\n{inversa}")

except ValueError as e:
    print("ERROR:", e)
```

```
ERROR: La matriz no es invertible.
```

d.

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ 6 & 7 & 0 & 0 \\ 9 & 11 & 1 & 0 \\ 5 & 4 & 1 & 1 \end{bmatrix}$$

```python
D = np.array(
    [[4, 0, 0, 0],
    [6, 7, 0, 0],
    [9, 11, 1, 0],
    [5, 4, 1, 1]]
)

try:
    aumentada, inversa = gauss_jordan_inversa(D)
    print(f"La matriz aumentada es:\n{aumentada}")
    print(f"\nPor lo tanto, la matriz inversa es:\n{inversa}")

except ValueError as e:
    print("ERROR:", e)
```

```
La matriz aumentada es:
[[ 1.          0.          0.          0.          0.25        0.
   0.          0.        ]
 [ 0.          1.          0.          0.         -0.21428571  0.14285714
   0.          0.        ]
 [ 0.          0.          1.          0.          0.10714286 -1.57142857
   1.          0.        ]
 [ 0.          0.          0.          1.         -0.5         1.
  -1.          1.        ]]

Por lo tanto, la matriz inversa es:
[[ 0.25        0.          0.          0.        ]
 [-0.21428571  0.14285714  0.          0.        ]
 [ 0.10714286 -1.57142857  1.          0.        ]
 [-0.5         1.         -1.          1.        ]]
```

1.3 Ejercicio 3

Resuelva los sistemas lineales 4×4 que tienen la misma matriz de coeficientes:

1.

$$\begin{aligned} x_1 - x_2 + 2x_3 - x_4 &= 6, \\ x_1 - x_3 + x_4 &= 4, \\ 2x_1 + x_2 + 3x_3 - 4x_4 &= -2, \\ -x_2 + x_3 + x_4 &= 5. \end{aligned}$$

2.

$$\begin{aligned} x_1 - x_2 + 2x_3 - x_4 &= 1, \\ x_1 - x_3 + x_4 &= 1, \\ 2x_1 + x_2 + 3x_3 - 4x_4 &= 2, \\ -x_2 + x_3 + x_4 &= -1. \end{aligned}$$

La matriz de coeficiente y las matrices L y U, después de descomponer A, son:

```python
A = np.array(
    [[1, -1, 2, -1],
     [1, 0, -1, 1],
     [2, 1, 3, -4],
     [0, -1, 1, -1]]
)

L, U = descomposicion_LU(A)

print('\nDescomposición de la matriz A')
print(f'- Matriz L\n{L}')
print(f'- Matriz U\n{U}')
```

```
[02-03 18:33:18][INFO]
[[ 1. -1.  2. -1.]
 [ 0.  1. -3.  2.]
 [ 0.  3. -1. -2.]
 [ 0. -1.  1. -1.]]
[02-03 18:33:18][INFO]
[[ 1. -1.  2. -1.]
 [ 0.  1. -3.  2.]
 [ 0.  0.  8. -8.]
 [ 0.  0. -2.  1.]]
[02-03 18:33:18][INFO]
[[ 1. -1.  2. -1.]
 [ 0.  1. -3.  2.]
 [ 0.  0.  8. -8.]
```

```
 [ 0.  0.  0. -1.]]
[02-03 18:33:18][INFO]
[[ 1. -1.  2. -1.]
 [ 0.  1. -3.  2.]
 [ 0.  0.  8. -8.]
 [ 0.  0.  0. -1.]]

Descomposición de la matriz A
- Matriz L
[[ 1.    0.    0.    0.  ]
 [ 1.    1.    0.    0.  ]
 [ 2.    3.    1.    0.  ]
 [ 0.   -1.   -0.25  1.  ]]
- Matriz U
[[ 1. -1.  2. -1.]
 [ 0.  1. -3.  2.]
 [ 0.  0.  8. -8.]
 [ 0.  0.  0. -1.]]
```

Ahora podemos resolver para cualquier vector **b** del sistema.

1. Primer Sistema

```python
b1 = np.array([6, 4, -2, 5])

x1 = resolver_LU(L, U, b1)
print(f'\nLa solución del sistema para b1 es:\n{x1}')
```

```
[02-03 18:33:26][INFO] Sustitución hacia adelante
[02-03 18:33:26][INFO] y =
[[ 6.]
 [-2.]
 [-8.]
 [ 1.]]
[02-03 18:33:26][INFO] Sustitución hacia atrás
[02-03 18:33:26][INFO] i = 2
[02-03 18:33:26][INFO] suma = [8.]
[02-03 18:33:26][INFO] U[i, i] = 8.0
[02-03 18:33:26][INFO] y[i] = [-8.]
[02-03 18:33:26][INFO] i = 1
[02-03 18:33:26][INFO] suma = [4.]
[02-03 18:33:26][INFO] U[i, i] = 1.0
```

```
[02-03 18:33:26][INFO] y[i] = [-2.]
[02-03 18:33:26][INFO] i = 0
[02-03 18:33:26][INFO] suma = [3.]
[02-03 18:33:26][INFO] U[i, i] = 1.0
[02-03 18:33:26][INFO] y[i] = [6.]

