PONTIFICIA UNIVERSIDAD JAVERIANA

PARCIAL 1. MÉTODO SOR

INTEGRANTES:

SARA JULIANA PEÑA GÓMEZ

DANIELA FLOREZ VILLAMIL

PROFESORA:

ANGIE TATIANA SUAREZ ROMERO
ANÁLISIS NÚMERICO

BOGOTÁ, MARZO 2025

1. INTRODUCCIÓN

   El siguiente informe consiste en la explicación detallada del funcionamiento del método SOR. Se hará una explicación teórica que proporciona las bases para la solución practica mediante una sencilla implementación en Python del método.

2. FUNCIONAMIENTO DEL MÉTODO SOR

   El método SOR es una modificación usada para mejorar la convergencia del método Gauss-Seidel considerando un promedio ponderado ω de las iteraciones anteriores y actuales. Se basa en la idea de tomar la dirección del método de Gauss-Seidel y aplicar un factor de relajación ω para ajustar la actualización de las soluciones en cada iteración.

   A partir del sistema de ecuaciones Ax= b, el método SOR actualiza cada componente x_i de la solución iterativamente utilizando la fórmula:

$$x_i^{(k)} = \omega \left\{ \left[-\sum_{\substack{j=1 \\ j<i}}^{n} (a_{ij}/a_{ii}) x_j^{(k)} - \sum_{\substack{j=1 \\ j>i}}^{n} (a_{ij}/a_{ii}) x_j^{(k-1)} + (b_i/a_{ii}) \right] \right\} + (1-\omega) x_i^{(k-1)}$$

   Tomado de: Métodos Numéricos y Computación de Ward Chaney y David Kincaid.

   Donde,

   - $x^{(k+1)}$ representa la nueva aproximación lineal.
   - $x^{(k)}$ es la aproximación anterior.
   - ω es el factor de relajación, que ajusta cuánto se "sobrepasa" el método Gauss-Seidel en la actualización.

   Se tienen las siguientes condiciones del método SOR:

   - Si $0 < \omega < 1$ se conoce como subrelajación. Se emplea para un sistema que no converge.
   - Si $1 < \omega < 2$ se llama sobre-relajación. Acelera la convergencia del sistema que ya converge.
   - Si $\omega = 1$ el resultado no se modifica.
   - Si $\omega > 2$ el método diverge.

   En esta implementación, se optó por utilizar la **norma infinita** como criterio de convergencia. Esta norma calcula el máximo valor absoluto de las diferencias entre la solución actual y la anterior, permitiendo identificar el cambio más significativo de una iteración a otra. Cuando dicho valor es menor que la tolerancia establecida, el método se considera convergente y se detiene.

   El uso de la norma infinito ofrece una mayor eficiencia computacional en comparación con otras normas, como la euclidiana, ya que solo evalúa el cambio máximo absoluto entre iteraciones en lugar de calcular la magnitud total del error.

En cuanto a la diferencia entre los métodos, el **método de Gauss-Seidel** puede considerarse un caso particular del **método SOR**. Cuando el parámetro de relajación es igual a ω =1, el método SOR se comporta exactamente como Gauss-Seidel, ya que no aplica ninguna modificación en la actualización de las soluciones. Sin embargo, con una elección adecuada de ω, el método SOR puede reducir significativamente el número de iteraciones necesarias para alcanzar una solución precisa, lo que representa su principal ventaja.

