Sistema de ecuaciones lineales: Métodos Directos

Todo sistema de ecuaciones lineales puede escribirse **matricialmente** como:

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ \vdots & \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= b_n \end{cases} \iff \boldsymbol{Ax} = \boldsymbol{b},$$

donde

$$\boldsymbol{A} := \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{R}^{n\times n} \quad \text{y} \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \in \mathbb{R}^n$$

son los datos y $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$ es el vector de incógnitas.

El sistema de ecuaciones lineales $\boldsymbol{Ax} = \boldsymbol{b}$ **tiene solución única** si y sólo si $\boldsymbol{A}$ es una matriz **no singular**. Recordemos que una matriz $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ es no singular si y sólo si se cumple cualquiera de estas condiciones:

1. $\boldsymbol{A}$ es **invertible**: $\exists \boldsymbol{A}^{-1} \in \mathbb{R}^{n\times n} : \ \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$;
2. $\det(\boldsymbol{A}) \neq 0$;
3. todas las filas (y columnas) de $\boldsymbol{A}$ son l.i.: $\text{rango}(\boldsymbol{A}) = n$.
4. 0 no es valor propio de $\boldsymbol{A}$: $0 \notin \sigma(\boldsymbol{A})$.

Si $\boldsymbol{A}$ es no singular, entonces

$$\boldsymbol{Ax} = \boldsymbol{b} \iff \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

Sin embargo, en general, **no es conveniente calcular la matriz inversa $\boldsymbol{A}^{-1}$ para resolver un sistema de ecuaciones**, pues hacerlo así resulta mucho más costoso computacionalmente.

Dificultades numéricas. Los siguientes aspectos deben tenerse en cuenta al diseñar un algoritmo para resolver un sistema de ecuaciones lineales:

- **Costo operacional.** El **tiempo de cálculo** del computador necesario para resolver el sistema debe ser lo menor posible.

  Una medida standard del costo operacional es la cantidad de operaciones aritméticas $(+,\ -,\ *,\ /)$ que requiere un algoritmo. Éste usualmente se expresa en **flop** *(floating point operations)*.

- **Costo de almacenamiento.** La cantidad de **posiciones de memoria** que requiere el computador para ejecutar un algoritmo (representación de los datos, variables auxiliares, etc.) también debe ser la menor posible.

- **Precisión de los resultados.** Los algoritmos deben ser **estables**, en el sentido de amplificar lo menos posible los errores de los datos y los de redondeo.

Costo operacional.

- Los sistemas que aparecen en muchas aplicaciones son de gran tamaño. Un sistema de 1000×1000 hoy se considera de tamaño moderado y en algunas aplicaciones deben resolverse sistemas de ecuaciones con cientos de miles de incógnitas.
- Hay métodos que en teoría permiten resolver cualquier sistema de ecuaciones lineales, pero que en la práctica requieren tiempos de cálculo prohibitivos.
- **Mal ejemplo: Regla de Cramer.** Este procedimiento permite calcular explícitamente la solución de un sistema $\boldsymbol{Ax} = \boldsymbol{b}$ mediante:

$$x_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})}, \qquad i = 1, \ldots, n,$$

  donde $\boldsymbol{A}_i$ es la matriz que se obtiene a partir de $\boldsymbol{A}$ reemplazándo en ésta su columna i-ésima por el segundo miembro $\boldsymbol{b}$.

  Si los determinantes se calculan mediante la fórmula recursiva usual de desarrollo por fila (o por columna), el costo operacional de la Regla de Cramer es de aproximadamente $(n+1)!$ **flop**.
- **Buen ejemplo: Método de Eliminación Gaussiana.** Este procedimiento se basa en el método algebraico de **transformaciones elementales**. Su costo operacional veremos que es de aproximadamente $\frac{2}{3}n^3$ **flop**.
- **Comparación:**

  En un computador de 1 Gflop (10^9 flop) por segundo:

n	10	15	20	100	1000	2000
Regla de Cramer						
flop	4×10^7	2×10^{13}	5×10^{19}	10^{160}	"∞"	"∞"
tiempo	0.04 s	5.5 horas	1500 años	"∞"	"∞"	"∞"
Eliminación Gaussiana						
flop	666	2250	5333	7×10^5	7×10^8	5×10^9
tiempo	0. s	0. s	0. s	0 s	0.73 s	4.88 s

Costo de almacenamiento.