La solución del sistema para b1 es:
[[ 3.]
 [-6.]
 [-2.]
 [-1.]]
```

2. Segundo Sistema

```python
b2 = np.array([1, 1, 2, -1])

x2 = resolver_LU(L, U, b2)
print(f"\nLa solución del sistema para b2 es:\n{x2}")
```

```
[02-03 18:33:29][INFO] Sustitución hacia adelante
[02-03 18:33:29][INFO] y =
[[ 1.]
 [ 0.]
 [ 0.]
 [-1.]]
[02-03 18:33:29][INFO] Sustitución hacia atrás
[02-03 18:33:29][INFO] i = 2
[02-03 18:33:29][INFO] suma = [-8.]
[02-03 18:33:29][INFO] U[i, i] = 8.0
[02-03 18:33:29][INFO] y[i] = [0.]
[02-03 18:33:29][INFO] i = 1
[02-03 18:33:29][INFO] suma = [-1.]
[02-03 18:33:29][INFO] U[i, i] = 1.0
[02-03 18:33:29][INFO] y[i] = [0.]
[02-03 18:33:29][INFO] i = 0
[02-03 18:33:29][INFO] suma = [0.]
[02-03 18:33:29][INFO] U[i, i] = 1.0
[02-03 18:33:29][INFO] y[i] = [1.]

La solución del sistema para b2 es:
[[1.]
 [1.]
```

```
 [1.]
 [1.]]
```

1.4 Ejercicio 4

Encuentre los valores de A que hacen que la siguiente matriz sea singular.

$$A = \begin{bmatrix} 1 & -1 & \alpha \\ 2 & 2 & 1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix}$$

Se tiene la matriz original:

$$A^{(0)} = \begin{bmatrix} 1 & -1 & \alpha \\ 2 & 2 & 1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix}$$

Se aplica la transformación en la segunda fila para hacer ceros en la primera columna:

$$-2 \cdot F1 + F2 \longrightarrow F2$$

Después de la transformación, la matriz queda como:

$$A^{(1)} = \begin{bmatrix} 1 & -1 & \alpha \\ 0 & 4 & -2\alpha + 1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix}$$

Esta nueva matriz, facilita el cálculo del determinante.

El determinante de la matriz

$$A^{(1)}$$

se calcula utilizando el método clásico, desarrollándolo por la primera columna:

$$A^{(1)} = \begin{bmatrix} 1 & -1 & \alpha \\ 0 & 4 & -2\alpha + 1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix}$$

Como la primera columna tiene dos ceros, el cálculo del determinante se simplifica considerablemente:

$$\det(A^{(1)}) = 1 \cdot \begin{vmatrix} 4 & -2\alpha+1 \\ \alpha & -\frac{3}{2} \end{vmatrix}$$

Ahora, calculamos el determinante de la submatriz 2×2:

$$\begin{vmatrix} 4 & -2\alpha+1 \\ \alpha & -\frac{3}{2} \end{vmatrix} = (4)(-\frac{3}{2}) - (-2\alpha+1)(\alpha)$$

$$= -6 - (-2\alpha^2 + \alpha)$$

$$= -6 + 2\alpha^2 - \alpha$$

Sustituyendo en la ecuación del determinante:

$$\det(A^{(1)}) = 1 \cdot (2\alpha^2 - \alpha - 6)$$

Por lo tanto, el determinante de $A^{(1)}$ es:

$$\det(A^{(1)}) = 2\alpha^2 - \alpha - 6.$$

Para que A sea **singular**:

$$2\alpha^2 - \alpha - 6 = 0$$

Factorando los términos y resolviendo la ecuación:

$$(2\alpha + 4)(2\alpha + 3) = 0$$

$$\alpha_1 = 2, \quad \alpha_2 = -\frac{3}{2}$$

Por lo tanto, la matriz A será singular cuando:

$$\alpha = 2 \quad \text{o} \quad \alpha = -\frac{3}{2}$$

1.5 Ejercicio 5

Resuelva los siguientes sistemas lineales:

a.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & -1 \\ 0 & -2 & 1 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

