ESCUELA POLITECNICA NACIONAL

Materia: Métodos Numéricos

Nombre: Stiv Quishpe

link repositorio

https://github.com/stiv001/Tarea-09.git

Eliminación gaussiana vs Gauss-Jordan

CONJUNTO DE EJERCICIOS

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.

a. $x_1 + 2x_2 = 0$, $x_1 - x_2 = 0$.

b. $x_1 + 2x_2 = 3$, $-2x_1 - 4x_2 = 6$.

c. $2x_1 + x_2 = -1$, $x_1 + x_2 = 2$, $x_1 - 3x_2 = 5$.

d. $2x_1 + x_2 + x_3 = 1$, $2x_1 + 4x_2 - x_3 = -1$.

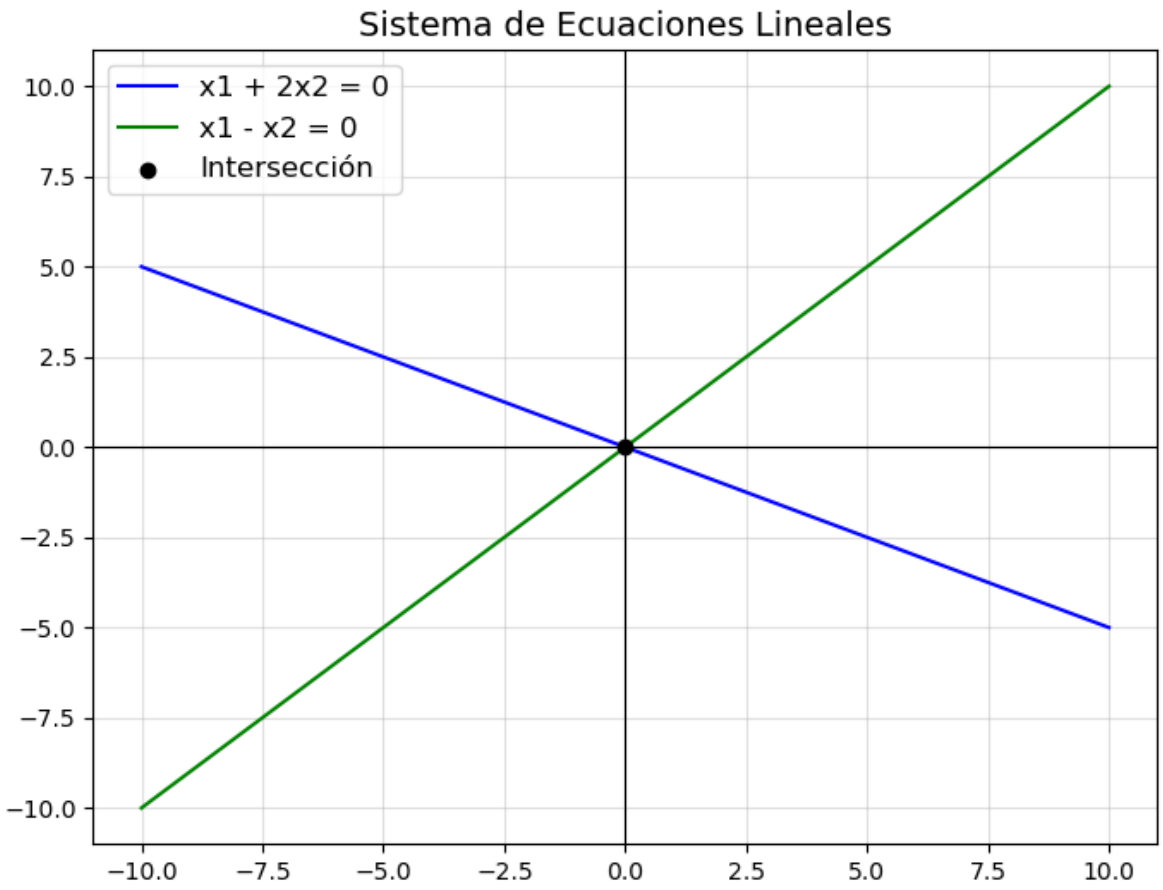
Figure 1: image.png

a. $x_1 + 2x_2 = 0$,
$x_1 - x_2 = 0$.