- En muchas aplicaciones los sistemas de ecuaciones lineales que deben resolverse involucran matrices de gran tamaño, pero tales que la mayor parte de sus entradas son nulas.

  Estas matrices se denominan **dispersas** o **ralas** (en inglés y en MATLAB, **sparse**) y existen técnicas para almacenarlas que sólo requieren una cantidad de posiciones de memoria aproximadamente igual al número de entradas no nulas de la matriz.
- Los métodos algebraicos usuales (por ejemplo el de **transformaciones elementales**) requieren modificar la matriz original del sistema y, muchas veces, destruyen el carácter disperso de la misma.
- Para evitar esto, estudiaremos también otros procedimientos (**métodos iterativos**) que no modifican la matriz del sistema, por lo que resultarán más convenientes desde el punto de vista del costo de almacenamiento.

  Estos métodos no se basan en calcular la solución exacta del sistema, sino en construir iterativamente aproximaciones cada vez mejores de la misma.

Precisión de los resultados.

- La resolución de un sistema de ecuaciones lineales en el computador involucra la propagación de **errores en los datos** y **errores de redondeo**. Por ello:

  1. Hay que disponer de alguna técnica que permita **predecir** cuando la resolución de un sistema de ecuaciones puede **propagar drásticamente estos errores**.
  2. Hay que diseñar métodos numéricos estables, que reduzcan la propagación de los errores de redondeo tanto como sea posible.
  3. Hay que diseñar técnicas computacionales que nos permitan, después de calcular la solución de un sistema de ecuaciones, **estimar a posteriori** la precisión de la solución calculada. Es decir, testear si el error con el que se la calculó está por debajo de una tolerancia aceptable.

Solución de sistemas con matriz triangular.

Dadas las matrices

$$\boldsymbol{L} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \quad \text{y} \quad \boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}$$

decimos que $\boldsymbol{L}$ es **triangular inferior** y $\boldsymbol{U}$ es **triangular superior**. Dado que

$$\det(\boldsymbol{L}) = l_{11}l_{22}\cdots l_{nn} \qquad \text{y} \qquad \det(\boldsymbol{U}) = u_{11}u_{22}\cdots u_{nn},$$

una matriz triangular es no singular si y sólo si sus términos diagonales son todos no nulos. Además notamos que la resolución de sistemas de ecuaciones lineales con matrices triangulares es muy sencilla y su costo operacional es bajo.

- Consideremos un sistema $\boldsymbol{Lx} = \boldsymbol{b}$ con matriz triangular inferior $\boldsymbol{L}$. Resolvemos mediante **sustitución progresiva**:

$$\begin{cases} l_{11}x_1 & = & b_1 & \Rightarrow & x_1 = b_1/l_{11} \\ l_{21}x_1 + l_{22}x_2 & = & b_2 & \Rightarrow & x_2 = (b_2 - l_{21}x_1)/l_{22} \\ \vdots & & \vdots & & \\ l_{n1}x_1 + \cdots + l_{nn}x_n & = & b_n & \Rightarrow & x_n = (b_n - l_{n1}x_1 - \cdots - l_{nn-1}x_{n-1})/l_{nn} \end{cases}$$

- **Algoritmo:**

Para $i = 1, \ldots, n$

$$x_i = \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j\right)$$

- **Costo operacional:** $\displaystyle\sum_{i=1}^{n}\left(1 + \sum_{j=1}^{i-1} 2\right) = \sum_{i=1}^{n}(2i-1) = \boldsymbol{n^2}$ **flop**.

De igual forma, se puede deducir el siguiente algoritmo para resolver un sistema $\boldsymbol{Ux} = \boldsymbol{b}$ con matriz triangular superior $\boldsymbol{U}$:

Para $i = n, n-1, \ldots, 1$

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

Método de Eliminación Gaussiana (M.E.G.)

