ESCUELA
POLITÉCNICA
NACIONAL

Métodos numéricos.

Nombre: Luis Enrique Pérez Señalin.

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

Respusta:

A: $\begin{bmatrix} -4 & 10 \\ 1 & 15 \end{bmatrix}$ B: $\begin{bmatrix} 11 & 4 & -8 \\ 6 & 13 & -12 \end{bmatrix}$

C: $\begin{bmatrix} -1 & 5 & -3 \\ 3 & 4 & -11 \\ -6 & -7 & -4 \end{bmatrix}$ D: $\begin{bmatrix} -2 & 1 \\ -14 & 7 \\ 6 & 1 \end{bmatrix}$

Métodos Numéricos Tarea09

Pregunta 1

```
import numpy as np

# Multiplicación de matrices usando numpy
a1 = np.array([
    [2, -3],
    [3, -1]
])
a2 = np.array([
    [1, 5],
    [2, 0]
])

b1 = np.array([
    [2, -3],
    [3, -1]
])
```

```
np.dot(a1,a2)
```

```
array([[-4, 10],
       [ 1, 15]])
```

```
np.dot(b1,b2)
```

```
array([[ 11,   4,  -8],
       [  6,  13, -12]])
```

```
np.dot(c1,c2)
```

```
array([[ -1,   5,  -3],
       [  3,   4, -11],
       [ -6,  -7,  -4]])
```

```
np.dot(d1,d2)
```

```
array([[ -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.** $\begin{bmatrix} 4 & 0 & 0 & 0 \\ 6 & 7 & 0 & 0 \\ 9 & 11 & 1 & 0 \\ 5 & 4 & 1 & 1 \end{bmatrix}$

Respuestas:

A: Es una matriz singular y no tiene inversa.
B: Es una matriz no singular y su inversa es:

$$\begin{bmatrix} -0.25 & 0.25 & 0.25 \\ 0.625 & -0.125 & -0.125 \\ 0.125 & -0.625 & 0.375 \end{bmatrix}$$

C: Es una matriz singular y no tiene inversa.
D: Es una matriz no singular y su inversa es:

$$\begin{bmatrix} 0.25 & 0 & 0 & 0 \\ -0.214 & 0.14 & -0 & -0 \\ 0.10 & -1.57 & 1 & -0 \\ -0.5 & 1 & -1 & 1 \end{bmatrix}$$

Pregunta 2

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

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

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

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

```
# Calcular y mostrar el determinante de cada matriz
det_a = np.linalg.det(a)
det_b = np.linalg.det(b)
det_c = np.linalg.det(c)
det_d = np.linalg.det(d)

print(f"Determinante de la matriz a: {det_a}")
print(f"Determinante de la matriz b: {det_b}")
print(f"Determinante de la matriz c: {det_c}")
print(f"Determinante de la matriz d: {det_d}")
```

```
Determinante de la matriz a: 0.0
Determinante de la matriz b: -8.000000000000002
Determinante de la matriz c: 0.0
Determinante de la matriz d: 28.00000000000001
```

```
inv_b = np.linalg.inv(b)
inv_b
```

```
array([[-0.25 ,  0.25 ,  0.25 ],
       [ 0.625, -0.125, -0.125],
       [ 0.125, -0.625,  0.375]])
```

```
inv_d = np.linalg.inv(d)
inv_d
```

```
array([[ 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 x 4 que tienen la misma matriz de coeficientes:

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

Respusta:
Utilizando descomposición LU de scipy resolvemos el ejercicio.

A: $[3 \quad -6 \quad -2 \quad -1]$
B: $[1 \quad 1 \quad 1 \quad 1]$

Pregunta 3

```
from scipy.linalg import lu, lu_solve, lu_factor

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

```
P, L, U = lu(A)

print("Matriz L:\n", L)
print("Matriz U:\n", U)
```

```
Matriz L:
 [[ 1.          0.          0.          0.        ]
 [ 0.5         1.          0.          0.        ]
 [ 0.5         0.33333333  1.          0.        ]
 [ 0.          0.66666667 -0.25        1.        ]]
Matriz U:
 [[ 2.          1.          3.         -4.        ]
 [ 0.         -1.5         0.5         1.        ]
 [ 0.          0.         -2.66666667  2.66666667]
 [ 0.          0.          0.         -1.        ]]
```