Figure 2: image.png

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

puntos_x = [0]
```

```
puntos_y = [0]

x = np.linspace(-10, 10, 500)

y1 = -x / 2
y2 = x

plt.figure(figsize=(8, 6))

plt.plot(x, y1, label="x1 + 2x2 = 0", color="blue")
plt.plot(x, y2, label="x1 - x2 = 0", color="g")

plt.scatter(puntos_x, puntos_y, color="k", label="Intersección", zorder=5)

plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)

plt.title("Sistema de Ecuaciones Lineales", fontsize=14)
plt.grid(alpha=0.4)
plt.legend(fontsize=12)

plt.show()
```



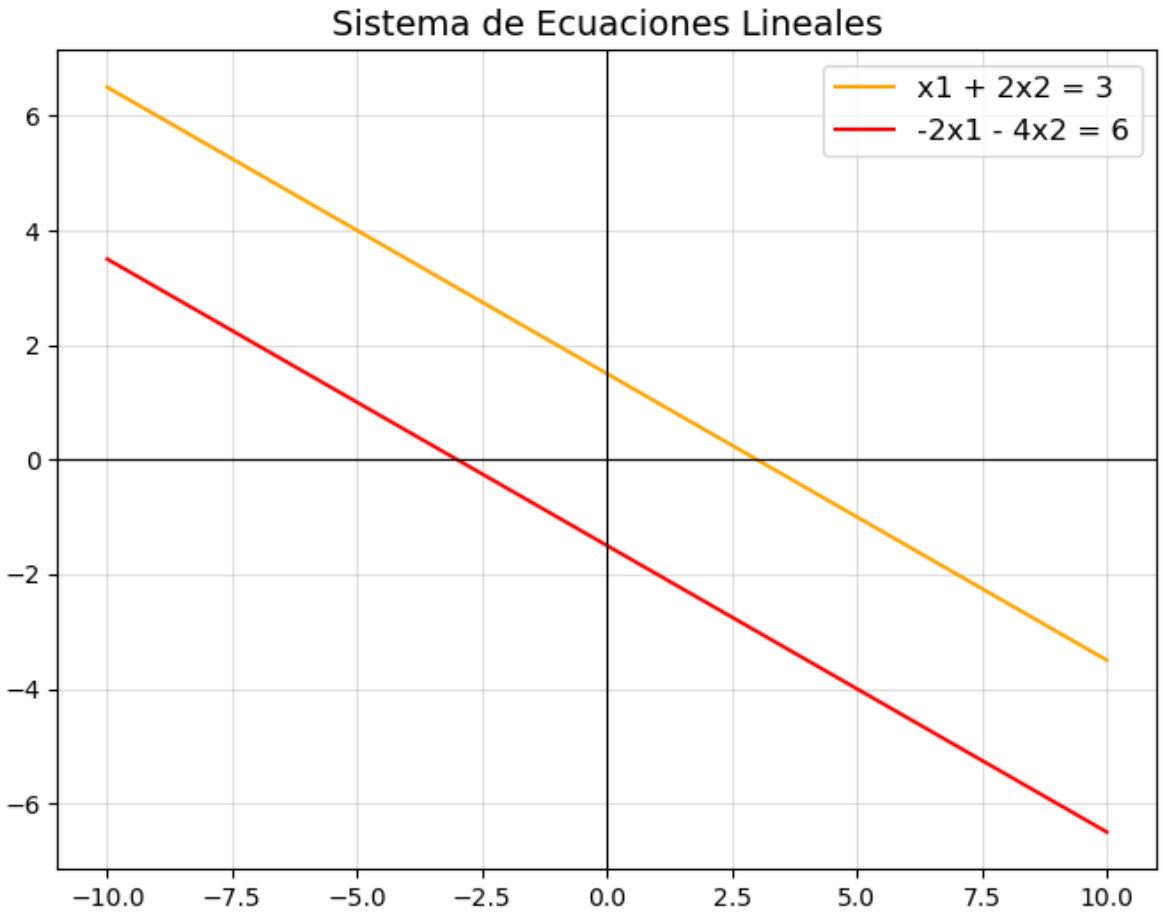
Las líneas no son ni paralelas ni completamente coincidentes, por lo tanto, tienen exactamente un punto de intersección, lo que significa que se cruzan en ese punto.

b. $$\begin{aligned} x_1 + 2x_2 &= 3, \\ -2x_1 - 4x_2 &= 6. \end{aligned}$$

Figure 3: image.png

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

x = np.linspace(-10, 10, 500)

y1 = (3 - x) / 2
y2 = (6 + 2 * x) / -4
```

```
plt.figure(figsize=(8, 6))

plt.plot(x, y1, label="x1 + 2x2 = 3", color="orange")
plt.plot(x, y2, label="-2x1 - 4x2 = 6", color="red")

plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)

plt.title("Sistema de Ecuaciones Lineales", fontsize=14)
plt.grid(alpha=0.4)
plt.legend(fontsize=12)

plt.show()
```



