

Modelos de predicción en edificación

Bloque 1: Ecuaciones diferenciales y sistemas dinámicos

Sección 2: Álgebra lineal numérica

Toni Guillamon

Dept. Matemàtiques - UPC

29 de Septiembre y 6 de octubre de 2020

Clasificación de sistemas lineales

$$Ax = b,$$

donde $A \in M_{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $m, n \in \mathbb{N}$. (m ecuaciones, n variables).

- **Compatible.** $\text{rg}A = \text{rg}(A|b)$.
 - Solución única (**determinado**). $\text{rg}A = \text{rg}(A|b) = n$.
 - Infinitas soluciones (**indeterminado**, k grados de libertad).
 $\text{rg}A = \text{rg}(A|b) = n - k$.
- **Incompatible.** $\text{rg}A \neq \text{rg}(A|b)$.

Clasificación de sistemas lineales: ejemplos I

- **Compatible.** $\text{rg}A = \text{rg}(A|b)$.
 - Solución única (**determinado**). $\text{rg}A = \text{rg}(A|b) = n$.

$$\begin{pmatrix} 2 & 3 \\ -1 & 3 \\ 1 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 3 \end{pmatrix}$$

$$m = 3, n = 2, \text{rg}A = \text{rg}(A|b) = 2 = n;$$

$$x_1 = 7/3, \quad x_2 = 1/9.$$

Clasificación de sistemas lineales: ejemplos II

- Infinitas soluciones (**indeterminado**, k grados de libertad).

$$\begin{pmatrix} 2 & 3 & 5 \\ -1 & 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

$$m = 2, n = 3, \operatorname{rg} A = \operatorname{rg}(A|b) = 2 = n - 1 \text{ (1 d.o.f);}$$

$$x_1 = 2/3 - x_3/3, \quad x_2 = 11/9 - 13/9 x_3, \quad x_3 = x_3.$$

Clasificación de sistemas lineales: ejemplos III

- Incompatible. $\text{rg}A \neq \text{rg}(A|b)$.

$$\begin{pmatrix} 2 & 3 \\ -1 & 3 \\ 1 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 3 \end{pmatrix}$$

$$m = 3, n = 3, \text{rg}A = 2 \neq \text{rg}(A|b) = 3.$$

Realmente, ¿debemos descartarlos?

Aparecen de manera muy natural en las aplicaciones (por ejemplo, cuando tenemos muchos datos); normalmente, se resuelven por el **método de los mínimos cuadrados** (*Least Squares Method*, LSM).

Matrices de grandes dimensiones

Por ejemplo, en **cálculo de estructuras** (estructuras reticulares, celosías).



Ejemplo: matrices de rigidez, una simple barra en 2D

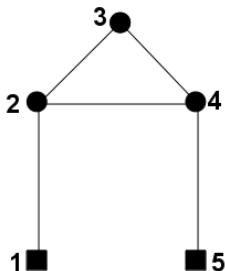


La matriz de rigidez (**local**) K^L se construye a partir de propiedades físico-geométricas de las barras: E (el módulo de Young), I (el momento de inercia), A (el área de la sección transversal) y L (la longitud).

Los desplazamientos en x_i e y_i y la rotación θ_i que sufre el nodo i bajo la acción de las cargas $\vec{f}$ (fuerzas y momentos) son las soluciones de un sistema lineal “local” (6×6) simétrico: $\vec{f}_L = K^L \vec{u}_L$,

$$\begin{pmatrix} f_{x1} \\ f_{y1} \\ m_1 \\ f_{x2} \\ f_{y2} \\ m_2 \end{pmatrix} = \begin{pmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \end{pmatrix}$$

Ejemplo: matrices de rigidez, una pórtico plano



Denotamos por $K^{r,s}$ el **global** (via rotación) **matriz de rigidez** de la barra que une los nodos r y s ; entonces, la **matriz de rigidez de la estructura entera** es una matriz (15×15) .

