

ANÁLISIS NUMÉRICO I/ANÁLISIS NUMÉRICO – 2021

Trabajo de Laboratorio N^o 6

Aclaración: Todas las matrices y vectores mencionados como entrada/salida de funciones en este laboratorio deberán considerarse arreglos de `numpy`, es decir clases `numpy.ndarray`.

1. Escribir dos funciones en `python` llamadas `soltrsup` y `soltrin` que resuelvan el sistema lineal $Ax = b$, donde A es una matriz triangular (superior e inferior, respectivamente). La entrada debe ser (A, b) con $A \in \mathbb{R}^{n \times n}$ matriz triangular y $b \in \mathbb{R}^n$, y la salida debe ser la solución x . Se debe imprimir un mensaje de error si la matriz es singular.
2.
 - a) Escribir una función llamada “`egauss`” que implemente el método de eliminación Gaussiana. Debe tener entrada (A, b) con $A \in \mathbb{R}^{n \times n}$ y $b \in \mathbb{R}^n$, con salida $[U, y]$ con $U \in \mathbb{R}^{n \times n}$ triangular superior e $y \in \mathbb{R}^n$.
 - b) Escribir una función llamada “`soleg`” que resuelva sistemas lineales $Ax = b$ usando eliminación Gaussiana y resolviendo el sistema triangular superior $Ux = y$ (usando `soltrsup`). Debe tener entrada (A, b) con $A \in \mathbb{R}^{n \times n}$ y $b \in \mathbb{R}^n$ y, la salida debe ser la solución x .
3. Escribir una función llamada “`sollu`” que resuelva sistemas lineales $Ax = b$ usando descomposición LU con pivoteo (para obtener dicha descomposición investigar el subpaquete de la librería `scipy`: “`linalg`”) para luego resolver $Ly = P^{-1}b$ (¿cómo se puede obtener la inversa de una matriz de pivoteo?) y $Ux = y$ usando `soltrin` y `soltrsup`. La salida debe ser la solución x y debe tener entrada (A, b) con $A \in \mathbb{R}^{n \times n}$ y $b \in \mathbb{R}^n$.
4. Comparar las soluciones dadas por `soleg` y `sollu` al resolver $Ax = b$ con

$$A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{y también } b_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

5. Escribir dos funciones llamadas “`jacobi`” y “`gseidel`” que resuelvan sistemas lineales $Ax = b$ usando los métodos de Jacobi y Gauss-Seidel, respectivamente. La salida debe ser $[x, k]$ donde x es la solución aproximada y k la cantidad de iteraciones realizadas. Debe tener entrada $(A, b, \text{err}, \text{mit})$ con $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, `err` tolerancia de error y `mit` cantidad máxima de iteraciones. El algoritmo debe parar si $\|x^{(k)} - x^{(k-1)}\|_\infty \leq \text{err}$ o $k \geq \text{mit}$.
6. Usar los métodos de Jacobi y Gauss-Seidel para resolver

$$(1) \quad \begin{bmatrix} 3 & 1 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 6 \end{bmatrix}, \quad (2) \quad \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}.$$

con una tolerancia de 10^{-11} para (1) y 10^{-4} para (2). ¿Cuántas iteraciones son necesarias en cada caso para alcanzar la precisión deseada?