ESCUELA POLITECNICA NACIONAL

MÉTODOS NUMÉRICOS - GR1CC

Darlin Joel Anacicha Sanchez

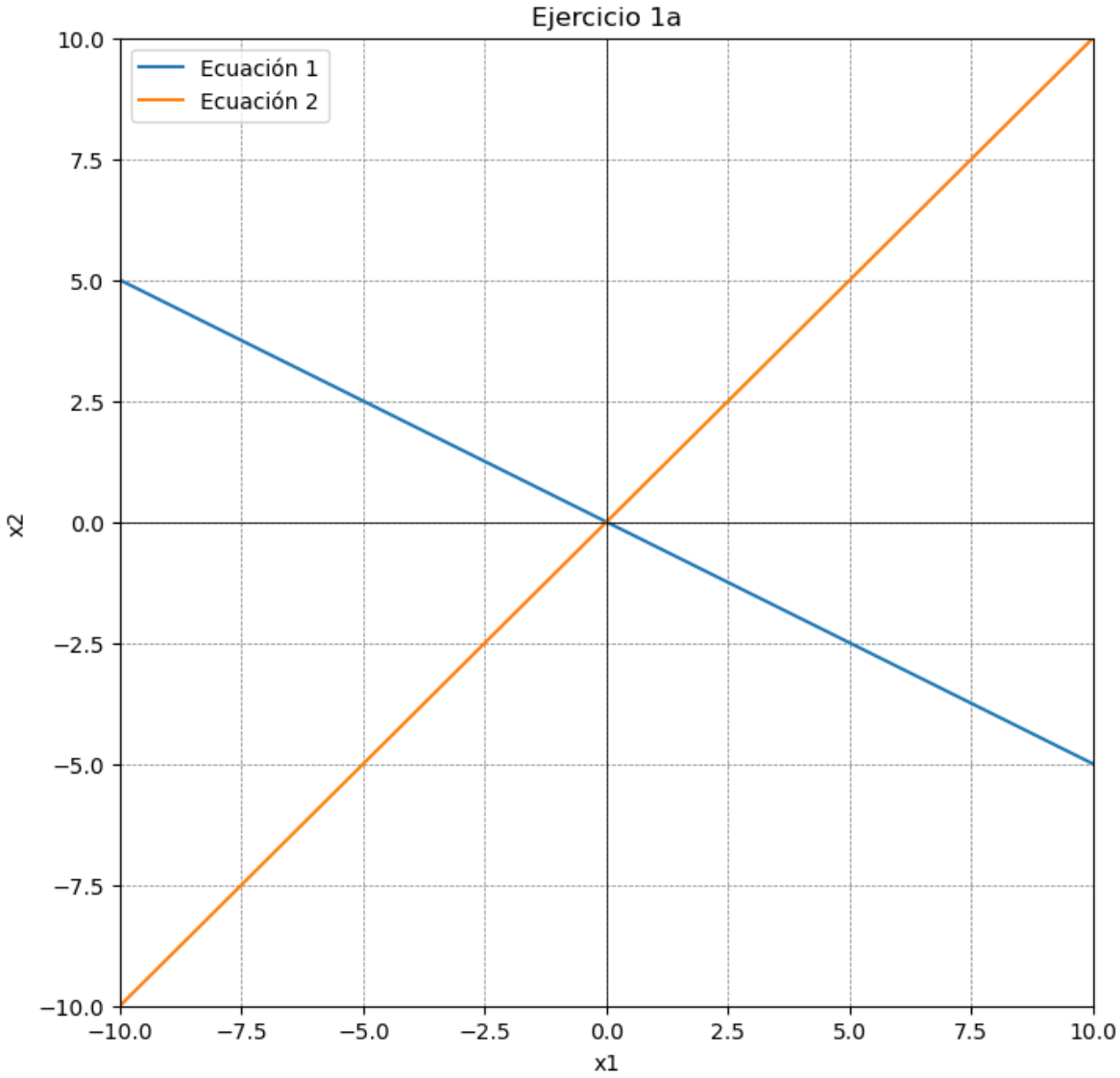
2024-07-27

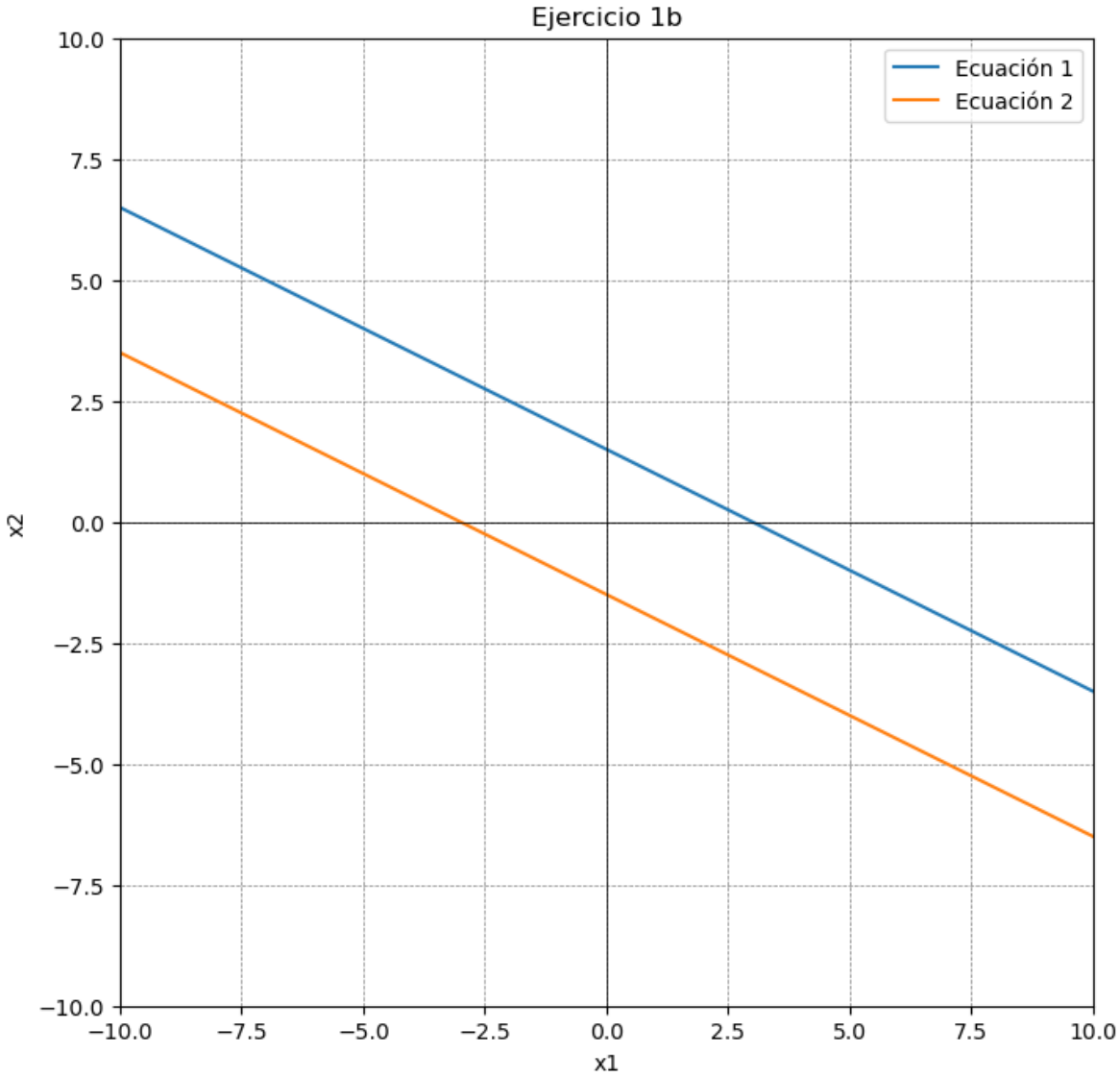
Ejercios Propuestos

