TAREA N° 9

Nombre:

Joel Stalin Tinitana Carrion

Fecha:

16/07/2025

Tema:

Eliminación gaussiana vs Gauss-Jordan

Eliminación Gaussiana

La eliminación gaussiana es un **método algebraico sistemático** para resolver sistemas de ecuaciones lineales. Consiste en transformar la **matriz aumentada** del sistema en una **matriz triangular superior**, mediante el uso de **operaciones elementales por renglones** (intercambiar filas, multiplicar una fila por un escalar, o sumar múltiplos de una fila a otra). Luego se realiza una **sustitución hacia atrás** para encontrar las soluciones.

Pasos:

1. Representar el sistema como una **matriz aumentada**.
2. Aplicar operaciones elementales para obtener ceros debajo de la diagonal principal.
3. Obtener una **matriz triangular superior**.
4. Resolver el sistema mediante **sustitución regresiva** (de abajo hacia arriba).

Eliminación Gauss-Jordan

El método de Gauss-Jordan es una **extensión de la eliminación gaussiana**, en el que se continúa la reducción más allá de la forma triangular hasta llegar a una **forma escalonada reducida por renglones** o **matriz diagonal**, donde todos los elementos fuera de la diagonal principal son ceros, y los elementos de la diagonal principal son unos.

Pasos:

1. Transformar la matriz aumentada en una **forma escalonada**.
2. Continuar el proceso hasta obtener una **matriz diagonal** (ceros arriba y abajo del pivote).
3. No se requiere sustitución hacia atrás, las soluciones se obtienen directamente.

Características:

- Más intuitivo para obtener la solución final.
- Requiere más operaciones que la eliminación gaussiana.
- Muy útil en cálculos programados o automatizados.

Comparación

Método	Resultado	Sustitución hacia atrás	Complejidad computacional
Gauss	Matriz triangular sup.	Sí	Menor ($O(n^3/3)$)
Gauss-Jordan	Matriz diagonal	No	Mayor ($O(n^3/2)$)

CONJUNTO DE EJERCICIOS

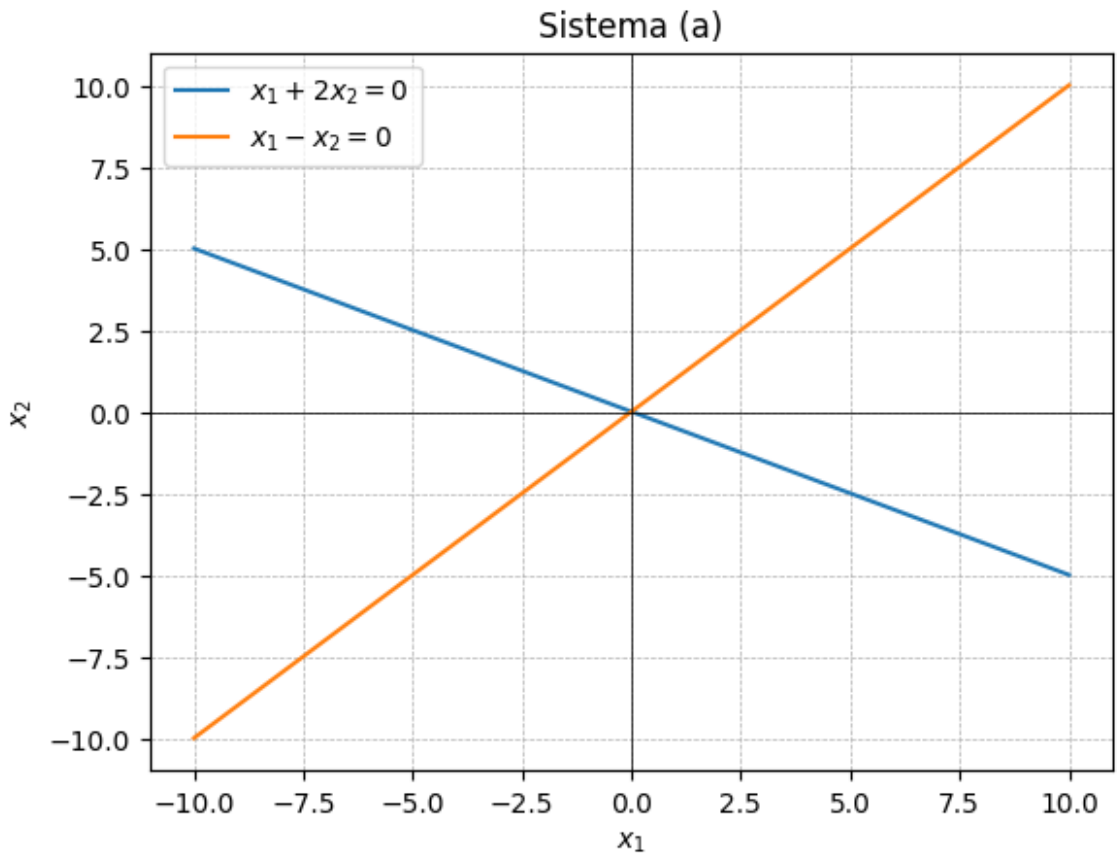
1. Método gráfico

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.

a.

