Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

# Análisis Numérico

## Evidencia 1:

"Método de Eliminación Gaussiana"


Alumno: Jesús Armando Espino Rodríguez

Matricula: 1844607


Profesora: María del Carmen Martínez Cejudo

Grupo: 031

Horario: 09:00 am a 10:00 am

# Método de Eliminación Gaussiana

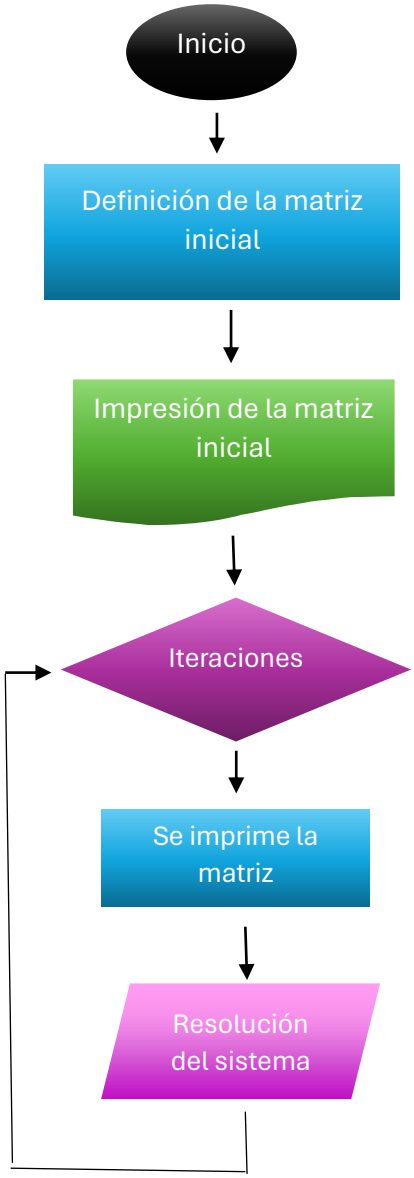
- El método de eliminación gaussiana, también conocido como eliminación por escalonamiento o eliminación en triángulo, es una técnica fundamental en el ámbito del álgebra lineal.

- Su objetivo principal es resolver sistemas de ecuaciones lineales mediante la transformación sistemática de la matriz de coeficientes en una forma triangular, lo que facilita enormemente la resolución de sistemas de ecuaciones.

- Esta técnica implica una serie de pasos organizados para eliminar variables de manera gradual, llevando la matriz de coeficientes a una forma escalonada o triangular. Este proceso simplifica el sistema de ecuaciones y permite resolverlo de manera eficiente utilizando métodos como la sustitución hacia atrás.

- La eliminación gaussiana es ampliamente utilizada en campos como la física, la ingeniería y la ciencia de datos, proporcionando un enfoque estructurado y poderoso para la resolución de problemas que involucran sistemas de ecuaciones lineales.

# Diagrama de Flujo




- El diagrama comenzaría con un nodo de inicio, indicando el inicio del programa.

- Se debe definir la matriz inicial que se utilizará para resolver el sistema de ecuaciones lineales.

- Se imprimiría la matriz inicial en la consola para mostrar el estado inicial del sistema.

- Se comienza la iteración del método de Gauss
- Por cada iteración se realiza la eliminación de Gauss

- Se imprime la matriz después de cada iteración para mostrar los cambios.

- Una vez que la matriz está en forma triangular superior, se resuelve el sistema de ecuaciones lineales.