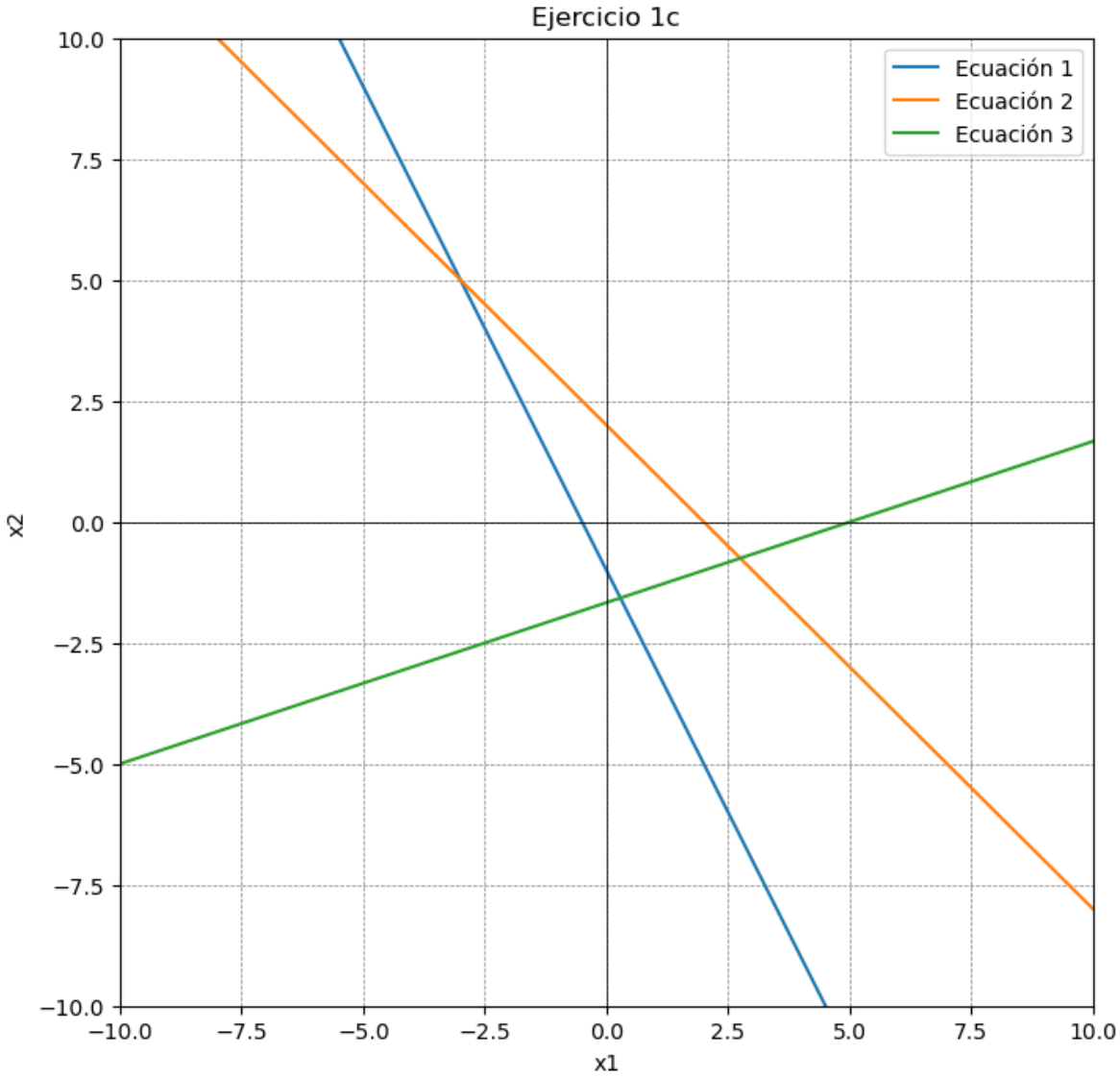
1. Para cada uno de los siguientes sistemas lineales, obtenga, de ser posible, una solución con métodos gráficos. Explique los resultados desde un punto de vista geométrico.