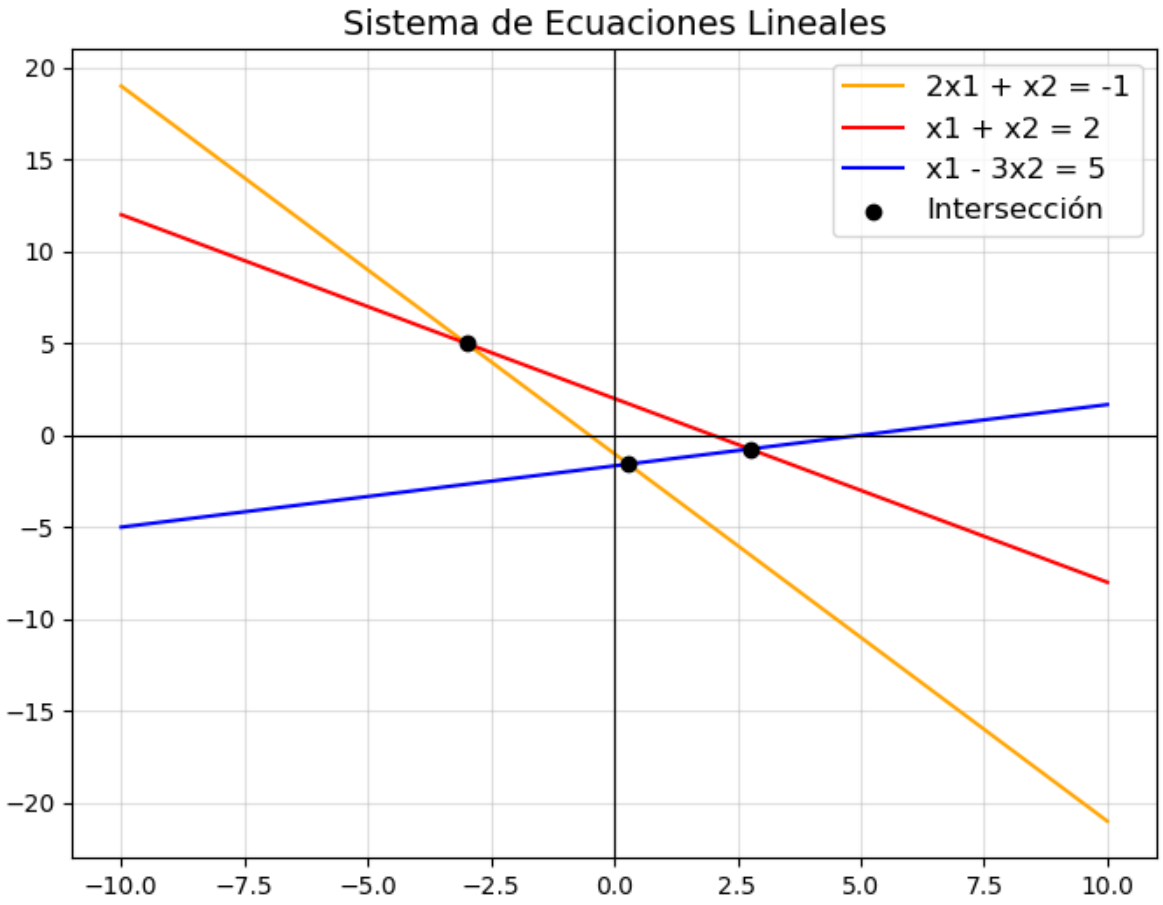
Este sistema de ecuaciones no tiene una solución ya que las funciones son paralelas y, por lo

tanto, nunca se cruzan en un punto.

$$\text{c. } \begin{aligned} 2x_1 + x_2 &= -1, \\ x_1 + x_2 &= 2, \\ x_1 - 3x_2 &= 5. \end{aligned}$$

