TAREA N° 10

Nombre:

Joel Stalin Tinitana Carrion

Fecha:

17/07/2025

Tema:

Descomposición LU

CONJUNTO DE EJERCICIOS

1. Realice la siguiente multiplicación matriz-matriz:

1.a

$$\left[\begin{matrix}2 & -3\\ 3 & -1\end{matrix}\right)\left[\begin{matrix}1 & 5\\ 2 & 0\end{matrix}\right)$$

```python
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]]
```

1.b

$$\left[\begin{matrix}2 & -3\\ 3 & -1\end{matrix}\right)\left[\begin{matrix}1 & 5 & -4\\ -3 & 2 & 0\end{matrix}\right)$$

```python
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]]
```

1.c

$$\left[\begin{matrix} 2 & -3 & 1 \\ 4 & 3 & 0 \\ 5 & 2 & -4 \end{matrix}\right)\left[\begin{matrix} 0 & 1 & -2 \\ 1 & 0 & -1 \\ 2 & 3 & -2 \end{matrix}\right)$$

```
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]]
```

1.d

$$\left[\begin{matrix} 2 & 1 & 2 \\ -2 & 3 & 0 \\ 2 & -1 & 3 \end{matrix}\right)\left[\begin{matrix} 1 & -2 \\ -4 & 1 \\ 0 & 2 \end{matrix}\right)$$

```
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)
```

```
        print(f"Determinante: {det:.6f}")

        # Usar tolerancia para evitar errores por redondeo
        if abs(det) > 1e-10:
            inversa = np.linalg.inv(matriz)
            print("La matriz es NO singular.")
            print("Inversa:")
            print(np.round(inversa, 4))  # redondeo a 4 decimales
            print()
        else:
            print("La matriz es SINGULAR (no tiene inversa).\n")
    except Exception as e:
        print(f"Error en el análisis - {e}\n")
```

2.a

$$\left[\begin{matrix} 4 & 2 & 6 \\ 3 & 0 & 7 \\ -2 & -1 & -3 \end{matrix}\right)$$

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

```
Determinante: 0.000000
La matriz es SINGULAR (no tiene inversa).
```

2.b

$$\left[\begin{matrix} 1 & 2 & 0 \\ 2 & 1 & -1 \\ 3 & 1 & 1 \end{matrix}\right)$$

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

```
Determinante: -8.000000
La matriz es NO singular.
Inversa:
[[-0.25   0.25   0.25 ]
 [ 0.625 -0.125 -0.125]
 [ 0.125 -0.625  0.375]]
```

2.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: 0.000000
La matriz es SINGULAR (no tiene inversa).
```

2.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: 28.000000
La matriz es NO singular.
Inversa:
[[ 0.25    0.      0.      0.    ]
 [-0.2143  0.1429 -0.     -0.    ]
 [ 0.1071 -1.5714  1.     -0.    ]
 [-0.5     1.     -1.      1.    ]]
```

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

Sistema 1

$$ x_1 - x_2 + 2x_3 - x_4 = 6 \\ x_1 - x_3 + x_4 = 4 \\ 2x_1 + x_2 + 3x_3 - 4x_4 = -2 \\ - x_2 + x_3 - x_4 = 5 $$

Sistema 2

$$ x_1 - x_2 + 2x_3 - x_4 = 1 \\ x_1 - x_3 + x_4 = 1 \\ 2x_1 + x_2 + 3x_3 - 4x_4 = 2 \\ - x_2 + x_3 - x_4 = -1 $$

```
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")

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}")
```

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

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

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

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

Una matriz es **singular** si:

$$det(A)=0$$

Cálculo del determinante:

$$det(A)=1\cdot \left| \begin{matrix} 2 & 1 \\ \alpha & -3/2 \end{matrix} ¿¿ - (-1)\cdot \left| \begin{matrix} 2 & 1 \\ 0 & -3/2 \end{matrix} ¿¿ + \alpha \cdot \left| \begin{matrix} 2 & 2 \\ 0 & \alpha \end{matrix} ¿¿$$

Primer menor:

$$\left| \begin{matrix} 2 & 1 \\ \alpha & -3/2 \end{matrix} ¿¿ = (2)(-1.5)-(1)(\alpha)=-3-\alpha$$

Segundo menor:

$$\left| \begin{matrix} 2 & 1 \\ 0 & -3/2 \end{matrix} ¿¿ = (2)(-1.5)-(1)(0)=-3$$

Tercer menor:

$$|\begin{matrix} 2 & 2 \\ 0 & \alpha \end{matrix} ¿¿=(2)(\alpha)-(2)(0)=2\alpha$$

Sustituimos en la expansión:

$$det(A)=1(-3-\alpha)+1(-3)+\alpha(2\alpha)$$

$$det(A)=(-3-\alpha)-3+2\alpha^2$$

$$det(A)=2\alpha^2-\alpha-6$$

Para que sea singular:

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

Resolvemos la ecuación cuadrática:

$$\alpha=\frac{-(-1)\pm\sqrt{(-1)^2-4(2)(-6)}}{2(2)}$$

$$\alpha=\frac{1\pm\sqrt{1+48}}{4}$$

$$\alpha=\frac{1\pm 7}{4}$$

Valores:

$$\alpha_1=\frac{1+7}{4}=\frac{8}{4}=2$$

$$\alpha_2=\frac{1-7}{4}=\frac{-6}{4}=-3/2$$

Respuesta:

$$\alpha=2\,\mathrm{o}\,\alpha=-3/2$$

5. Resuelva los siguientes sistemas lineales:

5.a

$$\left[\begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}\right)\left[\begin{matrix} 2 & 3 & -1 \\ 0 & -2 & 1 \\ 0 & 0 & 3 \end{matrix}\right)\left[\begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix}\right)=\left[\begin{matrix} 2 \\ -1 \\ 1 \end{matrix}\right)$$

```
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
```

5.b

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

```
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 $\ell_{ii}=1$ para todas las i:

```
from scipy.linalg import lu
```

```python
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")
```

6.a

$$\left[\begin{matrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{matrix}\right)$$

```python
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.]]
```

6.b

$$\left[\begin{matrix} 1.012 & -2.132 & 3.104 \\ -2.132 & 4.096 & -7.013 \\ 3.104 & -7.013 & 0.014 \end{matrix}\right)$$

```python
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.]]
```

6.c

$$\left[\begin{matrix} 2 & 0 & 0 & 0 \\ 1 & 1.5 & 0 & 0 \\ 0 & -3 & 0.5 & 0 \\ 2 & -2 & 1 & 1 \end{matrix}\right)$$

```
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.]]
```

6.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.]
 [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")
```

7.a

$$\begin{cases} 2x_1 - x_2 + x_3 = -1 \\ 3x_1 + 3x_2 + 9x_3 = 0 \\ 3x_1 + 3x_2 + 5x_3 = 4 \end{cases}$$

```
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
```

7.b

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

```
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
```

7.c

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

```
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
```

7.d

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

```
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
```