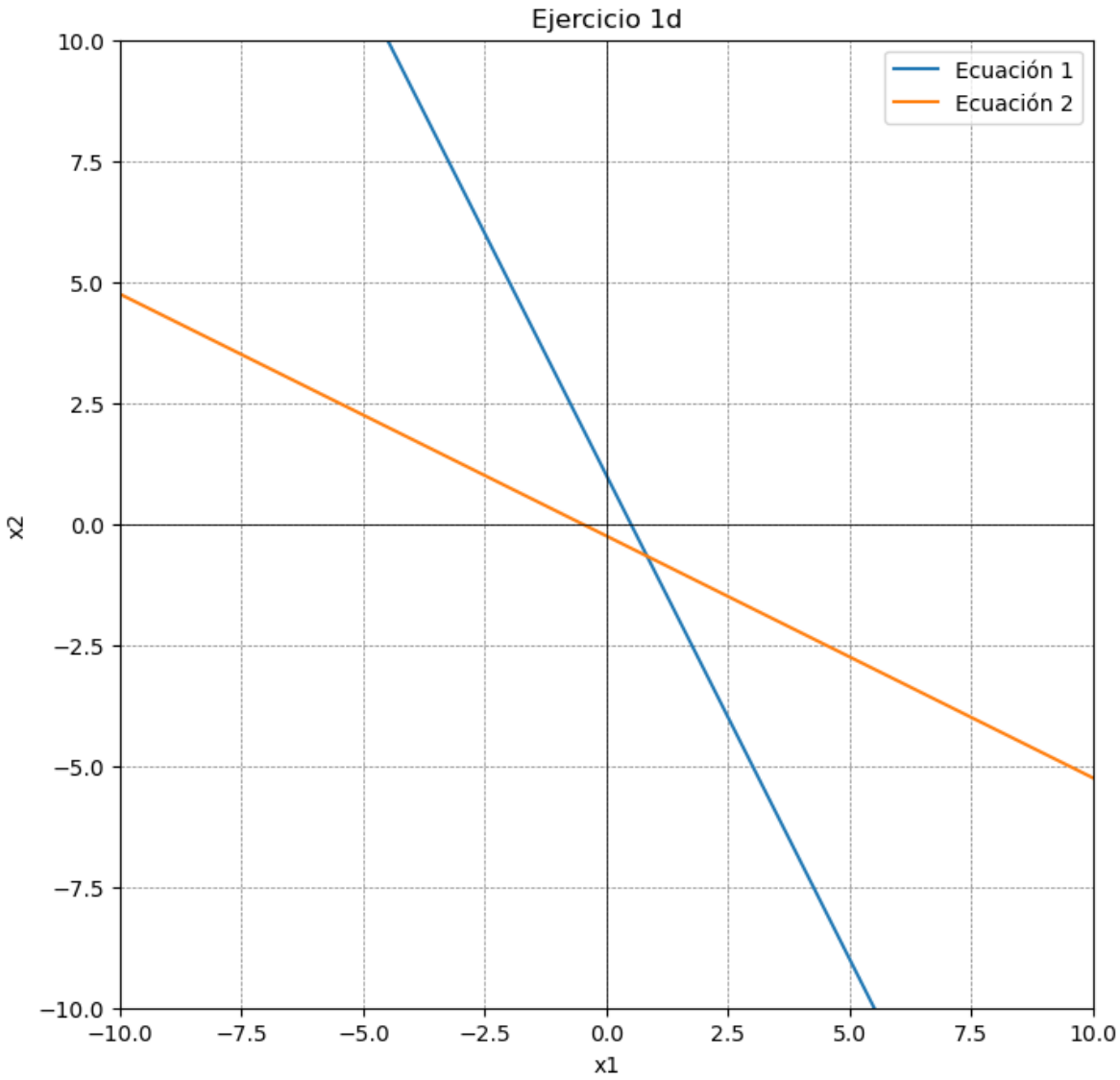
```
import numpy as np
import matplotlib.pyplot as plt

def graficar_sistema(A, b, titulo):
    """
    Grafica un sistema de ecuaciones lineales de la forma Ax = b.
    A: matriz de coeficientes.
    b: vector de términos independientes.
    titulo: título del gráfico.
    """
    x_vals = np.linspace(-10, 10, 400)
    plt.figure(figsize=(8, 8))

    for i in range(A.shape[0]):
        if A[i, 1] != 0:  # Si el coeficiente de x2 no es cero
            y_vals = (b[i] - A[i, 0] * x_vals) / A[i, 1]
            plt.plot(x_vals, y_vals, label=f'Ecuación {i+1}')
        else:  # Si el coeficiente de x2 es cero (x1 = b)
            plt.axvline(x=b[i]/A[i, 0], label=f'Ecuación {i+1}')

    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)
```

```
    plt.grid(color = 'gray', linestyle = '--', linewidth = 0.5)
    plt.legend()
    plt.title(titulo)
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.xlim(-10, 10)
    plt.ylim(-10, 10)
    plt.show()

# Ejercicio 1a
A1a = np.array([
    [1, 2],
    [1, -1]
], dtype=float)
b1a = np.array([0, 0], dtype=float)
graficar_sistema(A1a, b1a, 'Ejercicio 1a')

# Ejercicio 1b
A1b = np.array([
    [1, 2],
    [-2, -4]
], dtype=float)
b1b = np.array([3, 6], dtype=float)
graficar_sistema(A1b, b1b, 'Ejercicio 1b')

# Ejercicio 1c
A1c = np.array([
    [2, 1],
    [1, 1],
    [1, -3]
], dtype=float)
b1c = np.array([-1, 2, 5], dtype=float)
graficar_sistema(A1c, b1c, 'Ejercicio 1c')

# Ejercicio 1d
A1d = np.array([
    [2, 1, 1],
    [2, 4, -1]
], dtype=float)
b1d = np.array([1, -1], dtype=float)
graficar_sistema(A1d, b1d, 'Ejercicio 1d')
```


Ejercicio 1a
Ecuación 1
Ecuación 2
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
x2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x1


Ejercicio 1b
Ecuación 1
Ecuación 2
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
x2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x1


Ejercicio 1c
Ecuación 1
Ecuación 2
Ecuación 3
x1
x2
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



Conclusión: -Ejercicio 1a: Tiene una única solución en el origen (0,0).

-Ejercicio 1b: Tiene infinitas soluciones porque las ecuaciones son dependientes.

-Ejercicio 1c: No tiene solución porque las líneas no se intersectan en un único punto común.

-Ejercicio 1d: Las ecuaciones representan planos en 3D. Se necesitan más detalles para visualizar y determinar la solución exacta, pero en 2D, su representación no es posible.

2. Utilice la eliminación gaussiana con sustitución hacia atrás y aritmética de redondeo de dos dígitos para resolver los siguientes sistemas lineales. No reordene

las ecuaciones. (La solución exacta para cada sistema es $1 = -1, 2 = 2, 3 = 3$.)