Figure 4: image.png

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

x = np.linspace(-10, 10, 500)

y1 = -2 * x - 1
y2 = 2 - x
y3 = (x - 5) / 3

puntos_x = [-3, 2/7, 11/4]
puntos_y = [5, -11/7, -3/4]

plt.figure(figsize=(8, 6))

plt.plot(x, y1, label="2x1 + x2 = -1", color="orange")
plt.plot(x, y2, label="x1 + x2 = 2", color="red")
plt.plot(x, y3, label="x1 - 3x2 = 5", color="blue")

plt.scatter(puntos_x, puntos_y, color="k", label="Intersección", zorder=5)

plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)

plt.title("Sistema de Ecuaciones Lineales", fontsize=14)
plt.grid(alpha=0.4)
plt.legend(fontsize=12)

plt.show()
```



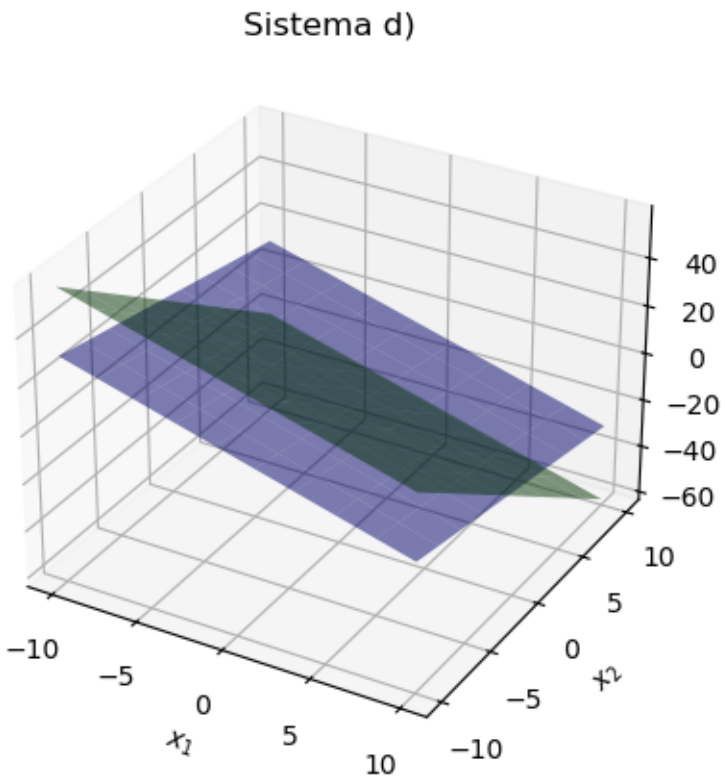
No hay un único punto donde las tres funciones se crucen. Además, este sistema de ecuaciones tiene tres ecuaciones con dos variables, lo que hace que no sea posible encontrar una solución.

$$\begin{aligned} \text{d. } & 2x_1 + \quad x_2 + x_3 = 1, \\ & 2x_1 + 4x_2 - x_3 = -1. \end{aligned}$$

Figure 5: image.png

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

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

```
def plane_z1(x, y):
    return 1 - 2*x - y

def plane_z2(x, y):
    return -1 - 2*x - 4*y

x_vals = np.linspace(-10, 10, 10)
y_vals = np.linspace(-10, 10, 10)
x_grid, y_grid = np.meshgrid(x_vals, y_vals)

z1_plane = plane_z1(x_grid, y_grid)
z2_plane = plane_z2(x_grid, y_grid)

ax.plot_surface(x_grid, y_grid, z1_plane, alpha=0.5, color='blue', edgecolor='none')
ax.plot_surface(x_grid, y_grid, z2_plane, alpha=0.5, color='green', edgecolor='none')

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

a. $-x_1 + 4x_2 + x_3 = 8,$
$\frac{5}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 = 1,$
$2x_1 + x_2 + 4x_3 = 11.$

b. $4x_1 + 2x_2 - x_3 = -5,$
$\frac{1}{9}x_1 + \frac{1}{9}x_2 - \frac{1}{3}x_3 = -1,$
$x_1 + 4x_2 + 2x_3 = 9,$

Figure 6: image.png

a. $-x_1 + 4x_2 + x_3 = 8,$
$\frac{5}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 = 1,$
$2x_1 + x_2 + 4x_3 = 11.$