A continuación, se presentará la implementación del método SOR y, posteriormente, su comparación con el método Gauss-Seidel. Esta comparación permitirá determinar cuál de los dos métodos ofrece un mejor desempeño en la resolución del sistema de ecuaciones dado.

```
def sor(A, b, w, x0, tol=1e-6, max_iter=100):  1 usage
    n = len(b) #Se comprueba la longitud de b para que coincida con la matriz A

    x = np.array(x0, dtype=float) #Se crea un arreglo para guardar las iteraciones.
```

El código define la función “sor (A, b, w, x0, tol=1e-6, max_iter=100)”, que implementa el método SOR para resolver sistemas de ecuaciones lineales. Dentro de la función, se obtiene la longitud de b para asegurarse de que coincida con la matriz A. Luego, se inicializa el vector x a partir de x0, convirtiéndolo en un arreglo de tipo “float” para almacenar las iteraciones.

```
for k in range(max_iter):
    x_old = x.copy() #Se hace comparaciones entre iteraciones

    for i in range(n):
        sum1 = sum(A[i][j] * x[j] for j in range(i)) #Calcula la suma de los elementos actualizados en cada iteración
        sum2 = sum(A[i][j] * x_old[j] for j in range(i + 1, n)) #Calcula la suma de los elementos que no estan actualizados en cada iteración
        x[i] = (1 - w) * x_old[i] + (w / A[i][i]) * (b[i] - sum1 - sum2) #Actualiza el valor del arreglo. Se aplica la formula SOR. Se agrega w.
```

Se inicia el bucle principal de iteraciones, ejecutándose hasta max_iter veces. En cada iteración, se almacena una copia del vector x anterior (x_old) para evaluar la convergencia. Luego, se recorre cada fila de la matriz A, dividiendo los cálculos en dos partes: la suma de los elementos **ya actualizados** y la de los **no actualizados**. Con estos valores, se aplica la fórmula del método SOR para actualizar x[i] en cada iteración.

```
    if np.linalg.norm(x - x_old, ord=np.inf) < tol: #S
        return x, k + 1 # Si la condición de converger
return x, max_iter # Si no se alcanza la convergencia
```

Para determinar si el método ha convergido, se evalúa la norma infinita, de la diferencia entre x y x_old, comparándola con la tolerancia “tol”. Si la diferencia es menor que tol, el algoritmo retorna la solución x y el número de iteraciones realizadas. Si no se alcanza la convergencia tras max_iter iteraciones, la función retorna x junto con el número máximo de iteraciones permitidas.

```python
w = 1.1  # Factor de relajación
x0 = np.zeros(len(b))  # Vector inicial

sol, iterations = sor(A, b, w, x0)
print("Solución:", sol)
print("Iteraciones:", iterations)
```

Se define un factor de relajación w = 1.1 y un vector inicial x0 lleno de ceros. Luego, se llama a la función sor con estos parámetros y se imprimen los resultados, mostrando la solución obtenida y el número de iteraciones necesarias para la convergencia. Este método es útil para resolver sistemas de ecuaciones grandes y dispersos, ya que mejora la convergencia en comparación con el método de Gauss-Seidel.

Comparación con el método de Gauss-Seidel:

```python
def gauss_seidel(A, b, x0, tol=1e-6, max_iter=100):  1 usage  Sara Peña
    n = len(b)
    x = np.array(x0, dtype=float)
    iter_list = []
    for k in range(max_iter):
        x_old = x.copy()
        for i in range(n):
            sum1 = sum(A[i][j] * x[j] for j in range(i))
            sum2 = sum(A[i][j] * x_old[j] for j in range(i + 1, n))
            x[i] = (b[i] - sum1 - sum2) / A[i][i]

        iter_list.append(np.linalg.norm(x - x_old, ord=np.inf))
        if iter_list[-1] < tol:
            return x, k + 1, iter_list
    return x, max_iter, iter_list
```

Durante la investigación del método SOR, se encontró que este es una extensión del método de Gauss-Seidel. Para comprender mejor su funcionamiento y verificar su eficiencia, se realizó una comparación resolviendo un mismo sistema de ecuaciones con ambos métodos. El objetivo era analizar de manera práctica cómo SOR puede reducir el número de iteraciones necesarias para alcanzar una solución.

