

ANyP, Práctica 13.

Métodos numéricos para resolver sistemas de ecuaciones lineales.

Eliminación gaussiana. Factorización LU.

Sea

$$A \mathbf{x} = \mathbf{b}$$

un sistema de n ecuaciones lineales con n incógnitas, donde A es una matriz cuadrada de orden n no singular, por lo tanto el sistema admite una solución y ésta es única. El procedimiento de eliminación gaussiana con una estrategia de pivoteo que permita el intercambio de filas (típicamente, el pivoteo parcial, llamado también pivoteo máximo de columna) muestra que la matriz de coeficientes del sistema admite una factorización de la forma

$$A = PLU.$$

L es una *matriz triangular inferior* con elementos diagonales iguales a la unidad y cuyos otros elementos bajo la diagonal son los multiplicadores utilizados durante la eliminación. U es una *matriz triangular superior* cuyos elementos son los coeficientes del sistema final equivalente, siendo los elementos de la diagonal no nulos. Si los intercambios de fila durante el proceso de eliminación son registrados en un vector $\mathbf{p}$ (cuyo valor inicial es $(1, 2, \dots, n)$), entonces la *matriz de permutación* P se obtiene a partir de la matriz identidad I de orden n permutando sucesivamente la columna i por la columna $j = \mathbf{p}(i)$ para $i = 1, 2, \dots, n$. Por ejemplo, si para $n = 4$, $\mathbf{p} = (3, 4, 3, 4)$, entonces

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Conocida la factorización LU , el sistema de ecuaciones lineales es equivalente a $PLU \mathbf{x} = \mathbf{b}$, el cual conduce a dos sistemas triangulares,

$$L \hat{\mathbf{b}} = P^{-1} \mathbf{b}, \quad U \mathbf{x} = \hat{\mathbf{b}}.$$

Así, para obtener el vector solución $\mathbf{x}$, se resuelve en primer lugar el primer sistema por *sustitución hacia adelante* para obtener $\hat{\mathbf{b}}$ y luego se resuelve el segundo sistema por *sustitución hacia atrás*.

Python proporciona un amplio conjunto de funciones para resolver sistemas lineales dependiendo de la naturaleza particular de la matriz de coeficientes.

En particular, la función `np.linalg.solve(A,b)` de la librería *NumPy*, resuelve el sistema $A \mathbf{x} = \mathbf{b}$ usando

la descomposición LU, pero no nos proporciona las matrices L , U y P .

Si además queremos conocer la descomposición LU, podemos utilizar la librería *scipy.linalg*, combinando las funciones `scipy.linalg.lu_factor` y `scipy.linalg.lu_solve`.

La función `scipy.linalg.lu_factor` nos devuelve la descomposición LU de una matriz A , pero en una única matriz. Si queremos recuperar a partir de la matriz que nos devuelve `scipy.linalg.lu_factor` las matrices L y U , podemos usar las funciones de *NumPy* `np.tril`, `np.eye` y `np.triu`. La función `scipy.linalg.lu_factor` además nos devuelve el vector $\mathbf{p}$. A partir de este vector podemos obtener la matriz P .

Finalmente, para conocer la solución del sistema, debemos usar la función `scipy.linalg.lu_solve`. Aquí debemos utilizar como argumentos de la función la matriz LU y el vector $\mathbf{p}$, obtenidos con la función `scipy.linalg.lu_factor`, y el vector $\mathbf{b}$.

Ejercicio 1. Utilizar las funciones mencionadas previamente que crea más conveniente para resolver los siguientes sistemas $A \mathbf{x} = \mathbf{b}$:

a)

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 5 \end{pmatrix}$$

b)

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4 \\ 12 \\ 17 \end{pmatrix}$$

c)

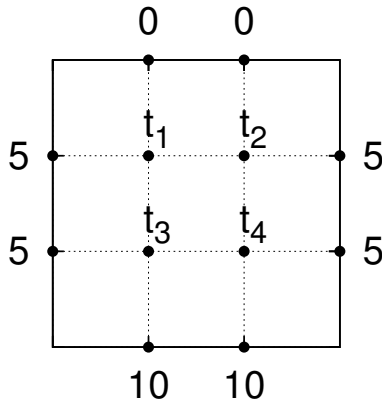
$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

En cada caso explicitar la factorización PLU de las matrices A .