- El **método de eliminación gaussiana** consiste en reducir mediante **transformaciones elementales** un sistema $\boldsymbol{Ax} = \boldsymbol{b}$ a otro **equivalente** (es decir, que tenga la misma solución), de la forma

$$\boldsymbol{Ux} = \tilde{\boldsymbol{b}},$$

donde $\boldsymbol{U}$ es una matriz **triangular superior**. Luego, el sistema resultante se resuelve por el algoritmo descrito para matrices triangulares.

- Denotemos el sistema original por $\boldsymbol{A}^{(1)}\boldsymbol{x} = \boldsymbol{b}^{(1)}$. El proceso empleado consiste en reemplazar las ecuaciones por combinaciones no triviales de las otras. Así, consideremos la matriz no singular $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ y supongamos que el elemento $a_{11}^{(1)}$ es no nulo. Consideremos los **multiplicadores**

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \qquad i = 2, \ldots, n,$$

donde $a_{ij}^{(1)}$ denota el elemento que está en la fila i y columna j de $A^{(1)}$.

- Es posible **eliminar la incógnita x_1 de la segunda ecuación en adelante**, por simple sustracción a la fila i, $i = 2, \ldots, n$, de la primera fila previamente multiplicada por m_{i1} y haciendo lo mismo para el vector $\boldsymbol{b}$:

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, \qquad i, j = 2, \ldots, n, \\ b_i^{(2)} &= b_i^{(1)} - m_{i1} b_1^{(1)}, \qquad i = 2, \ldots, n, \end{aligned}$$

donde $b_i^{(1)}$ denota la componente i-ésima del vector $b^{(1)}$.

- Así se obtiene un sistema $\boldsymbol{A}^{(2)}\boldsymbol{x} = \boldsymbol{b}^{(2)}$ **equivalente** al anterior:

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

- A continuación, si $a_{22}^{(2)} \neq 0$, podemos análogamente **eliminar la incógnita x_2 de la tercera ecuación en adelante**.

- Siguiendo con este proceso k-veces ($k \leq n$), se obtiene el sistema

$$\boldsymbol{A}^{(k)}\boldsymbol{x} = \boldsymbol{b}^{(k)}, \qquad 1 \leq k \leq n,$$

donde la matriz $A^{(k)}$ toma la siguiente forma:

$$\boldsymbol{A}^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

Para realizar este proceso hemos **supuesto** que $a_{ii}^{(i)} \neq 0$, $i = 1, \ldots, k-1$.

- Notemos que para $k = n$ obtenemos el sistema triangular superior

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}$$

- Los valores $a_{kk}^{(k)}$ son llamados **pivotes** y deben ser valores no nulos para $k = 1, \ldots, n-1$.
- Si la matriz original $\boldsymbol{A}$ es no singular, entonces también $a_{nn}^{(n)} \neq 0$ y el sistema triangular superior resultante puede resolverse por el algoritmo ya visto:

$x_n = b^{(n)}/a_{nn}^{(n)}$
Para $i = n-1, \ldots, 1$
$$x_i = \frac{1}{a_{ii}^{(i)}} \left(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j \right)$$

Algoritmo del M.E.G.

- Recordemos que en el paso k-ésimo se parte de la siguiente matriz:

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

Para $k = 1, \ldots, n-1$
  para $i = k+1, \ldots, n$
    $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$
    para $j = k+1, \ldots, n$
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$
    $b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$
$x_n = b^{(n)}/a_{nn}^{(n)}$
Para $i = n-1, \ldots, 1$
$$x_i = \frac{1}{a_{ii}^{(i)}} \left(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j \right)$$

- Observación: El algoritmo no precisa crear los ceros debajo de la diagonal de la matriz, pues éstos luego no se utilizan.

El **costo operacional** del método de eliminación gaussiana es de aproximadamente $\frac{\mathbf{2}}{\mathbf{3}}\boldsymbol{n^3}$ **flop**. Notamos que la mayor parte del costo corresponde a la **triangularización de la matriz**.

Factorización LU.

Aunque la eliminación Gauss representa una forma satisfactoria para resolver sistemas lineales $\boldsymbol{Ax} = \boldsymbol{b}$, resulta ineficiente cuando deben resolverse ecuaciones con los mismos coeficientes $\boldsymbol{A}$, pero con diferentes constantes del lado derecho. El método de descomposición LU separa el tiempo usado en las eliminaciones para la matriz $\boldsymbol{A}$ de las manipulaciones en el lado derecho $\boldsymbol{b}$. Una vez que $\boldsymbol{A}$ se ha *descompuesto*, múltiples vectores del lado derecho se pueden evaluar de manera eficiente.