```
lu_piv = lu_factor(A)

x1 = lu_solve(lu_piv, b1)

x2 = lu_solve(lu_piv, b2)

print("Solución del primer sistema:\n", x1)
print("Solución del segundo sistema:\n", x2)
```

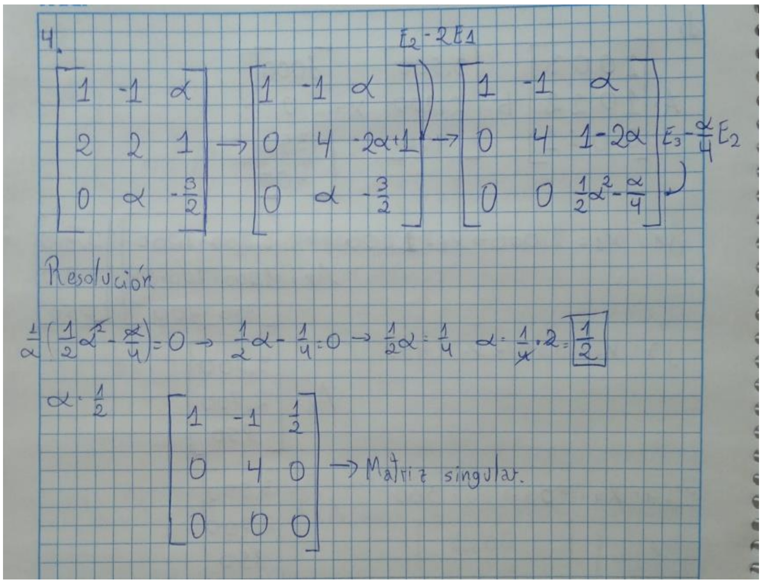
```
Solución del primer sistema:
 [ 3. -6. -2. -1.]
Solución del segundo sistema:
 [1. 1. 1. 1.]
```

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

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

Respuesta:

$\alpha = \dfrac{1}{2}$



4.

$$\begin{bmatrix} 1 & -1 & \alpha \\ 2 & 2 & 1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix} \xrightarrow{E_2 - 2E_1} \begin{bmatrix} 1 & -1 & \alpha \\ 0 & 4 & -2\alpha+1 \\ 0 & \alpha & -\frac{3}{2} \end{bmatrix} \xrightarrow{E_3 - \frac{\alpha}{4}E_2} \begin{bmatrix} 1 & -1 & \alpha \\ 0 & 4 & 1-2\alpha \\ 0 & 0 & \frac{1}{2}\alpha^2 - \frac{\alpha}{4} \end{bmatrix}$$

Resolución

$$\frac{1}{\alpha}\left(\frac{1}{2}\alpha^{\cancel{2}} - \frac{\cancel{\alpha}}{4}\right) = 0 \rightarrow \frac{1}{2}\alpha - \frac{1}{4} = 0 \rightarrow \frac{1}{2}\alpha = \frac{1}{4} \quad \alpha = \frac{1}{\cancel{4}} \cdot 2 = \boxed{\frac{1}{2}}$$

$\alpha = \frac{1}{2}$

$$\begin{bmatrix} 1 & -1 & \frac{1}{2} \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \text{Matriz singular.}$$

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

Respuesta:

Las matrices tienen forma de LU * x = b, entonces se resuelve utilizando descomposición LU.

A: $[-3 \quad 3 \quad 1]$

B: $[0.5 \quad -4.5 \quad 3.5]$

Pregunta 5

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

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

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

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

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

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

```
# Resolver el primer sistema L * y = b_a
lu_piv_a = lu_factor(A_a)  # Factoriza L en A_a
y_a = lu_solve(lu_piv_a, b_a)

# Resolver el sistema U * x = y_a para encontrar x
x_a = np.linalg.solve(U_a, y_a)

# Resolver el segundo sistema L * y = b_b
lu_piv_b = lu_factor(A_b)
y_b = lu_solve(lu_piv_b, b_b)

# Resolver el sistema U * x = y_b para encontrar x
x_b = np.linalg.solve(U_b, y_b)
```

```
# Imprimir resultados
print("Solución del sistema a:")
print(x_a)

print("\nSolución del sistema b:")
print(x_b)
```

```
Solución del sistema a:
[-3.  3.  1.]

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

Respuesta:
Tenemos que hacer una descomposición LU donde L tienen una diagonal de 1.

A:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix}$$

B:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -2.1 & 1 & 0 \\ 3.06 & 1.19 & 1 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 1.012 & -2.132 & 3.104 \\ 0 & -0.395 & -0.473 \\ 0 & 0 & -8.939 \end{bmatrix}$$

C:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 1 & -1.33 & 2 & 1 \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} 2.17 & 4.02 & -2.17 & 5.19 \\ 0 & 13.43 & -4.01 & 10.80 \\ 0 & 0 & -0.89 & 5.09 \\ 0 & 0 & 0 & 12.03 \end{pmatrix}$$

