# Sistemas de Ecuaciones Lineales III

- **Pivoteo:** Estrategia de pivoteo parcial.
- **Adaptación a matrices con estructuras particulares:** Matrices banda y tridiagonales. Método de Cholesky.

# Necesidad del pivoteo

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

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

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

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

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

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

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

- **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 $a_{11} = 0$ y, por lo tanto, $m_{21} = a_{21}^{(1)} / a_{11}^{(1)}$ y $m_{31} = a_{31}^{(1)} / a_{11}^{(1)}$ no están definidos.

# Necesidad del pivoteo (cont.)

- 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:

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

- 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

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

- 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:

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

- 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, $\left| a_{kk}^{(k)} \right| \ge \left| a_{ik}^{(k)} \right|$, $i = k, \ldots, n$. Por lo tanto, los multiplicadores no pueden pasar de $1$ en módulo:

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

# 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 $3 \times 3$:

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

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

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

# Factorización LU con estrategia de pivoteo parcial

- **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**.

# Factorización LU con pivoteo en MATLAB

- **Comando:** `[L,U]=lu(A)`
- L es una matriz **psicológicamente triangular inferior**
- U es una matriz triangular superior
- L*U = A.

```
>> A=[1 2 3;4 5 6;7 8 0];
>> [L,U]=lu(A)

L = 0.1429    1.0000         0
    0.5714    0.5000    1.0000
    1.0000         0         0

U = 7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000
>> L*U
ans = 1     2     3
      4     5     6
      7     8     0
```

# Factorización LU con pivoteo en MATLAB (cont.)

- **Comando:**
  `[L,U,P]=lu(A)`
- L es una matriz triangular inferior
- U es una matriz triangular superior
- P es una matriz de permutación
- L*U = P*A.

```
>> A=[1 2 3;4 5 6;7 8 0];
>> [L,U,P]=lu(A)

L = 1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000

U = 7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000

P =  0     0     1
     1     0     0
     0     1     0
```

# Factorización LU con pivoteo en MATLAB (cont.)

- **Comando:**
  $\boxed{\texttt{[L,U,P]=lu(A)}}$
- L es una matriz triangular inferior
- U es una matriz triangular superior
- P es una matriz de permutación
- L*U = P*A.

```
>> L*U

ans = 7     8     0
      1     2     3
      4     5     6

>> P*A

ans = 7     8     0
      1     2     3
      4     5     6
```

# Matrices banda

- Se dice que $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ es una **matriz banda** si $a_{ij} = 0$ cuando $|i-j| \geq \ell$, con $\ell \ll n$. Al número $\ell$ se lo llama el **ancho de banda** de la matriz.

$$\begin{array}{cc}
\begin{pmatrix}
\times & \cdots & \times & 0 & \cdots & 0 \\
\vdots & \ddots & & \ddots & \ddots & \vdots \\
\times & & \ddots & & \ddots & 0 \\
0 & \ddots & & \ddots & & \times \\
\vdots & \ddots & \ddots & & \ddots & \vdots \\
0 & \cdots & 0 & \times & \cdots & \times
\end{pmatrix} & \begin{array}{c} \\ \\ \\ \uparrow \\ \ell \\ \downarrow \end{array} \\
\qquad\qquad\qquad \longleftarrow \quad \ell \quad \longrightarrow &
\end{array}$$

- Las matrices banda son un caso especial de matrices dispersas y, como tales, deben almacenarse como **sparse** en MATLAB a fin de reducir el costo de almacenamiento.

- Estas matrices aparecen muy habitualmente en las aplicaciones; especialmente en la resolución de problemas de valores de contorno para ecuaciones diferenciales.

# Factorización LU de matrices banda

- Si una matriz banda $\boldsymbol{A}$ puede factorizarse LU sin necesidad de pivoteo, entonces las matrices triangulares $\boldsymbol{L}$ y $\boldsymbol{U}$ también son banda con el mismo ancho de banda que $\boldsymbol{A}$.

$$
\boldsymbol{A} = \underset{\boldsymbol{L} \qquad \longleftarrow \quad \ell \quad \longrightarrow}{\begin{pmatrix}
\times & 0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \ddots & \ddots & & & \vdots \\
\times & & \ddots & \ddots & & \vdots \\
0 & \ddots & & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & & \ddots & 0 \\
0 & \cdots & 0 & \times & \cdots & \times
\end{pmatrix}}
\underset{\boldsymbol{U}}{\begin{pmatrix}
\times & \cdots & \times & 0 & \cdots & 0 \\
0 & \ddots & & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & & \ddots & 0 \\
\vdots & & \ddots & \ddots & & \times \\
\vdots & & & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & 0 & \times
\end{pmatrix}}
\begin{array}{c} \\ \\ \\ \uparrow \\ \ell \\ \downarrow \end{array}
$$

- Por eso se dice que la factorización LU sin pivoteo **preserva la estructura banda de las matrices**.

