Método para matrices tridiagonales

Daniela Beltrán Saavedra / Nicolás Duarte Ospina

Una matriz tridiagonal se corresponde a un sistema de ecuaciones de la forma.

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

Donde $a_1 = 0$ y $c_n = 0$ lo que se puede representar matricialmente como.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

Para este tipo de sistemas se puede obtener con este algoritmo una solución con solo O (n) operaciones en vez de las O (n^3) que requiere la eliminación gaussiana.

El primer paso del método es modificar los coeficientes como sigue.

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\[2ex] \dfrac{c_i}{b_i - c'_{i-1} a_i} & ; \quad i = 2, 3, \ldots, n-1 \end{cases}$$

donde se marcan con superíndice ' los nuevos coeficientes.

De igual manera se opera:

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\[2ex] \dfrac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} & ; \quad i = 2, 3, \ldots, n. \end{cases}$$

a lo que se llama barrido hacia adelante. A continuación, se obtiene la solución por sustitución hacia atrás:

$$x_n = d'_n$$
$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \ldots, 1.$$

Algoritmo

Entradas:

1. columnas = int(input()): Número de columnas

2. aux = columnas: Auxiliar de columnas

3. matriz = []: Matriz tridiagonal

4. numero = int(input())

matriz[i][j] = numero: Números de la matriz

5. solu = int(input())

d[i] = solu: Solución de la matriz tridiagonal

Salida:

1. g: Arreglo con los nuevos valores calculados

Deducción de la formula

Se puede obtener dicho algoritmo usando la forma gauss de forma genérica. Suponiendo como incógnitas X1…., Xn y con ecuaciones a resolver:

$$\begin{aligned} b_1 x_1 + c_1 x_2 &= d_1; \qquad & i &= 1 \\ a_i x_{i-1} + b_i x_i + c_i x_{i+1} &= d_i; \qquad & i &= 2, \ldots, n-1 \\ a_n x_{n-1} + b_n x_n &= d_n; \qquad & i &= n. \end{aligned}$$

Se puede realizar la eliminación normalmente hasta tener:

$$(a_3 x_2 + b_3 x_3 + c_3 x_4)(b_2 b_1 - c_1 a_2) - ((b_2 b_1 - c_1 a_2) x_2 + c_2 b_1 x_3) a_3 = d_3 (b_2 b_1 - c_1 a_2) - (d_2 b_1 - d_1 a_2) a_3$$
$$(b_3 (b_2 b_1 - c_1 a_2) - c_2 b_1 a_3) x_3 + c_3 (b_2 b_1 - c_1 a_2) x_4 = d_3 (b_2 b_1 - c_1 a_2) - (d_2 b_1 - d_1 a_2) a_3 .$$

Entonces

Examinando el procedimiento, se pueden definir de forma recursiva:

$$\begin{aligned} \tilde{a}_i &= 0 \\ \tilde{b}_1 &= b_1 \\ \tilde{b}_i &= b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i \\ \tilde{c}_1 &= c_1 \\ \tilde{c}_i &= c_i \tilde{b}_{i-1} \\ \tilde{d}_1 &= d_1 \\ \tilde{d}_i &= d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i . \end{aligned}$$

Esto nos da un sistema de ecuaciones con las mismas incógnitas, resolviendo esta, podremos resolver de forma iterativa la anterior y así sucesivamente, todo en función de los originales.

Condición de convergencia

Para las condiciones de convergencia este método tiene una lista corta para comprobar.

- No deben existir divisiones por cero.
- Matriz triangular superior e inferior, cuadrada.
- Deben existir las 3 diagonales en la matriz.
- El método iterativo debe garantizar que se de una solución para cada incógnita.

Método en Python

Para el método en Python no existen paquetes para resolución de matrices tridiagonales, todo se implementó desde cero.

Ejercicios

- Compruebe el método con la matriz.

$$\begin{matrix} 2 & 6 & 0 \\ 1 & 2 & 1 \\ 0 & 4 & 2 \end{matrix}$$

Con soluciones $\begin{matrix} 8 \\ 4 \\ 6 \end{matrix}$.

Bibliografía

- SISTEMAS_NO_LINEALES_NEWTON.pdf (suministrado por la profesora).
- https://es.wikipedia.org/wiki/Algoritmo_para_matrices_tridiagonales.