# Codificación del programa

## Función: Gauss_elimination

```python
def gauss_elimination(matrix):
    n = len(matrix)            # Obtiene el tamaño de la matriz (número de filas)

    # Imprime la matriz inicial
    print("Matriz inicial:")
    print_matrix(matrix)

    # Aplica el método de eliminación de Gauss
    for i in range(n):
        # Hacer la columna i en la matriz triangular
        for j in range(i+1, n):
            # Calcula el factor de eliminación
            factor = matrix[j][i] / matrix[i][i]
            # Actualiza los elementos de la fila j
            for k in range(i, n+1):
                matrix[j][k] -= factor * matrix[i][k]
        # Imprime la matriz en cada iteración
        print(f"Iteración {i+1}:")
        print_matrix(matrix)
    # Resolver el sistema triangular superior
    solutions = [0] * n
    for i in range(n-1, -1, -1):
        solutions[i] = matrix[i][n] / matrix[i][i]
        for j in range(i-1, -1, -1):
            matrix[j][n] -= matrix[j][i] * solutions[i]
    return solutions
```

- En la función gauss_elimination se introduce una matriz como parámetro, para después inicializa una variable con la longitud de la matriz ingresada.
- Después entramos en una red de ciclos en los cuales se llevará a cabo la eliminación de gauss.
- En el primer ciclo entramos en la primera fila de la matriz, para posteriormente entrar en un segundo ciclo que divide entre el factor de eliminación para después entrar en el tercer ciclo y restar el valor generado de manera que se logra la forma triangular del método de eliminación gaussiana.
- En cada final del primer ciclo se imprime la matriz, con sus nuevos valores.
- Por último, resolvemos el sistema de ecuaciones despejando y comparando valores, los cuales estarán almacenados en el arreglo de solutions.

## Impresión de los cambios en la matriz

```python
def print_matrix(matrix):
    for row in matrix:
        print(row)
    print()
```

- En esta definimos una función que nos permite imprimir la matriz de manera ordenada

```
Matriz inicial:
[2, 1, 1, 8]
[1, 1, 2, 3]
[2, 2, 1, 3]

Iteración 1:
[2, 1, 1, 8]
[0.0, 0.5, 1.5, -1.0]
[0.0, 1.0, 0.0, -5.0]

Iteración 2:
[2, 1, 1, 8]
[0.0, 0.5, 1.5, -1.0]
[0.0, 0.0, -3.0, -3.0]

Iteración 3:
[2, 1, 1, 8]
[0.0, 0.5, 1.5, -1.0]
[0.0, 0.0, -3.0, -3.0]
```

- Aquí tenemos un ejemplo de las impresiones por iteraciones así de cómo es definida y mandada a llamar la función de print_matrix.
- Podemos ver como los cambios van generándose de manera que se logra la forma triangular del método de eliminación gaussiana

```python
# Ejemplo de uso
matrix = [
    [2, 1, 1, 8],
    [1, 1, 2, 3],
    [2, 2, 1, 3]
]

# Llama a la función gauss_elimination con la matriz de entrada
solutions = gauss_elimination(matrix)

# Imprime las soluciones encontradas
print("Las soluciones son:", solutions)
```

- Primero tenemos la declaración de una matriz (usada como ejemplo a lo largo del documento), en esta parte podemos definir diferentes matrices cambiando los valores ingresados.
- Posteriormente declaramos una variable solutions, a la cual le damos como valor el resultado de la función gauss_elimination, para después imprimir los resultados obtenidos.

```
Las soluciones son: [6.0, -5.0, 1.0]
```

- Aquí vemos el vector de soluciones, el cual nos muestra los valores obtenidos de las variables de la matriz ingresada.

## Conclusiones

- Durante la implementación del programa se presentaron varias dificultades, referentes a la lógica y manejo correcto de las matrices, fue un desafío un tanto interesante programar este método, pese a ser el mas sencillo de los métodos vistos en clase, este me hizo pensar mucho con los resultados que arrojaban los ciclos anidados, al final se logró con éxito la implementación y diseño de este.

- Fue entretenido elaborar este documento, se hizo con la intención de ser llamativo y de fácil comprensión, siendo un guía en cada paso de la elaboración e introducción del método.

- Espero que sea de su agrado y éxito a todos.

## Referencias Bibliográficas


- Kolman, Bernard, y Hill, David R. (2019). "Elementary Linear Algebra with Applications". Pearson.

- Python Documentation: https://docs.python.org/3/library/array.html