ESCUELA POLITECNICA NACIONAL

MÉTODOS NUMÉRICOS - GR1CC

Darlin Joel Anacicha Sanchez

2024-07-27

CONJUNTO DE EJERCICIOS

1. Realice las siguientes multiplicaciones matriz-matriz:

```
import numpy as np

# Ejercicio 1a
A1a = np.array([[2, -3], [3, -1], [4, 0]], dtype=np.float32)
B1a = np.array([[1, 5], [2, 0]], dtype=np.float32)
C1a = np.dot(A1a, B1a)
print(f"Resultado Ejercicio 1a:\n{C1a}")

# Ejercicio 1b
A1b = np.array([[2, -3], [3, -1], [2, 1]], dtype=np.float32)
B1b = np.array([[1, 5, -4], [0, 2, 3]], dtype=np.float32)
C1b = np.dot(A1b, B1b)
print(f"Resultado Ejercicio 1b:\n{C1b}")

# Ejercicio 1c
A1c = np.array([[2, -3], [3, -1], [4, 0]], dtype=np.float32)
B1c = np.array([[2, 3, 1], [1, 0, -1]], dtype=np.float32)
C1c = np.dot(A1c, B1c)
print(f"Resultado Ejercicio 1c:\n{C1c}")
```

```
Resultado Ejercicio 1a:
[[-4. 10.]
 [ 1. 15.]
 [ 4. 20.]]
```

```
Resultado Ejercicio 1b:
[[  2.   4. -17.]
 [  3.  13. -15.]
 [  2.  12.  -5.]]
Resultado Ejercicio 1c:
[[ 1.  6.  5.]
 [ 5.  9.  4.]
 [ 8. 12.  4.]]
```