Figure 7: image.png

```python
import numpy as np

Ab = np.array([
    [-1, 4, 1, 8],
    [5/3, 2/3, 2/3, 1],
    [2, 1, 4, 11]
], dtype=float)

A = Ab[:, :-1]
b = Ab[:, -1]

solucion = np.linalg.solve(A, b)

print("La solución del sistema es:")
print(f"x1 = {solucion[0]:.2f}")
print(f"x2 = {solucion[1]:.2f}")
print(f"x3 = {solucion[2]:.2f}")
```

```
La solución del sistema es:
x1 = -1.00
x2 = 1.00
x3 = 3.00
```

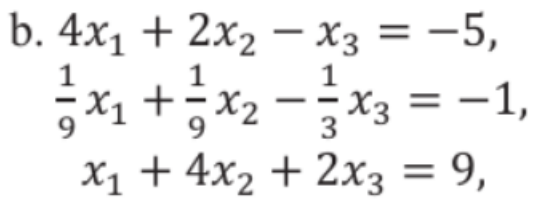


Figure 8: image.png

```python
import numpy as np

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

    for i in range(n):
        pivote = A[i, i]
```

```
        A[i, :] = np.round(A[i, :] / pivote, 2)
        b[i] = np.round(b[i] / pivote, 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_b = [
    [4, 2, -1],
    [1/9, 1/9, -1/3],
    [1, 4, 2]
]
b_b = [-5, -1, 9]

solucion_a = eliminacion_gaussiana_con_redondeo(A_b, b_b)

print("La solución del sistema es:")
print(f"x1 = {solucion_a[0]:.2f}")
print(f"x2 = {solucion_a[1]:.2f}")
print(f"x3 = {solucion_a[2]:.2f}")
```

```
La solución del sistema es:
x1 = -1.02
x2 = 1.02
x3 = 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:

a. $\begin{aligned} x_1 - x_2 + 3x_3 &= 2, \\ 3x_1 - 3x_2 + x_3 &= -1, \\ x_1 + x_2 &= 3. \end{aligned}$

b. $\begin{aligned} 2x_1 - 1.5x_2 + 3x_3 &= 1, \\ -x_1 + 2x_3 &= 3, \\ 4x_1 - 4.5x_2 + 5x_3 &= 1, \end{aligned}$

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. $\begin{aligned} x_1 + x_2 + x_4 &= 2, \\ 2x_1 + x_2 - x_3 + x_4 &= 1, \\ 4x_1 - x_2 - 2x_3 + 2x_4 &= 0, \\ 3x_1 - x_2 - x_3 + 2x_4 &= -3. \end{aligned}$

Figure 9: image.png

```python
import numpy as np

def eliminacion_gaussiana_con_pivoteo(A, b):
    n = len(b)
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    intercambios = []

    for i in range(n):
        fila_max = np.argmax(np.abs(A[i:, i])) + i
        if fila_max != i:
            A[[i, fila_max]] = A[[fila_max, i]]
            b[[i, fila_max]] = b[[fila_max, i]]
            intercambios.append((i, fila_max))

        pivote = A[i, i]
        A[i, :] /= pivote
        b[i] /= pivote

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

a. $$\begin{aligned} x_1 - x_2 + 3x_3 &= 2, \\ 3x_1 - 3x_2 + x_3 &= -1, \\ x_1 + x_2 &= 3. \end{aligned}$$

Figure 10: image.png

```
A = [
    [1, -1, 3],
    [3, -3, 1],
    [1, 1, 0]
]
b = [2, -1, 3]

solucion, intercambios = eliminacion_gaussiana_con_pivoteo(A, b)

print("La solución del sistema es:")
print(f"x1 = {solucion[0]:.2f}")
print(f"x2 = {solucion[1]:.2f}")
print(f"x3 = {solucion[2]:.2f}")
print("Intercambios realizados:", intercambios)
```

```
La solución del sistema es:
x1 = 1.19
x2 = 1.81
x3 = 0.88
Intercambios realizados: [(0, 1), (1, 2)]
```

b. $$\begin{aligned} 2x_1 - 1.5x_2 + 3x_3 &= 1, \\ -x_1 + 2x_3 &= 3, \\ 4x_1 - 4.5x_2 + 5x_3 &= 1, \end{aligned}$$