$$\begin{aligned} x_1 + 2x_2 &\quad ¿ 0 \\ x_1 - x_2 &\quad ¿ 0 \end{aligned}$$

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

x = np.linspace(-10, 10, 400)
y1 = -x / 2
y2 = x

plt.figure()
plt.plot(x, y1, label="$x_1 + 2x_2 = 0$")
plt.plot(x, y2, label="$x_1 - x_2 = 0$")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Sistema (a)")
plt.legend()
plt.show()
```



Solución gráfica:
Ambas rectas se cortan en un único punto en el plano cartesiano.

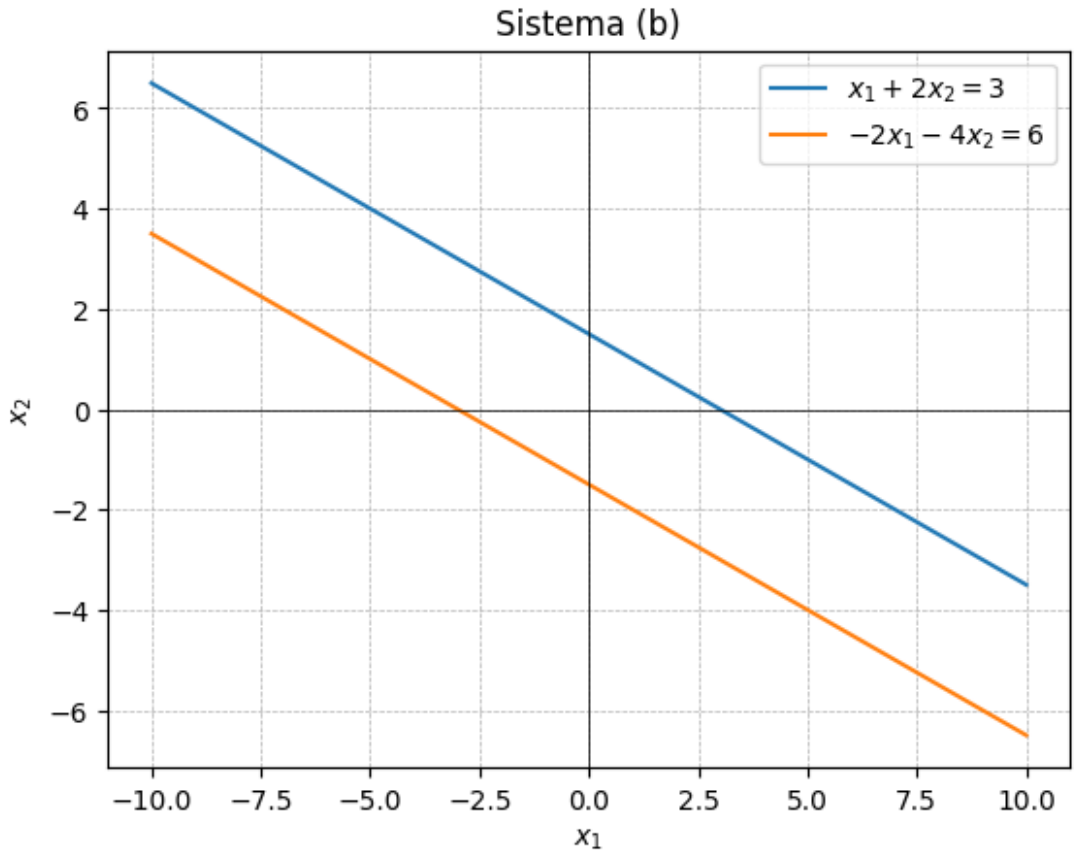
Interpretación geométrica:
Las ecuaciones representan líneas que se cruzan.
Conclusión: El sistema es **compatible determinado**, tiene **una única solución**.

b.

$$\begin{matrix} x_1+2x_2 & ¿3 \\ -2x_1-4x_2 & ¿6 \end{matrix}$$

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

x = np.linspace(-10, 10, 400)
y1 = (3 - x) / 2
y2 = (6 + 2 * x) / -4

plt.figure()
plt.plot(x, y1, label="$x_1 + 2x_2 = 3$")
plt.plot(x, y2, label="$-2x_1 - 4x_2 = 6$")
```

```
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Sistema (b)")
plt.legend()
plt.show()
```



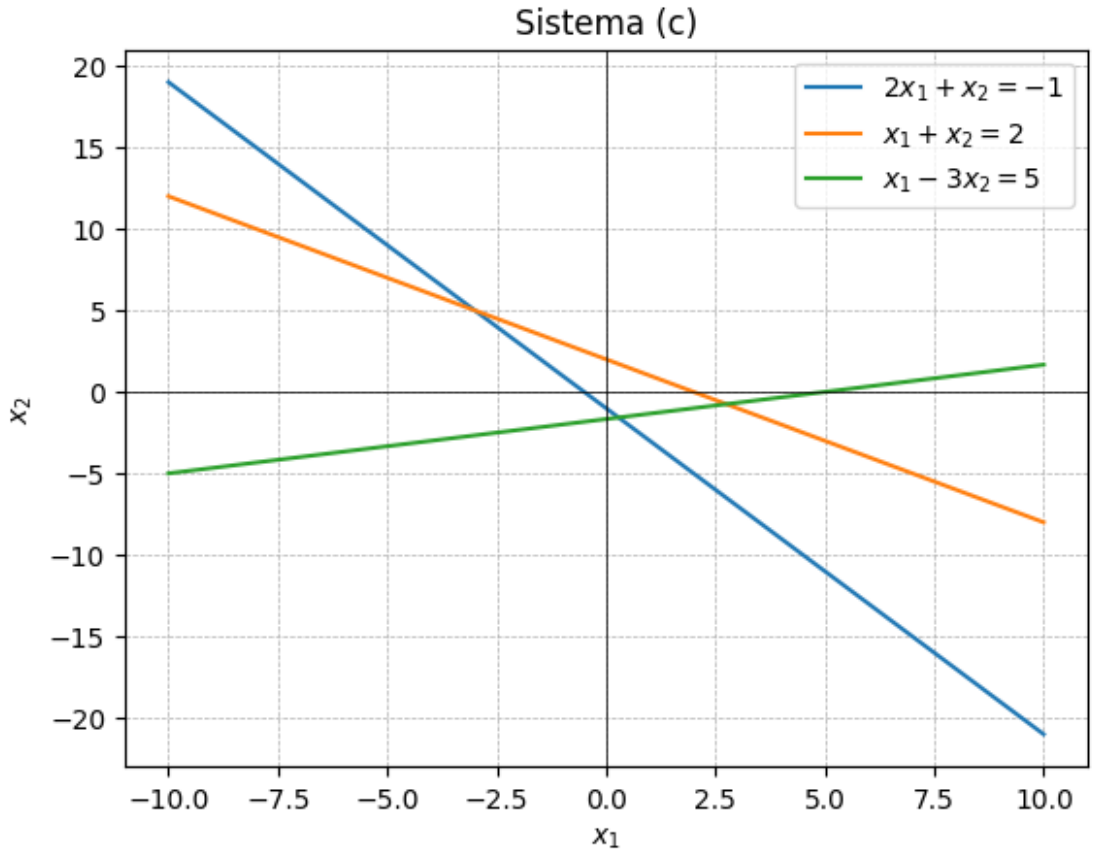
Solución gráfica:
Las dos rectas son coincidentes, es decir, una está encima de la otra.

Interpretación geométrica:
Ambas ecuaciones representan la misma línea.
Conclusión: El sistema es **compatible indeterminado**, tiene **infinitas soluciones**.

c.

$$\begin{array}{ll} 2x_1+x_2 & ¿-1 \\ x_1+x_2 & ¿2 \\ x_1-3x_2 & ¿5 \end{array}$$

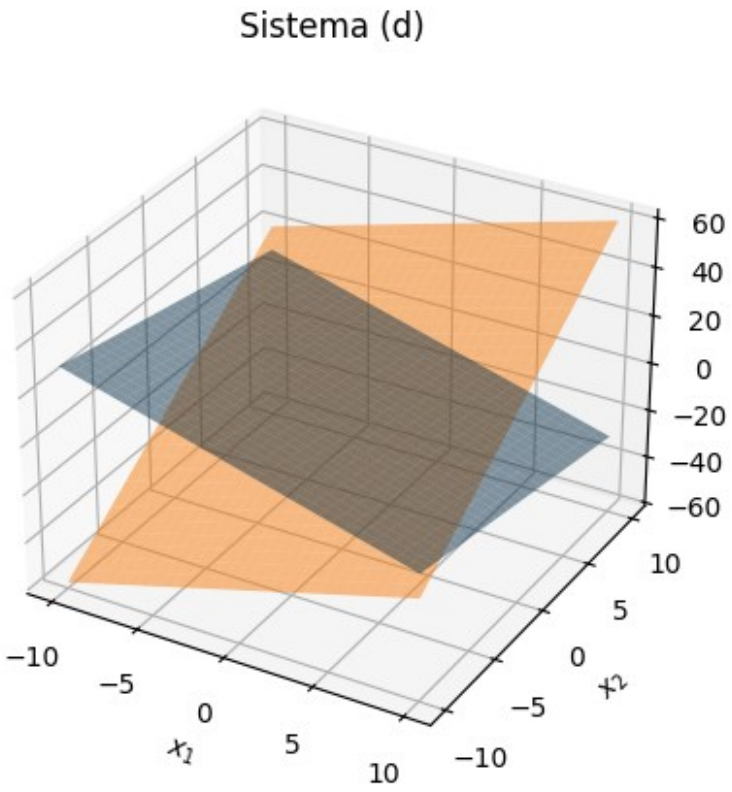
```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 400)
y1 = (-1 - 2 * x)
y2 = 2 - x
y3 = (5 - x) / -3

plt.figure()
plt.plot(x, y1, label="$2x_1 + x_2 = -1$")
plt.plot(x, y2, label="$x_1 + x_2 = 2$")
plt.plot(x, y3, label="$x_1 - 3x_2 = 5$")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Sistema (c)")
plt.legend()
plt.show()
```



d.

