

Método de Jacobi para resolver sistemas de ecuaciones lineales

Jonatan Enrique Badillo Tejeda y Javier Dolores Tolentino

May 7, 2024

Abstract

Este documento presenta una explicación del método de Jacobi para resolver sistemas de ecuaciones lineales, junto con dos implementaciones (secuencial y paralela) de este método en Java.

1 Explicación del método de Jacobi

El método de Jacobi es un algoritmo iterativo para resolver sistemas de ecuaciones lineales de la forma $Ax = b$ donde A es una matriz de coeficientes, x es el vector de incógnitas y b es el vector de términos independientes.

El método de Jacobi descompone la matriz A en una matriz diagonal D y el resto R , de modo que $A = D + R$. Luego, se resuelve iterativamente la ecuación $Dx = b - Rx$ hasta que la solución converja.

Para cada iteración k y para cada elemento i del vector x , se calcula:

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

Donde a_{ij} son los elementos de la matriz A , b_i son los elementos del vector b , y $x_j^{(k-1)}$ son los elementos del vector x en la iteración anterior.

El algoritmo se detiene cuando la norma del error (la suma de las diferencias absolutas entre las soluciones actuales y las de la iteración anterior) es menor que un criterio de convergencia predefinido:

$$\sum_{i=1}^n |x_i^{(k)} - x_i^{(k-1)}| < \epsilon$$

Donde ϵ es el criterio de convergencia.

2 Implementación en Java

```
public class JacobiSecuencial {
    public static void main(String[] args) {
        // Define tu matriz A y el vector B aquí
        double[][] A = {
            {4, -1, 0, 0},
            {-1, 4, -1, 0},
        }
```

```

        {0, -1, 4, -1},
        {0, 0, -1, 3}
    };

    double[] B = {15, 10, 10, 10};
    double[] X = new double[B.length]; // Vector de soluciones iniciales, inicializado a 0
    double[] lastX = new double[B.length]; // Vector para almacenar la solución de la iteración anterior
    double error = 1e-10; // Criterio de convergencia

    while (true) {
        for (int i = 0; i < A.length; i++) {
            double sum = B[i]; // Inicializa la suma con el elemento correspondiente del vector B
            for (int j = 0; j < A[i].length; j++) {
                if (j != i) sum -= A[i][j] * lastX[j]; // Resta los productos Aij*Xj para j != i
            }
            X[i] = 1/A[i][i] * sum; // Divide por el coeficiente diagonal Aii
        }

        // Calcula la norma del error como la suma de las diferencias absolutas entre la solución actual y la anterior
        double errorNorm = 0;
        for (int i = 0; i < X.length; i++) {
            errorNorm += Math.abs(X[i] - lastX[i]);
            lastX[i] = X[i]; // Actualiza lastX con la solución actual para la próxima iteración
        }

        // Si la norma del error es menor que el criterio de convergencia, termina el bucle
        if (errorNorm < error) break;
    }

    // Imprime el resultado
    for (double x : X) System.out.println(x);
}
}

```