Guía de laboratorio 3: Resolución de Sistemas de Ecuaciones Lineales.

1. Introducción

La resolución de sistemas de ecuaciones lineales constituye una de las herramientas más importantes en ingeniería, ciencias y economía. Existen diversos métodos para resolverlos, que se clasifican en directos (como Gauss y Gauss-Jordan) e iterativos (como Jacobi y Gauss-Seidel)..

2. Objetivos

- Comprender la formulación matemática de un sistema de ecuaciones lineales.
- Aplicar métodos directos (Gauss y Gauss-Jordan) y iterativos (Jacobi y Gauss-Seidel).
- Implementar los algoritmos en pseudocódigo y Python.
- Comparar la eficiencia y precisión de cada método.
- Relacionar las operaciones de sistemas de ecuaciones lineales con aplicaciones reales en ingeniería.
- Desarrollar habilidades de programación en Python orientadas a la solución de sistema de ecuaciones lineales con NumPy.

3. Materiales y Recursos

- Computadora personal o laboratorio de cómputo.
- Python 3.10+ instalado (preferible en Anaconda).
- Librería NumPy instalada (pip install numpy).
- Editor de código (Sublime Text, Visual Studio Code o Jupyter Notebook).
- Opcional (crear un ambiente virtual)

4. Ejercicios y Desarrollo

4.1 Método de Triangulación de Gauss

Ejercicio 1 (2 variables):

$$\begin{aligned} 2x + 3y &= 8 \\ x - y &= 1 \end{aligned}$$

Ejercicio 2 (3 variables):

$$\begin{aligned} 3x + 2y - z &= 1 \\ 2x - 2y + 4z &= -2 \\ -x + \tfrac{1}{2}y - z &= 0 \end{aligned}$$

a. **Realizar Pseudocódigo**
b. **Codificar en Python**

```python
# MÉTODO DE TRIANGULACIÓN DE GAUSS

import numpy as np

def gauss_elimination(A, b):
    n = len(b)
    M = np.hstack([A.astype(float), b.reshape(-1,1)])
```

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

        for i in range(k+1, n):
            factor = M[i][k] / M[k][k]
            M[i] = M[i] - factor * M[k]

    # Sustitución regresiva
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (M[i, -1] - np.dot(M[i,i+1:n], x[i+1:n])) / M[i,i]
    return x

# Ejemplo 1
A = np.array([[2, 3],
              [1, -1]])
b = np.array([8, 1])
print("Solución Ejercicio 1 (Gauss):", gauss_elimination(A, b))

# Ejemplo 2
A = np.array([[3, 2, -1],
              [2, -2, 4],
              [-1, 0.5, -1]])
b = np.array([1, -2, 0])
print("Solución Ejercicio 2 (Gauss):", gauss_elimination(A, b))
```

4.2 Método de Gauss-Jordan

Ejercicio 1 (2 variables):

$$x + 2y = 5$$
$$3x - y = 4$$

Ejercicio 2 (3 variables):

$$2x + y - z = 8$$
$$-3x - y + 2z = -11$$
$$-2x + y + 2z = -3$$

c. **Realizar Pseudocódigo**
d. **Codificar en Python**

```
#METODO DE GAUSS-JORDAN

import numpy as np
```

```python
def gauss_jordan(A, b):
    n = len(b)
    M = np.hstack([A.astype(float), b.reshape(-1,1)])

    for k in range(n):
        M[k] = M[k] / M[k][k]
        for i in range(n):
            if i != k:
                M[i] = M[i] - M[i][k] * M[k]
    return M[:, -1]

# Ejemplo 1
A = np.array([[1, 2],
              [3, -1]])
b = np.array([5, 4])
print("Solución Ejercicio 1 (Gauss-Jordan):", gauss_jordan(A, b))

# Ejemplo 2
A = np.array([[2, 1, -1],
              [-3, -1, 2],
              [-2, 1, 2]])
b = np.array([8, -11, -3])
print("Solución Ejercicio 2 (Gauss-Jordan):", gauss_jordan(A, b))
```

4.3 Método iterativo de Jacobi

Ejercicio 1 (2 variables):

$$10x + y = 9$$
$$x + 10y = 20$$

Ejercicio 2 (3 variables):

$$4x - y + z = 7$$
$$x + 5y - z = -8$$
$$2x + y + 6z = 6$$

e. **Realizar Pseudocódigo**
f. **Codificar en Python**

```python
# METODO ITERATIVO DE JACOBI
import numpy as np

def jacobi(A, b, tol=1e-6, max_iter=100):
    n = len(b)
    x = np.zeros(n)
    for _ in range(max_iter):
```

```
        x_new = np.zeros(n)
        for i in range(n):
            s = sum(A[i][j]*x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i][i]
        if np.linalg.norm(x_new - x, ord=np.inf) < tol:
            return x_new
        x = x_new
    return x

# Ejemplo 1
A = np.array([[10, 1],
              [1, 10]])
b = np.array([9, 20])
print("Solución Ejercicio 1 (Jacobi):", jacobi(A, b))

# Ejemplo 2
A = np.array([[4, -1, 1],
              [1, 5, -1],
              [2, 1, 6]])
b = np.array([7, -8, 6])
print("Solución Ejercicio 2 (Jacobi):", jacobi(A, b))
```

4.4 Método iterativo de Gauss-Seidel

Ejercicio 1 (2 variables):

$$\begin{aligned} 4x + y &= 15 \\ x + 3y &= 10 \end{aligned}$$

Ejercicio 2 (3 variables):

$$\begin{aligned} 3x + y + z &= 1 \\ x + 4y + z &= 2 \\ x + y + 5z &= 0 \end{aligned}$$

g. Realizar Pseudocódigo

h. Codificar en Python

```
# METODO DE GAUSS-SEIDEL
import numpy as np

def gauss_seidel(A, b, tol=1e-6, max_iter=100):
    n = len(b)
    x = np.zeros(n)
    for _ in range(max_iter):
        x_new = np.copy(x)
        for i in range(n):
            s = sum(A[i][j]*x_new[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i][i]
```

```
        if np.linalg.norm(x_new - x, ord=np.inf) < tol:
            return x_new
        x = x_new
    return x

# Ejemplo 1
A = np.array([[4, 1],
              [1, 3]])
b = np.array([15, 10])
print("Solución Ejercicio 1 (Gauss-Seidel):", gauss_seidel(A, b))

# Ejemplo 2
A = np.array([[3, 1, 1],
              [1, 4, 1],
              [1, 1, 5]])
b = np.array([1, 2, 0])
print("Solución Ejercicio 2 (Gauss-Seidel):", gauss_seidel(A, b))
```