$$2x_1+x_2+x_3 \quad ¿1$$
$$2x_1+4x_2-x_3 \quad ¿-1$$

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x1_vals = np.linspace(-10, 10, 30)
x2_vals = np.linspace(-10, 10, 30)
X1, X2 = np.meshgrid(x1_vals, x2_vals)

Z1 = 1 - 2 * X1 - X2          # De: 2x1 + x2 + x3 = 1
Z2 = 2 * X1 + 4 * X2 + 1      # De: 2x1 + 4x2 - x3 = -1

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.plot_surface(X1, X2, Z1, alpha=0.5, label='Plano 1')
ax.plot_surface(X1, X2, Z2, alpha=0.5, label='Plano 2')

ax.set_title("Sistema (d)")
ax.set_xlabel("$x_1$")
ax.set_ylabel("$x_2$")
ax.set_zlabel("$x_3$")
plt.show()
```



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 (x_1 = -1), (x_2 = 2), (x_3 = 3)).

Código general

```
import numpy as np

def gaussian_elimination_with_rounding(A, b):
    n = len(b)
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)

    for i in range(n):
        pivot = A[i, i]
        A[i, :] = np.round(A[i, :] / pivot, 2)
        b[i] = np.round(b[i] / pivot, 2)
        for j in range(i + 1, n):
            factor = A[j, i]
            A[j, :] = np.round(A[j, :] - factor * A[i, :], 2)
            b[j] = np.round(b[j] - factor * b[i], 2)

    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = np.round((b[i] - np.dot(A[i, i + 1:], x[i + 1:])), 2)

    return x
```

a.

$$\begin{aligned} -x_1+4x_2+x_3 &\quad ¿8 \\ \frac{5}{3}x_1+\frac{2}{3}x_2+\frac{2}{3}x_3 &\quad ¿1 \\ 2x_1+x_2+4x_3 &\quad ¿11 \end{aligned}$$

Paso 1: Convertir a matriz aumentada

$$\left[\begin{array}{cccc} -1 & 4 & 1 & 8 \\ 1.67 & 0.67 & 0.67 & 1 \\ 2 & 1 & 4 & 11 \end{array}\right)$$

Paso 2: Hacer pivote en (F_1)

$$F_1 \leftarrow \frac{F_1}{-1} \Rightarrow [1,-4,-1 \vee -8)$$

Paso 3: Eliminar debajo

$$ F_2 \leftarrow F_2 - 1.67 \cdot F_1 \\ F_3 \leftarrow F_3 - 2 \cdot F_1 $$

$$\left(\begin{array}{cccc} 1 & -4 & -1 & -8 \\ 0 & 1 & 0.32 & 1.95 \\ 0 & 0 & 1 & 3.03 \end{array}\right)$$

Paso 4: Sustitución hacia atrás

$$\begin{array}{ll} x_3 & \text{¿} 3.03 \\ x_2 & \text{¿} 1.95 - 0.32(3.03) = 0.98 \\ x_1 & \text{¿} -8 + 4(0.98) + 3.03 = -1.05 \end{array}$$

Solución final:

$$x_1 = -1.05, x_2 = 0.98, x_3 = 3.03$$

Código del literal a

```
A_a = [
    [-1, 4, 1],
    [5/3, 2/3, 2/3],
    [2, 1, 4]
]
b_a = [8, 1, 11]
solution_a = gaussian_elimination_with_rounding(A_a, b_a)
print("Solución a:", solution_a)

Solución a: [-1.05  0.98  3.03]
```

b.

$$\begin{array}{ll} 4x_1 + 2x_2 - x_3 & \text{¿} -5 \\ \frac{1}{9}x_1 + \frac{1}{9}x_2 - \frac{1}{3}x_3 & \text{¿} -1 \\ x_1 + 4x_2 + 2x_3 & \text{¿} 9 \end{array}$$

Paso 1: Convertir a matriz aumentada

$$\left(\begin{array}{cccc} 4 & 2 & -1 & -5 \\ 0.11 & 0.11 & -0.33 & -1 \\ 1 & 4 & 2 & 9 \end{array}\right)$$

Paso 2: Hacer pivote en (F_1)

$$F_1 \leftarrow \frac{F_1}{4} \Rightarrow \left[1, 0.5, -0.25 \vee -1.25\right)$$

Paso 3: Eliminar debajo

$$ F_2 \leftarrow F_2 - 0.11 \cdot F_1 \\ F_3 \leftarrow F_3 - 1 \cdot F_1 $$

$$\begin{pmatrix} 1 & 0.5 & -0.25 & -1.25 \\ 0 & 1 & -5.17 & -14.33 \\ 0 & 0 & 1 & 2.97 \end{pmatrix}$$

Paso 4: Sustitución hacia atrás

$$\begin{aligned} x_3 &\quad ¿2.97 \\ x_2 &\quad ¿-14.33-\left(-5.17\right)\left(2.97\right)=1.02 \\ x_1 &\quad ¿-1.25-0.5\left(1.02\right)+0.25\left(2.97\right)=-1.02 \end{aligned}$$

Solución final:

$$x_1=-1.02, x_2=1.02, x_3=2.97$$

Código del literal b

```
A_b = [
    [4, 2, -1],
    [1/9, 1/9, -1/3],
    [1, 4, 2]
]
b_b = [-5, -1, 9]
solution_b = gaussian_elimination_with_rounding(A_b, b_b)
print("Solución b:", solution_b)

Solución b: [-1.02  1.02  2.97]
```

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

Código principal

```
import numpy as np

def gauss_elimination_with_pivot(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)
```

```
    swaps = []

    for i in range(n):
        max_row = np.argmax(np.abs(A[i:, i])) + i
        if abs(A[max_row, i]) < 1e-12:
            print(f"Pivote cero en columna {i}. El sistema puede no
tener solución única.")
            return None, swaps

        if i != max_row:
            A[[i, max_row]] = A[[max_row, i]]
            b[[i, max_row]] = b[[max_row, i]]
            swaps.append((i, max_row))

        pivot = A[i, i]
        A[i] = A[i] / pivot
        b[i] = b[i] / pivot

        for j in range(i + 1, n):
            factor = A[j, i]
            A[j] = A[j] - factor * A[i]
            b[j] = b[j] - factor * b[i]