Figure 11: image.png

```
A = [
    [2, -1.5, 3],
    [-1, 0, 2],
    [4, -4.5, 5]
]
b = [1, 3, 1]

solucion, intercambios = eliminacion_gaussiana_con_pivoteo(A, b)

print("La solución del sistema es:")
print(f"x1 = {solucion[0]:.2f}")
print(f"x2 = {solucion[1]:.2f}")
print(f"x3 = {solucion[2]:.2f}")
print("Intercambios realizados:", intercambios)
```

```
La solución del sistema es:
x1 = -1.00
x2 = 0.00
x3 = 1.00
Intercambios realizados: [(0, 2)]
```

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

Figure 12: image.png

```
A = [
    [2, 0, 0, 0],
    [1, 1.5, 0, 0],
    [0, -3, 0.5, 0],
    [2, -2, 1, 1]
]
b = [3, 4.5, -6.6, 0.8]

solucion, intercambios = eliminacion_gaussiana_con_pivoteo(A, b)
```

```
print("La solución del sistema es:")
print(f"x1 = {solucion[0]:.2f}")
print(f"x2 = {solucion[1]:.2f}")
print(f"x3 = {solucion[2]:.2f}")
print("Intercambios realizados:", intercambios)
```

```
La solución del sistema es:
x1 = 1.50
x2 = 2.00
x3 = -1.20
Intercambios realizados: [(1, 2), (2, 3)]
```

d.
$$\begin{aligned} x_1 + x_2 + \quad\quad x_4 &= 2, \\ 2x_1 + x_2 - \ x_3 + \ x_4 &= 1, \\ 4x_1 - x_2 - 2x_3 + 2x_4 &= 0, \\ 3x_1 - x_2 - \ x_3 + 2x_4 &= -3. \end{aligned}$$

Figure 13: image.png

```
A = [
    [1, 1, 1, 1],
    [2, 1, -1, -1],
    [4, -1, 2, 0],
    [3, -1, -2, 2]
]
b = [2, 1, 0, -3]

solucion, intercambios = eliminacion_gaussiana_con_pivoteo(A, b)

print("La solución del sistema es:")
print(f"x1 = {solucion[0]:.2f}")
print(f"x2 = {solucion[1]:.2f}")
print(f"x3 = {solucion[2]:.2f}")
print(f"x3 = {solucion[3]:.2f}")
print("Intercambios realizados:", intercambios)
```

```
La solución del sistema es:
```

```
x1 = 0.03
x2 = 1.45
x3 = 0.67
x3 = -0.15
Intercambios realizados: [(0, 2), (2, 3)]
```

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

a. $\frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 = 9,$
$\frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = 8,$
$\frac{1}{2}x_1 + x_2 + 2x_3 = 8.$

b. $3.333x_1 + 15920x_2 - 10.333x_3 = 15913,$
$2.222x_1 + 16.71x_2 + 9.612x_3 = 28.544,$
$1.5611x_1 + 5.1791x_2 + 1.6852x_3 = 8.4254.$

c. $x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 = \frac{1}{6},$
$\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 = \frac{1}{7},$
$\frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 = \frac{1}{8},$
$\frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 = \frac{1}{9}.$

d. $2x_1 + x_2 - x_3 + x_4 - 3x_5 = 7,$
$x_1 + 2x_3 - x_4 + x_5 = 2,$
$-2x_2 - x_3 + x_4 - x_5 = -5,$
$3x_1 + x_2 - 4x_3 + 5x_5 = 6,$
$x_1 - x_2 - x_3 - x_4 + x_5 = -3.$

Figure 14: image.png

a. $\frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 = 9,$
$\frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = 8,$
$\frac{1}{2}x_1 + x_2 + 2x_3 = 8.$

Figure 15: image.png

```python
import numpy as np

A_a = np.array([
    [1/4, 1/5, 1/6],
    [1/3, 1/4, 1/5],
    [1/2, 1.0, 2.0]
], 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("La solución del sistema es:")
print(x_a)
print()
```

```
La solución del sistema es:
[-227.0767   476.92267 -177.69215]
```

