

### **Análisis Numérico**

#### **Proyecto 1: Método Montante**

**Grupo:** 037

**Alumno**

Carlos Enrique Castillo Mayorga

Maximiliano Rada Moreno

**Matricula**

1965541

1723609

Monterrey, Nuevo León, al 17 de noviembre del 2022

# Índice

Introducción.....	2
Marco teórico.....	3
1. Sistemas de ecuaciones lineales .....	3
2. Métodos de resolución de sistemas de ecuaciones.....	3
3. El método Montante .....	4
3.1. Principios fundamentales del método Montante .....	4
3.2. Proceso de aplicación del método Montante.....	4
4. Aplicaciones del método Montante.....	5
5. Ventajas y desventajas del método Montante.....	5
5.1. Ventajas.....	5
5.2. Desventajas.....	5
Código .....	6
Manual de usuario.....	13
Requisitos .....	13
Ejecución del Programa.....	13
Uso.....	13
Mensajes de Error.....	15
Ejemplo de ejecución.....	16
Ejecución exitosa.....	16
Ejecución matriz no invertible.....	17
Ejecución con matriz con diagonal con 0 .....	18
Ejecución con matriz con solo 0 .....	19
Calcular otra matriz .....	19
Diagrama de Flujo.....	20
Conclusión .....	21
Carlos Enrique Castillo Mayorga.....	21
Maximiliano Rada Moreno.....	21
Bibliografía.....	21

# Introducción

El método Montante es una técnica del análisis numérico utilizada para resolver sistemas de ecuaciones lineales de manera eficiente y precisa. Se considera una variante del método de eliminación de Gauss, con la diferencia fundamental de que evita las divisiones durante el proceso de eliminación, lo que reduce significativamente los errores de redondeo y mejora la estabilidad numérica en ciertos casos. Este método fue desarrollado en la década de 1970 por el matemático mexicano René Montante Pardo, quien lo diseñó con el objetivo de optimizar la resolución de sistemas de ecuaciones lineales sin comprometer la exactitud de los resultados.

El método Montante se basa en la transformación de la matriz aumentada del sistema mediante operaciones elementales sobre sus filas. A diferencia de otros métodos de eliminación, en lugar de utilizar divisiones, se emplea un esquema basado en multiplicaciones con un factor pivote, lo que permite mantener coeficientes enteros o racionales a lo largo del procedimiento. Este enfoque tiene la ventaja de preservar la precisión en los cálculos y evitar la propagación de errores numéricos, algo crucial en sistemas con coeficientes fraccionarios o números grandes.

Una de las principales aplicaciones del método Montante es en la resolución de sistemas de ecuaciones en álgebra lineal, especialmente en situaciones donde se requiere alta precisión en los cálculos. Además, su estructura facilita la implementación computacional, ya que el procedimiento es sistemático y no requiere el uso de fracciones en cada paso intermedio. Esto lo hace útil en criptografía, donde la exactitud de los valores es esencial, así como en modelos matemáticos aplicados en ingeniería, economía y otras áreas científicas.

Otra ventaja del método Montante es su facilidad de comprensión y aplicación en comparación con otros métodos como Gauss-Jordan o eliminación de Gauss, ya que el esquema de cálculos se mantiene uniforme en todas las etapas. Su estabilidad numérica lo convierte en una opción atractiva para problemas donde la acumulación de errores puede ser un problema crítico.

# Marco teórico

## 1. Sistemas de ecuaciones lineales

Un **sistema de ecuaciones lineales** es un conjunto de ecuaciones en las que las incógnitas aparecen solo con exponentes de primer grado y sin productos entre ellas. En términos generales, un sistema de ecuaciones lineales con  $n$  incógnitas se expresa de la siguiente manera:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots &\quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Donde  $a_{ij}$  representan los coeficientes del sistema,  $x_j$  son las incógnitas y  $b_i$  son los términos independientes. La resolución de estos sistemas es un problema fundamental en el álgebra lineal, con aplicaciones en diversas disciplinas como la economía, la ingeniería y la computación.

