

[Tarea 10] Ejercicios Unidad 04-C | Descomposición LU

Resuelva los ejercicios adjuntos.

1. Realice las siguientes multiplicaciones matriz-matriz:

a.

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

```
import numpy as np

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

resultado_a = np.dot(A, B)
print("Resultado:")
print(resultado_a)
```

Resultado:

```
[[ -4  10]
 [  1  15]]
```

b.

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

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

resultado_b = np.dot(A, B)
print("Resultado:")
print(resultado_b)
```

Resultado:

```
[[ 11   4  -8]
 [  6  13 -12]]
```

c.

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

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

```
resultado_c = np.dot(A, B)
print("Resultado:")
print(resultado_c)
```

Resultado:

```
[[ -1   5  -3]
 [  3   4 -11]
 [-6  -7  -4]]
```

d.

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

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

```
resultado_d = np.dot(A, B)
print("Resultado:")
print(resultado_d)
```

Resultado:

```
[[ -2   1]
 [-14   7]
 [  6   1]]
```

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

```

import numpy as np

def analizar_matriz(matriz):
    try:
        det = np.linalg.det(matriz)
        if det != 0:
            inversa = np.linalg.inv(matriz)
            print(f"Determinante: {det:.2f} (No singular)")
            print("Inversa:")
            print(inversa, "\n")
        else:
            print("Determinante: 0 (Singular)\n")
    except Exception as e:
        print(f"Error en el análisis - {e}\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]])
analizar_matriz(A)

```

Determinante: 0 (Singular)

b.

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

```

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

```

Determinante: -8.00 (No singular)

Inversa:

```

[[-0.25  0.25  0.25 ]
 [ 0.625 -0.125 -0.125]
 [ 0.125 -0.625  0.375]]

```

c.

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

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

Determinante: 0 (Singular)

d.

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

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

Determinante: 28.00 (No singular)

Inversa:

```
[[ 0.25      0.      0.      0.      ]
 [-0.21428571 0.14285714 -0.      -0.      ]
 [ 0.10714286 -1.57142857  1.      -0.      ]
 [-0.5       1.      -1.      1.      ]]
```

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

$$\begin{array}{ll} x_1 - x_2 + 2x_3 - x_4 = 6, & x_1 - x_2 + 2x_3 - x_4 = 1, \\ x_1 - x_3 + x_4 = 4, & x_1 - x_3 + x_4 = 1, \\ 2x_1 + x_2 + 3x_3 - 4x_4 = -2, & 2x_1 + x_2 + 3x_3 - 4x_4 = 2, \\ -x_2 + x_3 - x_4 = 5, & -x_2 + x_3 - x_4 = -1. \end{array}$$

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

```

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

x1 = np.linalg.solve(A, b1)

print("Primer sistema:")
print(f"x1 = {x1[0]:.2f}, x2 = {x1[1]:.2f}, x3 = {x1[2]:.2f}, x4 = {x1[3]:.2f}\n")

```

Primer sistema:
x1 = 3.00, x2 = -6.00, x3 = -2.00, x4 = -1.00

```

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

x2 = np.linalg.solve(A, b2)

print("Segundo sistema:")
print(f"x1 = {x2[0]:.2f}, x2 = {x2[1]:.2f}, x3 = {x2[2]:.2f}, x4 = {x2[3]:.2f}")

```

Segundo sistema:
x1 = 1.00, x2 = 1.00, x3 = 1.00, x4 = 1.00

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

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

5. Resuelva los siguientes sistemas lineales:

a.

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

```

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

B = np.array([

```

```

    [2, 3, -1],
    [0, -2, 1],
    [0, 0, 3]
])

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

x_a = np.linalg.solve(A @ B, b)
print("Literal a:")
print(f"x1 = {x_a[0]:.2f}, x2 = {x_a[1]:.2f}, x3 = {x_a[2]:.2f}\n")

```

Literal a:
x1 = -3.00, x2 = 3.00, x3 = 1.00

b.

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

```

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

x = np.linalg.solve(A @ B, b)
print("Literal b:")
print(f"x1 = {x[0]:.2f}, x2 = {x[1]:.2f}, x3 = {x[2]:.2f}")

```

Literal b:
x1 = 0.50, x2 = -4.50, x3 = 3.50

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

```
from scipy.linalg import lu

def factorizar_lu(matriz):
    P, L, U = lu(matriz)
    print("Matriz L:")
    print(L)
    print("Matriz U:")
    print(U)
    print("Matriz P (permuta):")
    print(P)
    print("\n")
```

a.

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

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

```
Matriz L:
[[ 1.          0.          0.          ]
 [ 0.66666667  1.          0.          ]
 [ 1.          -0.          1.          ]]
Matriz U:
[[ 3.  3.  9.]
 [ 0. -3. -5.]
 [ 0.  0. -4.]]
Matriz P (permuta):
[[0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]]
```

b.

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

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

Matriz L:

```
[[ 1.          0.          0.          ]
 [-0.68685567  1.          0.          ]
 [ 0.32603093 -0.21424728  1.          ]]
```

Matriz U:

```
[[ 3.104      -7.013      0.014      ]
 [ 0.         -0.72091881 -7.00338402]
 [ 0.          0.         1.59897957]]
```

Matriz P (permuta):

```
[[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

c.