Pregunta 6

```python
def lu_decomposition(A):
    n = A.shape[0]
    L = np.zeros_like(A, dtype=np.float64)
    U = np.zeros_like(A, dtype=np.float64)

    for i in range(n):
        # Set diagonal of L to 1
        L[i, i] = 1.0

        for j in range(i, n):
            U[i, j] = A[i, j] - L[i, :i].dot(U[:i, j])

        for j in range(i+1, n):
            L[j, i] = (A[j, i] - L[j, :i].dot(U[:i, i])) / U[i, i]

    return L, U

# Definir las matrices
A_a = np.array([
    [2, -1, 1],
    [3, 3, 9],
    [3, 3, 5]
])

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

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

A_d = 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]
])

# Factorización LU
L_a, U_a = lu_decomposition(A_a)
L_b, U_b = lu_decomposition(A_b)
L_c, U_c = lu_decomposition(A_c)
L_d, U_d = lu_decomposition(A_d)

# Imprimir resultados
print("Matriz L_a:\n", L_a)
print("Matriz U_a:\n", U_a)
print("\nMatriz L_b:\n", L_b)
print("Matriz U_b:\n", U_b)
print("\nMatriz L_c:\n", L_c)
print("Matriz U_c:\n", U_c)
print("\nMatriz L_d:\n", L_d)
print("Matriz U_d:\n", U_d)
```

```
Matriz L_a:
 [[1.  0.  0. ]
 [1.5 1.  0. ]
 [1.5 1.  1. ]]
Matriz U_a:
 [[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]

Matriz L_b:
 [[ 1.          0.          0.        ]
 [-2.10671937  1.          0.        ]
 [ 3.06719368  1.19775553  1.        ]]
Matriz U_b:
 [[ 1.012      -2.132       3.104     ]
 [ 0.         -0.39552569 -0.47374308]
 [ 0.          0.         -8.93914077]]

Matriz L_c:
 [[ 1.          0.          0.          0.        ]
 [ 0.5         1.          0.          0.        ]
 [ 0.         -2.          1.          0.        ]
 [ 1.         -1.33333333  2.          1.        ]]
Matriz U_c:
 [[2.  0.  0.  0. ]
 [0.  1.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  1. ]]

Matriz L_d:
 [[ 1.          0.          0.          0.        ]
 [-1.84919103  1.          0.          0.        ]
 [-0.45964332 -0.25012194  1.          0.        ]
 [ 2.76866152 -0.30794361 -5.35228302  1.        ]]
Matriz U_d:
 [[ 2.1756      4.0231     -2.1732      5.1967    ]
 [ 0.         13.43948042 -4.01866194 10.80699101]
 [ 0.          0.         -0.89295239  5.09169403]
 [ 0.          0.          0.         12.03612803]]
```

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

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

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

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

Pregunta 7

```python
import numpy as np

def eliminacion_gaussiana_con_lu(A, b):
    """
    Realiza la eliminación gaussiana con descomposición LU para resolver un sistema de ecuacione

    Args:
        A: Matriz de coeficientes (numpy.ndarray).
        b: Vector de términos independientes (numpy.ndarray).

    Returns:
        Vector solución (numpy.ndarray).
    """
    n = len(A)
    L = np.eye(n, dtype=np.float64)
    U = A.astype(np.float64)  # Asegurarse de que U sea float64

    # Realizar la descomposición LU
    for i in range(n):
        # Encontrar el pivote y realizar la eliminación
        for j in range(i+1, n):
            factor = U[j, i] / U[i, i]
            U[j, i:] -= factor * U[i, i:]
            L[j, i] = factor

    # Sustitución hacia adelante para resolver L * y = b
    y = np.zeros_like(b, dtype=np.float64)
    for i in range(n):
        y[i] = b[i] - np.dot(L[i, :i], y[:i])

    # Sustitución hacia atrás para resolver U * x = y
    x = np.zeros_like(y, dtype=np.float64)
    for i in range(n-1, -1, -1):
        x[i] = (y[i] - np.dot(U[i, i+1:], x[i+1:])) / U[i, i]

    return x
```

```python
# Sistema a
A_a = np.array([
    [2, -1, 1],
    [3, 3, 9],
    [3, 3, 5]
])
b_a = np.array([-1, 0, 4])

# Sistema b
A_b = np.array([
    [1.012, -2.132, 3.104],
    [-2.132, 4.096, -7.013],
    [3.104, -7.013, 0.014]
])
b_b = np.array([1.984, -5.049, -3.895])

# Sistema c
A_c = np.array([
    [2, 0, 0, 0],
    [1, 1.5, 0, 0],
    [0, -3, 0.5, 0],
    [2, -2, 1, 1]
])
b_c = np.array([3, 4.5, -6.6, 0.8])

# Sistema d
A_d = 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_d = np.array([17.102, -6.1593, 3.0004, 0])
```

```python
x_a = eliminacion_gaussiana_con_lu(A_a, b_a)
x_b = eliminacion_gaussiana_con_lu(A_b, b_b)
x_c = eliminacion_gaussiana_con_lu(A_c, b_c)
x_d = eliminacion_gaussiana_con_lu(A_d, b_d)


print("Solución del sistema a:",x_a)
print("\nSolución del sistema b:",x_b)
print("\nSolución del sistema c:",x_c)
print("\nSolución del sistema d:",x_d)
```

```
Solución del sistema a: [ 1.  2. -1.]

Solución del sistema b: [1. 1. 1.]

Solución del sistema c: [ 1.5  2.  -1.2  3. ]

Solución del sistema d: [2.9398512  0.0706777  5.67773512 4.37981223]
```