Para introducir esta factorizacón consideramos el siguiente ejemplo:

$$\boldsymbol{A} = \begin{pmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{pmatrix} \overset{\substack{m_{21}=-2 \\ m_{31}=4}}{\longrightarrow} \begin{pmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 6 & -3 \end{pmatrix} \overset{m_{32}=3}{\longrightarrow} \begin{pmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{pmatrix}$$

$$\boldsymbol{U} := \begin{pmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{pmatrix} \quad \boldsymbol{L} := \begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{pmatrix}$$

Donde m_{ij} está definido en el algoritmo del M.E.G. Notemos que

$$\boldsymbol{LU} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{pmatrix} = \begin{pmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{pmatrix} = \boldsymbol{A}$$

Si la matriz $\boldsymbol{A}$ es tal que la etapa de eliminación del M.E.G. se puede llevar a cabo (es decir si todos los pivotes $a_{ii}^{(i)} \neq 0$, $i = 1, \ldots, n-1$), entonces

$$\boldsymbol{A} = \boldsymbol{LU},$$

donde:

- $\boldsymbol{U}$ es la **matriz triangular superior que resulta de la eliminación** y
- $\boldsymbol{L}$ es la **matriz triangular inferior de los multiplicadores** m_{ij}:

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{nn-1} & 1 \end{pmatrix}$$

Solución de sistemas mediante factorización LU.

- Si $\boldsymbol{A} = \boldsymbol{LU}$, entonces

$$\boldsymbol{Ax} = \boldsymbol{b} \iff \boldsymbol{L}(\boldsymbol{Ux}) = \boldsymbol{b} \iff \begin{cases} \boldsymbol{Ly} = \boldsymbol{b}, \\ \boldsymbol{Ux} = \boldsymbol{y}. \end{cases}$$

  Por lo tanto, resolver un sistema $\boldsymbol{Ax} = \boldsymbol{b}$ es equivalente a:

  1. resolver $\boldsymbol{Ly} = \boldsymbol{b}$ y, luego,
  2. resolver $\boldsymbol{Ux} = \boldsymbol{y}$.

- Como estos sistemas son triangulares (inferior y superior, respectivamente), el costo operacional de resolver los dos sistemas es $2n^2$ flop.
- Factorizar la matriz $\boldsymbol{A} = \boldsymbol{LU}$ consiste simplemente en:
  - **triangularizar $\boldsymbol{A}$ por eliminación gaussiana** y
  - **almacenar la matriz triangular $\boldsymbol{L}$ de multiplicadores**.

  Por lo tanto, el costo de factorizar la matriz $\boldsymbol{A} = \boldsymbol{LU}$ es $\approx \frac{2}{3}n^3$ flop.
- Como $2n^2 \ll \frac{2}{3}n^3$, el costo operacional total para resolver un sistema mediante factorización LU es $\approx \frac{2}{3}n^3$ flop.

Muchas veces deben resolverse varios sistemas de ecuaciones con la misma matriz y distintos segundos miembros $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m$ (por ejemplo, para calcular $\boldsymbol{A}^{-1}$). En tal caso, conviene primero factorizar la matriz $\boldsymbol{A} = \boldsymbol{LU}$ (**una sola vez!**) y luego resolver los pares de sistemas triangulares para cada segundo miembro:

$$\begin{cases} \boldsymbol{Ly}^1 = \boldsymbol{b}^1, \\ \boldsymbol{Ux}^1 = \boldsymbol{y}^1, \end{cases} \quad \cdots \quad \begin{cases} \boldsymbol{Ly}^m = \boldsymbol{b}^m, \\ \boldsymbol{Ux}^m = \boldsymbol{y}^m. \end{cases}$$

Así, la parte más costosa del proceso (la factorización: $\frac{2}{3}n^3$) se hace una sola vez y sólo se repite la parte menos costosa (la solución de los sistemas triangulares: $2n^2$).