3. Utilice el algoritmo de eliminación gaussiana para resolver, de ser posible, los siguientes sistemas lineales, y determine si se necesitan intercambios de fila:

```
%reload_ext autoreload
%autoreload 2

import sys
import os
import numpy as np

# Añadir el directorio 'src' al sys.path
sys.path.append(os.path.abspath('./src'))

from linear_sist_methods import eliminacion_gaussiana, matriz_aumentada

def verificar_determinante(A):
    det = np.linalg.det(A)
    if det == 0:
        raise ValueError("La matriz de coeficientes es singular. No existe una solución ún
    else:
        print(f"Determinante: {det}")

def redondeo_dos_digitos(matrix):
    """Redondea los elementos de una matriz a dos dígitos."""
    return np.round(matrix, decimals=2)

# Ejercicio 2a
A2a = np.array([
    [3, 4, 1],
    [5, 2, 3],
    [3, 1, 4]
], dtype=float)
b2a = np.array([8, 1, 11], dtype=float)

# Ejercicio 2b
A2b = np.array([
    [4, 1, -2],
    [1, 5, 1],
    [9, 7, 3]
], dtype=float)
```

```
b2b = np.array([-5, -1, -1], dtype=float)

# Ejercicio 2c
A2c = np.array([
    [1, 1, 3],
    [1, 4, 4],
    [4, 2, 3]
], dtype=float)
b2c = np.array([-1, 3, 9], dtype=float)

# Ejercicio 3a
A3a = np.array([
    [1, -2, 3],
    [3, 3, 2],
    [1, 1, 2]
], dtype=float)
b3a = np.array([2, -1, 3], dtype=float)

# Ejercicio 3b
A3b = np.array([
    [2, -1.5, 3],
    [-1, 2, 3],
    [4, 4.5, 5]
], dtype=float)
b3b = np.array([1, 3, 1], dtype=float)

# Ejercicio 3c
A3c = np.array([
    [2, 0, 0],
    [1.15, -3, 0.5],
    [3.02, 0.53, 0]
], dtype=float)
b3c = np.array([3, 4.5, -6.6], dtype=float)

# Ejercicio 3d
A3d = np.array([
    [1, 1, 2, 0],
    [2, 1, 4, 0],
    [3, 1, 1, 0],
    [2, -1, -3, 0]
], dtype=float)
```

```
b3d = np.array([2, 6, 5, 3], dtype=float)

# Aplicando redondeo a dos dígitos
A2a = redondeo_dos_digitos(A2a)
b2a = redondeo_dos_digitos(b2a)

A2b = redondeo_dos_digitos(A2b)
b2b = redondeo_dos_digitos(b2b)

A2c = redondeo_dos_digitos(A2c)
b2c = redondeo_dos_digitos(b2c)

A3a = redondeo_dos_digitos(A3a)
b3a = redondeo_dos_digitos(b3a)

A3b = redondeo_dos_digitos(A3b)
b3b = redondeo_dos_digitos(b3b)

A3c = redondeo_dos_digitos(A3c)
b3c = redondeo_dos_digitos(b3c)

A3d = redondeo_dos_digitos(A3d)
b3d = redondeo_dos_digitos(b3d)

# Resolviendo ejercicios y mostrando resultados
ejercicios = [
    ('Ejercicio 2a', A2a, b2a),
    ('Ejercicio 2b', A2b, b2b),
    ('Ejercicio 2c', A2c, b2c),
    ('Ejercicio 3a', A3a, b3a),
    ('Ejercicio 3b', A3b, b3b),
    ('Ejercicio 3c', A3c, b3c),
    ('Ejercicio 3d', A3d, b3d),
]

for nombre, A, b in ejercicios:
    print(f"\nSolución {nombre} - Eliminación Gaussiana:")
    try:
        verificar_determinante(A)
        solucion = eliminacion_gaussiana(matriz_aumentada(A, b))
        print(f"Solución: {solucion}")
```

```
    except ValueError as e:
        print(e)
```

```
Solución Ejercicio 2a - Eliminación Gaussiana:
Determinante: -29.99999999999999
[07-27 22:14:40][INFO]
[[  3.           4.           1.           8.        ]
 [  0.          -4.66666667   1.33333333 -12.33333333]
 [  0.          -3.           3.           3.        ]]
[07-27 22:14:40][INFO]
[[  3.           4.           1.           8.        ]
 [  0.          -3.           3.           3.        ]
 [  0.           0.          -3.33333333 -17.        ]]
Solución: [-4.5  4.1  5.1]

Solución Ejercicio 2b - Eliminación Gaussiana:
Determinante: 113.99999999999993
[07-27 22:14:40][INFO]
[[  1.   5.   1.  -1.]
 [  0. -19.  -6.  -1.]
 [  0. -38.  -6.   8.]]
[07-27 22:14:40][INFO]
[[  1.   5.   1.  -1.]
 [  0. -19.  -6.  -1.]
 [  0.   0.   6.  10.]]
Solución: [-0.29824561 -0.47368421  1.66666667]

Solución Ejercicio 2c - Eliminación Gaussiana:
Determinante: -25.000000000000007
[07-27 22:14:40][INFO]
[[ 1.  1.  3. -1.]
 [ 0.  3.  1.  4.]
 [ 0. -2. -9. 13.]]
[07-27 22:14:40][INFO]
[[  1.    1.    3.   -1. ]
 [  0.   -2.   -9.   13. ]
 [  0.    0.  -12.5  23.5]]
Solución: [ 2.68  1.96 -1.88]

Solución Ejercicio 3a - Eliminación Gaussiana:
Determinante: 12.000000000000005
```

```
[07-27 22:14:40][INFO]
[[ 1. -2.  3.  2.]
 [ 0.  9. -7. -7.]
 [ 0.  3. -1.  1.]]
[07-27 22:14:40][INFO]
[[  1.  -2.   3.   2.]
 [  0.   3.  -1.   1.]
 [  0.   0.  -4. -10.]]
Solución: [-3.16666667  1.16666667  2.5       ]

Solución Ejercicio 3b - Eliminación Gaussiana:
Determinante: -70.00000000000003
[07-27 22:14:40][INFO]
[[-1.   2.   3.   3. ]
 [ 0.   2.5  9.   7. ]
 [ 0.  12.5 17.  13. ]]
[07-27 22:14:40][INFO]
[[ -1.    2.    3.    3. ]
 [  0.    2.5   9.    7. ]
 [  0.    0.  -28.  -22. ]]
Solución: [-0.7        -0.02857143  0.78571429]

Solución Ejercicio 3c - Eliminación Gaussiana:
Determinante: -0.53
[07-27 22:14:40][INFO]
[[  1.15        -3.           0.5          4.5       ]
 [  0.           5.2173913   -0.86956522  -4.82608696]
 [  0.           8.40826087  -1.31304348 -18.4173913 ]]
[07-27 22:14:40][INFO]
[[  1.15        -3.           0.5          4.5       ]
 [  0.           5.2173913   -0.86956522  -4.82608696]
 [  0.           0.           0.08833333 -10.63975   ]]
Solución: [   1.5   -21.   -120.45]

Solución Ejercicio 3d - Eliminación Gaussiana:
La matriz de coeficientes es singular. No existe una solución única.
```