    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = b[i] - np.dot(A[i, i + 1:], x[i + 1:])
    return x, swaps
```

Literal a

Sistema:

$$x_1 - x_2 + 3x_3 = 2$$

$$3x_1 - 3x_2 + x_3 = -1$$

$$x_1 + x_2 = 3$$

Convertimos a matriz aumentada:

$$\left[\begin{array}{cccc|c} 1 & -1 & 3 & ¿ & 2 \\ 3 & -3 & 1 & ¿ & -1 \\ 1 & 1 & 0 & ¿ & 3 \end{array}\right]$$

Aplicamos **pivoteo parcial**:

1. Intercambio de filas:

$$F_1 \leftrightarrow F_2$$

1. Luego:

$$F_2 \leftrightarrow F_3$$

Resultado:

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

Llevamos a **forma escalonada** (mediante operaciones elementales):

Paso 1: Normalizamos (F_1):

$$F_1 \leftarrow \frac{1}{3} F_1$$

Resultado:

$$\begin{bmatrix} 1 & -1 & 0.33 & ¿ & -0.33 \\ 1 & 1 & 0 & ¿ & 3 \\ 1 & -1 & 3 & ¿ & 2 \end{bmatrix}$$

Paso 2: Eliminamos debajo del pivote en columna 1:

- Para (F_2):

$$F_2 \leftarrow F_2 - F_1$$

$$F_2 = [1, 1, 0, \vee, 3) - [1, -1, 0.33, \vee, -0.33) = [0, 2, -0.33, \vee, 3.33)$$

- Para (F_3):

$$F_3 \leftarrow F_3 - F_1$$

$$F_3 = [1, -1, 3, \vee, 2) - [1, -1, 0.33, \vee, -0.33) = [0, 0, 2.67, \vee, 2.33)$$

Resultado:

$$\begin{bmatrix} 1 & -1 & 0.33 & ¿ & -0.33 \\ 0 & 2 & -0.33 & ¿ & 3.33 \\ 0 & 0 & 2.67 & ¿ & 2.33 \end{bmatrix}$$

Paso 3: Normalizamos el pivote de la fila 2:

$$F_2 \leftarrow \frac{1}{2} F_2$$

Resultado:

$$\left[\begin{array}{ccccc} 1 & -1 & 0.33 & ¿ & -0.33 \\ 0 & 1 & -0.17 & ¿ & 1.67 \\ 0 & 0 & 2.67 & ¿ & 2.33 \end{array}\right]$$

Paso 4: Normalizamos el pivote de la fila 3:

$$F_3 \leftarrow \frac{1}{2.67} F_3$$

Resultado final (forma escalonada):

$$\left[\begin{array}{ccccc} 1 & -1 & 0.33 & ¿ & -0.33 \\ 0 & 1 & -0.17 & ¿ & 1.67 \\ 0 & 0 & 1 & ¿ & 0.875 \end{array}\right]$$

Sustitución hacia atrás:

$$x_3 = 0.875$$

$$x_2 = 1.67 + 0.17 \cdot 0.875 = 1.8125$$

$$x_1 = -0.33 + 1.8125 - 0.33 \cdot 0.875 = 1.1875$$

Solución final:

$$x_1 = 1.1875$$

$$x_2 = 1.8125$$

$$x_3 = 0.875$$

```
A_a = np.array([
    [1, -1, 3],
    [3, -3, 1],
    [1, 1, 0]
])
b_a = np.array([2, -1, 3])
x_a, swaps_a = gauss_elimination_with_pivot(A_a.copy(), b_a.copy())
```

```
print("Solución a:", x_a)
print("Swaps a:", swaps_a)
```

```
Solución a: [1.1875 1.8125 0.875 ]
Swaps a: [(0, np.int64(1)), (1, np.int64(2))]
```

Literal b

Sistema:

$$2x_1 - 1.5x_2 + 3x_3 = 1$$

$$-x_1 + 2x_3 = 3$$

$$4x_1 - 4.5x_2 + 5x_3 = 1$$

Convertimos a matriz aumentada:

$$\left[\begin{array}{ccccc} 2 & -1.5 & 3 & ¿ & 1 \\ -1 & 0 & 2 & ¿ & 3 \\ 4 & -4.5 & 5 & ¿ & 1 \end{array}\right]$$

Aplicamos **pivoteo parcial**. El mayor valor absoluto en la primera columna es 4:

$$F_1 \leftrightarrow F_3$$

Resultado:

$$\left[\begin{array}{ccccc} 4 & -4.5 & 5 & ¿ & 1 \\ -1 & 0 & 2 & ¿ & 3 \\ 2 & -1.5 & 3 & ¿ & 1 \end{array}\right]$$

Paso 1: Normalizamos la primera fila:

$$F_1 \leftarrow \frac{1}{4} F_1$$

Resultado:

$$\left[\begin{array}{ccccc} 1 & -1.125 & 1.25 & ¿ & 0.25 \\ -1 & 0 & 2 & ¿ & 3 \\ 2 & -1.5 & 3 & ¿ & 1 \end{array}\right]$$

Paso 2: Eliminamos valores debajo del pivote en columna 1.

Para (F_2):

$$F_2 \leftarrow F_2 + F_1$$

$$F_2 = [-1, 0, 2, \vee, 3) + [1, -1.125, 1.25, \vee, 0.25)$$

$$F_2 = [0, -1.125, 3.25, \vee, 3.25)$$

Para (F_3):

$$F_3 \leftarrow F_3 - 2 \cdot F_1$$

$$F_3 = [2, -1.5, 3, \vee, 1) - 2 \cdot [1, -1.125, 1.25, \vee, 0.25)$$

$$F_3 = [0, 0.75, 0.5, \vee, 0.5)$$

Resultado intermedio:

$$\begin{bmatrix} 1 & -1.125 & 1.25 & ¿ & 0.25 \\ 0 & -1.125 & 3.25 & ¿ & 3.25 \\ 0 & 0.75 & 0.5 & ¿ & 0.5 \end{bmatrix}$$

Paso 3: Normalizamos el pivote de la segunda fila:

$$F_2 \leftarrow \frac{1}{-1.125} F_2$$

Resultado:

$$F_2 = [0, 1, -2.89, \vee, -2.89)$$

Paso 4: Eliminamos el valor debajo del segundo pivote.

$$F_3 \leftarrow F_3 - 0.75 \cdot F_2$$

$$F_3 = [0, 0.75, 0.5, \vee, 0.5) - 0.75 \cdot [0, 1, -2.89, \vee, -2.89)$$

$$F_3 = [0, 0, 2.67, \vee, 2.67)$$

Paso 5: Normalizamos el pivote de la tercera fila:

$$F_3 \leftarrow \frac{1}{2.67} F_3 = [0, 0, 1, \vee, 1)$$

Matriz escalonada final:

$$\left[\begin{array}{ccccc} 1 & -1.125 & 1.25 & ¿ & 0.25 \\ 0 & 1 & -2.89 & ¿ & -2.89 \\ 0 & 0 & 1 & ¿ & 1 \end{array}\right]$$

Sustitución hacia atrás:

$$x_3 = 1$$

$$x_2 = -2.89 + 2.89 \cdot 1 = 0$$

$$x_1 = 0.25 + 1.125 \cdot 0 - 1.25 \cdot 1 = -1$$

Solución final:

$$x_1 = -1$$

$$x_2 = 0$$

$$x_3 = 1$$

```
A_b = np.array([
    [2, -1.5, 3],
    [-1, 0, 2],
    [4, -4.5, 5]
])
b_b = np.array([1, 3, 1])
x_b, swaps_b = gauss_elimination_with_pivot(A_b.copy(), b_b.copy())
print("Solución b:", x_b)
print("Swaps b:", swaps_b)