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

```python
from src.linear_sist_methods import descomposicion_LU, resolver_LU

def lu_inverse(A):
    n = A.shape[0]
    L, U = descomposicion_LU(A)
    inv_A = np.zeros_like(A, dtype=np.float32)

    for i in range(n):
        e = np.zeros((n, 1))
        e[i] = 1
        inv_A[:, i] = resolver_LU(L, U, e).flatten()

    return inv_A

# Ejercicio 2a
A2a = np.array([[4, 2, 6], [3, 1, 7], [2, 1, 3]], dtype=np.float32)
if np.linalg.det(A2a) != 0:
    inv_A2a = lu_inverse(A2a)
    print(f"Inversa Ejercicio 2a:\n{inv_A2a}")
else:
    print("La matriz del Ejercicio 2a es singular.")

# Ejercicio 2b
A2b = np.array([[1, 2, 1], [1, 3, 4], [1, 0, -2]], dtype=np.float32)
if np.linalg.det(A2b) != 0:
    inv_A2b = lu_inverse(A2b)
    print(f"Inversa Ejercicio 2b:\n{inv_A2b}")
else:
    print("La matriz del Ejercicio 2b es singular.")
```

```python
# Ejercicio 2c
A2c = np.array([[1, 0, 1], [1, 2, 2], [2, 1, 3]], dtype=np.float32)
if np.linalg.det(A2c) != 0:
    inv_A2c = lu_inverse(A2c)
    print(f"Inversa Ejercicio 2c:\n{inv_A2c}")
else:
    print("La matriz del Ejercicio 2c es singular.")

# Ejercicio 2d
A2d = np.array([[4, 7, 0], [1, 6, 3], [2, 0, 1]], dtype=np.float32)
if np.linalg.det(A2d) != 0:
    inv_A2d = lu_inverse(A2d)
    print(f"Inversa Ejercicio 2d:\n{inv_A2d}")
else:
    print("La matriz del Ejercicio 2d es singular.")
```

```
La matriz del Ejercicio 2a es singular.
[08-15 10:25:05][INFO]
[[ 1.  2.  1.]
 [ 0.  1.  3.]
 [ 0. -2. -3.]]
[08-15 10:25:05][INFO]
[[1. 2. 1.]
 [0. 1. 3.]
 [0. 0. 3.]]
[08-15 10:25:05][INFO]
[[1. 2. 1.]
 [0. 1. 3.]
 [0. 0. 3.]]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[ 1.]
 [-1.]
 [-3.]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [-3.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [-1.]
[08-15 10:25:05][INFO] i = 0
```

```
[08-15 10:25:05][INFO] suma = [3.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [1.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[0.]
 [1.]
 [2.]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [2.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [1.]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [-1.33333333]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[0.]
 [0.]
 [1.]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [1.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [-1.66666667]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [0.]
Inversa Ejercicio 2b:
[[-2.          1.3333334   1.6666666 ]
 [ 2.         -1.         -1.        ]
 [-1.          0.6666667   0.33333334]]
[08-15 10:25:05][INFO]
[[1. 0. 1.]
 [0. 2. 1.]
 [0. 1. 1.]]
[08-15 10:25:05][INFO]
[[1.  0.  1. ]
 [0.  2.  1. ]
 [0.  0.  0.5]]
```

```
[08-15 10:25:05][INFO]
[[1.  0.  1. ]
 [0.  2.  1. ]
 [0.  0.  0.5]]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[ 1. ]
 [-1. ]
 [-1.5]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [-3.]
[08-15 10:25:05][INFO] U[i, i] = 2.0
[08-15 10:25:05][INFO] y[i] = [-1.]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [-3.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [1.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[ 0. ]
 [ 1. ]
 [-0.5]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [-1.]
[08-15 10:25:05][INFO] U[i, i] = 2.0
[08-15 10:25:05][INFO] y[i] = [1.]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [-1.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[0.]
 [0.]
 [1.]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [2.]
[08-15 10:25:05][INFO] U[i, i] = 2.0
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] i = 0
```

```
[08-15 10:25:05][INFO] suma = [2.]
[08-15 10:25:05][INFO] U[i, i] = 1.0
[08-15 10:25:05][INFO] y[i] = [0.]
Inversa Ejercicio 2c:
[[ 4.  1. -2.]
 [ 1.  1. -1.]
 [-3. -1.  2.]]
[08-15 10:25:05][INFO]
[[ 4.    7.    0.  ]
 [ 0.    4.25  3.  ]
 [ 0.   -3.5   1.  ]]
[08-15 10:25:05][INFO]
[[4.         7.         0.        ]
 [0.         4.25       3.        ]
 [0.         0.         3.47058824]]
[08-15 10:25:05][INFO]
[[4.         7.         0.        ]
 [0.         4.25       3.        ]
 [0.         0.         3.47058824]]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[ 1.        ]
 [-0.25      ]
 [-0.70588235]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [-0.61016949]
[08-15 10:25:05][INFO] U[i, i] = 4.25
[08-15 10:25:05][INFO] y[i] = [-0.25]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [0.59322034]
[08-15 10:25:05][INFO] U[i, i] = 4.0
[08-15 10:25:05][INFO] y[i] = [1.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[0.        ]
 [1.        ]
 [0.82352941]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [0.71186441]
[08-15 10:25:05][INFO] U[i, i] = 4.25
[08-15 10:25:05][INFO] y[i] = [1.]
```

```
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [0.47457627]
[08-15 10:25:05][INFO] U[i, i] = 4.0
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] Sustitución hacia adelante
[08-15 10:25:05][INFO] y =
[[0.]
 [0.]
 [1.]]
[08-15 10:25:05][INFO] Sustitución hacia atrás
[08-15 10:25:05][INFO] i = 1
[08-15 10:25:05][INFO] suma = [0.86440678]
[08-15 10:25:05][INFO] U[i, i] = 4.25
[08-15 10:25:05][INFO] y[i] = [0.]
[08-15 10:25:05][INFO] i = 0
[08-15 10:25:05][INFO] suma = [-1.42372881]
[08-15 10:25:05][INFO] U[i, i] = 4.0
[08-15 10:25:05][INFO] y[i] = [0.]
Inversa Ejercicio 2d:
[[ 0.10169491 -0.11864407  0.3559322 ]
 [ 0.08474576  0.06779661 -0.20338982]
 [-0.20338982  0.23728813  0.2881356 ]]
```