El método de Gauss-Seidel, como se observa en la imagen, es un algoritmo iterativo utilizado para resolver sistemas de ecuaciones lineales. Su funcionamiento se basa en actualizar los valores de la

solución en cada iteración, empleando los valores recién calculados dentro de la misma iteración. Esto le permite mejorar la convergencia en comparación con el método de Jacobi, aunque en ciertos sistemas de ecuaciones su velocidad de convergencia puede ser limitada.

3. EJEMPLO

Dado el siguiente sistema de ecuaciones:

$$
\begin{aligned}
4x_1 - x_2 &= 0,\\
-x_1 + 4x_2 - x_3 &= 5,\\
-x_2 + 4x_3 &= 0,\\
4x_4 - x_5 &= 6,\\
-x_4 + 4x_5 - x_6 &= -2,\\
-x_5 + 4x_6 &= 6
\end{aligned}
$$

Se plantea la siguiente matriz A y el vector b para solucionar.

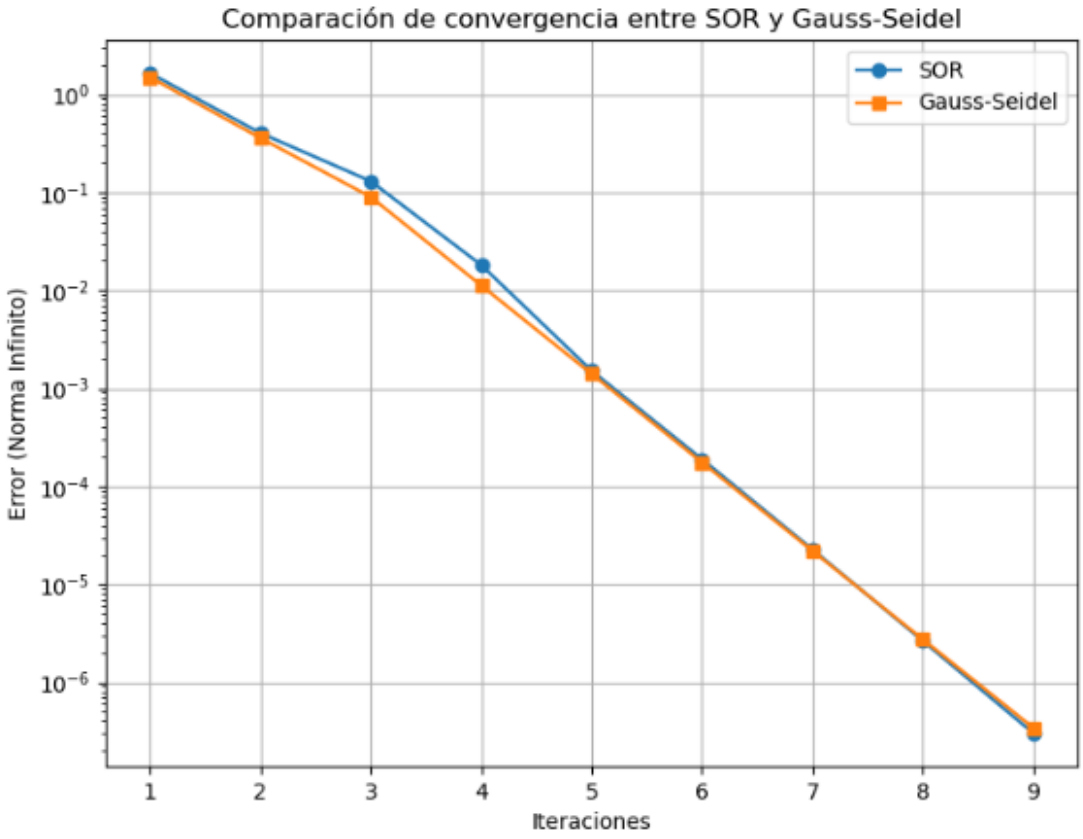
```
A = np.array( object: [[4, -1, 0, 0, 0, 0],
                       [1, 4, -1, 0, 0, 0],
                       [0, -1, 4, 0, 0, 0],
                       [0, 0, 0, 4, -1, 0],
                       [0, 0, 0, -1, 4, -1],
                       [0, 0, 0, 0, -1, 4]], dtype=float)

b = np.array( object: [0, 5, 0, 6, -2, 6], dtype=float)
```