Ejercicio 2. Considere una placa cuadrada cuya temperatura en los bordes es de 0°C en el lado superior, 10°C en el lado inferior y 5°C en cada uno de los lados laterales. Suponga que se traza una malla sobre la placa y se numeran los puntos de intersección como se indica en la figura. Suponiendo que la temperatura en un punto interior de la malla es igual al promedio de las temperaturas de los cuatro puntos

circundantes, se obtiene para el punto 1,

$$t_1 = \frac{1}{4}(5 + 0 + t_2 + t_3)$$



- Escriba el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ que resulta de considerar las temperaturas en los cuatro puntos interiores de la malla.
- Resuelva el sistema anterior para obtener la distribución de temperatura en tales puntos.

Métodos iterativos.

Un método iterativo para resolver el sistema lineal $A\mathbf{x} = \mathbf{b}$ comienza con una aproximación inicial $\mathbf{x}^{(0)}$ a la solución $\mathbf{x}$, y genera una sucesión de vectores $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ que se espera converja a $\mathbf{x}$. Los dos métodos iterativos más conocidos son el *método de Jacobi* y el *método de Gauss-Seidel* cuyas implementaciones algorítmicas se describen a continuación.

🚩 Método de Jacobi

Dada A de $n \times n$ con elementos diagonales no nulos. Escoger $\mathbf{x}^{(0)}$ de orden n , usualmente $\mathbf{x}^{(0)} = \mathbf{0}$.

Para $k = 1, 2, \dots$

Tomar

$$x_i^{(k)} = \frac{b_i - \sum_{\substack{j=1 \\ (j \neq i)}}^n a_{ij} x_j^{(k-1)}}{a_{ii}}, \quad \text{para } i = 1, \dots, n.$$

🚩 Método de Gauss-Seidel

Dada A de $n \times n$ con elementos diagonales no nulos. Escoger $\mathbf{x}^{(0)}$ de orden n , usualmente $\mathbf{x}^{(0)} = \mathbf{0}$.

Para $k = 1, 2, \dots$

Tomar

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)}}{a_{ii}}, \quad \text{para } i = 1, \dots, n.$$

Como en todo método iterativo, debe establecerse un criterio de paro para las iteraciones, el cual, en este contexto, puede ser

$$\frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|}{\|\mathbf{x}^{(k)}\|} < \epsilon,$$

para cierta tolerancia ϵ prefijada.

Ejercicio 3. Escribir sendos programas en Python para las implementaciones del método de Jacobi y del método de Gauss-Seidel.

Ejercicio 4. Utilizar los programas anteriores para resolver los siguientes sistemas de ecuaciones:

a)

$$\begin{aligned} 3.8x_1 + 1.6x_2 + 0.9x_3 &= 15.5 \\ -0.7x_1 + 5.4x_2 + 1.6x_3 &= 10.3 \\ 1.5x_1 + 1.1x_2 - 3.2x_3 &= 3.5 \end{aligned}$$

b)

$$\begin{aligned} x_1 &+ x_3 &= 4 \\ -x_1 + x_2 &= 1 \\ x_1 + 2x_2 - 3x_3 &= -4 \end{aligned}$$

c)

$$\begin{aligned} x_1 + 0.5x_2 + 0.5x_3 &= 2 \\ 0.5x_1 + x_2 + 0.5x_3 &= 2 \\ 0.5x_1 + 0.5x_2 + x_3 &= 2 \end{aligned}$$

Los sistemas del ejercicio anterior muestran que los métodos iterativos no siempre son convergentes. La condición *necesaria y suficiente* para la convergencia es dada como sigue.

Sea A la matriz cuadrada de orden n del sistema. Denotemos por D a la matriz diagonal de orden n formada con los elementos de la diagonal principal de A y ceros en los demás elementos. Sea $-L$ la matriz triangular inferior de orden n formada con los elementos de A situados bajo la diagonal principal y ceros en los demás elementos. Sea $-U$ la matriz triangular superior de orden n formada con los elementos situados por arriba de la diagonal principal y ceros en los demás elementos. Entonces

$$A = D - L - U.$$

Sea

$$T = \begin{cases} D^{-1}(L + U) & \text{para Jacobi,} \\ (D - L)^{-1}U & \text{para Gauss-Seidel.} \end{cases}$$

Entonces el método iterativo respectivo converge para cualquier elección del vector inicial, si y solo si $\rho(T) < 1$, donde $\rho(T)$ denota el radio espectral de T . Además, la velocidad de convergencia del método

depende de $\rho(T)$, cuanto menor sea este valor más rápida será la convergencia.