## 2. Métodos de resolución de sistemas de ecuaciones

Existen varios métodos para resolver sistemas de ecuaciones lineales, los cuales pueden clasificarse en **métodos exactos y aproximados**. Entre los métodos exactos se encuentran:

- **Método de sustitución:** Se despeja una incógnita en términos de las demás y se sustituye en las ecuaciones restantes.
- **Método de igualación:** Se despejan dos ecuaciones en función de la misma variable y se igualan.
- **Método de eliminación de Gauss:** Se utiliza la matriz aumentada del sistema y se aplican operaciones elementales para convertirla en una matriz escalonada.
- **Método de Gauss-Jordan:** Similar al método de Gauss, pero en este caso se lleva la matriz aumentada a la forma reducida por filas.
- **Regla de Cramer:** Se usa determinantes para encontrar las soluciones del sistema cuando la matriz de coeficientes es cuadrada y tiene determinante distinto de cero.
- **Método Montante:** Se basa en una variante del método de eliminación que evita divisiones durante el proceso de reducción de la matriz aumentada.

### 3. El método Montante

El **método Montante** es una técnica desarrollada por René Montante en la década de 1970 para resolver sistemas de ecuaciones lineales mediante un procedimiento basado en multiplicaciones, evitando el uso de divisiones.

#### 3.1. Principios fundamentales del método Montante

El **método Montante** se basa en la transformación progresiva de la matriz aumentada del sistema utilizando un esquema de pivoteo especial. Se caracteriza por lo siguiente:

1. **Uso de la matriz aumentada:** Se parte de la representación matricial del sistema de ecuaciones.
2. **Eliminación de coeficientes sin divisiones:** En lugar de dividir, se usa un esquema de productos entre los coeficientes y el pivote actual.
3. **Preservación de coeficientes enteros o racionales:** Esto evita errores numéricos y facilita el manejo de fracciones en cálculos algebraicos.
4. **Estructura sistemática:** La resolución sigue un patrón uniforme en cada iteración, lo que facilita su implementación computacional.

#### 3.2. Proceso de aplicación del método Montante

El procedimiento para aplicar el método Montante consiste en los siguientes pasos:

1. Se construye la *matriz aumentada* del sistema.
2. Se elige el primer pivote (generalmente  $a_{11}$  si es distinto de cero).
3. Se transforman los elementos de la matriz mediante la regla:

$$a'_{ij} = \frac{a_{ij} \cdot p_{k-1} - a_{ik} \cdot a_{jk}}{p_{k-1}}$$

Donde  $p_{k-1}$  es el pivote anterior y los valores se actualizan en cada iteración.

4. Se repite el proceso hasta obtener una matriz diagonal con los coeficientes de las incógnitas en la diagonal principal.
5. Se extraen las soluciones del sistema a partir de la matriz resultante.

Este procedimiento garantiza una solución exacta sin la necesidad de operaciones que introduzcan errores de redondeo.

## 4. Aplicaciones del método Montante

El método Montante es útil en diversas áreas, entre ellas:

- **Álgebra lineal:** Resolución de sistemas de ecuaciones en modelos matemáticos.
- **Computación y algoritmos:** Implementación en programas de resolución numérica de ecuaciones.
- **Criptografía:** Uso en cálculos donde se requiere preservar la exactitud de los valores.
- **Ingeniería y economía:** Modelado de sistemas con múltiples variables.

## 5. Ventajas y desventajas del método Montante

### 5.1. Ventajas

- Mayor estabilidad numérica: Reduce los errores de redondeo al evitar divisiones sucesivas.
- Fácil implementación: Sigue un esquema sistemático y uniforme.
- Mayor precisión: Mantiene coeficientes enteros o racionales durante todo el proceso.

### 5.2. Desventajas

- Menos eficiente en grandes sistemas: Para sistemas con muchas ecuaciones, otros métodos pueden ser más rápidos.
- Dependencia de pivotes adecuados: Si un pivote es cero, es necesario intercambiar filas para evitar errores.

