Métodos Numericos

Descomposicion LU

José Sarango

Tabla de Contenidos

Conjunto de ejercicios

1. Realice las siguientes multiplicaciones matriz-matriz:

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

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

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

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
import numpy as np
# a)
A = np.array([[2,-3],[3,-1]])
A1 = np.array([[1,5],[2,0]])
Ar = np.dot(A, A1)
# b)
B = np.array([[2,-3],[3,-1]])
B1 = np.array([[1,5,-4],[-3,2,0]])
Br = np.dot(B, B1)
# c)
C = np.array([[2,-3,1],[4,3,0],[5,2,-4]])
C1 = np.array([[0,1,-2],[1,0,-1],[2,3,-2]])
Cr = np.dot(C, C1)
# d)
D = np.array([[2,1,2],[-2,3,0],[2,-1,3]])
D1 = np.array([[1,-2],[-4,1],[0,2]])
Dr = np.dot(D, D1)
print("Resultado de la multiplicación del literal a) :\n", Ar)
print("Resultado de la multiplicación del literal b) :\n", Br)
print("Resultado de la multiplicación del literal c) :\n", Cr)
print("Resultado de la multiplicación del literal d) :\n", Dr)
```

```
Resultado de la multiplicación del literal a) :
 [[-4 10]
 [ 1 15]]
Resultado de la multiplicación del literal b) :
 [[ 11   4  -8]
 [  6  13 -12]]
Resultado de la multiplicación del literal c) :
 [[ -1   5  -3]
 [  3   4 -11]
 [ -6  -7  -4]]
Resultado de la multiplicación del literal d) :
 [[ -2   1]
 [-14   7]
 [  6   1]]
```

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

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

c. $\begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 2 & -4 & -2 \\ 2 & 1 & 1 & 5 \\ -1 & 0 & -2 & -4 \end{bmatrix}$ **d.** $\begin{bmatrix} 4 & 0 & 0 & 0 \\ 6 & 7 & 0 & 0 \\ 9 & 11 & 1 & 0 \\ 5 & 4 & 1 & 1 \end{bmatrix}$

```python
import numpy as np

# Definimos las matrices
matrices = [np.array([[4,2,6],[3,0,7],[-2,-1,-3]]), np.array([[1,2,0],[2,1,-1],[3,1,1]]),np.a

for i, M in enumerate(matrices):
    det = np.linalg.det(M)
    if det != 0:
        inv_M = np.linalg.inv(M)
        print(f"Matriz {i+1} es no singular y su inversa es:\n", inv_M)
    else:
        print(f"Matriz {i+1} es singular")
```

```
Matriz 1 es singular
Matriz 2 es no singular y su inversa es:
 [[-0.25   0.25   0.25 ]
 [ 0.625 -0.125 -0.125]
 [ 0.125 -0.625  0.375]]
Matriz 3 es singular
Matriz 4 es no singular y su inversa es:
 [[ 2.50000000e-01 -2.46716228e-17  0.00000000e+00  0.00000000e+00]
 [-2.14285714e-01  1.42857143e-01 -0.00000000e+00 -0.00000000e+00]
 [ 1.07142857e-01 -1.57142857e+00  1.00000000e+00 -0.00000000e+00]
 [-5.00000000e-01  1.00000000e+00 -1.00000000e+00  1.00000000e+00]]
```

3.Resuelva los sistemas lineales 4 x 4 que tienen la misma matriz de coeficientes:

$\boldsymbol{a.} x_1 - x_2 + 2x_3 - x_4 = 6,$
$x_1 - x_3 + x_4 = 4,$
$2x_1 + x_2 + 3x_3 - x_4 = -2.$
$ -x_2+x_3-x_4=5;$

$\boldsymbol{b.} x_1 - x_2 + 2x_3 - x_4 = 1,$
$x_1 - x_3 + x_4 = 1,$
$2x_1 + x_2 + 3x_3 - 4x_4 = 2,$
$-x_2 + x_3 - x_4 = -1;$

```
import numpy as np

# a)
A = np.array([[1,-1,2,-1],[1,0,-1,1],[2,1,3,-4],[0,-1,1,-1]])
b = np.array([6,4,-2,5])
# b)
B = np.array([[1,-1,2,-1],[1,0,-1,1],[2,1,3,-4],[0,-1,1,-1]])
b1 = np.array([1,1,2,-1])