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

```
A_c = np.array([
    [2, 0, 0, 0],
    [1, 1.5, 0, 0],
    [0, -3, 0.5, 0],
    [2, -2, 1, 1]
])
factorizar_lu(A)
```


Matriz L:

```
[[ 1.          0.          0.          ]
 [-0.68685567  1.          0.          ]
 [ 0.32603093 -0.21424728  1.          ]]
```

Matriz U:

```
[[ 3.104      -7.013      0.014      ]
 [ 0.         -0.72091881 -7.00338402]
 [ 0.          0.         1.59897957]]
```

Matriz P (permuta):

```
[[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

d.

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

```
A = np.array([
    [2.1756, 4.0231, -2.1732, 5.1967],
    [-4.0231, 6.0000, 0.0000, 1.1973],
    [-1.0000, -5.2107, 1.1111, 0.0000],
    [6.0235, 7.0000, 0.0000, -4.1561]
])
factorizar_lu(A)
```

Matriz L:

```
[[ 1.          0.          0.          0.          ]
 [-0.66790072  1.          0.          0.          ]
 [ 0.36118536  0.14002434  1.          0.          ]
 [-0.16601644 -0.37924771 -0.5112737  1.          ]]
```

Matriz U:

```
[[ 6.0235      7.          0.         -4.1561      ]
 [ 0.          10.67530506  0.         -1.57856219]
 [ 0.          0.         -2.1732     6.91885959]
 [ 0.          0.          0.         2.24878393]]
```

Matriz P (permuta):

```
[[0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 0.]]
```

```
[1. 0. 0. 0.]
```

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.

```
from scipy.linalg import lu_factor, lu_solve

def resolver_sistema_con_lu(A, b):
    lu, piv = lu_factor(A)
    x = lu_solve((lu, piv), b)

    print("Solución:")
    for i, val in enumerate(x):
        print(f"x{i+1} = {val:.4f}")
    print("\n")
```

a.

$$2x_1 - x_2 + x_3 = -1,$$

$$3x_1 + 3x_2 + 9x_3 = 0,$$

$$3x_1 + 3x_2 + 5x_3 = 4.$$

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

Solución:

x1 = 1.0000

x2 = 2.0000

x3 = -1.0000

b.

$$1.012x_1 - 2.132x_2 + 3.104x_3 = 1.984,$$

$$-2.132x_1 + 4.096x_2 - 7.013x_3 = -5.049,$$

$$3.104x_1 - 7.013x_2 + 0.014x_3 = -3.895.$$

```

A = np.array([
    [1.012, -2.132, 3.104],
    [-2.132, 4.096, -7.013],
    [3.104, -7.013, 0.014]
])
b = np.array([1.984, -5.049, -3.895])
resolver_sistema_con_lu(A, b)

```

Solución:

```

x1 = 1.0000
x2 = 1.0000
x3 = 1.0000

```

c.

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

```

A = np.array([
    [2, 0, 0, 0],
    [1, 1.5, 0, 0],
    [0, -3, 0.5, 0],
    [2, -2, 1, 1]
])
b = np.array([3, 4.5, -6.6, 0.8])
resolver_sistema_con_lu(A, b)

```

Solución:

```

x1 = 1.5000
x2 = 2.0000
x3 = -1.2000
x4 = 3.0000

```

d.

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

```
A = 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]
])
b = np.array([17.102, -6.1593, 3.0004, 0.0000])
resolver_sistema_con_lu(A, b)
```

Solución:

x1 = 2.9399

x2 = 0.0707

x3 = 5.6777

x4 = 4.3798

Declaración de uso de IA

En la preparación de este contenido, se utilizó ChatGPT para generar las instrucciones de los ejercicios, comprender el código, realizar correcciones y asistir en la presentación de las gráficas, con el objetivo de optimizar el proceso de elaboración y mantener la responsabilidad del producto final en el criterio del autor.

Link del repositorio <https://github.com/carol230/MetodosNumericos>