# Código

## Método Montante

```
import copy

def metodo_montante(matriz, rest):
    #Obtenemos el tamano de la matriz
    n = len(rest)

    #Generamos una matriz identidad de tamano n
    identidad = [[1 if i == j else 0 for j in range(n)] for i in range(n)]

    #Declaramos el primer pivote
    pivot_ant = 1

    #Creamos una iteracion para resolver la matriz
    for k in range(n):

        #Guardamos el valor de la posicion (k,k) que sera con el que estemos trabajando (pivote actual)
        actual = matriz[k][k]

        #Guardamos la columna k con la cual tambien estaremos trabajando
        otras = [fila[k] for fila in matriz]

        #Creamos otra iteracion en la cual haremos que todas las pocisiones en la columna k a excepcion de (k,k) sean 0
        for j in range(n):
            if k != j:
                matriz[j][k] = 0

        #Creamos otra iteracion donde los valores por arriba de (k,k) en la diagonal sean igual a este
        for i in range(k):
            matriz[i][i] = actual
```

```

# Modificamos los valores de la matriz usando el método de Montante
for i in range(n):
    if i != k:
        for j in range(k+1, n):
            matriz[i][j] = (actual * matriz[i][j] - otras[i] * matriz[k][j]) // pivotе_ant

#Realizamos el mismo procedimiento pero con la matriz identidad
for i in range(n):
    if i != k:
        for j in range(n):
            identidad[i][j] = (actual * identidad[i][j] - otras[i] * identidad[k][j]) // pivotе_ant

#Modificamos el pivotе anterior
pivotе_ant = actual

#Una vez obtenida la determinante y la Adjunta obtenemos la inversa
Determinante = pivotе_ant
Adjunta = identidad
inversa = copy.deepcopy(Adjunta)
for i in range(n):
    for j in range(n):
        inversa[i][j] = inversa[i][j] / Determinante

Valores_resultantes = [0] * n

for i in range(n):
    for j in range(n):
        Valores_resultantes[i] = Valores_resultantes[i] + (rest[j] * Adjunta[i][j])/Determinante

return Valores_resultantes, Determinante, Adjunta, inversa

```

## Cambio de lugar de las Filas

```
def pivoteo(matriz, actual, siguiente):
```

```
#Realizamos el cambio de filas en la matriz
```

```
matriz[actual], matriz[siguiente] = matriz[siguiente], matriz[actual]
```

```
return matriz
```

## Validación si la matriz tiene algun 0 en la Diagonal

```
def validar (matriz):
```

```
n = len(matriz)
```

```
for i in range(n):
```

```
    if matriz[i][i] == 0:
```

```
        return True
```

```
return False
```

## Función principal para resolver la matriz

```
def solucionar_matriz(matriz, rest):
```

```
n = len(rest)
```

```
#Validamos que la matriz no algun 0 en la diagonal
```

```
cero = validar(matriz)
```

```
#Si llegara a tener un 0 en la diagonal realizamos movimientos en las filas
```

```
actual = 0 #Iniciamos el contador de la casilla que vamos a cambiar
```

```
siguiente = actual + 1 #Iniciamos el contador de la casilla con la cual la vamos a cambiar
```

```
iteraciones = 0 #iniciamos el contador de iteraciones para realizar los cambios
```

```
while cero and iteraciones < n:
```

```
#Llamamos a la funcion pivoteo que realizara los cambios en la matriz
```

```
matriz = pivoteo(matriz, actual, siguiente)
```

```

#Volvemos a validar matriz
cero = validar(matriz)

#Si hay 0 avanzamos en la matriz para realizar cambios
if cero:

    #Si siguiente no ha llegado al ultima fila de la matriz avanzamos
    if siguiente + 1 < n:
        actual = actual +1
        siguiente = actual +1

    #Si no volvemos a cambiar desde la primer fila e incrementamos las iteraciones
    else:
        actual = 0
        siguiente = actual +1
        iteraciones += 1

    #Salimos de la funcion

#Validamos que si podemos realizar el metodo motante
if cero:
    print("ERROR: la matriz ingresada no es apta para el metodo montante, ya que no hay una combinacion de filas con una diagonal que no tenga 0, intente nuevamente con otra matriz")
    return False

else:
    #Se calcula el resultado por el metodo montante
    resultado = metodo_montante(matriz, rest)
    return resultado