4. Use el algoritmo de eliminación gaussiana y la aritmética computacional de precisión de 32 bits para resolver los siguientes sistemas lineales.

5. Dado el sistema lineal:

a. Encuentre el valor(es) de para los que el sistema no tiene soluciones.

b. Encuentre el valor(es) de para los que el sistema tiene un número infinito de soluciones.

c. Suponga que existe una única solución para una a determinada, encuentre la solución.

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

def verificar_determinante(A):
    det = np.linalg.det(A)
    if det == 0:
        raise ValueError("La matriz de coeficientes es singular. No existe una solución ún
    else:
        print(f"Determinante: {det}")
    return det

# Ejercicio 4a
A4a = np.array([
    [1/3, 1/2, 1/4],
    [1/3, 1/3, 1/3],
    [1/2, 1/4, 1/3]
], dtype=np.float32)
b4a = np.array([9, 8, 8], dtype=np.float32)

# Ejercicio 4b
A4b = np.array([
    [3.333, 15920, -10.333],
    [2.222, 16.71, 9.612],
    [1.5611, 5.1791, 1.6852]
], dtype=np.float32)
b4b = np.array([15913, 28.544, 8.4254], dtype=np.float32)

# Ejercicio 4c
A4c = np.array([
    [1, 1/2, 1/3, 1/4],
    [1/2, 1/3, 1/4, 1/5],
    [1/3, 1/4, 1/5, 1/6],
    [1/4, 1/5, 1/6, 1/7]
], dtype=np.float32)
b4c = np.array([1, 1/2, 1/3, 1/4], dtype=np.float32)

# Ejercicio 4d
A4d = np.array([
```

```
    [2, 1, -2, -3, 7],
    [1, 2, -4, 4, 5],
    [-2, -3, 4, -5, -5],
    [3, 1, -2, 4, -5],
    [1, -1, 1, -1, 3]
], dtype=np.float32)
b4d = np.array([1, 2, -5, 6, -3], dtype=np.float32)

# Resolviendo ejercicio 4
print("Solución ejercicio 4a - Eliminación Gaussiana:")
verificar_determinante(A4a)
print(eliminacion_gaussiana(matriz_aumentada(A4a, b4a)))

print("Solución ejercicio 4b - Eliminación Gaussiana:")
verificar_determinante(A4b)
print(eliminacion_gaussiana(matriz_aumentada(A4b, b4b)))

print("Solución ejercicio 4c - Eliminación Gaussiana:")
verificar_determinante(A4c)
print(eliminacion_gaussiana(matriz_aumentada(A4c, b4c)))

print("Solución ejercicio 4d - Eliminación Gaussiana:")
verificar_determinante(A4d)
print(eliminacion_gaussiana(matriz_aumentada(A4d, b4d)))

# Ejercicio 5
def resolver_ejercicio_5(alpha):
    A5 = np.array([
        [1, -1, -alpha],
        [-1, 2, -3],
        [alpha, 1, 1]
    ], dtype=np.float32)
    b5 = np.array([-2, 3, 2], dtype=np.float32)

    print(f"\nResolviendo para   = {alpha}")
    try:
        det = verificar_determinante(A5)
        if det != 0:
            solucion = eliminacion_gaussiana(matriz_aumentada(A5, b5))
            print(solucion)
        else:
```

```
            print("El sistema tiene infinitas soluciones o no tiene solución.")
    except ValueError as e:
        print(e)

print("\nSolución ejercicio 5:")

# Parte a: Valores de  para los que el sistema no tiene soluciones
print("\nParte a:")
for alpha in np.linspace(-5, 5, 11):
    resolver_ejercicio_5(alpha)

# Parte b: Valores de  para los que el sistema tiene un número infinito de soluciones
print("\nParte b:")
# Se puede encontrar el rango para alfa donde el determinante es cero
for alpha in np.linspace(-5, 5, 11):
    A5 = np.array([
        [1, -1, -alpha],
        [-1, 2, -3],
        [alpha, 1, 1]
    ], dtype=np.float32)
    det = np.linalg.det(A5)
    if det == 0:
        print(f"  = {alpha} hace que el sistema tenga un número infinito de soluciones")

# Parte c: Una solución para un valor específico de
print("\nParte c:")
resolver_ejercicio_5(1)  # Ejemplo con  = 1
```

```
Solución ejercicio 4a - Eliminación Gaussiana:
Determinante: 0.016203703358769417
[07-27 22:14:41][INFO]
[[ 0.33333334  0.5         0.25        9.        ]
 [ 0.         -0.16666666  0.08333334 -1.        ]
 [ 0.         -0.5        -0.04166666 -5.5       ]]
[07-27 22:14:41][INFO]
[[ 0.33333334  0.5         0.25        9.        ]
 [ 0.         -0.16666666  0.08333334 -1.        ]
 [ 0.          0.         -0.29166672 -2.4999998 ]]
[ 5.14285857 10.28571432  8.57142639]
Solución ejercicio 4b - Eliminación Gaussiana:
Determinante: 179350.1875
[07-27 22:14:41][INFO]
```