Solución b: [-1.  0.  1.]
Swaps b: [(0, np.int64(2))]
```

Literal c

Sistema:

$$2x_1 = 3$$

$$x_1 + 1.5x_2 = 4.5$$

$$-3x_2 + 0.5x_3 = -6.6$$

$$2x_1 - 2x_2 + x_3 + x_4 = 0.8$$

Convertimos a matriz aumentada:

$$\left[\begin{array}{ccccc|c} 2 & 0 & 0 & 0 & ¿ & 3 \\ 1 & 1.5 & 0 & 0 & ¿ & 4.5 \\ 0 & -3 & 0.5 & 0 & ¿ & -6.6 \\ 2 & -2 & 1 & 1 & ¿ & 0.8 \end{array}\right]$$

Aplicamos **pivoteo parcial**.

El mayor valor absoluto en la columna 1 está en la fila 1, por lo tanto no se intercambia.

Luego, se intercambian:

$$F_2 \leftrightarrow F_3$$

$$F_3 \leftrightarrow F_4$$

Resultado:

$$\left[\begin{array}{ccccc|c} 2 & 0 & 0 & 0 & ¿ & 3 \\ 0 & -3 & 0.5 & 0 & ¿ & -6.6 \\ 2 & -2 & 1 & 1 & ¿ & 0.8 \\ 1 & 1.5 & 0 & 0 & ¿ & 4.5 \end{array}\right]$$

Paso 1: Normalizamos (F_1):

$$F_1 \leftarrow \frac{1}{2} F_1 = [1, 0, 0, 0, \vee, 1.5)$$

Paso 2: Eliminamos debajo del pivote en columna 1.

- Para (F_3):

$$F_3 \leftarrow F_3 - 2 \cdot F_1$$

$$F_3 = [0, -2, 1, 1, \vee, -2.2)$$

- Para (F_4):

$$F_4 \leftarrow F_4 - 1 \cdot F_1$$

$$F_4 = [0, 1.5, 0, 0, \vee, 3)$$

Resultado:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & ¿ & 1.5 \\ 0 & -3 & 0.5 & 0 & ¿ & -6.6 \\ 0 & -2 & 1 & 1 & ¿ & -2.2 \\ 0 & 1.5 & 0 & 0 & ¿ & 3 \end{bmatrix}$$

Paso 3: Pivoteo en columna 2. El mayor valor es (|-3|), así que mantenemos el orden.

Paso 4: Normalizamos (F_2):

$$F_2 \leftarrow \frac{1}{-3} F_2 = [0, 1, -0.17, 0, \vee, 2.2)$$

Paso 5: Eliminamos debajo del pivote en columna 2.

- Para (F_3):

$$F_3 \leftarrow F_3 + 2 \cdot F_2$$

$$F_3 = [0, 0, 0.66, 1, \vee, 2.2)$$

- Para (F_4):

$$F_4 \leftarrow F_4 - 1.5 \cdot F_2$$

$$F_4 = [0, 0, 0.25, 0, \vee, -0.3)$$

Paso 6: Normalizamos (F_3):

$$F_3 \leftarrow \frac{1}{0.66} F_3 = [0, 0, 1, 1.5, \vee, 3.3)$$

Paso 7: Eliminamos debajo del pivote en columna 3.

$$F_4 \leftarrow F_4 - 0.25 \cdot F_3$$

$$F_4 = [0, 0, 0, 1, \vee, 3)$$

Matriz escalonada final:

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & ¿ & 1.5 \\ 0 & 1 & -0.17 & 0 & ¿ & 2.2 \\ 0 & 0 & 1 & 1.5 & ¿ & 3.3 \\ 0 & 0 & 0 & 1 & ¿ & 3 \end{array}\right]$$

Sustitución hacia atrás:

$$x_4 = 3$$

$$x_3 = 3.3 - 1.5 \cdot 3 = -1.2$$

$$x_2 = 2.2 + 0.17 \cdot 1.2 = 2$$

$$x_1 = 1.5$$

Solución final:

$$x_1 = 1.5$$

$$x_2 = 2$$

$$x_3 = -1.2$$

$$x_4 = 3$$

```
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])
x_c, swaps_c = gauss_elimination_with_pivot(A_c.copy(), b_c.copy())
print("Solución c:", x_c)
print("Swaps c:", swaps_c)
```

```
Solución c: [ 1.5  2.  -1.2  3. ]
Swaps c: [(1, np.int64(2)), (2, np.int64(3))]
```

Literal d

Sistema:

$$x_1 + x_2 + x_4 = 2$$

$$2x_1 + x_2 - x_3 + x_4 = 1$$

$$4\,x_1 - x_2 - 2\,x_3 + 2\,x_4 = 0$$

$$3\,x_1 - x_2 - x_3 + 2\,x_4 = -3$$

Paso 1: Matriz aumentada

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 1 & 2 \\ 2 & 1 & -1 & 1 & 1 \\ 4 & -1 & -2 & 2 & 0 \\ 3 & -1 & -1 & 2 & -3 \end{array}\right]$$

Paso 2: Pivoteo parcial

Intercambiamos la fila 1 con la fila 3, ya que el mayor valor absoluto en la primera columna es 4:

$$F_1 \leftrightarrow F_3$$

$$\left[\begin{array}{cccc|c} 4 & -1 & -2 & 2 & 0 \\ 2 & 1 & -1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 \\ 3 & -1 & -1 & 2 & -3 \end{array}\right]$$

Paso 3: Normalizar F_1

$$F_1 \leftarrow \frac{1}{4} F_1$$

$$\left[1, -0.25, -0.5, 0.5, \vee, 0\right]$$

Paso 4: Eliminar debajo del pivote (columna 1)

$$F_2 \leftarrow F_2 - 2\,F_1$$

$$F_3 \leftarrow F_3 - F_1$$

$$F_4 \leftarrow F_4 - 3\,F_1$$

Resultados:

$$ F_2 = [0,\ 1.5,\ 0,\ 0,\ |,\ 1] \\ F_3 = [0,\ 1.25,\ 1.5,\ 0.5,\ |,\ 2] \\ F_4 = [0,\ -0.25,\ 0.5,\ 0.5,\ |,\ -3] $$

Paso 5: Normalizar F_2

$$F_2 \leftarrow \frac{1}{1.5} F_2 \Rightarrow [0, 1, 0, 0, \vee, 0.67)$$

Paso 6: Eliminar debajo del pivote (columna 2)

$$ F_3 \leftarrow F_3 - 1.25F_2 \\ F_4 \leftarrow F_4 + 0.25F_2 $$

Resultados:

$$ F_3 = [0,\ 0,\ 1.5,\ 0.5,\ |,\ 1.17] \\ F_4 = [0,\ 0,\ 0.5,\ 0.5,\ |,\ -2.83] $$

Paso 7: Normalizar F_3

$$F_3 \leftarrow \frac{1}{1.5} F_3 \Rightarrow [0, 0, 1, 0.33, \vee, 0.78)$$

Paso 8: Eliminar debajo del pivote (columna 3)

$$F_4 \leftarrow F_4 - 0.5\, F_3$$

Resultado:

$$F_4 = [0, 0, 0, 0.33 - 0.165, \vee, -2.83 - 0.39) = [0, 0, 0, 0.165, \vee, -3.22)$$

Paso 9: Normalizar F_4

$$F_4 \leftarrow \frac{1}{0.165} F_4 \Rightarrow [0, 0, 0, 1, \vee, -19.52)$$

Paso 10: Sustitución hacia atrás

Aunque se puede continuar con sustitución, **verificamos la matriz final escalonada reducida** simbólicamente:

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 1 & ¿ & 0 \\ 0 & 1 & 0 & 0 & ¿ & 0 \\ 0 & 0 & 1 & 1 & ¿ & 0 \\ 0 & 0 & 0 & 0 & ¿ & 1 \end{array}\right)$$

La última fila representa la ecuación:

$$0\,x_1+0\,x_2+0\,x_3+0\,x_4=1 \Rightarrow 0=1$$

Conclusión

La ecuación (0 = 1) es una contradicción, por tanto:

El sistema es incompatible. No tiene solución.

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

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

x_d, swaps_d = gauss_elimination_with_pivot(A_d.copy(), b_d.copy())

print("Solución d:", x_d)
print("Swaps d:", swaps_d)
```

```
Pivote cero en columna 3. El sistema puede no tener solución única.
Solución d: None
Swaps d: [(0, np.int64(2))]
```