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

```python
# Ejercicio 3a
A3 = np.array([[1, -2, 2, -3], [1, 1, -3, 4], [2, 1, -3, -4], [1, 2, 2, 1]], dtype=np.floa
b3a = np.array([6, 4, -2, 1], dtype=np.float32)
L3, U3 = descomposicion_LU(A3)
sol3a = resolver_LU(L3, U3, b3a)
print(f"Solución Ejercicio 3a: {sol3a}")

# Ejercicio 3b
b3b = np.array([1, 4, -2, 2], dtype=np.float32)
sol3b = resolver_LU(L3, U3, b3b)
print(f"Solución Ejercicio 3b: {sol3b}")

# Ejercicio 3c
b3c = np.array([1, 4, -2, 1], dtype=np.float32)
sol3c = resolver_LU(L3, U3, b3c)
print(f"Solución Ejercicio 3c: {sol3c}")
```

```python
# Ejercicio 3d
b3d = np.array([1, 4, -2, -2], dtype=np.float32)
sol3d = resolver_LU(L3, U3, b3d)
print(f"Solución Ejercicio 3d: {sol3d}")
```

```
[08-15 10:25:08][INFO]
[[ 1. -2.  2. -3.]
 [ 0.  3. -5.  7.]
 [ 0.  5. -7.  2.]
 [ 0.  4.  0.  4.]]
[08-15 10:25:08][INFO]
[[ 1.         -2.          2.         -3.        ]
 [ 0.          3.         -5.          7.        ]
 [ 0.          0.          1.33333333 -9.66666667]
 [ 0.          0.          6.66666667 -5.33333333]]
[08-15 10:25:08][INFO]
[[ 1.         -2.          2.         -3.        ]
 [ 0.          3.         -5.          7.        ]
 [ 0.          0.          1.33333333 -9.66666667]
 [ 0.          0.          0.         43.        ]]
[08-15 10:25:08][INFO]
[[ 1.         -2.          2.         -3.        ]
 [ 0.          3.         -5.          7.        ]
 [ 0.          0.          1.33333333 -9.66666667]
 [ 0.          0.          0.         43.        ]]
[08-15 10:25:08][INFO] Sustitución hacia adelante
[08-15 10:25:08][INFO] y =
[[  6.        ]
 [ -2.        ]
 [-10.66666667]
 [ 51.        ]]
[08-15 10:25:08][INFO] Sustitución hacia atrás
[08-15 10:25:08][INFO] i = 2
[08-15 10:25:08][INFO] suma = [-11.46511628]
[08-15 10:25:08][INFO] U[i, i] = 1.3333333333333334
[08-15 10:25:08][INFO] y[i] = [-10.66666667]
[08-15 10:25:08][INFO] i = 1
[08-15 10:25:08][INFO] suma = [5.30813953]
[08-15 10:25:08][INFO] U[i, i] = 3.0
[08-15 10:25:08][INFO] y[i] = [-2.]
[08-15 10:25:08][INFO] i = 0
[08-15 10:25:08][INFO] suma = [2.51162791]
```

```
[08-15 10:25:08][INFO] U[i, i] = 1.0
[08-15 10:25:08][INFO] y[i] = [6.]
Solución Ejercicio 3a: [[ 3.48837209]
 [-2.43604651]
 [ 0.59883721]
 [ 1.18604651]]
[08-15 10:25:08][INFO] Sustitución hacia adelante
[08-15 10:25:08][INFO] y =
[[ 1.]
 [ 3.]
 [-9.]
 [42.]]
[08-15 10:25:08][INFO] Sustitución hacia atrás
[08-15 10:25:08][INFO] i = 2
[08-15 10:25:08][INFO] suma = [-9.44186047]
[08-15 10:25:08][INFO] U[i, i] = 1.3333333333333334
[08-15 10:25:08][INFO] y[i] = [-9.]
[08-15 10:25:08][INFO] i = 1
[08-15 10:25:08][INFO] suma = [5.18023256]
[08-15 10:25:08][INFO] U[i, i] = 3.0
[08-15 10:25:08][INFO] y[i] = [3.]
[08-15 10:25:08][INFO] i = 0
[08-15 10:25:08][INFO] suma = [-0.81395349]
[08-15 10:25:08][INFO] U[i, i] = 1.0
[08-15 10:25:08][INFO] y[i] = [1.]
Solución Ejercicio 3b: [[ 1.81395349]
 [-0.72674419]
 [ 0.33139535]
 [ 0.97674419]]
[08-15 10:25:08][INFO] Sustitución hacia adelante
[08-15 10:25:08][INFO] y =
[[ 1.]
 [ 3.]
 [-9.]
 [41.]]
[08-15 10:25:08][INFO] Sustitución hacia atrás
[08-15 10:25:08][INFO] i = 2
[08-15 10:25:08][INFO] suma = [-9.21705426]
[08-15 10:25:08][INFO] U[i, i] = 1.3333333333333334
[08-15 10:25:08][INFO] y[i] = [-9.]
[08-15 10:25:08][INFO] i = 1
[08-15 10:25:08][INFO] suma = [5.86046512]
[08-15 10:25:08][INFO] U[i, i] = 3.0
```

```
[08-15 10:25:08][INFO] y[i] = [3.]
[08-15 10:25:08][INFO] i = 0
[08-15 10:25:08][INFO] suma = [-0.62790698]
[08-15 10:25:08][INFO] U[i, i] = 1.0
[08-15 10:25:08][INFO] y[i] = [1.]
Solución Ejercicio 3c: [[ 1.62790698]
 [-0.95348837]
 [ 0.1627907 ]
 [ 0.95348837]]
[08-15 10:25:08][INFO] Sustitución hacia adelante
[08-15 10:25:08][INFO] y =
[[ 1.]
 [ 3.]
 [-9.]
 [38.]]
[08-15 10:25:08][INFO] Sustitución hacia atrás
[08-15 10:25:08][INFO] i = 2
[08-15 10:25:08][INFO] suma = [-8.54263566]
[08-15 10:25:08][INFO] U[i, i] = 1.3333333333333334
[08-15 10:25:08][INFO] y[i] = [-9.]
[08-15 10:25:08][INFO] i = 1
[08-15 10:25:08][INFO] suma = [7.90116279]
[08-15 10:25:08][INFO] U[i, i] = 3.0
[08-15 10:25:08][INFO] y[i] = [3.]
[08-15 10:25:08][INFO] i = 0
[08-15 10:25:08][INFO] suma = [-0.06976744]
[08-15 10:25:08][INFO] U[i, i] = 1.0
[08-15 10:25:08][INFO] y[i] = [1.]
Solución Ejercicio 3d: [[ 1.06976744]
 [-1.63372093]
 [-0.34302326]
 [ 0.88372093]]
```