```
[[ 1.5611000e+00  5.1791000e+00  1.6852000e+00  8.4253998e+00]
 [ 0.0000000e+00  9.3383007e+00  7.2133622e+00  1.6551662e+01]
 [ 0.0000000e+00  1.5908942e+04 -1.3930958e+01  1.5895012e+04]]
[07-27 22:14:41][INFO]
[[ 1.5611000e+00  5.1791000e+00  1.6852000e+00  8.4253998e+00]
 [ 0.0000000e+00  9.3383007e+00  7.2133622e+00  1.6551662e+01]
 [ 0.0000000e+00  0.0000000e+00 -1.2302779e+04 -1.2302777e+04]]
[0.99999975 1.00000009 0.99999982]
Solución ejercicio 4c - Eliminación Gaussiana:
Determinante: 1.653475862894993e-07
[07-27 22:14:41][INFO]
[[ 0.25        0.2         0.16666667  0.14285715  0.25      ]
 [ 0.         -0.06666666 -0.08333334 -0.0857143   0.        ]
 [ 0.         -0.01666668 -0.02222224 -0.02380954  0.        ]
 [ 0.         -0.3        -0.33333334 -0.3214286   0.        ]]
[07-27 22:14:41][INFO]
[[ 2.5000000e-01  2.0000000e-01  1.6666667e-01  1.4285715e-01
   2.5000000e-01]
 [ 0.0000000e+00 -1.6666681e-02 -2.2222236e-02 -2.3809537e-02
   0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  5.5555180e-03  9.5237717e-03
   0.0000000e+00]
 [ 0.0000000e+00  2.9802322e-08  6.6666603e-02  1.0714275e-01
   0.0000000e+00]]
[07-27 22:14:41][INFO]
[[ 2.5000000e-01  2.0000000e-01  1.6666667e-01  1.4285715e-01
   2.5000000e-01]
 [ 0.0000000e+00 -1.6666681e-02 -2.2222236e-02 -2.3809537e-02
   0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  5.5555180e-03  9.5237717e-03
   0.0000000e+00]
 [ 0.0000000e+00  2.9802322e-08  0.0000000e+00 -7.1431771e-03
   0.0000000e+00]]
[ 1. -0.  0. -0.]
Solución ejercicio 4d - Eliminación Gaussiana:
Determinante: -508.0
[07-27 22:14:41][INFO]
[[  1.   2.  -4.   4.   5.   2.]
 [  0.  -3.   6. -11.  -3.  -3.]
 [  0.   1.  -4.   3.   5.  -1.]
 [  0.  -5.  10.  -8. -20.   0.]
 [  0.  -3.   5.  -5.  -2.  -5.]]
[07-27 22:14:41][INFO]
```

```
[[  1.   2.  -4.   4.   5.   2.]
 [  0.   1.  -4.   3.   5.  -1.]
 [  0.   0.  -6.  -2.  12.  -6.]
 [  0.   0. -10.   7.   5.  -5.]
 [  0.   0.  -7.   4.  13.  -8.]]
[07-27 22:14:41][INFO]
[[  1.         2.        -4.         4.         5.         2.      ]
 [  0.         1.        -4.         3.         5.        -1.      ]
 [  0.         0.        -6.        -2.        12.        -6.      ]
 [  0.         0.         0.        10.333333 -15.         5.      ]
 [  0.         0.         0.         6.333333  -1.        -1.      ]]
[07-27 22:14:41][INFO]
[[  1.         2.        -4.         4.         5.         2.      ]
 [  0.         1.        -4.         3.         5.        -1.      ]
 [  0.         0.        -6.        -2.        12.        -6.      ]
 [  0.         0.         0.         6.333333  -1.        -1.      ]
 [  0.         0.         0.         0.       -13.368422   6.631579]]
[ 0.70078729  2.53543319  0.08661423 -0.23622048 -0.49606296]

Solución ejercicio 5:

Parte a:

Resolviendo para  = -5.0
Determinante: 34.0
[07-27 22:14:41][INFO]
[[ 1. -1.  5. -2.]
 [ 0.  1.  2.  1.]
 [ 0. -4. 26. -8.]]
[07-27 22:14:41][INFO]
[[ 1. -1.  5. -2.]
 [ 0.  1.  2.  1.]
 [ 0.  0. 34. -4.]]
[-0.17647059  1.23529412 -0.11764706]

Resolviendo para  = -4.0
Determinante: 20.0
[07-27 22:14:41][INFO]
[[ 1. -1.  4. -2.]
 [ 0.  1.  1.  1.]
 [ 0. -3. 17. -6.]]
[07-27 22:14:41][INFO]
[[ 1. -1.  4. -2.]
```

```
 [ 0.  1.  1.  1.]
 [ 0.  0. 20. -3.]]
[-0.24999997  1.15000001 -0.15000001]

Resolviendo para  = -3.0
Determinante: 10.0
[07-27 22:14:41][INFO]
[[ 1. -1.  3. -2.]
 [ 0.  1.  0.  1.]
 [ 0. -2. 10. -4.]]
[07-27 22:14:41][INFO]
[[ 1. -1.  3. -2.]
 [ 0.  1.  0.  1.]
 [ 0.  0. 10. -2.]]
[-0.39999999  1.         -0.2       ]

Resolviendo para  = -2.0
Determinante: 4.0
[07-27 22:14:41][INFO]
[[ 1. -1.  2. -2.]
 [ 0.  1. -1.  1.]
 [ 0. -1.  5. -2.]]
[07-27 22:14:41][INFO]
[[ 1. -1.  2. -2.]
 [ 0.  1. -1.  1.]
 [ 0.  0.  4. -1.]]
[-0.75  0.75 -0.25]

Resolviendo para  = -1.0
Determinante: 2.0
[07-27 22:14:41][INFO]
[[ 1. -1.  1. -2.]
 [ 0.  1. -2.  1.]
 [ 0.  0.  2.  0.]]
[07-27 22:14:41][INFO]
[[ 1. -1.  1. -2.]
 [ 0.  1. -2.  1.]
 [ 0.  0.  2.  0.]]
[-1.  1.  0.]