```python
L1 = np.array(
    [[1, 0, 0],
     [2, 1, 0],
     [-1, 0, 1]]
)

L2 = np.array(
    [[2, 3, -1],
     [0, -2, 1],
     [0, 0, 3]]
)

b1 = np.array([2, -1, 1])

x1 = resolver_LU(L1, L2, b1)
print(f'\nLa solución del sistema es:\n{x1}')
```

```
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[ 2.]
 [-5.]
 [ 3.]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 1
[02-03 17:43:50][INFO] suma = [1.]
[02-03 17:43:50][INFO] U[i, i] = -2
[02-03 17:43:50][INFO] y[i] = [-5.]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [8.]
[02-03 17:43:50][INFO] U[i, i] = 2
[02-03 17:43:50][INFO] y[i] = [2.]

La solución del sistema es:
[[-3.]
```

```
 [ 3.]
 [ 1.]]
```

b.

$$\begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix}$$

```python
L2 = np.array(
    [[2, 0, 0],
     [-1, 1, 0],
     [3, 2, -1]]
)

U2 = np.array(
    [[1, 1, 1],
     [0, 1, 2],
     [0, 0, 1]]
)

b2 = np.array([-1, 3, 0])

x2 = resolver_LU(L2, U2, b2)
print(f"\nLa solución del sistema es:\n{x2}")
```

```
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[-0.5]
 [ 2.5]
 [ 3.5]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 1
[02-03 17:43:50][INFO] suma = [7.]
[02-03 17:43:50][INFO] U[i, i] = 1
[02-03 17:43:50][INFO] y[i] = [2.5]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [-1.]
[02-03 17:43:50][INFO] U[i, i] = 1
[02-03 17:43:50][INFO] y[i] = [-0.5]

La solución del sistema es:
[[ 0.5]
```

```
 [-4.5]
 [ 3.5]]
```

1.6 Ejercicio 6

Factorice las siguientes matrices en la descomposición LU mediante el algoritmo de factorización LU con $l_{ii} = 1$ para todas las i.

a.

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

```python
X = np.array(
    [[2, -1, 1],
     [3, 3, 9],
     [3,3, 5]]
)

L1, U1 = descomposicion_LU(X)

print("\nDescomposición de la matriz")
print(f"- Matriz L\n{L1}")
print(f"- Matriz U\n{U1}")
```

```
[02-03 18:36:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   4.5  3.5]]
[02-03 18:36:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[02-03 18:36:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]

Descomposición de la matriz
- Matriz L
[[1.  0.  0. ]
 [1.5 1.  0. ]
```

```
 [1.5 1.  1. ]]
- Matriz U
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
```

b.

$$\begin{bmatrix} 1.012 & -2.132 & 3.104 \\ -2.132 & 4.096 & -7.013 \\ 3.104 & -7.013 & 0.014 \end{bmatrix}$$

```python
Y = np.array(
    [[1.012, -2.132, 3.104],
     [-2.132, 4.096, -7.013],
     [3.104, -7.013, 0.014]]
)

print(Y)

L2, U2 = descomposicion_LU(Y)

print("\nDescomposición de la matriz")
print(f"- Matriz L\n{L2}")
print(f"- Matriz U\n{U2}")
```

```
[[ 1.012 -2.132  3.104]
 [-2.132  4.096 -7.013]
 [ 3.104 -7.013  0.014]]
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.         -0.47374308 -9.50656917]]
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]

Descomposición de la matriz
```

```
- Matriz L
[[ 1.          0.          0.        ]
 [-2.10671937  1.          0.        ]
 [ 3.06719368  1.19775553  1.        ]]
- Matriz U
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
```

c.

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 1.5 & 0 & 0 \\ 0 & -3 & 0.5 & 0 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

```python
Z = np.array(
    [[2, 0, 0, 0],
     [1, 1.5, 0, 0],
     [0, -3, 0.5, 0],
     [2, -2, 1, 1]]
)

L3, U3 = descomposicion_LU(Z)

print("\nDescomposición de la matriz")
print(f"- Matriz L\n{L3}")
print(f"- Matriz U\n{U3}")
```

```
[02-03 17:43:50][INFO]
[[ 2.   0.   0.   0. ]
 [ 0.   1.5  0.   0. ]
 [ 0.  -3.   0.5  0. ]
 [ 0.  -2.   1.   1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  1.  1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
```

```
 [0.  0.  0.  1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]