Ejemplo: matrices de rigidez

Denotamos por $K^{r,s}$ el **global** (via rotación) **matriz de rigidez** de la barra que une los nodos r y s ; entonces, la **matriz de rigidez de la estructura entera** es una matriz (15×15) .

$$K^0 = \begin{pmatrix} K_{1,1}^{1,2} & K_{1,2}^{1,2} & 0 & 0 & 0 \\ K_{2,1}^{1,1} & K_{2,2}^{1,2} + K_{1,1}^{2,3} + K_{1,1}^{2,4} & K_{1,2}^{2,3} & K_{1,2}^{2,4} & 0 \\ 0 & K_{2,1}^{2,3} & K_{2,2}^{2,3} + K_{1,1}^{3,4} & K_{1,2}^{3,4} & \\ 0 & K_{2,1}^{2,4} & K_{2,1}^{3,4} & K_{2,2}^{2,4} + K_{2,2}^{3,4} + K_{1,1}^{4,5} & K_{1,2}^{4,5} \\ 0 & & & K_{2,1}^{4,5} & K_{2,2}^{4,5} \end{pmatrix}$$

En general, se necesita resolver un sistema lineal $(3n) \times (3n)$ $\vec{f} = K^0 \vec{u}$, donde n es el número de nodos. Por lo tanto, para valores de n elevados, la elección de un **método numérico apropiado** es crucial.

La dimensión puede reducirse ligeramente según el número de empotramientos y voladizos.

Resolución numérica de sistemas lineales I

Nos gustaría resolver $Ax = b$, donde x contiene las incógnitas y A y b son datos conocidos. Suponemos que A es una matriz cuadrada y el sistema se escribe como:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Resolución de $Ax = b$ para **valores de n elevados**?

Los **métodos** más útiles consisten en factorizar la matriz A como el producto de dos matrices que poseen ciertas propiedades *deseables*:

Resolución numérica de sistemas lineales II

- el método LU se basa en descomponer $A = L U$, siendo L una matriz **triangular inferior** y U una matriz **triangular superior**. Este método se fundamenta en el proceso de eliminación de Gauss.

$$L = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \ddots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix}; \quad U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}.$$

Resolución numérica de sistemas lineales III

- el método QR se basa en descomponer $A = QR$, siendo Q una **matriz ortogonal** y R una matriz **triangular superior**. Este método se fundamenta en el proceso de ortogonalización de Gram-Schmidt.

$$Q^t Q = \text{Id}; \quad R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{pmatrix}.$$

Método LU

El método LU se basa en descomponer $A = LU$, donde L es una matriz **triangular inferior** y U una matriz **triangular superior**.

De manera general, se descompone como

$$PA = LU, \quad P \text{ (matriz permutación),}$$

para permitir la permutación de filas de A y así usar el máximo **pivote** en cada paso, lo que da **estabilidad numérica**. En estas notas, consideramos $P = \text{Id}$. Ver http://en.wikipedia.org/wiki/LU_decomposition para más información.

Veamos un ejemplo, primero en esta exposición, y después en Matlab.

Algoritmo de Doolittle (secundario) I

Creamos una sucesión de matrices $A^{(n)}$ tal que:

$$A^{(n)} = L_n A^{(n-1)}, \quad \text{con } A^{(0)} = A,$$

y:

$$L_n = \begin{pmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{n+1,n} & \ddots & & \\ & & \vdots & & \ddots & \\ 0 & & l_{N,n} & & & 1 \end{pmatrix}.$$

con $l_{i,n} := -\frac{a_{i,n}^{(n-1)}}{a_{n,n}^{(n-1)}}$ para $i = n+1, \dots, N$.

Después de $N-1$ pasos,

$U := A^{(N-1)}$ es triangular superior y

Algoritmo de Doolittle (secundario) II

$L := L_1^{-1} \dots L_{N-1}^{-1}$ es triangular inferior.