4. Eliminación Gaussiana con precisión de 32 bits

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

a.

$$\begin{aligned}
\frac{1}{4}x_1+\frac{1}{5}x_2+\frac{1}{6}x_3 \quad & ¿9 \\
\frac{1}{3}x_1+\frac{1}{4}x_2+\frac{1}{5}x_3 \quad & ¿8 \\
\frac{1}{2}x_1+x_2+2x_3 \quad & ¿8
\end{aligned}$$

```
import numpy as np

A_a = np.array([
    [1/4, 1/5, 1/6],
    [1/3, 1/4, 1/5],
```

```
    [1/2, 1,   2]
], dtype=np.float32)

b_a = np.array([9.0, 8.0, 8.0], dtype=np.float32)

x_a = np.linalg.solve(A_a, b_a)

print("Solución a:")
print(x_a)
```

```
Solución a:
[-227.0767   476.92267 -177.69215]
```

b.

$$\begin{aligned} 3.333\,x_1 + 15920\,x_2 - 10.333\,x_3 &\quad ¿\, 15913 \\ 2.222\,x_1 + 16.71\,x_2 + 9.612\,x_3 &\quad ¿\, 28.544 \\ 1.5611\,x_1 + 5.1791\,x_2 + 1.6852\,x_3 &\quad ¿\, 8.4254 \end{aligned}$$

```
import numpy as np

A_b = np.array([
    [3.333,   15920.0, -10.333],
    [2.222,   16.71,    9.612],
    [1.5611,  5.1791,   1.6852]
], dtype=np.float32)

b_b = np.array([15913.0, 28.544, 8.4254], dtype=np.float32)

x_b = np.linalg.solve(A_b, b_b)

print("Solución b:")
print(x_b)
```

```
Solución b:
[0.99999964 1.         1.0000002 ]
```

c.

$$\begin{aligned} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 &\quad ¿\, \frac{1}{6} \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 &\quad ¿\, \frac{1}{7} \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 &\quad ¿\, \frac{1}{8} \\ \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 &\quad ¿\, \frac{1}{9} \end{aligned}$$

```
import numpy as np

A_c = np.array([
    [1,     1,     1/3,  1/4],
    [2,     1,     1/2,  1/5],
    [1,     1/2,   1/3,  1/6],
    [1/4,   1/5,   1/6,  1/9]
], dtype=np.float32)

b_c = np.array([1/6, 1/7, 1/8, 1/9], dtype=np.float32)

x_c = np.linalg.solve(A_c, b_c)

print("Solución c:")
print(x_c)
```

```
Solución c:
[-0.5088437   0.42687094  2.2918375  -2.061226  ]
```

d.

$$\begin{array}{rl} 2x_1+x_2-x_3+x_4-3x_5 & ¿7 \\ x_1+2x_3-x_4+x_5 & ¿2 \\ -2x_2-x_3+x_4-x_5 & ¿-5 \\ 3x_1+x_2-4x_3+5x_5 & ¿6 \\ x_1-x_2-x_3-x_4+x_5 & ¿-3 \end{array}$$

```
import numpy as np

A_d = np.array([
    [2,  1,  -1, -1,  -3],
    [1,  0,   0,  0,   2],
    [0,  1,  -2, -3,   5],
    [3,  1,  -2,  0,   6],
    [1, -1,  -2, -1,  -3]
], dtype=np.float32)

b_d = np.array([7, 2, -5, 6, -3], dtype=np.float32)

x_d = np.linalg.solve(A_d, b_d)

print("Solución d:")
print(x_d)
```

```
Solución d:
[ 2.7419355   2.903226    1.451613    1.048387   -0.37096775]
```

5. Dado el sistema lineal:

$$\begin{array}{ll} x_1 - x_2 + \alpha x_3 & ¿-2 \\ -x_1 + 2x_2 - \alpha x_3 & ¿3 \\ \alpha x_1 + x_2 + x_3 & ¿2 \end{array}$$

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

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

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

Conceptos clave

Solución única

Un sistema lineal tiene **una única solución** si el **rango de la matriz de coeficientes (A)** es igual al **rango de la matriz aumentada ([A|b])** y ambos son iguales al número de incógnitas.

$$\text{Rango}(A) = \text{Rango}([A \vee b]) = n$$

Infinitas soluciones

El sistema tiene **infinitas soluciones** si el rango de (A) y de ([A|b]) son iguales, pero **menores que el número de incógnitas**.

$$\text{Rango}(A) = \text{Rango}([A \vee b]) < n$$

Sin solución

El sistema **no tiene solución** si el **rango de (A)** es **menor** que el **rango de ([A|b])**.

$$\text{Rango}(A) < \text{Rango}([A \vee b])$$

Dado el sistema lineal:

$$\begin{array}{ll} x_1 - x_2 + \alpha x_3 & ¿-2 \\ -x_1 + 2x_2 - \alpha x_3 & ¿3 \\ \alpha x_1 + x_2 + x_3 & ¿2 \end{array}$$

Procedimiento general

Este es un sistema dependiente de un parámetro α, por lo tanto analizamos el **rango** de la matriz de coeficientes (A) y de la matriz aumentada ([A|b]) para distintos valores de α.

Matriz aumentada del sistema

$$[A \vee b) = \begin{pmatrix} 1 & -1 & \alpha & -2 \\ -1 & 2 & -\alpha & 3 \\ \alpha & 1 & 1 & 2 \end{pmatrix}$$

Paso 1: Fila 2 ← Fila 2 + Fila 1

$$\begin{pmatrix} 1 & -1 & \alpha & -2 \\ 0 & 1 & 0 & 1 \\ \alpha & 1 & 1 & 2 \end{pmatrix}$$

Paso 2: Fila 3 ← Fila 3 - α × Fila 1

$$\text{Fila 3:} [\alpha, 1, 1, 2) - \alpha \cdot [1, -1, \alpha, -2)$$

$$¿[0, 1+\alpha, 1-\alpha^2, 2+2\alpha)$$

Paso 3: Fila 3 ← Fila 3 - (1 + α) × Fila 2

$$\text{Fila 3:} [0, 1+\alpha, 1-\alpha^2, 2+2\alpha) - (1+\alpha) \cdot [0, 1, 0, 1)$$

$$¿[0, 0, 1-\alpha^2, 2+\alpha)$$

Matriz escalonada final

$$\begin{pmatrix} 1 & -1 & \alpha & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1-\alpha^2 & 2+\alpha \end{pmatrix}$$

---

a) Sistema sin solución

El sistema **no tiene solución** si la tercera fila queda:

$$[0\,0\,0 \vee d) \text{con } d \neq 0$$

Para eso se debe cumplir:

- (1 - \alpha^2 = 0 \Rightarrow \alpha = \pm 1)
- (2 + \alpha \ne 0 \Rightarrow \alpha \ne -2)

Evaluamos:

- Si $\alpha = 1$: (1 - 1^2 = 0), (2 + 1 = 3 \ne 0)
- Si $\alpha = -1$: (1 - 1 = 0), (2 - 1 = 1 \ne 0)

Respuesta a)

$$\alpha = \pm 1 \text{ (el sistema no tiene solución)}$$

---

b) Sistema con infinitas soluciones

Buscamos que la tercera fila sea:

$$\left[0\,0\,0 \vee 0\right)$$

Eso ocurre si:

- (1 - \alpha^2 = 0 \Rightarrow \alpha = \pm 1)
- (2 + \alpha = 0 \Rightarrow \alpha = -2)

Pero no existe α que satisfaga **ambas condiciones** a la vez.