RESULTADOS:

```
Solución SOR: [0.31249997 1.25       0.3125     1.57142858 0.28571429 1.57142857]
Iteraciones SOR: 9
Solución Gauss-Seidel: [0.3125     1.25       0.3125     1.57142852 0.28571426 1.57142857]
Iteraciones Gauss-Seidel: 9
```

Esta imagen presenta las soluciones obtenidas por ambos métodos. La solución final de ambos métodos es prácticamente la misma, con ligeras diferencias en la precisión de algunos valores debido a la naturaleza iterativa de los métodos.

Ambos métodos convergen en 9 iteraciones, lo que sugiere que, en este caso particular, el factor de sobre-relajación del método SOR no logró una reducción significativa en el número de iteraciones en comparación con Gauss-Seidel.



Gráfica de convergencia

- En el eje **X** se representan las **iteraciones**.
- En el eje **Y** se muestra el **error (Norma Infinito)** en escala logarítmica.

Se comparan dos métodos:

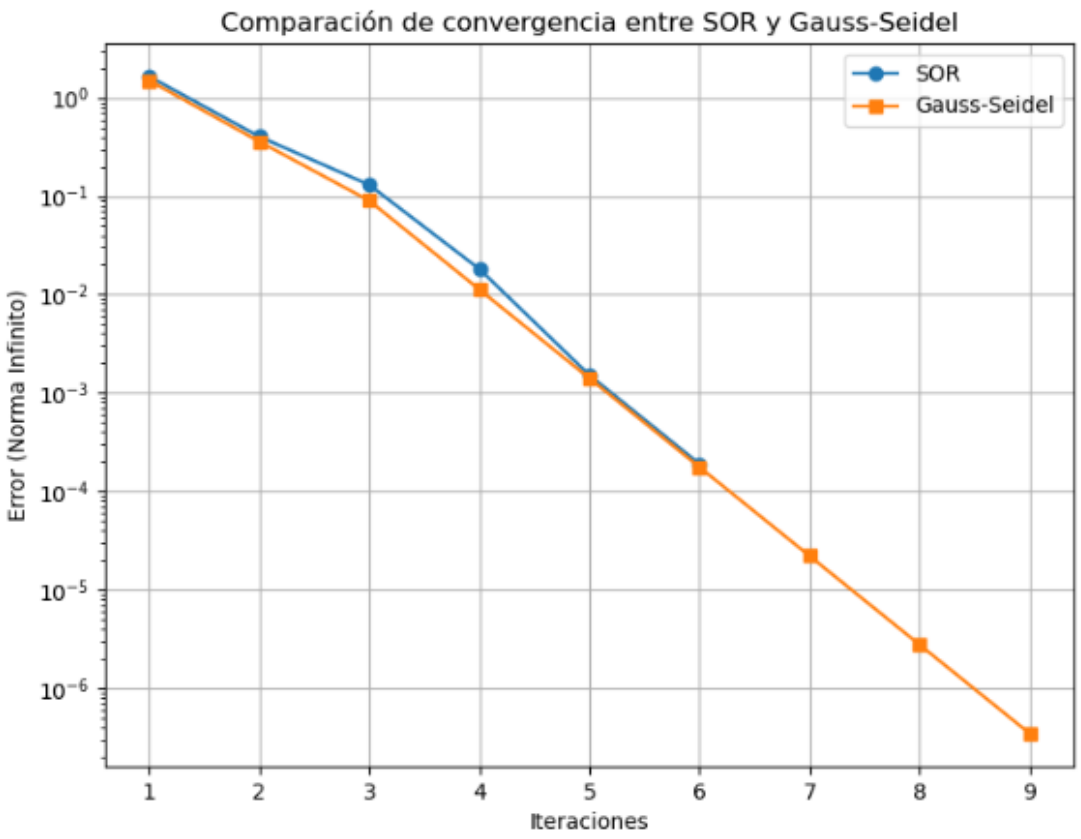
- **SOR (línea azul con círculos)**
- **Gauss-Seidel (línea naranja con cuadrados)**
- Se observa que ambos métodos convergen de manera similar, aunque SOR puede mostrar una reducción más rápida del error en ciertas iteraciones.