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

Figure 16: image.png

```
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("La solución del sistema es:")
print(x_b)
print()
```

```
La solución del sistema es:
[0.99999964 1.         1.0000002 ]
```

$$
\begin{aligned}
\text{c.}\quad & x_1 + \tfrac{1}{2}x_2 + \tfrac{1}{3}x_3 + \tfrac{1}{4}x_4 = \tfrac{1}{6},\\
& \tfrac{1}{2}x_1 + \tfrac{1}{3}x_2 + \tfrac{1}{4}x_3 + \tfrac{1}{5}x_4 = \tfrac{1}{7},\\
& \tfrac{1}{3}x_1 + \tfrac{1}{4}x_2 + \tfrac{1}{5}x_3 + \tfrac{1}{6}x_4 = \tfrac{1}{8},\\
& \tfrac{1}{4}x_1 + \tfrac{1}{5}x_2 + \tfrac{1}{6}x_3 + \tfrac{1}{7}x_4 = \tfrac{1}{9}.
\end{aligned}
$$

Figure 17: image.png

```
A_c = np.array([
    [ 1.0,     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)

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("La solución del sistema es:")
print(x_c)
print()
```

```
La solución del sistema es:
[-1.0240577e-03  3.4946290e-01 -1.8894023e+00  2.4910402e+00]
```

$$
\begin{aligned}
\text{d.}\quad & 2x_1 + x_2 - x_3 + x_4 - 3x_5 = 7,\\
& x_1 + 2x_3 - x_4 + x_5 = 2,\\
& -2x_2 - x_3 + x_4 - x_5 = -5,\\
& 3x_1 + x_2 - 4x_3 + 5x_5 = 6,\\
& x_1 - x_2 - x_3 - x_4 + x_5 = -3.
\end{aligned}
$$

Figure 18: image.png

```
A_d = np.array([
    [ 2.0,  1.0, -1.0,  1.0, -3.0],
    [ 1.0,  0.0,  2.0, -1.0,  1.0],
    [ 0.0, -2.0, -1.0,  1.0, -1.0],
    [ 3.0,  1.0, -4.0,  0.0,  5.0],
    [ 1.0, -2.0, -1.0, -1.0,  1.0]
], dtype=np.float32)

b_d = np.array([ 7.0, 2.0, -5.0, 6.0, -3.0], dtype=np.float32)

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

print("La solución del sistema es:")
print(x_d)
print()
```

```
La solución del sistema es:
[ 1.8974359   2.4615386   0.02564103 -0.46153846 -0.41025642]
```

5. Dado el sistema lineal:

$$\begin{aligned} x_1 - x_2 + \alpha x_3 &= -2, \\ -x_1 + 2x_2 - \alpha x_3 &= 3, \\ \alpha x_1 + x_2 + x_3 &= 2. \end{aligned}$$

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 a determinada, encuentre la solución.

Figure 19: image.png

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

Método de Gauss.

$$\begin{bmatrix} 1 & -1 & \alpha & -2 \\ -1 & 2 & -\alpha & 3 \\ \alpha & 1 & 1 & 2 \end{bmatrix} \equiv F1{+}F2{->}F2 \begin{bmatrix} 1 & -1 & -\alpha & -2 \\ 0 & 1 & 0 & 1 \\ \alpha & 1 & 1 & 2 \end{bmatrix} \equiv (-x{-}1)F2{+}F3{->}F3 \begin{bmatrix} 1 & -1 & -\alpha & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -\alpha^2+1 & \alpha+ \end{bmatrix}$$

Para que el sistema no tenga solución, se debe cumplir que $-\alpha^2 + 1 = 0$ y que $\alpha + 1 \neq 0$,A continuación, verificaremos estas condiciones:

$$-\alpha^2 + 1 = 0 - \alpha^2 = -1\alpha^2 = 1\alpha = \pm 1$$

y

$$\alpha + 1 \neq 0 \alpha \neq -1$$

por lo tanto

$$\alpha = 1$$

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

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

Para cumplir con la condición debemos garantizar que $-\alpha^2 + 1 = 0$ y $\alpha + 1 = 0$ por ende

$\alpha = \pm 1 \wedge \alpha = -1 \therefore \alpha = -1$

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

Para que el sistema tenga una unica solución $\alpha \neq \pm 1$ por lo que escogeremos el valor de $\alpha = 0$. Al reemplazar el valor de α en la matriz resultante obtendremos lo siguiente:

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

De donde se obtienen las ecuaciones:

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

reemplazando:

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

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

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ \vdots \quad & \quad \vdots \quad \quad \vdots \quad \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{aligned}$$