Resolviendo para  = 0.0
Determinante: 4.0
[07-27 22:14:41][INFO]
```

```
[[ 1. -1. -0. -2.]
 [ 0.  1. -3.  1.]
 [ 0.  1.  1.  2.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -0. -2.]
 [ 0.  1. -3.  1.]
 [ 0.  0.  4.  1.]]
[-0.25  1.75  0.25]

Resolviendo para  = 1.0
Determinante: 10.0
[07-27 22:14:41][INFO]
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  2.  2.  4.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  0. 10.  2.]]
[1.49011612e-08 1.80000001e+00 2.00000003e-01]

Resolviendo para  = 2.0
Determinante: 20.0
[07-27 22:14:41][INFO]
[[ 1. -1. -2. -2.]
 [ 0.  1. -5.  1.]
 [ 0.  3.  5.  6.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -2. -2.]
 [ 0.  1. -5.  1.]
 [ 0.  0. 20.  3.]]
[0.05000004 1.75000003 0.15000001]

Resolviendo para  = 3.0
Determinante: 34.0
[07-27 22:14:41][INFO]
[[ 1. -1. -3. -2.]
 [ 0.  1. -6.  1.]
 [ 0.  4. 10.  8.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -3. -2.]
 [ 0.  1. -6.  1.]
 [ 0.  0. 34.  4.]]
```

```
[0.05882353 1.70588236 0.11764706]

Resolviendo para  = 4.0
Determinante: 52.0
[07-27 22:14:41][INFO]
[[ 1. -1. -4. -2.]
 [ 0.  1. -7.  1.]
 [ 0.  5. 17. 10.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -4. -2.]
 [ 0.  1. -7.  1.]
 [ 0.  0. 52.  5.]]
[0.05769233 1.67307694 0.09615385]

Resolviendo para  = 5.0
Determinante: 74.0
[07-27 22:14:41][INFO]
[[ 1. -1. -5. -2.]
 [ 0.  1. -8.  1.]
 [ 0.  6. 26. 12.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -5. -2.]
 [ 0.  1. -8.  1.]
 [ 0.  0. 74.  6.]]
[0.05405401 1.64864862 0.08108108]

Parte b:

Parte c:

Resolviendo para  = 1
Determinante: 10.0
[07-27 22:14:41][INFO]
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  2.  2.  4.]]
[07-27 22:14:41][INFO]
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  0. 10.  2.]]
[1.49011612e-08 1.80000001e+00 2.00000003e-01]
```

```python
import numpy as np

def resolver_ejercicio_5(alpha):
    A5 = np.array([
        [1, -1, -alpha],
        [-1, 2, -3],
        [alpha, 1, 1]
    ], dtype=np.float32)
    b5 = np.array([-2, 3, 2], dtype=np.float32)

    det = np.linalg.det(A5)
    if det == 0:
        return alpha, det, "El sistema tiene un número infinito de soluciones o no tiene s
    else:
        return alpha, det, eliminacion_gaussiana(matriz_aumentada(A5, b5))

alphas_infinite_or_no_solution = []
alphas_unique_solution = []
for alpha in np.linspace(-5, 5, 11):
    result = resolver_ejercicio_5(alpha)
    if result[1] == 0:
        alphas_infinite_or_no_solution.append(result[0])
    else:
        alphas_unique_solution.append(result)

print("\nParte a y b:")
print(f"Valores de  para los que el sistema tiene un número infinito de soluciones o no t

print("\nParte c:")
for result in alphas_unique_solution:
    print(f"Para  = {result[0]}: Solución = {result[2]}")
```

```
[07-27 22:14:43][INFO]
[[ 1. -1.  5. -2.]
 [ 0.  1.  2.  1.]
 [ 0. -4. 26. -8.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  5. -2.]
 [ 0.  1.  2.  1.]
 [ 0.  0. 34. -4.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  4. -2.]
```

```
 [ 0.  1.  1.  1.]
 [ 0. -3. 17. -6.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  4. -2.]
 [ 0.  1.  1.  1.]
 [ 0.  0. 20. -3.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  3. -2.]
 [ 0.  1.  0.  1.]
 [ 0. -2. 10. -4.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  3. -2.]
 [ 0.  1.  0.  1.]
 [ 0.  0. 10. -2.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  2. -2.]
 [ 0.  1. -1.  1.]
 [ 0. -1.  5. -2.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  2. -2.]
 [ 0.  1. -1.  1.]
 [ 0.  0.  4. -1.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  1. -2.]
 [ 0.  1. -2.  1.]
 [ 0.  0.  2.  0.]]
[07-27 22:14:43][INFO]
[[ 1. -1.  1. -2.]
 [ 0.  1. -2.  1.]
 [ 0.  0.  2.  0.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -0. -2.]
 [ 0.  1. -3.  1.]
 [ 0.  1.  1.  2.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -0. -2.]
 [ 0.  1. -3.  1.]
 [ 0.  0.  4.  1.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  2.  2.  4.]]
[07-27 22:14:43][INFO]
```

```
[[ 1. -1. -1. -2.]
 [ 0.  1. -4.  1.]
 [ 0.  0. 10.  2.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -2. -2.]
 [ 0.  1. -5.  1.]
 [ 0.  3.  5.  6.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -2. -2.]
 [ 0.  1. -5.  1.]
 [ 0.  0. 20.  3.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -3. -2.]
 [ 0.  1. -6.  1.]
 [ 0.  4. 10.  8.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -3. -2.]
 [ 0.  1. -6.  1.]
 [ 0.  0. 34.  4.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -4. -2.]
 [ 0.  1. -7.  1.]
 [ 0.  5. 17. 10.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -4. -2.]
 [ 0.  1. -7.  1.]
 [ 0.  0. 52.  5.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -5. -2.]
 [ 0.  1. -8.  1.]
 [ 0.  6. 26. 12.]]
[07-27 22:14:43][INFO]
[[ 1. -1. -5. -2.]
 [ 0.  1. -8.  1.]
 [ 0.  0. 74.  6.]]

Parte a y b:
Valores de  para los que el sistema tiene un número infinito de soluciones o no tiene soluci

Parte c:
Para  = -5.0: Solución = [-0.17647059  1.23529412 -0.11764706]
Para  = -4.0: Solución = [-0.24999997  1.15000001 -0.15000001]
Para  = -3.0: Solución = [-0.39999999  1.         -0.2       ]
```

```
Para   = -2.0: Solución = [-0.75  0.75 -0.25]
Para   = -1.0: Solución = [-1.  1.  0.]
Para   = 0.0: Solución = [-0.25  1.75  0.25]
Para   = 1.0: Solución = [1.49011612e-08 1.80000001e+00 2.00000003e-01]
Para   = 2.0: Solución = [0.05000004 1.75000003 0.15000001]
Para   = 3.0: Solución = [0.05882353 1.70588236 0.11764706]
Para   = 4.0: Solución = [0.05769233 1.67307694 0.09615385]
Para   = 5.0: Solución = [0.05405401 1.64864862 0.08108108]
```

EJERCICIOS APLICADOS

6. Suponga que en un sistema biológico existen n especies de animales y m fuentes de alimento. Si representa la población de las j-ésimas especies, para cada = 1, , ; ; representa el suministro diario disponible del i-ésimo alimento y representa la cantidad del i-ésimo alimento.

b. ¿Cuál es el número máximo de animales de cada especie que se podría agregar de forma individual al sistema con el suministro de alimento que cumpla con el consumo?

c. Si la especie 1 se extingue, ¿qué cantidad de incremento individual de las especies restantes se podría soportar?

d. Si la especie 2 se extingue, ¿qué cantidad de incremento individual de las especies restantes se podría soportar?