```

## Main (Captura de la matriz)

```
def main():
    op = 1
    while op == 1:
        evaluador1 = False
        #Se ingresa el tamaño de la matriz
        print("Escriba el tamaño de la matriz:")
        #Se comprueba que el valor sea un numero entero positivo
        while evaluador1 == False:
            try:
                n = int(input())
                if n > 0:
                    evaluador1 = True
                else:
                    print("ERROR: Debe de ser un numero positivo: Intente nuevamente:")
            except ValueError:
                print("ERROR: Debe ingresar un numero entero positivo. Intente nuevamente:")
        #Se crea la Matriz junto a los valores de sus funciones
        matriz = [[0 for _ in range(n)] for _ in range(n)]
        valores = [0 for _ in range(n)]

        #Se evalua que la matriz sea invertible
        while True:
            for i in range(n):
                for j in range(n+1):
                    evaluador2 = False
                    if(j<n):
                        print(f"Ingrese el valor de X [{i+1}][{j+1}]:")
                        #se comprueba que los valores dentro de la matriz sean numeros
                        while evaluador2 == False:
                            try:
```

```

    matriz [i][j] = float(input())

    evaluador2 = True

except ValueError:

    print("ERROR: Debe ingresar un numero. Intente nuevamente:")

else:

    print(f"Ingrese el valor de la funcion {i+1}:")

    #se evalua que los valores de los resultados sean numeros

    while evaluador2 == False:

        try:

            valores [i] = float(input())

            evaluador2 = True

        except ValueError:

            print("ERROR: Debe ingresar un numero. Intente nuevamente:")

try:

    resultado, determinante, adjunta, inversa = solucionar_matriz(matriz, valores)

    break

except ZeroDivisionError:

    print("ERROR: la matriz ingresada no es invertible por ende no es apta para el metodo montante, intente nuevamente con otra matriz")

except TypeError:

    print("")

if resultado:

    print("El resultado es el siguiente")

    for i in range(n):

        print(f"\nX{i+1} = {resultado[i]}")

    print(f"\nEl determinante de la matriz es:{determinante}\n")

    print("\nLa adjunta de la matriz es:\n")

    for j in range(n):

        for k in range(n):

```

```
print(f"[{adjunta[j][k]}]", end=" ")
print("")
print("\nla inversa de la matriz es:\n")
for p in range(n):
    for q in range(n):
        print(f"[{inversa[p][q]}]", end=" ")
    print("")

print('¿Desea calcular otra matriz?')
opcion = input("Si / No\n")
if opcion == 'si' or opcion == 'Si' or opcion == 'SI' or opcion == 'sI':
    op = 1
else:
    op = 0

if __name__ == "__main__":
    main()
```

# Manual de usuario

## Requisitos

Python 3 instalado en el sistema.

Conocimientos básicos sobre matrices y sistemas de ecuaciones lineales.

## Ejecución del Programa

Para ejecutar el programa, simplemente ejecute el archivo Python en una terminal o entorno de desarrollo:

`python nombre_del_archivo.py`

## Uso

1. **Ingresar el tamaño de la matriz:** El usuario debe ingresar un número entero positivo que represente la dimensión de la matriz cuadrada.

*Ejemplo:*

```
Escriba el tamaño de la matriz:  
3
```

2. **Ingresar los coeficientes de la matriz:** El programa solicitará los valores de cada elemento de la matriz, verificando que sean números válidos.

*Ejemplo:*

```
Ingrese el valor de X [1][1]:  
2  
Ingrese el valor de X [1][2]:  
5  
Ingrese el valor de X [1][3]:  
4
```

3. **Ingresar los valores de la función (vector de resultados):** Cada ecuación del sistema tiene un resultado, que también debe ser ingresado como un número.

*Ejemplo:*

```
Ingrese el valor de la funcion i:  
12
```

4. **Cálculo del resultado:** El programa aplicará el Método de Montante para calcular los valores de las variables y mostrará los resultados en pantalla.

*Ejemplo:*

```
El resultado es el siguiente
X1 = 1.0
X2 = 3.0

El determinante de la matriz es:4.0

La adjunta de la matriz es:

[3.0] [-1.0]
[-5.0] [3.0]

la inversa de la matriz es:

[0.75] [-0.25]
[-1.25] [0.75]
```

5. **Desea ingresar otra matriz:** Al terminar de calcular el resultado de evaluar una matriz y cuando la matriz no es invertible, se le pregunta a el usuario si quiere volver a realizar el proceso.

