Método de Gauss-Seidel

El método de Gauss-Seidel es un método iterativo para resolver sistemas de ecuaciones lineales, similar al método de Jacobi.

Este método puede aplicarse a cualquier sistema de ecuaciones lineales que produzca una matriz de coeficientes con los elementos de su diagonal no-nulos. Sin embargo, la convergencia del método solo se garantiza si la matriz es diagonalmente dominante o si es simétrica y definida positiva. Lógicamente, la matriz debe ser cuadrada, i.e., el sistema debe tener tantas ecuaciones como incógnitas, para que exista solución única.

El método parte de una aproximación inicial y se repite el proceso hasta llegar a una solución con un margen de error tan pequeño como se quiera. Dado un sistema de ecuaciones lineales en notación matricial $Ax = b,$ se busca la solución del mismo donde:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

El sistema de ecuaciones se puede reescribir como:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ (L_* + U)\mathbf{x} &= \mathbf{b} \\ L_*\mathbf{x} + U\mathbf{x} &= \mathbf{b} \\ L_*\mathbf{x} &= \mathbf{b} - U\mathbf{x} \end{aligned}$$

A continuación se despeja el término de la parte izquierda de la expresión, utilizando el valor previo de x que aparece en la parte derecha de la ecuación, lo cual se puede expresar analiticamente como:

$$\mathbf{x}^{(k+1)} = L_*^{-1} \left(\mathbf{b} - U\mathbf{x}^{(k)} \right).$$

Sin embargo, aprovechando la forma triangular de L_*, los elementos de $x^{(k+1)}$ se pueden calcular secuencialmente para cada fila i usando sustitución directa:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

Tenga en cuenta que las dos sumatorias que se utilizan en la fórmula se pueden expresar mediante una sola:

$$\sum_{j \neq i} a_{ij} x_j$$

que utiliza el valor x_j calculado más recientemente. El procedimiento continúa hasta que los cambios realizados en una iteración son mínimos, i.e., están por debajo de cierta tolerancia, como un residuo suficientemente pequeño.

La diferencia entre este método y el de Jacobi es que, en este último, las mejoras a las aproximaciones no se utilizan hasta completar las iteraciones

El cálculo de $x^{(k+1)}$ utiliza los elementos de $x^{(k+1)}$ que ya han sido calculados, y solo los elementos de $x^{(k)}$ que no se han calculado en la $(k+1)$-ésima iteración. Esto significa que, a diferencia del método de Jacobi, solo se requiere un vector de almacenamiento, ya que los elementos se pueden sobrescribir a medida que se calculan, lo que puede ser ventajoso para problemas muy grandes.

Sin embargo, a diferencia del método de Jacobi, los cálculos para cada elemento son generalmente mucho más difíciles de implementar en paralelo, ya que pueden tener una ruta crítica muy larga y, por lo tanto, son más factibles para matrices dispersas. Además, los valores en cada iteración dependen del orden de las ecuaciones originales.

Las propiedades de convergencia del método de Gauss-Seidel dependen de la matriz A. El procedimiento converge si:

- A es simétrica definida positiva, o
- A tiene diagonal estrictamente dominante.

El método de Gauss-Seidel converge a veces incluso si estas condiciones no se cumplen.

Algoritmo Método de Gauss-Seidel

While $\frac{\|x_k - x_{k-1}\|_{L\infty}}{\|x\|_{L\infty}} > tolerance$:

For $i = 1, 2, \cdots, n$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

Implementación del Método de Gauss-Seidel en Python

In [*]:

```
def gauss_seidel(A, b, tolerance=1e-10, max_iterations=10000):

    x = np.zeros_like(b, dtype=np.double)

    #Iterate
    for k in range(max_iterations):
        print('iteration:', k)

        x_old  = x.copy()

        #Loop over rows
        for i in range(A.shape[0]):
            x[i] = (b[i] - np.dot(A[i,:i], x[:i]) - np.dot(A[i,(i+1):], x_old[(i+1):]))

        #Stop condition
        if np.linalg.norm(x - x_old, ord=np.inf) / np.linalg.norm(x, ord=np.inf) < toler
            break

    return x
```

Ejemplo: a continuación vamos a aplicar el método de Gauss-Seidel al sistema de ecuaciones lineales $Ax = b$ donde:

$$A = \begin{bmatrix} 16 & 3 \\ 7 & -11 \end{bmatrix} \quad \text{y} \quad b = \begin{bmatrix} 11 \\ 13 \end{bmatrix},$$

In [*]:

```
import numpy as np

A = np.array([[16,3], [7, -11]])

b = np.array([11, 13])
```

In [*]:

```
gauss_seidel(A, b, tolerance=1e-10, max_iterations=20)
```

Bibliografía

- Método de Gauss-Seidel (https://es.wikipedia.org/wiki/M%C3%A9todo_de_Gauss-Seidel)
- Iterative Methods for Solving Linear Systems of Equations UTexas (https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_IterativeSolvers.html)
- Gauss–Seidel method (https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method)

In [ ]:

```
```