# Resolución del sistema lineal
x = np.linalg.solve(A, b)
x1 = np.linalg.solve(B, b1)
print("Solución del sistema lineal a) :", x)
print("Solución del sistema lineal b) :", x1)
```

```
Solución del sistema lineal a) : [ 3. -6. -2. -1.]
Solución del sistema lineal b) : [1. 1. 1. 1.]
```

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

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

```
import sympy as sp
alpha = sp.symbols('alpha')
A = sp.Matrix([
 [1, -1, alpha],
 [2, 2, 1],
```

```
 [0, alpha, -3/2]
])
det_A = A.det()
alpha_solutions = sp.solve(det_A, alpha)
print("Valores de alpha que hacen que la matriz A sea singular:")
print(alpha_solutions)
```

```
Valores de alpha que hacen que la matriz A sea singular:
[-1.50000000000000, 2.00000000000000]
```

5.Resuelva los siguientes sistemas lineales:

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

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

Se calcula la multiplicacion de cada matriz quedando asi para el sistema a:

```
import numpy as np
A = np.array([[1,0,0],[2,1,0],[-1,0,1]])
A1 = np.array([[2,3,-1],[0,-2,1],[0,0,3]])
Ar = np.dot(A, A1)
print("Resultado de la multiplicación del literal a) :\n", Ar)
```

```
Resultado de la multiplicación del literal a) :
 [[ 2  3 -1]
 [ 4  4 -1]
 [-2 -3  4]]
```

sistema b)

```python
import numpy as np
# b)
A = np.array([[2,0,0],[-1,1,0],[3,2,-1]])
A1 = np.array([[1,1,1],[0,1,2],[0,0,1]])
Ar = np.dot(A, A1)
print("Resultado de la multiplicación del literal b) :\n", Ar)
```

```
Resultado de la multiplicación del literal b) :
 [[ 2  2  2]
 [-1  0  1]
 [ 3  5  6]]
```

Luego calculamos la soluciones del sistema, usando descomposicon LU

```python
import logging
from sys import stdout
from datetime import datetime
import numpy as np

logging.basicConfig(
    level=logging.INFO,
    format="[%(asctime)s][%(levelname)s] %(message)s",
    stream=stdout,
    datefmt="%m-%d %H:%M:%S",
)
logging.info(datetime.now())


# #####################################################################
def eliminacion_gaussiana(A: np.ndarray) -> np.ndarray:
    if not isinstance(A, np.ndarray):
        logging.debug("Convirtiendo A a numpy array.")
        A = np.array(A)
    assert A.shape[0] == A.shape[1] - 1, "La matriz A debe ser de tamaño n-by-(n+1)."
    n = A.shape[0]

    for i in range(0, n - 1):  # loop por columna

        # --- encontrar pivote
        p = None  # default, first element
        for pi in range(i, n):
```

```
        if A[pi, i] == 0:
            # must be nonzero
            continue

        if p is None:
            # first nonzero element
            p = pi
            continue

        if abs(A[pi, i]) < abs(A[p, i]):
            p = pi

    if p is None:
        # no pivot found.
        raise ValueError("No existe solución única.")

    if p != i:
        # swap rows
        logging.debug(f"Intercambiando filas {i} y {p}")
        _aux = A[i, :].copy()
        A[i, :] = A[p, :].copy()
        A[p, :] = _aux

    # --- Eliminación: loop por fila
    for j in range(i + 1, n):
        m = A[j, i] / A[i, i]
        A[j, i:] = A[j, i:] - m * A[i, i:]

    logging.info(f"\n{A}")

if A[n - 1, n - 1] == 0:
    raise ValueError("No existe solución única.")

    print(f"\n{A}")
# --- Sustitución hacia atrás
solucion = np.zeros(n)
solucion[n - 1] = A[n - 1, n] / A[n - 1, n - 1]

for i in range(n - 2, -1, -1):
    suma = 0
    for j in range(i + 1, n):
        suma += A[i, j] * solucion[j]
```

```python
        solucion[i] = (A[i, n] - suma) / A[i, i]

    return solucion


# ######################################################################
def descomposicion_LU(A: np.ndarray) -> tuple[np.ndarray, np.ndarray]:
    A = np.array(
        A, dtype=float
    )  # convertir en float, porque si no, puede convertir como entero

    assert A.shape[0] == A.shape[1], "La matriz A debe ser cuadrada."
    n = A.shape[0]