```python
import numpy as np

# Matriz de coeficientes
A = np.array([
    [1, 2, 0, 3],
    [1, 0, 2, 2],
    [0, 0, 1, 1]
], dtype=np.float32)

# Vector de población de animales
x = np.array([1000, 500, 350, 400], dtype=np.float32)

# Vector de alimentos disponibles
b = np.array([3500, 2700, 900], dtype=np.float32)

# Parte a: Verificar si hay suficiente alimento
consumo_total = A @ x
```

```
suficiente = np.all(consumo_total <= b)

print("Parte a: Consumo total y verificación de suficiente alimento")
print(f"Consumo total: {consumo_total}")
print(f"Suficiente alimento: {suficiente}")

# Parte b: Máximo número de animales que se puede agregar
def max_animales(A, b):
    # Usamos la inversa de A para resolver el sistema
    # Primero verificamos si A es cuadrada y no singular
    try:
        inv_A = np.linalg.inv(A)
        max_x = inv_A @ b
        return max_x
    except np.linalg.LinAlgError as e:
        return str(e)

max_x = max_animales(A, b)
print("\nParte b: Número máximo de animales adicionales")
print(max_x)

# Parte c: Extinción de la especie 1
x_ext1 = x.copy()
x_ext1[0] = 0
consumo_total_ext1 = A @ x_ext1
suficiente_ext1 = np.all(consumo_total_ext1 <= b)

print("\nParte c: Extinción de la especie 1")
print(f"Consumo total sin especie 1: {consumo_total_ext1}")
print(f"Suficiente alimento sin especie 1: {suficiente_ext1}")

# Parte d: Extinción de la especie 2
x_ext2 = x.copy()
x_ext2[1] = 0
consumo_total_ext2 = A @ x_ext2
suficiente_ext2 = np.all(consumo_total_ext2 <= b)

print("\nParte d: Extinción de la especie 2")
print(f"Consumo total sin especie 2: {consumo_total_ext2}")
print(f"Suficiente alimento sin especie 2: {suficiente_ext2}")
```

```
Parte a: Consumo total y verificación de suficiente alimento
Consumo total: [3200. 2500.  750.]
Suficiente alimento: True

Parte b: Número máximo de animales adicionales
Last 2 dimensions of the array must be square

Parte c: Extinción de la especie 1
Consumo total sin especie 1: [2200. 1500.  750.]
Suficiente alimento sin especie 1: True

Parte d: Extinción de la especie 2
Consumo total sin especie 2: [2200. 2500.  750.]
Suficiente alimento sin especie 2: True
```

EJERCICIOS TEÓRICOS

7. Repita el ejercicio 4 con el método Gauss-Jordan.

```python
import numpy as np

def gauss_jordan(A, b):
    # Combine A and b into an augmented matrix
    Ab = np.hstack([A, b.reshape(-1, 1)])
    rows, cols = Ab.shape

    for i in range(rows):
        # Make the diagonal contain all 1's
        Ab[i] = Ab[i] / Ab[i, i]

        # Make the other rows contain 0's
        for j in range(rows):
            if i != j:
                Ab[j] = Ab[j] - Ab[j, i] * Ab[i]

    return Ab[:, -1]

# Ejercicio 4a
A4a = np.array([
    [1/4, 1/3, 1/2],
    [1/3, 1/4, 1/3],
```

```
    [1/2, 1, 1/3]
], dtype=np.float32)
b4a = np.array([9, 8, 8], dtype=np.float32)
sol4a = gauss_jordan(A4a, b4a)
print(f"Solución ejercicio 4a - Gauss-Jordan: {sol4a}")

# Ejercicio 4b
A4b = np.array([
    [3.3333, 15920, -10.333],
    [2.2222, 16.71, 9.6123],
    [1.5611, 5.1791, 1.6852]
], dtype=np.float32)
b4b = np.array([15913, 28.544, 8.4254], dtype=np.float32)
sol4b = gauss_jordan(A4b, b4b)
print(f"Solución ejercicio 4b - Gauss-Jordan: {sol4b}")

# Ejercicio 4c
A4c = np.array([
    [1, 1/2, 1/3, 1/4],
    [1, 1/3, 1/4, 1/5],
    [1, 1/4, 1/5, 1/6],
    [1, 1/5, 1/6, 1/7]
], dtype=np.float32)
b4c = np.array([1/6, 1/7, 1/8, 1/9], dtype=np.float32)
sol4c = gauss_jordan(A4c, b4c)
print(f"Solución ejercicio 4c - Gauss-Jordan: {sol4c}")

# Ejercicio 4d
A4d = np.array([
    [2, 1, -1, -3, 0],
    [1, 2, 0, 4, 1],
    [-2, 0, 1, -1, 5],
    [3, 1, -2, -4, 3],
    [1, -1, -3, 4, -1]
], dtype=np.float32)
b4d = np.array([7, 2, -5, 6, -3], dtype=np.float32)
sol4d = gauss_jordan(A4d, b4d)
print(f"Solución ejercicio 4d - Gauss-Jordan: {sol4d}")
```

```
Solución ejercicio 4a - Gauss-Jordan: [12.461542  -2.7692337 13.615386 ]
Solución ejercicio 4b - Gauss-Jordan: [1.0001469 1.0000001 0.9999925]
Solución ejercicio 4c - Gauss-Jordan: [ 0.00793662  0.11904685 -0.9523757   1.6666608 ]
```

```
Solución ejercicio 4d - Gauss-Jordan: [ 0.06896544  3.2413795  -0.931036   -0.8965517  -0.965
```