Método de Jacobi

Los métodos directos son variantes de la eliminación gaussiana. Estos métodos utilizan los elementos de matriz individuales directamente, a través de operaciones matriciales. Los mismos permiten resolver ecuaciones lineales con un alto nivel de precisión, pero pueden ser lentos cuando operan sobre matrices dispersas grandes. La velocidad de resolución de un sistema lineal con un método directo depende en gran medida de la densidad y el tamaño de la matriz de coeficientes.

Los métodos iterativos producen una solución aproximada al sistema lineal después de un número finito de pasos. Estos métodos son útiles para grandes sistemas de ecuaciones en los que es razonable sacrificar la precisión por un tiempo de ejecución más corto. Los métodos iterativos utilizan la matriz de coeficientes solo indirectamente, a través de un producto matriz-vector o un operador lineal abstracto.

Los métodos iterativos se pueden usar con cualquier matriz, pero normalmente se aplican a matrices dispersas grandes para las que las soluciones directas son lentas. La velocidad de resolución de un sistema lineal con un método indirecto no depende tanto de la densidad de coeficientes en la matriz como con el método directo. Sin embargo, el uso de un método iterativo normalmente requiere ajustar los parámetros para cada problema específico.

La mayoría de los algoritmos iterativos que resuelven ecuaciones lineales siguen un proceso similar:

1. Comienzar con una estimación inicial del vector solución x_0, el cual suele ser un vector de ceros, a menos que se tenga una estimación mejor.
2. Calcular la norma residual $res = norm(b - Ax_0)$.
3. Comparar el residuo con la tolerancia especificada. Si $res \leq tol$, finaliza el cálculo y devuelve la respuesta calculada para x_0.
4. Aplicar Ax_0 y actualizar la magnitud y la dirección del vector x_0 según el valor del residual y otras cantidades calculadas. Este es el paso donde se realiza la mayor parte del cálculo.
5. Repita los pasos 2 a 4 hasta que el valor de x_0 sea lo suficientemente bueno para satisfacer la tolerancia.

Los métodos iterativos difieren en la forma en que actualizan la magnitud y la dirección de x_0 en el Paso 4, y algunos tienen criterios de convergencia ligeramente diferentes en los Pasos 2 y 3, pero la descripción anterior captura el proceso básico que siguen todos los solucionadores iterativos.

El método de Jacobi es un método iterativo, usado para resolver sistemas de ecuaciones lineales del tipo $A\mathbf{x} = \mathbf{b}$. El método permite determinar las soluciones de sistemas de ecuaciones lineales con diagonal estrictamente dominante. Cada elemento de la diagonal se resuelve y se introduce un valor aproximado. Luego, el proceso se itera hasta que converge.

Este algoritmo es una versión simplificada del algoritmo de valores propios de Jacobi, un método iterativo para el cálculo de los valores y vectores propios de una matriz simétrica real (un proceso conocido como diagonalización).

La base del método consiste en construir una sucesión convergente definida iterativamente. El límite de esta sucesión es precisamente la solución del sistema. A efectos prácticos si el algoritmo se detiene después de un número finito de pasos se llega a una aproximación al valor de x de la solución del sistema.

La sucesión se construye descomponiendo la matriz del sistema A en la forma siguiente:

$$A = D + L + U = D + R$$

donde D, es una matriz diagonal, L es una matriz triangular inferior y U es una matriz triangular superior, y $R = L + U$.

Partiendo de $A\mathbf{x} = \mathbf{b}$, podemos reescribir dicha ecuación como:

$$\mathrm{D}\mathbf{x} + \mathrm{R}\mathbf{x} = \mathbf{b}$$

Luego,

$$\mathbf{x} = \mathrm{D}^{-1}\left(\mathbf{b} - \mathrm{R}\mathbf{x}\right)$$

Si $a_{ii} \neq 0$ para todo i, por la regla iterativa, el Método de Jacobi se puede expresar de la forma:

$$\mathbf{x}^{(k+1)} = \mathrm{D}^{-1}\left(\mathbf{b} - \mathrm{R}\mathbf{x}^{(k)}\right)$$

donde k es el contador de iteración. Finalmente tenemos:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)}\right), i = 1, 2, 3, \ldots$$

El método de Jacobi siempre converge si la matriz A es de diagonal estrictamente dominante.