    L = np.zeros((n, n), dtype=float)

    for i in range(0, n):  # loop por columna

        # --- deterimnar pivote
        if A[i, i] == 0:
            raise ValueError("No existe solución única.")

        # --- Eliminación: loop por fila
        L[i, i] = 1
        for j in range(i + 1, n):
            m = A[j, i] / A[i, i]
            A[j, i:] = A[j, i:] - m * A[i, i:]

            L[j, i] = m

        logging.info(f"\n{A}")

    if A[n - 1, n - 1] == 0:
        raise ValueError("No existe solución única.")

    return L, A


# ######################################################################
def resolver_LU(L: np.ndarray, U: np.ndarray, b: np.ndarray) -> np.ndarray:
    n = L.shape[0]
```

```python
    # --- Sustitución hacia adelante
    logging.info("Sustitución hacia adelante")

    y = np.zeros((n, 1), dtype=float)

    y[0] = b[0] / L[0, 0]

    for i in range(1, n):
        suma = 0
        for j in range(i):
            suma += L[i, j] * y[j]
        y[i] = (b[i] - suma) / L[i, i]

    logging.info(f"y = \n{y}")

    # --- Sustitución hacia atrás
    logging.info("Sustitución hacia atrás")
    sol = np.zeros((n, 1), dtype=float)

    sol[-1] = y[-1] / U[-1, -1]

    for i in range(n - 2, -1, -1):
        logging.info(f"i = {i}")
        suma = 0
        for j in range(i + 1, n):
            suma += U[i, j] * sol[j]
        logging.info(f"suma = {suma}")
        logging.info(f"U[i, i] = {U[i, i]}")
        logging.info(f"y[i] = {y[i]}")
        sol[i] = (y[i] - suma) / U[i, i]

    logging.debug(f"x = \n{sol}")
    return sol


def matriz_aumentada(A: np.ndarray, b: np.ndarray) -> np.ndarray:
    if not isinstance(A, np.ndarray):
        logging.debug("Convirtiendo A a numpy array.")
        A = np.array(A, dtype=float)
    if not isinstance(b, np.ndarray):
        b = np.array(b, dtype=float)
    assert A.shape[0] == b.shape[0], "Las dimensiones de A y b no coinciden."
```

```python
    return np.hstack((A, b.reshape(-1, 1)))
```

```
[07-27 11:59:05][INFO] 2024-07-27 11:59:05.635669
```

```python
if __name__ == "__main__":
    A = [[2, 3, -1], [4, 4, -1], [-2, -3, 4]]
    b = [2, -1, 1]
    A1=[[2,2,2],[-1,0,1],[3,5,6]]
    b1=[-1,3,0]
    Ab = matriz_aumentada(A, b)
    solucion = eliminacion_gaussiana(Ab)
    Ab1 = matriz_aumentada(A1, b1)
    solucion1 = eliminacion_gaussiana(Ab1)
    print("Solución usando eliminación gaussiana del literal a) :", solucion)
    print("Solución usando eliminación gaussiana del literal b) :", solucion1)
```

```
[07-27 11:59:06][INFO]
[[ 2.  3. -1.  2.]
 [ 0. -2.  1. -5.]
 [ 0.  0.  3.  3.]]
[07-27 11:59:06][INFO]
[[ 2.  3. -1.  2.]
 [ 0. -2.  1. -5.]
 [ 0.  0.  3.  3.]]
[07-27 11:59:06][INFO]
[[-1.  0.  1.  3.]
 [ 0.  2.  4.  5.]
 [ 0.  5.  9.  9.]]
[07-27 11:59:06][INFO]
[[-1.   0.   1.   3. ]
 [ 0.   2.   4.   5. ]
 [ 0.   0.  -1.  -3.5]]
Solución usando eliminación gaussiana del literal a) : [-3.  3.  1.]
Solución usando eliminación gaussiana del literal b) : [ 0.5 -4.5  3.5]
```

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

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

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

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

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

```python
if __name__ == "__main__":
    A = np.array([[2, -1, 1], [3, 3, 9], [3, 3, 5]])
    B = np.array([[1.012,-2.132,3.104],[-2.132,4.096,-7.013],[3.104,-7.013,0.014]])
    C = np.array([[2,0,0,0],[1,1.5,0,0],[0,-3,0.5,0],[2,-2,1,1]])
    D = np.array([[2.1756,4.0231,-2.1732,5.1967],[-4.0231,6.0000,0,1.1973],[-1.0000,-5.2107,