Necesidad del pivoteo.

El algoritmo de eliminación gaussiana (o el de factorización LU) sólo puede llevarse a cabo **todos los pivotes son no nulos:**

$$a_{kk}^{(k)} \neq 0.$$

Para $k = 1, \ldots, n-1$
  para $i = k+1, \ldots, n$
    $m_{ik} = a_{ik}^{(k)} / \boldsymbol{a_{kk}^{(k)}}$
    para $j = k+1, \ldots, n$
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

Ejemplo. El sistema de ecuaciones siguiente tiene matriz no singular pues su determinante es 1:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix}$$

Sin embargo el algoritmo anterior no puede aplicarse pues $\boldsymbol{a_{11} = 0}$ y, por lo tanto, $m_{21} = a_{21}^{(1)}/a_{11}^{(1)}$ y $m_{31} = a_{31}^{(1)}/a_{11}^{(1)}$ no están definidos.
Para poder resolver el sistema, debe **intercambiarse la primera ecuación con cualquiera de las otras de manera de evitar el pivote cero**. Por ejemplo, asi:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix} \quad \longrightarrow \quad \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 0 \end{pmatrix}$$

Por otra parte, puede demostrarse que la **estabilidad** del método de eliminación gaussiana en cuanto a **propagación de errores de redondeo** se deteriora si los multiplicadores m_{ij} son números muy grandes en módulo. Una forma de evitar ambos inconvenientes, pivotes nulos y multiplicadores grandes

en módulo, es realizar en cada paso el intercambio de ecuaciones que produzca el pivote mayor posible en módulo. Esto estrategia se denomina **pivoteo parcial**.

Estrategia de pivoteo parcial.

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & \boldsymbol{a_{kk}^{(k)}} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \boldsymbol{a_{nk}^{(k)}} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

En el paso k-ésimo se revisa el vector

$$\begin{pmatrix} a_{kk}^{(k)} \\ \vdots \\ a_{nk}^{(k)} \end{pmatrix}$$

y se busca la fila l en la que aparece la entrada mayor en módulo:

$$k \leq l \leq n : \ \left|a_{lk}^{(k)}\right| = \max_{k\leq i\leq n} \left\{\left|a_{ik}^{(k)}\right|\right\}.$$

- Luego, si $l \neq k$, se intercambia esa fila con la k-ésima.
- Si la matriz es no singular, siempre habrá una entrada no nula en ese vector, por lo que así se evitan los pivotes nulos.
- Además, despues del intercambio, $\left|a_{kk}^{(k)}\right| \geq \left|a_{ik}^{(k)}\right|$, $i = k, \ldots, n$. Por lo tanto, los multiplicadores no pueden pasar de 1 en módulo:

$$|m_{ik}| = \left|a_{ik}^{(k)}\right| / \left|a_{kk}^{(k)}\right| \leq 1, \qquad i = k, \ldots, n.$$

Matrices de permutación.

Si hay intercambios de filas, las matrices triangulares $\boldsymbol{L}$ y $\boldsymbol{U}$ que se obtienen por el **método de eliminación gaussiana con estrategia de pivoteo parcial**, ya no factorizan a $\boldsymbol{A}$, sino que factorizan a la matriz que se obtiene después de aplicar a $\boldsymbol{A}$ todos los intercambios de filas que tuvieron lugar.

Se llama **matriz de permutación** a toda matriz que se obtenga intercambiado filas de $\boldsymbol{I}$. Por ejemplo, las siguientes son todas las matrices de permutación 3×3:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Los intercambios de filas de una matriz se obtienen **multiplicando a izquierda** por una matriz de permutación. Por ejemplo:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & 2 & 3 \end{pmatrix}$$

Teorema. Si $\boldsymbol{A}$ es una matriz no singular, entonces existen matrices no singulares $\boldsymbol{L}$ triangular inferior y $\boldsymbol{U}$ triangular superior y una matriz de permutación $\boldsymbol{P}$, tales que

$$\boldsymbol{LU} = \boldsymbol{PA}.$$

- Estas matrices pueden obtenerse mediante el **método de eliminación gaussiana con estrategia de pivoteo parcial**.
- Si se debe resolver un sistema $\boldsymbol{Ax} = \boldsymbol{b}$, se procede así:

$$\boldsymbol{Ax} = \boldsymbol{b} \iff \boldsymbol{PAx} = \boldsymbol{Pb} \iff \boldsymbol{L}(\boldsymbol{Ux}) = \boldsymbol{Pb} \iff \begin{cases} \boldsymbol{Ly} = \boldsymbol{Pb}, \\ \boldsymbol{Ux} = \boldsymbol{y}. \end{cases}$$

- El método de eliminación gaussiana con estrategia de pivoteo parcial resulta **estable respecto a la propagación de errores de redondeo**.