Se dice que una matriz cuadrada es de diagonal estrictamente dominante si, para cada fila de la matriz, la magnitud del coeficiente de la diagonal es mayor a la suma de las magnitudes de todos los demás coeficientes (no diagonales) en dicha fila.

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{for all } i$$

donde a_{ij} denota el coeficiente correspondiente a la fila i, columna j.

Para verificar la convergencia del método se calcula el radio espectral (ρ):

$$\rho(D^{-1}R) < 1.$$

El método puede converger incluso si esta condición no se satisface

Algoritmo Método de Jacobi

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

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

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

In [2]:

```
import numpy as np

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

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

    T = A - np.diag(np.diagonal(A))

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

        x_old  = x.copy()

        x[:] = (b - np.dot(T, x)) / np.diagonal(A)

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

    return x
```

In [3]:

```
A = np.array([[2,1],[5,7]])
b = np.array([11,13])
xest = jacobi(A,b,tolerance=0.1,max_iterations=10)
print('x estimada', xest)
```

```
iteration: 0
iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
x estimada [ 6.78717201 -3.07543732]
```

In [4]:

```
A = np.array([[2,1],[5,7]])
b = np.array([11,13])
xest = jacobi(A,b,tolerance=0.01,max_iterations=20)
print('x estimada', xest)
```

```
iteration: 0
iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
iteration: 6
iteration: 7
iteration: 8
iteration: 9
x estimada [ 7.06979235 -3.20349966]
```

In [5]:

```
A = np.array([[2,1],[5,7]])
b = np.array([11,13])
xest = jacobi(A,b,tolerance=0.001,max_iterations=20)
print('x estimada', xest)
```

```
iteration: 0
iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
iteration: 6
iteration: 7
iteration: 8
iteration: 9
iteration: 10
iteration: 11
iteration: 12
iteration: 13
iteration: 14
x estimada [ 7.10991707 -3.21845776]
```

In [6]:

```
A = np.array([[2,1],[5,7]])
b = np.array([11,13])
xest = jacobi(A,b,tolerance=0.0001,max_iterations=100)
print('x estimada', xest)
```

```
iteration: 0
iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
iteration: 6
iteration: 7
iteration: 8
iteration: 9
iteration: 10
iteration: 11
iteration: 12
iteration: 13
iteration: 14
iteration: 15
iteration: 16
iteration: 17
iteration: 18
x estimada [ 7.11095881 -3.22174206]
```

tolerance	max_iterations	# iterations	results
0.1	10	5	[6.78717201 -3.07543732]
0.01	20	9	[7.06979235 -3.20349966]
0.001	20	14	[7.10991707 -3.21845776]

In [7]:

```
np.matmul(A,xest)-b
```

Out[7]:

```
array([0.00017556, 0.00259962])
```

Como vemos, el primer coeficiente tiene los 3 primeros dígitos iguales a cero, mientras que el segundo coeficiente solo tiene 2 hasta el momento, aunque solo hemos necesitado 18 iteraciones.

Bibliografía

- Iterative Methods for Linear Systems (https://es.mathworks.com/help/matlab/math/iterative-methods-for-linear-systems.html)
- Jacobi method (https://en.wikipedia.org/wiki/Jacobi_method)
- Método de Jacobi (https://es.wikipedia.org/wiki/M%C3%A9todo_de_Jacobi)
- Jacobi eigenvalue algorithm (https://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm)
- Matriz de diagonal estrictamente dominante (https://es.wikipedia.org/wiki/Matriz_de_diagonal_estrictamente_dominante)
- Iterative Methods for SolvingLinear Systems of Equations (https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_IterativeSolvers.html)

In [ ]:

```

```