    L, U = descomposicion_LU(A)
    L1, U1 = descomposicion_LU(B)
    L2, U2 = descomposicion_LU(C)
    L3, U3 = descomposicion_LU(D)
    print("Solucion del literal a)\n:")
    print("Matriz L (triangular inferior):")
    print(L)
    print("\nMatriz U (triangular superior):")
    print(U)
    print("Solucion del literal b)\n:")
    print("Matriz L (triangular inferior):")
    print(L1)
    print("\nMatriz U (triangular superior):")
    print(U1)

    print("Solucion del literal c)\n:")
    print("Matriz L (triangular inferior):")
    print(L2)
    print("\nMatriz U (triangular superior):")
    print(U2)

    print("Solucion del literal d)\n:")
    print("Matriz L (triangular inferior):")
    print(L3)
```

```python
    print("\nMatriz U (triangular superior):")
    print(U3)
```

```
[07-27 11:59:07][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   4.5  3.5]]
[07-27 11:59:07][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[07-27 11:59:07][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[07-27 11:59:07][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.         -0.47374308 -9.50656917]]
[07-27 11:59:07][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[07-27 11:59:07][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[07-27 11:59:07][INFO]
[[ 2.   0.   0.   0. ]
 [ 0.   1.5  0.   0. ]
 [ 0.  -3.   0.5  0. ]
 [ 0.  -2.   1.   1. ]]
[07-27 11:59:07][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  1.  1. ]]
[07-27 11:59:07][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
```

```
[07-27 11:59:07][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
[07-27 11:59:07][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]]
[07-27 11:59:07][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]]
[07-27 11:59:07][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]]
[07-27 11:59:07][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]]
Solucion del literal a)
:
Matriz L (triangular inferior):
[[1.  0.  0. ]
 [1.5 1.  0. ]
 [1.5 1.  1. ]]

Matriz U (triangular superior):
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
Solucion del literal b)
:
Matriz L (triangular inferior):
[[ 1.          0.          0.        ]
 [-2.10671937  1.          0.        ]
 [ 3.06719368  1.19775553  1.        ]]
```

```
Matriz U (triangular superior):
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
Solucion del literal c)
:
Matriz L (triangular inferior):
[[ 1.          0.          0.          0.        ]
 [ 0.5         1.          0.          0.        ]
 [ 0.         -2.          1.          0.        ]
 [ 1.         -1.33333333  2.          1.        ]]

Matriz U (triangular superior):
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
Solucion del literal d)
:
Matriz L (triangular inferior):
[[ 1.          0.          0.          0.        ]
 [-1.84919103  1.          0.          0.        ]
 [-0.45964332 -0.25012194  1.          0.        ]
 [ 2.76866152 -0.30794361 -5.35228302  1.        ]]

Matriz U (triangular superior):
[[ 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]]
```

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. $2x_1 - x_2 + x_3 = -1$
$3x_1 + 3x_2 + 9x_3 = 0$
$3x_1 + 3x_2 + 5x_3 = 4$

b. $1.012x_1 - 2.132x_2 + 3.104x_3 = 1.984$
$-2.132x_1 + 4.096x_2 - 7.013x_3 = -5.049$
$3.104x_1 - 7.013x_2 + 0.014x_3 = -3.895$

c. $2x_1 = 3$
$x_1 + 1.5x_2 = 4.5$
$-3x_2 + 0.5x_3 = -6.6$
$2x_1 - 2x_2 + x_3 + x_4 = 0.8$

d. $2.1756x_1 + 4.0231x_2 - 2.1732x_3 + 5.1967x_4 = 17.102$
$-4.0231x_1 + 6.0000x_2 + 1.1973x_4 = -6.1593$
$-1.0000x_1 - 5.2107x_2 + 1.1111x_3 = 3.0004$
$6.0235x_1 + 7.0000x_2 - 4.1561x_4 = 0.0000$

```python
import numpy as np
import logging
from sys import stdout
from datetime import datetime
logging.basicConfig(
    level=logging.INFO,
    format="[%(asctime)s][%(levelname)s] %(message)s",
    stream=stdout,
    datefmt="%m-%d %H:%M:%S",
)
logging.info(datetime.now())

def descomposicion_LU(A: np.ndarray) -> tuple[np.ndarray, np.ndarray]:

    A = np.array(
        A, dtype=float
    )  # convertir en float, porque si no, puede convertir como entero

    assert A.shape[0] == A.shape[1], "La matriz A debe ser cuadrada."
    n = A.shape[0]

    L = np.zeros((n, n), dtype=float)

    for i in range(n):  # loop por columna
        L[i, i] = 1  # Set diagonal of L to 1

        for j in range(i+1, n):
            if A[i, i] == 0:
                raise ValueError("No existe solución única.")