Descomposición de la matriz
- Matriz L
[[ 1.          0.          0.          0.        ]
 [ 0.5         1.          0.          0.        ]
 [ 0.         -2.          1.          0.        ]
 [ 1.         -1.33333333  2.          1.        ]]
- Matriz U
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
```

d.

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

```python
W = np.array(
    [[2.1756, 4.0231, -2.1732, 5.1967],
     [-4.0231, 6.0000, 0, 1.1973],
     [-1.0000, -5.2107, 1.1111, 1.1111],
     [6.0235, 7.0000, 0, -4.1561]]
)

L4, U4 = descomposicion_LU(W)

print("\nDescomposición de la matriz")
print(f"- Matriz L\n{L4}")
print(f"- Matriz U\n{U4}")
```

```
[02-03 17:43:50][INFO]
[[  2.1756       4.0231      -2.1732       5.1967    ]
 [  0.          13.43948042  -4.01866194  10.80699101]
 [  0.          -3.36150897   0.11220314   3.49972842]
```

```
 [  0.          -4.13860216   6.01685521 -18.54400331]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  6.20279403e+00]
 [ 0.00000000e+00  0.00000000e+00  4.77933394e+00 -1.52160595e+01]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  6.20279403e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.79830497e+01]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  6.20279403e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.79830497e+01]]

Descomposición de la matriz
- Matriz L
[[ 1.          0.          0.          0.        ]
 [-1.84919103  1.          0.          0.        ]
 [-0.45964332 -0.25012194  1.          0.        ]
 [ 2.76866152 -0.30794361 -5.35228302  1.        ]]
- Matriz U
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  6.20279403e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.79830497e+01]]
```

1.7 Ejercicio 7

Modifique el algoritmo de eliminación gaussiana de tal forma que se pueda utilizar para resolver un sistema lineal usando la descomposición LU y, a continuación, resuelva los siguientes sistemas lineales.

a.

$$
\begin{aligned}
2x_1 - x_2 + x_3 &= -1,\\
3x_1 + 3x_2 + 9x_3 &= 0,\\
3x_1 + 3x_2 + 5x_3 &= 4.
\end{aligned}
$$

```python
A1 = np.array(
    [[2, -1, 1],
     [3, 3, 9],
     [3, 3, 5]]
)

b1 = np.array([-1, 0, 4])

L1, U1 = descomposicion_LU(A1)

X1 = resolver_LU(L1, U1, b1)

print(f"\nLa solución del sistema es:\n{X1}")
```

```
[02-03 17:43:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   4.5  3.5]]
[02-03 17:43:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[02-03 17:43:50][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[-1. ]
 [ 1.5]
 [ 4. ]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 1
[02-03 17:43:50][INFO] suma = [-7.5]
[02-03 17:43:50][INFO] U[i, i] = 4.5
[02-03 17:43:50][INFO] y[i] = [1.5]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [-3.]
[02-03 17:43:50][INFO] U[i, i] = 2.0
[02-03 17:43:50][INFO] y[i] = [-1.]

La solución del sistema es:
```

```
[[ 1.]
 [ 2.]
 [-1.]]
```

b.

$$\begin{aligned} 1.012x_1 - 2.132x_2 + 3.104x_3 &= 1.984, \\ -2.132x_1 + 4.096x_2 - 7.013x_3 &= -5.049, \\ 3.104x_1 - 7.013x_2 + 0.014x_3 &= -3.895. \end{aligned}$$

```
A2 = np.array(
    [[1.012, -2.132, 3.104],
    [-2.132, 4.096, -7.013],
    [3.104, -7.013, 0.014]]
)

b2 = np.array([1.984, -5.049, -3.895])

L2, U2 = descomposicion_LU(A2)

X2 = resolver_LU(L2, U2, b2)

print(f"\nLa solución del sistema es:\n{X2}")
```

```
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.         -0.47374308 -9.50656917]]
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[02-03 17:43:50][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[ 1.984     ]
 [-0.86926877]
 [-8.93914077]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 1
```

```
[02-03 17:43:50][INFO] suma = [-0.47374308]
[02-03 17:43:50][INFO] U[i, i] = -0.3955256916996053
[02-03 17:43:50][INFO] y[i] = [-0.86926877]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [0.972]
[02-03 17:43:50][INFO] U[i, i] = 1.012
[02-03 17:43:50][INFO] y[i] = [1.984]

La solución del sistema es:
[[1.]
 [1.]
 [1.]]
```

c.

$$
\begin{aligned}
2x_1 &= 3, \\
x_1 + 1.5x_2 &= 4.5, \\
-3x_2 + 0.5x_3 &= -6.6, \\
2x_1 - 2x_2 + x_3 + x_4 &= 0.8.
\end{aligned}
$$

