

Resolución del Método de Gauss-Jordan usando Shiny en Python

Ronaldo Carlos Mamani Mena

Resumen

Este documento muestra el desarrollo de una aplicación interactiva para resolver sistemas de ecuaciones lineales utilizando el método de Gauss-Jordan. Se parte desde un código funcional en consola y se extiende a una interfaz gráfica con la herramienta **Shiny** para Python, resaltando sus ventajas frente a otros entornos como R. El archivo fuente se encuentra en: E:\METODOS_OPTIM\shiny.py.

1. Código Primitivo en Consola

A continuación, se presenta el código original implementado en Python, ejecutado mediante consola, que solicita al usuario ingresar el sistema de ecuaciones y devuelve la solución utilizando el método de Gauss-Jordan.

```
def gj(m):
    n = len(m)
    for j in range(n):
        if m[j][j] == 0:
            try:
                m[j], m[next(i for i in range(j+1,n) if m[i][j] != 0)] = \
                    m[next(i for i in range(j+1,n) if m[i][j] != 0)
                      ], m[j]
            except:
                raise ValueError("No solución única ")
        m[j] = [x / m[j][j] for x in m[j]]
        for i in range(n):
```

```

        if i != j:
            m[i] = [a - m[i][j] * b for a, b in zip(m[i], m[j])]
    return [round(r[-1], 6) for r in m]

try:
    n = int(input("N inc gnitas: "))
    m = [list(map(float, input(f"E{i+1}: ").split())) for i in range
          (n)]
    if any(len(r) != n + 1 for r in m):
        raise ValueError("Tama o incorrecto")
    r = input(" Restringir valores? (s/n): ").lower() == "s"
    rest = [tuple(map(float, input(f"x{i+1} min,max: ").split(", ")))
             if r else None for i in range(n)]
    s = gj(m)
    print("Soluci n:", *[f"x{i+1}={v}" for i, v in enumerate(s)])
    if r:
        for i, (v, t) in enumerate(zip(s, rest)):
            if t and not(t[0] <= v <= t[1]):
                print(f"error x{i+1}={v} fuera de [{t[0]},{t[1]}]")
except Exception as e:
    print(f"Error: {e}")

```

2. Ejecución del Código Original

```
PS C:\METODOS_OPTIM > Gauss.py ...
1 def gj(m):
7   m[j]=x/m[j][j]for x in m[j]
8   for i in range(n):
9     if i!=j:m[i]=[a-m[i][j]*b for a,b in zip(m[i],m[j])]
10  return [round(r[-1],6)for r in m]
11
12 try:
13   n=int(input("N incógnitas: "))
14   m=[list(map(float,input(f"E{i+1}: ").split()))for i in range(n)]
15   if any(len(r)!=n+1 for r in m):raise ValueError("Tamaño incorrecto")
16   r=input("¿Restringir valores? (s/n): ").lower()=="s"
17   rest=tuple(map(float,input(f"x{i+1} min,max: ").split(","))if r else None for i in range(n))
18   s=gj(m)
19   print("Solución:",*[f"x{i+1}={v}"for i,v in enumerate(s)])
20   if r:
21     for i,(v,t)in enumerate(zip(s,rest)):
22       if t and not(t[0]<=v<=t[1]):print(f"error x{i+1}={v} fuera de [{t[0]},{t[1]}]")
23 except Exception as e:print(f"Error: {e}")
```

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python312/python.exe e:/METODOS_OPTIM/SHINYY.py
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python312/python.exe e:/METODOS_OPTIM/Gauss.py
N incógnitas: 3
E1: 2 3 -1 5
E2: 4 1 2 6
E3: -2 5 -3 -4
¿Restringir valores? (s/n): n
Solución: x1=2.875 x2=-1.0 x3=-2.25
PS C:\Users\LENOVO>
```

Funcionamiento del

código original en consola

3. Transición a Shiny con Interfaz Gráfica

Para facilitar el uso de este método sin necesidad de conocimientos en línea de comandos, se utilizó **Shiny para Python**. Esta herramienta permite construir interfaces web interactivas de manera sencilla.

3.1. Pasos de Implementación

1. Instalar la biblioteca Shiny:

```
pip install shiny
```

2. Definir la lógica del método (`gj()`).
3. Crear la interfaz con `shiny.ui` (campos para ingresar ecuaciones y restricciones).
4. Crear el servidor con `shiny.render` y lógica reactiva.
5. Personalizar el estilo visual con CSS.
6. Ejecutar la app con:

```
shiny run --reload shiny.py
```

4. Ventajas de Usar Shiny en Python

Aunque originalmente Shiny fue desarrollado para R, su versión para Python ofrece muchas ventajas en contextos educativos y científicos como este:

- **Compatibilidad con librerías científicas:** Python permite usar NumPy, SymPy, scikit-learn, entre otros, en conjunto con Shiny.
- **Mayor adopción en ingeniería:** Python es más común en carreras técnicas, facilitando su integración en proyectos más amplios.
- **Curva de aprendizaje más baja:** Para quienes ya usan Python, no es necesario aprender otro lenguaje como R.
- **Extensibilidad:** Es fácil combinar Shiny con bases de datos, APIs, y otros servicios Python modernos.

Según la documentación oficial (*Posit, 2023*), la versión de Shiny para Python busca mantener la misma lógica reactiva que en R, pero con mayor compatibilidad hacia entornos modernos de desarrollo en ciencia de datos y machine learning.

5. Vista de la Aplicación Final

The screenshot shows a web-based application interface for solving a system of linear equations. On the left, there is a sidebar with a 'Número de incógnitas' (Number of unknowns) input field set to 3, a checkbox for '¿Restringir valores?' (Restrict values?), and a blue 'Resolver' (Solve) button. The main area is titled 'Método de Gauss-Jordan' and contains three input fields for equations, each with a label 'Ecuación 1 (separa con espacios)', 'Ecuación 2 (separa con espacios)', and 'Ecuación 3 (separa con espacios)'. The inputs contain the values '2 3 -1 5', '4 1 2 6', and '-2 5 -3 -4' respectively. Below these inputs is a section titled 'Resultado' (Result) which displays the solution: 'x1 = 2.875', 'x2 = -1.0', and 'x3 = -2.25'. It also shows 'Pasos intermedios' (Intermediate steps) for 'Paso 1' and 'Paso 2', each followed by a 4x4 matrix of values.

Interfaz

gráfica final construida con Shiny