Sistemas de Ecuaciones Lineales II

- **Factorización LU:** Eliminación Gaussiana. Relación con la factorización LU.

Solución de sistemas con matriz triangular

- Dadas

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

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

- Dado que

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

  una matriz triangular es no singular si y sólo si sus términos diagonales son todos no nulos.

- La resolución de sistemas de ecuaciones lineales con matrices triangulares es muy sencilla y su costo operacional es bajo.

Solución de sistemas con matriz triangular (cont.)

- Consideremos un sistema $\boldsymbol{Lx} = \boldsymbol{b}$ con matriz triangular inferior $\boldsymbol{L}$.

  Procedemos por **sustitución progresiva**:

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

- 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:

$$\sum_{i=1}^{n} \left(1 + \sum_{j=1}^{i-1} 2 \right) = \sum_{i=1}^{n} (2i - 1) = n^2 \text{ flop}.$$

Solución de sistemas con matriz triangular (cont.)

- Ejercicio:

  1. Deducir el siguiente algoritmo para resolver un sistema $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$ con matriz triangular superior $\boldsymbol{U}$:

$$\begin{array}{|l} \text{Para } i = n, n-1, \ldots, 1 \\ \quad x_i = \dfrac{1}{u_{ii}} \left(b_i - \displaystyle\sum_{j=i+1}^{n} u_{ij} x_j \right) \end{array}$$

  2. Calcular su costo operacional.

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

$$\boldsymbol{U}\boldsymbol{x} = \tilde{\boldsymbol{b}},$$

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

Método de Eliminación Gaussiana (cont.)

- 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:

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

Método de Eliminación Gaussiana (cont.)

- 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 un número finito de veces 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 $a_{ii}^{(i)} \neq 0$, $i = 1, \ldots, k-1$.

Método de Eliminación Gaussiana (cont.)

- 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:

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

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

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 $j = k+1, \ldots, n$

      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

    $b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$

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

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

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

Costo Operacional del M.E.G.

- Costo del paso de eliminación:

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

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

    $m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$

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

      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

    $b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$

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

$$= \sum_{k=1}^{n-1} (n-k)\left[2(n-k)+3\right]$$

$$= \left(\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n \right) \text{ flop}$$

- Costo de la solución de sistema triangular superior: n^2 flop.
- Costo operacional total: $\left(\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n \right)$ flop.

Costo Operacional del M.E.G. (cont.)

n	Eliminación	Sist. Triang.	Total M.E.G.	$\frac{2}{3}n^3$	% Elim	% S. T.
10	705	100	805	666	87.58	12.42
20	5510	400	5910	5333	93.23	6.77
30	18415	900	19315	18000	95.34	4.66
40	43420	1600	45020	42666	96.45	3.55
50	84525	2500	87025	83333	97.13	2.87
100	671550	10000	681550	666666	98.53	1.47
200	5353100	40000	5393100	5333333	99.26	0.74
300	18044650	90000	18134650	18000000	99.50	0.50
400	42746200	160000	42906200	42666666	99.63	0.37
500	83457750	250000	83707750	83333333	99.70	0.30
600	144179300	360000	144539300	144000000	99.75	0.25
700	228910850	490000	229400850	228666666	99.79	0.21
800	341652400	640000	342292400	341333333	99.81	0.19
900	486403950	810000	487213950	486000000	99.83	0.17
1000	667165500	1000000	668165500	666666666	99.85	0.15

Costo Operacional del M.E.G. (cont.)

- Para n grande, los términos proporcionales a n^2 (y a n) resultan despreciables respecto a los proporcionales a n^3.

  Por esa razón, **el costo operacional del método de eliminación gaussiana** se dice que es de aproximadamente

$$\boxed{\frac{2}{3}n^3 \text{ flop}}$$

- Notemos que la mayor parte del costo corresponde a la **triangularización de la matriz**:

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

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

    $m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$

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

      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

Costo operacional: $\frac{2}{3}n^3$ flop.

Factorización LU

- **Ejemplo:**

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

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

Notemos que

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

Factorización LU (cont.)

- 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

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U},$$

  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 A por eliminación gaussiana** y
  - **almacenar la matriz triangular 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.

Solución de sistemas mediante factorizac. LU (cont.)

- 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:

$$\begin{cases} \boldsymbol{L}\boldsymbol{y}^1 = \boldsymbol{b}^1, \\ \boldsymbol{U}\boldsymbol{x}^1 = \boldsymbol{y}^1, \end{cases} \quad \cdots \quad \begin{cases} \boldsymbol{L}\boldsymbol{y}^m = \boldsymbol{b}^m, \\ \boldsymbol{U}\boldsymbol{x}^m = \boldsymbol{y}^m. \end{cases}$$

- 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: $2n^2$).

- Hay algoritmos (**Crout**, **Doolitle**) que permiten obtener directamente la factorización LU de una matriz, pero el costo es el mismo que el de hacerlo por eliminación gaussiana.

Solución de sistemas mediante factorizac. LU (cont.)

- **Algoritmo (por eliminación gaussiana):**

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

| para $i = k+1, \ldots, n$

| | $m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$

| | para $j = k+1, \ldots, n$

| | | $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

Para $i = 1, \ldots, n$

| $y_i = b_i - \sum_{j=1}^{i-1} m_{ij} y_j$

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

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

- **Observación:** La matriz triangular superior puede calcularse **utilizando las mismas posiciones de memoria** en las que inicialmente está almacenada $\boldsymbol{A}$.
- A fin de no tener que utilizar una matriz más para guardar $\boldsymbol{L}$, pueden almacenarse **cada multiplicador m_{ij} en lugar de la entrada a_{ij}** que se hace cero en ese paso.
- Ésta no es la forma en que procede MATLAB, pues este software no pretende optimizar el costo de almacenamiento.
- Sin embargo, hay mucho software que utiliza este truco a fin de evitar tener que usar memoria adicional para almacenar los multiplicadores.