```python
A3 = np.array(
    [[2, 0, 0, 0],
     [1, 1.5, 0, 0],
     [0, -3, 0.5, 0],
     [2, -2, 1, 1]]
)

b3 = np.array([3, 4.5, -6.6, 0.8])

L3, U3 = descomposicion_LU(A3)

X3 = resolver_LU(L3, U3, b3)

print(f"\nLa solución del sistema c es:\n{X3}")
```

```
[02-03 17:43:50][INFO]
[[ 2.   0.   0.   0. ]
 [ 0.   1.5  0.   0. ]
 [ 0.  -3.   0.5  0. ]
 [ 0.  -2.   1.   1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
```

```
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  1.  1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
[02-03 17:43:50][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[ 3. ]
 [ 3. ]
 [-0.6]
 [ 3. ]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 2
[02-03 17:43:50][INFO] suma = [0.]
[02-03 17:43:50][INFO] U[i, i] = 0.5
[02-03 17:43:50][INFO] y[i] = [-0.6]
[02-03 17:43:50][INFO] i = 1
[02-03 17:43:50][INFO] suma = [0.]
[02-03 17:43:50][INFO] U[i, i] = 1.5
[02-03 17:43:50][INFO] y[i] = [3.]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [0.]
[02-03 17:43:50][INFO] U[i, i] = 2.0
[02-03 17:43:50][INFO] y[i] = [3.]

La solución del sistema c es:
[[ 1.5]
 [ 2. ]
 [-1.2]
 [ 3. ]]
```

d.

$$\begin{aligned}
2.1756x_1 + 4.0231x_2 - 2.1732x_3 + 5.1967x_4 &= 17.102,\\
-4.0231x_1 + 6.0000x_2 + 1.1973x_4 &= -6.1593,\\
-1.0000x_1 - 5.2107x_2 + 1.1111x_3 &= 3.0004,\\
6.0235x_1 + 7.0000x_2 - 4.1561x_4 &= 0.0000.
\end{aligned}$$

```python
A4 = np.array(
    [
        [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],
    ]
)

b4 = np.array([17.102, -6.1593, 3.0004, 0.0000])

L4, U4 = descomposicion_LU(A4)

X4 = resolver_LU(L4, U4, b4)

print(f"\nLa solución del sistema es:\n{X4}")
```

```
[02-03 17:43:50][INFO]
[[  2.1756       4.0231      -2.1732       5.1967    ]
 [  0.          13.43948042  -4.01866194  10.80699101]
 [  0.          -3.36150897   0.11220314   2.38862842]
 [  0.          -4.13860216   6.01685521 -18.54400331]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  5.09169403e+00]
 [ 0.00000000e+00  0.00000000e+00  4.77933394e+00 -1.52160595e+01]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  5.09169403e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.20361280e+01]]
[02-03 17:43:50][INFO]
[[ 2.17560000e+00  4.02310000e+00 -2.17320000e+00  5.19670000e+00]
 [ 0.00000000e+00  1.34394804e+01 -4.01866194e+00  1.08069910e+01]
 [ 0.00000000e+00  4.44089210e-16 -8.92952394e-01  5.09169403e+00]
```

```
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.20361280e+01]]
[02-03 17:43:50][INFO] Sustitución hacia adelante
[02-03 17:43:50][INFO] y =
[[17.102     ]
 [25.46556496]
 [17.23071662]
 [52.71598078]]
[02-03 17:43:50][INFO] Sustitución hacia atrás
[02-03 17:43:50][INFO] i = 2
[02-03 17:43:50][INFO] suma = [22.30066378]
[02-03 17:43:50][INFO] U[i, i] = -0.8929523938192969
[02-03 17:43:50][INFO] y[i] = [17.23071662]
[02-03 17:43:50][INFO] i = 1
[02-03 17:43:50][INFO] suma = [24.51569341]
[02-03 17:43:50][INFO] U[i, i] = 13.439480423791139
[02-03 17:43:50][INFO] y[i] = [25.46556496]
[02-03 17:43:50][INFO] i = 0
[02-03 17:43:50][INFO] suma = [10.70605972]
[02-03 17:43:50][INFO] U[i, i] = 2.1756
[02-03 17:43:50][INFO] y[i] = [17.102]

La solución del sistema es:
[[2.9398512 ]
 [0.0706777 ]
 [5.67773512]
 [4.37981223]]
```

GitHub: Tarea10 - @mateobtw18