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

```python
import sympy as sp

# Ejercicio 4
alpha = sp.symbols('alpha')
A4 = sp.Matrix([[1, -1, alpha], [2, 2, 0], [0, alpha, -1/2]])

# Determinante
```

```
det_A4 = A4.det()
print(f"Determinante de la matriz A en términos de  : {det_A4}")

# Resolver para  donde el determinante es 0
alpha_values = sp.solve(det_A4, alpha)
print(f"Valores de  que hacen la matriz singular: {alpha_values}")
```

```
Determinante de la matriz A en términos de  : 2*alpha**2 - 2.0
Valores de  que hacen la matriz singular: [-1.00000000000000, 1.00000000000000]
```

5. Resuelva los siguientes sistemas lineales:

```
import numpy as np
from src.linear_sist_methods import matriz_aumentada, eliminacion_gaussiana

def resolver_sistema_eliminacion_gaussiana(A, b):
    Ab = matriz_aumentada(A, b)
    sol = eliminacion_gaussiana(Ab)
    return sol

# Ejercicio 5a
A5a = np.array([[1, 0, 0], [2, 1, 0], [-1, 0, 1]], dtype=np.float32)
b5a = np.array([2, -1, 0], dtype=np.float32)
sol5a = resolver_sistema_eliminacion_gaussiana(A5a, b5a)
print(f"Solución Ejercicio 5a:\n{sol5a}")

# Ejercicio 5b
A5b = np.array([[2, 0, 0], [-1, 1, 0], [3, -2, 1]], dtype=np.float32)
b5b = np.array([-1, 3, 0], dtype=np.float32)
sol5b = resolver_sistema_eliminacion_gaussiana(A5b, b5b)
print(f"Solución Ejercicio 5b:\n{sol5b}")
```

```
[08-15 10:25:10][INFO]
[[ 1.  0.  0.  2.]
 [ 0.  1.  0. -5.]
 [ 0.  0.  1.  2.]]
[08-15 10:25:10][INFO]
[[ 1.  0.  0.  2.]
 [ 0.  1.  0. -5.]
 [ 0.  0.  1.  2.]]
```

```
Solución Ejercicio 5a:
[ 2. -5.  2.]
[08-15 10:25:10][INFO]
[[-1.  1.  0.  3.]
 [ 0.  2.  0.  5.]
 [ 0.  1.  1.  9.]]
[08-15 10:25:10][INFO]
[[ -1.   1.   0.   3.]
 [  0.   1.   1.   9.]
 [  0.   0.  -2. -13.]]
Solución Ejercicio 5b:
[-0.5  2.5  6.5]
```