En particular,

$$L = \begin{pmatrix} 1 & & & & & 0 \\ -l_{2,1} & \ddots & & & & \\ & & 1 & & & \\ \vdots & & -l_{n+1,n} & \ddots & & \\ & & \vdots & & 1 & \\ -l_{N,1} & & -l_{N,n} & & -l_{N,N-1} & 1 \end{pmatrix}.$$

Claramente, es necesario que $a_{n,n}^{(n-1)} \neq 0$ en cada paso. Si no, se podrían permutar filas y obtener una descomposición $PA = LU$, donde P es una matriz de permutación.

Ejemplo:

$$L = \begin{pmatrix} 3 & 1 & 4 \\ -2 & -4 & 1 \\ 1 & 1 & -2 \end{pmatrix}.$$

Ventajas de la descomposición $A = LU$

Supongamos que A es una matriz cuadrada $n \times n$.

- La resolución de un sistema lineal $Ax = b$ se reduce a la **resolución de 2 sistemas triangulares**; cada uno de ellos se resuelve por sustitución y requiere solo $O(n^2)$ operaciones en punto flotante:
 - Paso 1: Resolver $Ly = b$.
 - Paso 2: Resolver $Ux = y$.
 - Conclusión: Como $Ax = LUx = Ly = b$, entonces x es el solución.
- **Eficiencia computacional**: Teniendo en cuenta la descomposición LU y la resolución de los dos sistemas lineales triangulares, el proceso completo requiere $2n^2/3$ operaciones en punto flotante.
-

$$\det A = \det U = \prod_{i=1}^n u_{ii}.$$

Ventajas de la descomposición $A = L U$ (secundarias)

- Para calcular A^{-1} , la **matriz inversa** de A , resolvemos n sistemas $A x_i = e_i$, donde e_i es el vector unitario con un 1 en la posición i -ésima. Observar que los n sistemas comparten matriz, A , así que solo se precisa 1 sola descomposición LU .
- Si A es una matriz **simétrica**, es posible escoger U como la traspuesta conjugada de L , de manera que

$$A = LL^T \quad (\text{descomposición de Cholesky}).$$

La descomposición de Cholesky siempre existe y es única. Por ejemplo, las matrices de rigidez son simétricas.

- Para **grandes matrices dispersas** se han desarrollado algoritmos de factorización. La idea es tener en cuenta la estructura dispersa L y U de antemano y calcular solo los términos no nulos. Como regla de oro, el coste de computación viene determinado por el número de a_{ij} no nulos.

Vectores y valores propios

Dada una matriz $A \in M_{n \times n}(\mathbb{R})$, un vector $v \in \mathbb{R}^n$ se llama **vector propio** (*eigenvector*) de A si

$$A v = \lambda v,$$

para algun valor $\lambda \in \mathbb{R}$.

A λ se le denomina **valor propio** (*eigenvalue*) asociado a v .

Propiedades:

- Una matriz A puede tener, a lo máximo, n vectores propios y n valores propios. Se calculan a partir del **polinomio característico** de A , $p_A(x) := \det(A - x \text{Id})$; en concreto, son las soluciones de

$$\det(p_A(x)) = 0.$$

Vectores y valores propios (secundario)

- En el caso en que A tiene n valores propios reales y **diagonaliza**, admite una descomposición

$$A = V D V^{-1},$$

donde la i -ésima columna de V es el i -és vector propio (para $i = 1, \dots, n$) y D es una matriz diagonal tal que d_{ii} es el i -ésimo valor propio de A .

- Un algoritmo numérico para encontrar el valor propio con el máximo valor absoluto (sea λ_1) es el **método de la potencia**:

Tomando $b_{k+1} = \frac{A b_k}{\|A b_k\|}$ a partir de un vector inicial b_0 , el cociente

$$\eta_k := \frac{b_k^T A b_k}{b_k^T b_k}$$

converge a λ_1 .

Referencias

- Golub, G.H., van Loan, C.F., “Matrix Computaciones”. 1992.
- Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T
“Numerical recipes: The art de scientific computing”. Cambridge Univ. Press, 1989.