Condiciones suficientes, pero *no* necesarias para la convergencia de los métodos, que se deducen de lo anterior, son las siguientes.

Si para alguna norma natural $\|T\| < 1$ entonces el método iterativo correspondiente converge.

Si A es una matriz *estrictamente diagonal dominante*, esto es,

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad \text{para } i = 1, \dots, n$$

entonces tanto el método de Jacobi como el método de Gauss-Seidel convergen.

Ejercicio 5. Justificar la convergencia (o divergencia) de los métodos iterativos para los sistemas del ejercicio anterior.

Estimación del error. Número de condición.

En la práctica, una solución *numérica* $\hat{\mathbf{x}}$ de un sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$ tendrá cierto error $\|\mathbf{x} - \hat{\mathbf{x}}\|$, el cual, por su puesto no puede ser calculado. Ahora bien, siempre podemos calcular el *vector residual* $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$. Puesto que $\mathbf{r} = \mathbf{0}$ implica que $\hat{\mathbf{x}}$ es la solución exacta del sistema, resulta natural sugerir que si $\|\mathbf{r}\|$ es pequeña, entonces $\hat{\mathbf{x}}$ es una solución aproximada precisa. Sin embargo, esto no es siempre el caso. La exactitud de la solución calculada depende también del *número de condición* $\kappa(A) = \|A\| \|A^{-1}\|$ de acuerdo a la siguiente desigualdad

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

Esta desigualdad nos dice que el error relativo en una solución numérica puede ser tan grande como $\kappa(A)$ veces su residuo relativo. De este modo, solo si $\kappa(A) \simeq 1$ el error relativo y el residuo relativo son del mismo tamaño, y el residuo podrá ser utilizado con seguridad como una estimación del error. En esta situación se dice que la matriz A del sistema es una *matriz bien condicionada*. Si, por el contrario, $\kappa(A) \gg 1$, el intervalo en que se puede situar el error relativo es muy amplio y por lo tanto no se puede asegurar que la solución numérica es precisa, aún cuando el vector residual sea pequeño.

En esta situación, se dice que la matriz A es una *matriz mal condicionada*.

Ejercicio 6. Mostrar que para toda matriz no singular, $\kappa(A) \geq 1$.

Ejercicio 7. Sea $A\mathbf{x} = \mathbf{b}$ el sistema lineal dado por

$$A = \begin{pmatrix} 0.89 & 0.53 \\ 0.47 & 0.28 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0.36 \\ 0.19 \end{pmatrix}.$$

Dada la solución aproximada $\hat{\mathbf{x}} = (-11.5, 20)^t$, calcule el vector residual y estime el error relativo. Sabiendo que la solución exacta es $\mathbf{x} = (1, -1)^t$, calcule el error exacto. Compare los resultados.

Ejercicio 8. Dado el sistema $A\mathbf{x} = \mathbf{b}$, suponga que el término $\mathbf{b}$ se perturba por una cantidad $\delta\mathbf{b}$. Mostrar que el efecto de esta perturbación sobre la solución será $\mathbf{x} + \delta\mathbf{x}$, con $\delta\mathbf{x}$ acotada por

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}.$$

Si ahora es la matriz A la que es perturbada por una cantidad δA , mostrar que el efecto de tal perturbación sobre la solución será $\mathbf{x} + \delta\mathbf{x}$ con $\delta\mathbf{x}$ acotada por

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x} + \delta\mathbf{x}\|} \leq \kappa(A) \frac{\|\delta A\|}{\|A\|}.$$

Ejercicio 9. Sea

$$A = \begin{pmatrix} 6 & 13 & -17 \\ 13 & 29 & -38 \\ -17 & -38 & 50 \end{pmatrix}$$

Si se desea resolver el sistema $A\mathbf{x} = \mathbf{b}$, donde $\mathbf{b} = (1.9358, -2.3412, 3.5718)^t$ tiene sus elementos redondeados a 5 dígitos, ¿qué puede decirse del error relativo en la solución?

Ejercicio 10. Realizar un programa para calcular la solución del siguiente sistema de ecuaciones no lineales, en la región $[1.0, 3.0] \times [1.0, 3.0]$, aplicando el método de Newton.

$$\begin{aligned} x^2 + y^2 &= 10 \\ x^2 - y^2 &= 1 \end{aligned}$$

En cada paso iterativo, resolver el sistema lineal utilizando algún método de los disponibles en Python.