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

```python
import numpy as np
from src.linear_sist_methods import descomposicion_LU

def print_large_matrix(matrix, name):
    print(name)
    rows, cols = matrix.shape
    for row in range(rows):
        print(matrix[row, :])

# Ejercicio 6a
A6a = np.array([[2, -1, 1], [3, 3, 9], [3, 3, 5]], dtype=np.float32)
L6a, U6a = descomposicion_LU(A6a)
print_large_matrix(L6a, "Matriz L (Ejercicio 6a):")
print_large_matrix(U6a, "Matriz U (Ejercicio 6a):")

# Ejercicio 6b
A6b = np.array([[1.012, -2.132, 3.104], [-2.132, 4.096, -7.013], [3.104, -7.013, 0.014]],
L6b, U6b = descomposicion_LU(A6b)
print_large_matrix(L6b, "Matriz L (Ejercicio 6b):")
print_large_matrix(U6b, "Matriz U (Ejercicio 6b):")

# Ejercicio 6c
A6c = np.array([[2, 0, 0], [1, 1.5, 0], [2, -2, 1]], dtype=np.float32)
L6c, U6c = descomposicion_LU(A6c)
print_large_matrix(L6c, "Matriz L (Ejercicio 6c):")
print_large_matrix(U6c, "Matriz U (Ejercicio 6c):")
```

```
# Ejercicio 6d
A6d = np.array([[2.1756, 4.0231, -2.1732, 5.1967], [-4.0231, 6.0000, 0, 1.1973], [-1.0000,
L6d, U6d = descomposicion_LU(A6d)
print_large_matrix(L6d, "Matriz L (Ejercicio 6d):")
print_large_matrix(U6d, "Matriz U (Ejercicio 6d):")
```

```
[08-15 10:25:11][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   4.5  3.5]]
[08-15 10:25:11][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
[08-15 10:25:11][INFO]
[[ 2.  -1.   1. ]
 [ 0.   4.5  7.5]
 [ 0.   0.  -4. ]]
Matriz L (Ejercicio 6a):
[1. 0. 0.]
[1.5 1.  0. ]
[1.5 1.  1. ]
Matriz U (Ejercicio 6a):
[ 2. -1.  1.]
[0.  4.5 7.5]
[ 0.  0. -4.]
[08-15 10:25:11][INFO]
[[ 1.01199996 -2.13199997  3.10400009]
 [ 0.         -0.39552553 -0.47374277]
 [ 0.         -0.47374277 -9.50657006]]
[08-15 10:25:11][INFO]
[[ 1.01199996e+00 -2.13199997e+00  3.10400009e+00]
 [ 0.00000000e+00 -3.95525525e-01 -4.73742767e-01]
 [ 0.00000000e+00 -5.55111512e-17 -8.93914219e+00]]
[08-15 10:25:11][INFO]
[[ 1.01199996e+00 -2.13199997e+00  3.10400009e+00]
 [ 0.00000000e+00 -3.95525525e-01 -4.73742767e-01]
 [ 0.00000000e+00 -5.55111512e-17 -8.93914219e+00]]
Matriz L (Ejercicio 6b):
[1. 0. 0.]
[-2.10671941  1.          0.        ]
```

```
[3.06719387 1.19775523 1.        ]
Matriz U (Ejercicio 6b):
[ 1.01199996 -2.13199997  3.10400009]
[ 0.         -0.39552553 -0.47374277]
[ 0.00000000e+00 -5.55111512e-17 -8.93914219e+00]
[08-15 10:25:11][INFO]
[[ 2.   0.   0. ]
 [ 0.   1.5  0. ]
 [ 0.  -2.   1. ]]
[08-15 10:25:11][INFO]
[[2.  0.  0. ]
 [0.  1.5 0. ]
 [0.  0.  1. ]]
[08-15 10:25:11][INFO]
[[2.  0.  0. ]
 [0.  1.5 0. ]
 [0.  0.  1. ]]
Matriz L (Ejercicio 6c):
[1. 0. 0.]
[0.5 1.  0. ]
[ 1.         -1.33333333  1.        ]
Matriz U (Ejercicio 6c):
[2. 0. 0.]
[0.  1.5 0. ]
[0. 0. 1.]
[08-15 10:25:11][INFO]
[[  2.17560005   4.0230999   -2.17319989   5.1967001 ]
 [  0.          13.43947987  -4.01866155  10.80699068]
 [  0.          -3.3615091    0.11220318   2.58592841]
 [  0.          -4.50843938   6.2166339  -19.02172818]]
[08-15 10:25:11][INFO]
[[ 2.17560005e+00  4.02309990e+00 -2.17319989e+00  5.19670010e+00]
 [ 0.00000000e+00  1.34394799e+01 -4.01866155e+00  1.08069907e+01]
 [ 0.00000000e+00 -4.44089210e-16 -8.92952342e-01  5.28899414e+00]
 [ 0.00000000e+00  0.00000000e+00  4.86852429e+00 -1.53963898e+01]]
[08-15 10:25:11][INFO]
[[ 2.17560005e+00  4.02309990e+00 -2.17319989e+00  5.19670010e+00]
 [ 0.00000000e+00  1.34394799e+01 -4.01866155e+00  1.08069907e+01]
 [ 0.00000000e+00 -4.44089210e-16 -8.92952342e-01  5.28899414e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.34400836e+01]]
[08-15 10:25:11][INFO]
[[ 2.17560005e+00  4.02309990e+00 -2.17319989e+00  5.19670010e+00]
 [ 0.00000000e+00  1.34394799e+01 -4.01866155e+00  1.08069907e+01]
 [ 0.00000000e+00 -4.44089210e-16 -8.92952342e-01  5.28899414e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.34400836e+01]]
Matriz L (Ejercicio 6d):
[1. 0. 0. 0.]
[-1.84919094  1.          0.          0.        ]
[-0.45964331 -0.25012196  1.          0.        ]
[ 2.86059001 -0.33546234 -5.45216588  1.        ]
Matriz U (Ejercicio 6d):
[ 2.17560005  4.0230999  -2.17319989  5.1967001 ]
[ 0.         13.43947987 -4.01866155 10.80699068]
[ 0.00000000e+00 -4.44089210e-16 -8.92952342e-01  5.28899414e+00]
[ 0.          0.          0.         13.44008364]
```