Por último, se decidió modificar el índice de tolerancia. En el caso anterior este se encontrada en $1e-6$ y para este se modificó a $1e-3$.

```
Solución SOR: [0.31252031 1.24999875 0.31249762 1.57142569 0.28571831 1.57142831]
Iteraciones SOR: 6
Solución Gauss-Seidel: [0.3125     1.25       0.3125     1.57142852 0.28571426 1.57142857]
Iteraciones Gauss-Seidel: 9
```

En esta imagen, se observa una diferencia en el número de iteraciones requeridas por cada método para encontrar la solución. El método SOR logró converger en **6 iteraciones**, mientras que el método Gauss-

Seidel mantuvo las **9 iteraciones**. Ambas soluciones son muy similares, aunque SOR muestra una mayor precisión en los decimales. Esta diferencia en el rendimiento se visualiza con mayor claridad en la siguiente gráfica.



La siguiente gráfica muestra que el método SOR tiene una mejor convergencia al lograr encontrar la solución en 3 iteraciones menos que el Gauss-Seidel. Adicionalmente, se ve que este mantiene una reducción más rápida del error en ciertas iteraciones como en el caso anterior.

4. CONCLUSIONES

A partir de los resultados obtenidos, se dan dos conclusiones.

En primer lugar, el método SOR demostró ser más eficiente en la primera ejecución, alcanzando la solución en solo 6 iteraciones, mientras que en la segunda ejecución requirió 9 iteraciones, igualando el número de iteraciones del método Gauss-Seidel. Esto sugiere que el desempeño de SOR depende de la selección del factor de relajación ωy de la tolerancia establecida, lo que influye en la rapidez con la que el método converge a la solución. Cuando ω se elige correctamente, SOR puede reducir significativamente el número de iteraciones necesarias en comparación con Gauss-Seidel.

Por otro lado, aunque ambos métodos llegan a soluciones muy similares, se observa que SOR maneja los decimales con una mayor precisión. Sin embargo, más allá de la precisión en los valores

obtenidos, la ventaja principal de SOR radica en su capacidad para optimizar el tiempo de cómputo al disminuir la cantidad de iteraciones requeridas. Esto lo convierte en una opción más eficiente cuando se trabaja con sistemas de ecuaciones de mayor tamaño o con restricciones de recursos computacionales.

5. REFERENCIAS

- *Métodos Númericos*. (s/f). Studocu.com. Recuperado de https://www.studocu.com/es-mx/document/universidad-nacional-autonoma-de-mexico/matematicas-propedeutico/metodo-sor/13504182
- Mendoza, A. (2017, abril 26). *Cálculo Número*. http://chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://venus.ceride.gov.ar/cursos/moodledata/31/moddata/assignment/195/7392/Mendoza_Agustin_TP_03.pdf
- Perez, M. (2014). *Cálculo Número*. Google.com. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://venus.ceride.gov.ar/cursos/moodledata/25/moddata/assignment/150/1112/TP_3_Corregido_-Perez_Miguel_A.pdf
- Burden, R., & Faires, J. D. (2001). *Analisis Numerico* (7a ed.). Cengage Learning Editores S.A. de C.V.
- University of Texas Austin Ward Cheney, & Kincaid, D. (2012). *Metodos Numericos y Computacion*. Cengage Learning Editores.
- Sauer, T. (2017). *Numerical Analysis* (3a ed.). Pearson.