*Ejemplo:*

```
¿Desea calcular otra matriz?
Si / No
si
```

## Mensajes de Error

- Si el usuario ingresa un valor no numérico, se mostrará un mensaje de error y se solicitará nuevamente el dato.

*Casos de aparición :*

```
Escriba el tamaño de la matriz:  
dos  
ERROR: Debe ingresar un numero entero positivo. Intente nuevamente:  
  
Escriba el tamaño de la matriz:  
-2  
ERROR: Debe de ser un numero positivo. Intente nuevamente:  
  
Ingrese el valor de X [1][1]:  
dos punto uno  
ERROR: Debe ingresar un numero. Intente nuevamente:  
  
Ingrese el valor de la funcion 1:  
cuatro  
ERROR: Debe ingresar un numero. Intente nuevamente:
```

- Si la matriz ingresada no es invertible, se mostrará un mensaje indicando que no es apta para el método y se pedirá una nueva matriz.

*Casos de aparición:*

```
Ingrese el valor de X [1][1]:  
1  
Ingrese el valor de X [1][2]:  
2  
Ingrese el valor de X [1][3]:  
3  
Ingrese el valor de la funcion 1:  
6  
Ingrese el valor de X [2][1]:  
4  
Ingrese el valor de X [2][2]:  
5  
Ingrese el valor de X [2][3]:  
6  
Ingrese el valor de la funcion 2:  
34  
Ingrese el valor de X [3][1]:  
7  
Ingrese el valor de X [3][2]:  
8  
Ingrese el valor de X [3][3]:  
9  
Ingrese el valor de la funcion 3:  
67  
ERROR: la matriz ingresada no es invertible por ende no es apta para el metodo montante, intente nuevamente con otra matriz
```

# Ejemplo de ejecución

## Ejecución exitosa

```
Escriba el tamaño de la matriz:  
3  
Ingrese el valor de X [1][1]:  
3  
Ingrese el valor de X [1][2]:  
-2  
Ingrese el valor de X [1][3]:  
1  
Ingrese el valor de la función 1:  
2  
Ingrese el valor de X [2][1]:  
4  
Ingrese el valor de X [2][2]:  
3  
Ingrese el valor de X [2][3]:  
-5  
Ingrese el valor de la función 2:  
4  
Ingrese el valor de X [3][1]:  
2  
Ingrese el valor de X [3][2]:  
1  
Ingrese el valor de X [3][3]:  
-1  
Ingrese el valor de la función 3:  
3  
El resultado es el siguiente  
X1 = 1.3125  
X2 = 1.5625  
X3 = 1.1875
```

---

```
El determinante de la matriz es:16.0
```

```
La adjunta de la matriz es:
```

```
[2.0] [-1.0] [7.0]  
[-6.0] [-5.0] [19.0]  
[-2.0] [-7.0] [17.0]
```

```
la inversa de la matriz es:
```

```
[0.125] [-0.0625] [0.4375]  
[-0.375] [-0.3125] [1.1875]  
[-0.125] [-0.4375] [1.0625]
```

## Ejecución matriz no invertible

```
umerico/Proyecto 1 Montante.py"
Escriba el tamaño de la matriz:
3
Ingrese el valor de X [1][1]:
1
Ingrese el valor de X [1][2]:
2
Ingrese el valor de X [1][3]:
3
Ingrese el valor de la funcion 1:
4
Ingrese el valor de X [2][1]:
5
Ingrese el valor de X [2][2]:
6
Ingrese el valor de X [2][3]:
7
Ingrese el valor de la funcion 2:
8
Ingrese el valor de X [3][1]:
9
Ingrese el valor de X [3][2]:
10
Ingrese el valor de X [3][3]:
11
Ingrese el valor de la funcion 3:
12
ERROR: la matriz ingresada no es invertible por ende no es apta para el metodo montante, intente nuevamente con otra matriz
```