Matrices definidas positivas.

Una matriz simétrica $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ se dice **definida positiva** si

$$\boldsymbol{x}^{\mathrm{t}}\boldsymbol{Ax} > 0 \qquad \forall \boldsymbol{x} \in \mathbb{R}^n : \ \boldsymbol{x} \neq 0.$$

Estas matrices también aparecen muy habitualmente, por ejemplo, al ajustar parámetros de un modelo por cuadrados mínimos o al resolver problemas de valores de contorno para ecuaciones diferenciales.

Teorema. Sea $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ una matriz **simétrica**. $\boldsymbol{A}$ es definida positiva si y sólo si se cumple cualquiera de las siguientes condiciones:

1. los valores propios de $\boldsymbol{A}$ son todos positivos;
2. los determinantes de las submatrices principales de $\boldsymbol{A}$ son todos positivos;
3. existe una matriz $\boldsymbol{L}$, triangular inferior y no singular, tal que $\boldsymbol{A} = \boldsymbol{LL}^{\mathrm{t}}$.

Esta última propiedad nos dice que si la matriz es simétrica y definida positiva, siempre puede obtenerse una factorización en matrices triangulares **sin necesidad de pivoteo**.

Además, **no hace falta calcular la matriz triangular superior**, pues es la transpuesta de la triangular inferior. Veremos que esto reduce el costo operacional a la mitad.

Método de Cholesky.

- Se aplica solamente a **matrices simétricas y definidas positivas**.
- Se basa en calcular directamente la matriz $\boldsymbol{L}$ tal que $\boldsymbol{A} = \boldsymbol{LL}^{\mathrm{t}}$.
- Se procede como en el caso de matrices tridiagonales y se obtiene el siguiente algoritmo:

Para $j = 1, \ldots, n$

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$$

para $i = j+1, \ldots, n$

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$$

El costo operacional es:

$$\sum_{j=1}^{n} \left[\sum_{k=1}^{j-1} 2 + \sum_{i=j+1}^{n} \left(1 + \sum_{k=1}^{j-1} 2 \right) \right]$$
$\approx \frac{1}{3}n^3$ flop,

$+ n$ raíces cuadradas.

Para resolver un sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ con **matriz simétrica y definida positiva** por el **método de Cholesky**, una vez calculada $\boldsymbol{L}$, se tiene:

$$\boldsymbol{Ax} = \boldsymbol{b} \iff \boldsymbol{L}(\boldsymbol{L}^{\mathrm{t}}\boldsymbol{x}) = \boldsymbol{b} \iff \begin{cases} \boldsymbol{Ly} = \boldsymbol{b}, \\ \boldsymbol{L}^{\mathrm{t}}\boldsymbol{x} = \boldsymbol{y}. \end{cases}$$

Por otro lado, para resolver los sistemas $\boldsymbol{Ly} = \boldsymbol{b}$ y $\boldsymbol{L}^{\mathrm{t}}\boldsymbol{x} = \boldsymbol{y}$, se utiliza el algoritmo que ya conocemos para matrices triangulares, cuyo costo operacional es de $\approx 2n^2$.

De lo anterior notamos que el costo operacional total del método de Cholesky es de $\approx \frac{1}{3}n^3$. Vale decir, **aproximadamente la mitad que el del M.E.G.**

Además, se demuestra que si la matriz es **simétrica y definida positiva**, los métodos de factorización son estables respecto a la propagación de errores de redondeo **sin necesidad de estrategia de pivoteo**. En particular se tiene que el método de Cholesky es estable respecto a la propagación de errores de redondeo.

DMH/VAD.