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

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

Figure 20: image.png

a. Si $x = (x_j) = [1000, 500, 350, 400]$ y $b = (b_i) = [3500, 2700, 900]$.¿Existe suficiente alimento para satisfacer el consumo promedio diario?

```
import numpy as np

A = np.array([
    [1, 2, 0],
    [1, 0, 2],
    [0, 0, 1]
], dtype=float)

x = np.array([1000, 500, 350], dtype=float)

b = np.array([3500, 2700, 900], dtype=float)

consumo_actual = A @ x

print("Consumo total actual de cada recurso:")
print(consumo_actual)

print("Disponibilidad de cada recurso (b):")
print(b)

sobran = b - consumo_actual

if np.all(sobran >= 0):
    print("\n(a) Sí: el alimento alcanza para cubrir el consumo promedio diario.")
```

```
    print("    Sobran cantidades (recurso a recurso):", sobran)
else:
    print("\n(a) No: el alimento NO alcanza para cubrir el consumo en alguna fuente.")
```

```
Consumo total actual de cada recurso:
[2000. 1700.  350.]
Disponibilidad de cada recurso (b):
[3500. 2700.  900.]

(a) Sí: el alimento alcanza para cubrir el consumo promedio diario.
    Sobran cantidades (recurso a recurso): [1500. 1000.  550.]
```

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?

```
import numpy as np

max_adicionales = np.zeros_like(x)

for j in range(len(x)):
    numeradores = []
    for i in range(A.shape[0]):
        if A[i, j] > 0:
            remanente_i = b[i] - consumo_actual[i]
            cota = remanente_i / A[i, j]
            numeradores.append(cota)

    if numeradores:
        Ymax = np.min(numeradores)
    else:
        Ymax = float('inf')

    max_adicionales[j] = max(0, np.floor(Ymax))

for j in range(len(x)):
    print(f"   - Especie {j+1}: se pueden agregar hasta {max_adicionales[j]} individuos extra
```

```
   - Especie 1: se pueden agregar hasta 1000.0 individuos extra (por separado)
   - Especie 2: se pueden agregar hasta 750.0 individuos extra (por separado)
   - Especie 3: se pueden agregar hasta 500.0 individuos extra (por separado)
```

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

```
import numpy as np

x_ext1 = x.copy()
x_ext1[0] = 0

consumo_ext1 = A @ x_ext1
remanente_ext1 = b - consumo_ext1

print("Nuevo consumo de recursos:")
print(consumo_ext1)
print("Nuevo remanente de cada recurso:")
print(remanente_ext1)

print("Incrementos posibles de manera individual para especies 2 y 3:")
for j in [1, 2]:
    numeradores = []
    for i in range(A.shape[0]):
        if A[i, j] > 0:
            cota = (b[i] - consumo_ext1[i]) / A[i, j]
            numeradores.append(cota)
    if numeradores:
        Ymax = np.min(numeradores)
    else:
        Ymax = float('inf')
    print(f"   - Especie {j+1}: se pueden agregar aproximadamente {np.floor(Ymax)} individuos
```

```
Nuevo consumo de recursos:
[1000.  700.  350.]
Nuevo remanente de cada recurso:
[2500. 2000.  550.]
Incrementos posibles de manera individual para especies 2 y 3:
   - Especie 2: se pueden agregar aproximadamente 1250.0 individuos extra.
   - Especie 3: se pueden agregar aproximadamente 550.0 individuos extra.
```

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

```
import numpy as np

x_ext2 = x.copy()
x_ext2[1] = 0

consumo_ext2 = A @ x_ext2
remanente_ext2 = b - consumo_ext2

print("Nuevo consumo de recursos:")
print(consumo_ext2)
print("Nuevo remanente de cada recurso:")
print(remanente_ext2)

print("Incrementos posibles de manera individual para especies 1 y 3:")
for j in [0, 2]:
    numeradores = []
    for i in range(A.shape[0]):
        if A[i, j] > 0:
            cota = (b[i] - consumo_ext2[i]) / A[i, j]
            numeradores.append(cota)
    if numeradores:
        Ymax = np.min(numeradores)
    else:
        Ymax = float('inf')
    print(f"   - Especie {j+1}: se pueden agregar aproximadamente {np.floor(Ymax)} individuos
```

```
Nuevo consumo de recursos:
[1000. 1700.  350.]
Nuevo remanente de cada recurso:
[2500. 1000.  550.]
Incrementos posibles de manera individual para especies 1 y 3:
   - Especie 1: se pueden agregar aproximadamente 1000.0 individuos extra.
   - Especie 3: se pueden agregar aproximadamente 500.0 individuos extra.
```