            # Calcula el multiplicador
            m = A[j, i] / A[i, i]
            L[j, i] = m
```

```python
            # Resta la fila multiplicada de la fila actual
            A[j, i:] = A[j, i:] - m * A[i, i:]

        logging.info(f"\n{A}")

    return L, A  # A se convierte en U


def resolver_LU(L: np.ndarray, U: np.ndarray, b: np.ndarray) -> np.ndarray:

    n = L.shape[0]

    # --- Sustitución hacia adelante para resolver Ly = b
    y = np.zeros(n)

    for i in range(n):
        suma = sum(L[i, j] * y[j] for j in range(i))
        y[i] = (b[i] - suma) / L[i, i]

    # --- Sustitución hacia atrás para resolver Ux = y
    x = np.zeros(n)

    for i in range(n - 1, -1, -1):
        suma = sum(U[i, j] * x[j] for j in range(i + 1, n))
        x[i] = (y[i] - suma) / U[i, i]

    return x
```

```
[07-27 11:59:08][INFO] 2024-07-27 11:59:08.465107
```

```python
A = np.array([[2, -1, 1],[3, 3, 9],[3, 3, 5]])
b = np.array([-1, 0, 4])

B = np.array([[1.012,-2.132,3.104],[-2.132,4.096,-7.013],[3.104,-7.013,0.014]])
b1 = np.array([1.984,-5.049,-3.895])

C = np.array([[2,0,0,0],[1,1.5,0,0],[0,-3,0.5,0],[2,-2,1,1]])
b2 = np.array([3,4.5,-6.6,0.8])

D = np.array([[2.1756,4.0231,-2.1732,5.1967], [-4.0231,6.0000,0,1.1973],[-1.0000,-5.2107,1.1
b3 = np.array([17.102,-6.1593,3.0004,0.0000])
```

```
#Literal a)
L, U = descomposicion_LU(A)
solucion = resolver_LU(L, U, b)
#Literal b)
L, U = descomposicion_LU(B)
solucion1 = resolver_LU(L, U, b1)
#Literal c)
L, U = descomposicion_LU(C)
solucion2 = resolver_LU(L, U, b2)
#Literal d)
L, U = descomposicion_LU(D)
solucion3 = resolver_LU(L, U, b3)

print("\nSolución del sistema literal a) :\n")
print(solucion)
print("\nSolución del sistema literal b) :\n")
print(solucion1)
print("\nSolución del sistema literal c) :\n")
print(solucion2)
print("\nSolución del sistema literal d) :\n")
print(solucion3)
```

```
[07-27 11:59:08][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   4.5  3.5]]
[07-27 11:59:08][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[07-27 11:59:08][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[07-27 11:59:08][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.         -0.47374308 -9.50656917]]
[07-27 11:59:08][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
```

```
 [ 0.          0.         -8.93914077]]
[07-27 11:59:08][INFO]
[[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]
[07-27 11:59:08][INFO]
[[ 2.   0.   0.   0. ]
 [ 0.   1.5  0.   0. ]
 [ 0.  -3.   0.5  0. ]
 [ 0.  -2.   1.   1. ]]
[07-27 11:59:08][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  1.  1. ]]
[07-27 11:59:08][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
[07-27 11:59:08][INFO]
[[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]
[07-27 11:59:08][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]]
[07-27 11:59:08][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]]
[07-27 11:59:08][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]]
[07-27 11:59:08][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]]

Solución del sistema literal a) :

[ 1.  2. -1.]

Solución del sistema literal b) :

[1. 1. 1.]

Solución del sistema literal c) :

[ 1.5  2.  -1.2  3. ]

Solución del sistema literal d) :

[2.9398512  0.0706777  5.67773512 4.37981223]
```

Link del repositorio de Github: https://github.com/armando-2002/Metodos_Numericos.git