## Ejecución con matriz con diagonal con 0

```
Escriba el tamaño de la matriz:  
3  
Ingrese el valor de X [1][1]:  
0  
Ingrese el valor de X [1][2]:  
-2  
Ingrese el valor de X [1][3]:  
1  
Ingrese el valor de la función 1:  
1  
Ingrese el valor de X [2][1]:  
4  
Ingrese el valor de X [2][2]:  
0  
Ingrese el valor de X [2][3]:  
-5  
Ingrese el valor de la función 2:  
4  
Ingrese el valor de X [3][1]:  
2  
Ingrese el valor de X [3][2]:  
1  
Ingrese el valor de X [3][3]:  
0  
Ingrese el valor de la función 3:  
3  
El resultado es el siguiente  
X1 = 2.333333333333335  
X2 = -0.6666666666666667  
X3 = 1.6666666666666665
```

El determinante de la matriz es: 24.0

La adjunta de la matriz es:

```
[1.0] [10.0] [5.0]  
[-2.0] [4.0] [-10.0]  
[-4.0] [8.0] [4.0]
```

la inversa de la matriz es:

```
[0.04166666666666664] [0.4166666666666667] [0.208333333333334]  
[-0.0833333333333333] [0.1666666666666666] [-0.4166666666666667]  
[-0.1666666666666666] [0.333333333333333] [0.1666666666666666]
```

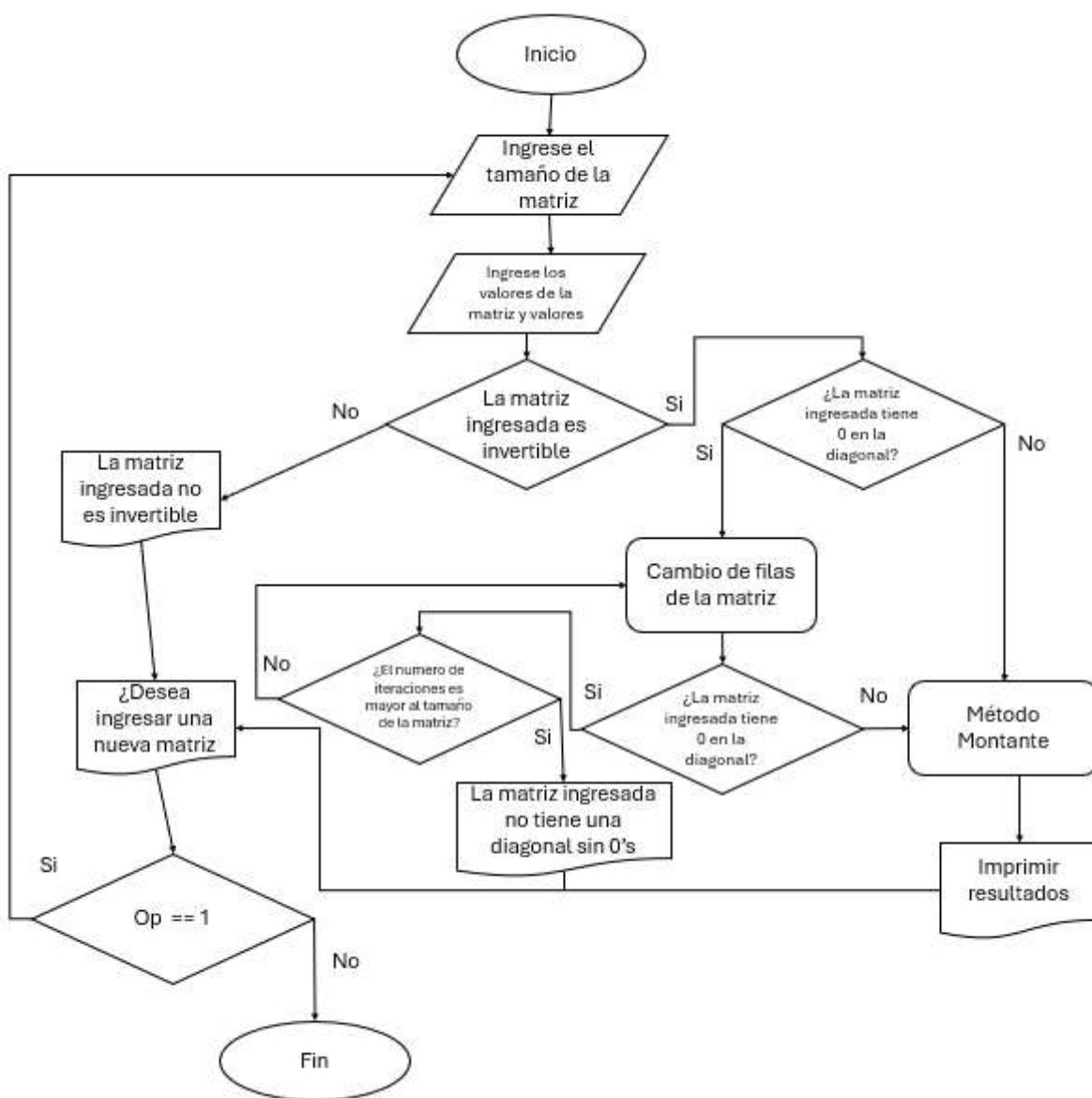
## Ejecución con matriz con solo 0

```
Escriba el tamaño de la matriz:  
3  
Ingrese el valor de X [1][1]:  
0  
Ingrese el valor de X [1][2]:  
0  
Ingrese el valor de X [1][3]:  
0  
Ingrese el valor de la función 1:  
0  
Ingrese el valor de X [2][1]:  
0  
Ingrese el valor de X [2][2]:  
0  
Ingrese el valor de X [2][3]:  
0  
Ingrese el valor de la función 2:  
0  
Ingrese el valor de X [3][1]:  
0  
Ingrese el valor de X [3][2]:  
0  
Ingrese el valor de X [3][3]:  
0  
Ingrese el valor de la función 3:  
0  
ERROR: la matriz ingresada no es apta para el método montante, ya que no hay una combinación de filas con una diagonal que no tenga 0, intente nuevamente con otra matriz
```