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.

```python
import numpy as np

def descomposicion_LU(A):
    n = len(A)
    L = np.zeros((n, n))
    U = np.zeros((n, n))

    for i in range(n):
        L[i][i] = 1  # Matriz L tiene 1s en la diagonal principal

        for j in range(i, n):
            sum_u = sum(L[i][k] * U[k][j] for k in range(i))
            U[i][j] = A[i][j] - sum_u

        for j in range(i+1, n):
            sum_l = sum(L[j][k] * U[k][i] for k in range(i))
            L[j][i] = (A[j][i] - sum_l) / U[i][i]

    return L, U

def resolver_LU(A, b):
    L, U = descomposicion_LU(A)

    # Sustitución hacia adelante para resolver Ly = b
    n = len(b)
    y = np.zeros(n)
    for i in range(n):
        y[i] = b[i] - sum(L[i][j] * y[j] for j in range(i))

    # Sustitución hacia atrás para resolver Ux = y
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (y[i] - sum(U[i][j] * x[j] for j in range(i+1, n))) / U[i][i]

    return x

# Ejercicio 7a
A7a = np.array([[2, -1, 1], [3, 3, 9], [3, 3, 5]], dtype=np.float32)
b7a = np.array([-1, 0, 4], dtype=np.float32)
x7a = resolver_LU(A7a, b7a)
print(f"Solución Ejercicio 7a: {x7a}")

# Ejercicio 7b
A7b = np.array([[1.012, -2.132, 3.104], [-2.132, 4.096, -7.013], [3.104, -7.013, 0.014]],
b7b = np.array([1.984, -5.049, -3.895], dtype=np.float32)
x7b = resolver_LU(A7b, b7b)
print(f"Solución Ejercicio 7b: {x7b}")

# Ejercicio 7c
A7c = np.array([[2, 0, 0, 0], [1, 1.5, 0, 0], [0, -3, 0.5, 0], [2, -2, 1, 1]], dtype=np.fl
b7c = np.array([3, 4.5, -6.6, 0.8], dtype=np.float32)
x7c = resolver_LU(A7c, b7c)
print(f"Solución Ejercicio 7c: {x7c}")

# Ejercicio 7d
A7d = np.array([[2.1756, 4.0231, -2.1732, 5.1967], [-4.0231, 6.0000, 0, 1.1973], [-1.0000,
b7d = np.array([17.102, -6.1593, 3.0004, 0], dtype=np.float32)
x7d = resolver_LU(A7d, b7d)
print(f"Solución Ejercicio 7d: {x7d}")
```

```
Solución Ejercicio 7a: [ 1.  2. -1.]
Solución Ejercicio 7b: [1.0000016  1.00000072 0.99999993]
Solución Ejercicio 7c: [ 1.5         2.         -1.19999981  2.99999982]
Solución Ejercicio 7d: [ 2.68543558 -0.02123014  4.31002601  3.98551468]
```