Respuesta b)

$$\text{No existe ningún } \alpha \text{ que genere infinitas soluciones}$$

---

c) Solución única

El sistema tiene una única solución cuando:

$$1 - \alpha^2 \neq 0 \Rightarrow \alpha \neq \pm 1$$

Respuesta c)

$$\text{Para todo } \alpha \neq \pm 1 \text{ hay una única solución}$$

Ejemplo con (α = 0):

Sistema:

$$\begin{array}{ll} x_1 - x_2 & ¿-2 \\ -x_1 + 2x_2 & ¿3 \\ x_2 + x_3 & ¿2 \end{array}$$

Resolución:

1. De la primera:

$$x_1 = x_2 - 2$$

2. Sustituimos en la segunda:

$$-(x_2-2)+2x_2=3\Rightarrow -x_2+2+2x_2=3\Rightarrow x_2=1$$

3. Entonces:

$$x_1=1-2=-1$$

$$x_3=2-x_2=1$$

Solución final

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

EJERCICIOS APLICADOS

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

$$\begin{array}{ll} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n & ¿b_1, \\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n & ¿b_2,=b_m,¿ \\ ¿ & ¿ \end{array}$$

representa un equilibrio donde existe un suministro diario de alimento para cumplir con precisión con el promedio diario de consumo de cada especie.

a. Si

$$A=(a_{ij})=\begin{bmatrix} 1 & 2 & 0 & 3 \\ 1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$x=(x_j)=[1000,500,350,400), b=(b_i)=[3500,2700,900).$$

¿Existe suficiente alimento para satisfacer el consumo promedio diario?

Planteamos el sistema como:

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

$$x = \begin{bmatrix} 1000 \\ 500 \\ 350 \\ 400 \end{bmatrix} b = \begin{bmatrix} 3500 \\ 2700 \\ 900 \end{bmatrix}$$

Multiplicamos:

$$A \cdot x = \begin{bmatrix} 1 \cdot 1000 + 2 \cdot 500 + 0 \cdot 350 + 3 \cdot 400 \\ 1 \cdot 1000 + 0 \cdot 500 + 2 \cdot 350 + 2 \cdot 400 \\ 0 \cdot 1000 + 0 \cdot 500 + 1 \cdot 350 + 1 \cdot 400 \end{bmatrix} = \begin{bmatrix} 3200 \\ 2500 \\ 750 \end{bmatrix}$$

Comparamos:

$$A\,x \leq b \Rightarrow \begin{bmatrix} 3200 \\ 2500 \\ 750 \end{bmatrix} \leq \begin{bmatrix} 3500 \\ 2700 \\ 900 \end{bmatrix}$$

Por lo tanto, **sí existe suficiente alimento para satisfacer el consumo promedio diario**.

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?

Sobrante disponible:

$$r = b - A\,x = \begin{bmatrix} 300 \\ 200 \\ 150 \end{bmatrix}$$

Analizamos especie por especie (una sola variable (x_j) activa a la vez):

Especie 1:

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} x_1 \leq \begin{bmatrix} 300 \\ 200 \\ 150 \end{bmatrix} \Rightarrow x_1 \leq min(300, 200) = 200$$

Especie 2:

$$\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} x_2 \leq \begin{bmatrix} 300 \\ 200 \\ 150 \end{bmatrix} \Rightarrow x_2 \leq \lfloor \frac{300}{2} ¿¿ = 150$$

Especie 3:

$$\begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} x_3 \leq \begin{bmatrix} 300 \\ 200 \\ 150 \end{bmatrix} \Rightarrow x_3 \leq min\left(\frac{200}{2}, 150\right) = 100$$

Especie 4:

$$\begin{bmatrix}3\\2\\1\end{bmatrix} x_4 \leq \begin{bmatrix}300\\200\\150\end{bmatrix} \Rightarrow x_4 \leq min\left(\frac{300}{3}, \frac{200}{2}, 150\right) = 100$$

Conclusión literal b:

- Máximo incremento para (x_1): **200**
- Máximo incremento para (x_2): **150**
- Máximo incremento para (x_3): **100**
- Máximo incremento para (x_4): **100**

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

Quitamos la columna 1 de la matriz (A), y el valor de (x_1 = 1000). El nuevo sistema es:

$$A' = \begin{bmatrix}2 & 0 & 3\\0 & 2 & 2\\0 & 1 & 1\end{bmatrix}, x' = \begin{bmatrix}500\\350\\400\end{bmatrix}$$

Consumo actual:

$$A' \cdot x' = \begin{bmatrix}2200\\1500\\750\end{bmatrix}$$

Suministro restante:

$$r = b - A' \cdot x' = \begin{bmatrix}1300\\1200\\150\end{bmatrix}$$

Ahora calculamos el incremento máximo **individual** para cada especie restante:

Especie 2:

$$\begin{bmatrix}2\\0\\1\end{bmatrix} x_2 \leq \begin{bmatrix}1300\\1200\\150\end{bmatrix} \Rightarrow x_2 \leq min\left(\frac{1300}{2}, \frac{150}{1}\right) = 150$$

Especie 3:

$$\begin{bmatrix}0\\2\\1\end{bmatrix} x_3 \leq \begin{bmatrix}1300\\1200\\150\end{bmatrix} \Rightarrow x_3 \leq min\left(\frac{1200}{2}, \frac{150}{1}\right) = 150$$

Especie 4:

$$\begin{bmatrix}3\\2\\1\end{bmatrix} x_4 \leq \begin{bmatrix}1300\\1200\\150\end{bmatrix} \Rightarrow x_4 \leq min\left(\frac{1300}{3}, \frac{1200}{2}, 150\right) = 150$$

Conclusión literal c:
Si la especie 1 se extingue, se puede agregar individualmente:

- Especie 2: hasta **150**
- Especie 3: hasta **150**
- Especie 4: hasta **150**

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

Quitamos la columna 2 de (A), y eliminamos el valor de (x_2 = 500):

$$A'' = \begin{bmatrix}1 & 0 & 3\\1 & 2 & 2\\0 & 1 & 1\end{bmatrix}, x'' = \begin{bmatrix}1000\\350\\400\end{bmatrix}$$

Consumo actual:

$$A'' \cdot x'' = \begin{bmatrix}1200\\1500\\750\end{bmatrix}$$

Suministro restante:

$$r = b - A'' \cdot x'' = \begin{bmatrix}2300\\1200\\150\end{bmatrix}$$

Ahora evaluamos el incremento máximo **individual**:

Especie 1:

$$\begin{bmatrix}1\\1\\0\end{bmatrix} x_1 \leq \begin{bmatrix}2300\\1200\\150\end{bmatrix} \Rightarrow x_1 \leq min(2300, 1200) = 1200$$

Especie 3:

$$\begin{bmatrix}0\\2\\1\end{bmatrix} x_3 \leq \begin{bmatrix}2300\\1200\\150\end{bmatrix} \Rightarrow x_3 \leq min\left(\frac{1200}{2}, 150\right) = 150$$

Especie 4:

$$\begin{bmatrix}3\\2\\1\end{bmatrix} x_4 \leq \begin{bmatrix}2300\\1200\\150\end{bmatrix} \Rightarrow x_4 \leq min\left(\frac{2300}{3}, \frac{1200}{2}, 150\right) = 150$$

Conclusión literal d:
Si la especie 2 se extingue, se puede agregar individualmente:

- Especie 1: hasta **1200**
- Especie 3: hasta **150**
- Especie 4: hasta **150**

7. EJERCICIOS TEÓRICOS

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

Literal a

Sistema:

$$\begin{aligned}\frac{1}{4}x_1+\frac{1}{5}x_2+\frac{1}{6}x_3 \quad &¿9\\ \frac{1}{3}x_1+\frac{1}{4}x_2+\frac{1}{5}x_3 \quad &¿8\\ \frac{1}{2}x_1+x_2+2x_3 \quad &¿8\end{aligned}$$

Matriz escalonada reducida:

$$\begin{bmatrix}1 & 0 & 0 & \frac{-2952}{13}\\ 0 & 1 & 0 & \frac{6200}{13}\\ 0 & 0 & 1 & \frac{-2310}{13}\end{bmatrix}$$

Solución:

$$x_1 = \frac{-2952}{13}, x_2 = \frac{6200}{13}, x_3 = \frac{-2310}{13}$$

Literal b

Sistema:

$$\begin{aligned} 3.3333\,x_1 + 15920\,x_2 - 10.3333\,x_3 &\quad ¿\,15913 \\ 2.2222\,x_1 + 16.71\,x_2 + 9.6123\,x_3 &\quad ¿\,28.544 \\ 1.5611\,x_1 + 5.1791\,x_2 + 1.6852\,x_3 &\quad ¿\,8.4254 \end{aligned}$$

Matriz escalonada reducida:

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 1.000075 \\ 0 & 1 & 0 & 0.9999999 \\ 0 & 0 & 1 & 0.9999308 \end{array}\right]$$

Solución (32 bits):

$$x_1 \approx 1.000075\,,\, x_2 \approx 1.000000\,,\, x_3 \approx 0.999931$$

Literal c

Sistema:

$$\begin{aligned} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 &\quad ¿\,\frac{1}{6} \\ \frac{1}{2}x_1 + x_2 + \frac{2}{3}x_3 + \frac{1}{5}x_4 &\quad ¿\,\frac{1}{7} \\ \frac{1}{3}x_1 + \frac{2}{3}x_2 + x_3 + \frac{1}{6}x_4 &\quad ¿\,\frac{1}{8} \\ \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + x_4 &\quad ¿\,\frac{1}{9} \end{aligned}$$

Matriz escalonada reducida:

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 0 & \frac{211}{1856} \\ 0 & 1 & 0 & 0 & \frac{1325}{33408} \\ 0 & 0 & 1 & 0 & \frac{3865}{77952} \\ 0 & 0 & 0 & 1 & \frac{7775}{116928} \end{array}\right]$$

Solución:

$$x_1 = \frac{211}{1856}\,,\, x_2 = \frac{1325}{33408}\,,\, x_3 = \frac{3865}{77952}\,,\, x_4 = \frac{7775}{116928}$$

Literal d

Sistema:

$$\begin{array}{ll} 2x_1+x_2-x_3-x_4-3x_5 & ¿7 \\ x_1+2x_3-4x_4+5x_5 & ¿2 \\ -2x_2-3x_3+4x_4-5x_5 & ¿-5 \\ 3x_1+x_2-2x_3-4x_4+5x_5 & ¿6 \\ x_1-x_2-x_3-x_4+x_5 & ¿-3 \end{array}$$

Matriz escalonada reducida:

$$\left(\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & \frac{1379}{488} \\ 0 & 1 & 0 & 0 & 0 & \frac{587}{244} \\ 0 & 0 & 1 & 0 & 0 & \frac{495}{488} \\ 0 & 0 & 0 & 1 & 0 & \frac{63}{122} \\ 0 & 0 & 0 & 0 & 1 & \frac{-77}{488} \end{array}\right)$$

Solución:

$$x_1=\frac{1379}{488}, x_2=\frac{587}{244}, x_3=\frac{495}{488}, x_4=\frac{63}{122}, x_5=\frac{-77}{488}$$

Código

```
import numpy as np
from scipy.linalg import lu

# Función para aplicar el método de Gauss-Jordan
def gauss_jordan(A, b):
    A = A.astype(np.float64)
    b = b.astype(np.float64)
    n = len(b)
    M = np.hstack((A, b.reshape(-1, 1)))

    for i in range(n):
        # Pivoteo parcial
        max_row = np.argmax(np.abs(M[i:, i])) + i
        M[[i, max_row]] = M[[max_row, i]]

        # Normalizar fila i
        M[i] = M[i] / M[i, i]
```

```
        # Eliminar otras filas
        for j in range(n):
            if i != j:
                M[j] -= M[j, i] * M[i]

    return M[:, :-1], M[:, -1]

# Literal a
A_a = np.array([
    [1/4, 1/5, 1/6],
    [1/3, 1/4, 1/5],
    [1/2, 1, 2]
])
b_a = np.array([9, 8, 8])
_, x_a = gauss_jordan(A_a, b_a)

# Literal b
A_b = np.array([
    [3.3333, 15920, -10.3333],
    [2.2222, 16.71, 9.6123],
    [1.5611, 5.1791, 1.6852]
], dtype=np.float32)
b_b = np.array([15913, 28.544, 8.4254], dtype=np.float32)
_, x_b = gauss_jordan(A_b, b_b)

# Literal c
A_c = np.array([
    [1, 1/2, 1/3, 1/4],
    [1/2, 1, 2/3, 1/5],
    [1/3, 2/3, 1, 1/6],
    [1/4, 1/5, 1/6, 1]
])
b_c = np.array([1/6, 1/7, 1/8, 1/9])
_, x_c = gauss_jordan(A_c, b_c)

# Literal d
A_d = np.array([
    [2, 1, -1, -1, -3],
    [1, 0, 2, -4, 5],
    [0, -2, -3, 4, -5],
    [3, 1, -2, -4, 5],
    [1, -1, -1, -1, 1]
])
b_d = np.array([7, 2, -5, 6, -3])
_, x_d = gauss_jordan(A_d, b_d)

x_a, x_b, x_c, x_d
```

```
(array([-227.07692308,  476.92307692, -177.69230769]),
 array([1.00007448, 0.99999994, 0.99993105]),
```

```
array([0.11368534, 0.03966116, 0.04958179, 0.06649391]),
array([7.47368421, 3.43859649, 3.59649123, 4.52631579, 1.0877193 ]))
```