## Calcular otra matriz

¿Desea calcular otra matriz?  
Si / No

# Diagrama de Flujo



# Conclusión

## Carlos Enrique Castillo Mayorga

En este proyecto aprendí sobre métodos numéricos y como usar programación para resolver problemas matemáticos. Fue interesante ver como los conceptos teóricos se pueden aplicar en la práctica. Al principio me pareció un poco complicado, pero con el tiempo logré entender mejor como funciona el método Montante para resolver sistemas de ecuaciones. Al final, pude hacer un programa que funciona, lo cual me da mucha satisfacción.

## Maximiliano Rada Moreno

Durante el desarrollo del proyecto pudimos observar el funcionamiento del método montante, cuáles son sus beneficios y cuáles son sus desventajas frente a otro tipo de métodos, así mismo desarrollamos un proyecto en el lenguaje Python con el cual implementamos este método para la resolución de matrices  $n \times n$ , validando que estas pudieran ser resueltas, pudiendo ser inversas y hubiera una combinación de sus filas en la cual no aparezcan 0's en la diagonal, cabe resaltar que por este método además de la solución de cada una de las variables poder obtener las determinante, la matriz inversa, y la matriz adjunta. La parte compleja de realizar este proyecto no fue el lenguaje o la resolución de la matriz a través del método escogido, la verdadera dificultad que llegamos a tener al realizar el código fue la validación de la matriz, que esta fuera ideal para resolver a través de este método, ya que tuvimos que validar varias cosas para que esto pudiera ser resuelto de esta manera, fuera de eso el manejo del código en la resolución del método montante fue fácil. Otro de los problemas que llegamos a tener fue que al tener que obtener las soluciones, al manejar decimales y no fracciones obtenemos resultados aproximados a los esperados, sin embargo, no podemos optar por métodos de redondeo ya que afectaría a las soluciones con soluciones decimales.

# Bibliografía

**Montante Pardo, R.** (1976). *Un método para la solución de sistemas de ecuaciones lineales*. Universidad Autónoma de Nuevo León.

**Chapra, S. C., & Canale, R. P.** (2020). *Métodos numéricos para ingenieros* (8<sup>a</sup> ed.). McGraw-Hill.