## Matrices tridiagonales

- Un caso extremo de matrices banda es el de las **matrices tridiagonales** ($\ell = 2$):

$$\boldsymbol{A} = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix} \in \mathbb{R}^{n\times n}$$

- Estas matrices aparecen también muy habitualmente, por ejemplo, al interpolar por splines o al resolver problemas de valores de contorno para ecuaciones diferenciales ordinarias.

# Factorización LU de matrices tridiagonales

- Cuando se cumplen las siguientes desigualdades,

$$
\begin{aligned}
&|b_1| > |c_1|, \\
&|b_i| \geq |a_i| + |c_i|, \qquad i = 2, \ldots, n-1, \\
&|b_n| > |a_n|,
\end{aligned}
$$

las matrices tridiagonales pueden factorizarse LU **sin necesidad de pivoteo** y este procedimiento resulta **estable respecto a la propagación de errores de redondeo**:

$$
\boldsymbol{A} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha_2 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n & 1 \end{pmatrix}}_{\boldsymbol{L}} \underbrace{\begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \gamma_{n-1} \\ 0 & \cdots & \cdots & 0 & \beta_n \end{pmatrix}}_{\boldsymbol{U}}
$$

- Las entradas $\alpha_i$, $\beta_i$ y $\gamma_i$ de las matrices $\boldsymbol{L}$ y $\boldsymbol{U}$ pueden calcularse muy fácilmente:

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha_2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n & 1 \end{pmatrix} \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & \gamma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \beta_{n-1} & \gamma_{n-1} \\ 0 & \cdots & \cdots & 0 & \beta_n \end{pmatrix}$$

$$\begin{aligned} 1\beta_1 = b_1 \quad &\Longrightarrow \quad \beta_1 = b_1 \\ 1\gamma_1 = c_1 \quad &\Longrightarrow \quad \gamma_1 = c_1 \\ \alpha_2\beta_1 = a_2 \quad &\Longrightarrow \quad \alpha_2 = a_2/\beta_1 \\ \alpha_2\gamma_1 + 1\beta_2 = b_2 \quad &\Longrightarrow \quad \beta_2 = b_2 - \alpha_2\gamma_1 \\ 1\gamma_2 = c_2 \quad &\Longrightarrow \quad \gamma_2 = c_2 \\ \cdots \quad & \quad\quad \cdots \\ \alpha_n\beta_{n-1} = a_n \quad &\Longrightarrow \quad \alpha_n = a_n/\beta_{n-1} \\ \alpha_n\gamma_{n-1} + 1\beta_n = b_n \quad &\Longrightarrow \quad \beta_n = b_n - \alpha_n\gamma_{n-1} \end{aligned}$$

## Factorización LU de matrices tridiagonales (cont.)

- Así obtenemos el siguiente algoritmo (**Algoritmo de Thomas**):

$$\left.\begin{array}{rcl}
1\beta_1 = b_1 & \Longrightarrow & \beta_1 = b_1 \\
1\gamma_1 = c_1 & \Longrightarrow & \gamma_1 = c_1 \\
\alpha_2\beta_1 = a_2 & \Longrightarrow & \alpha_2 = a_2/\beta_1 \\
\alpha_2\gamma_1 + 1\beta_2 = b_2 & \Longrightarrow & \beta_2 = b_2 - \alpha_2\gamma_1 \\
1\gamma_2 = c_2 & \Longrightarrow & \gamma_2 = c_2 \\
\cdots & & \cdots \\
\alpha_n\beta_{n-1} = a_n & \Longrightarrow & \alpha_n = a_n/\beta_{n-1} \\
\alpha_n\gamma_{n-1} + 1\beta_n = b_n & \Longrightarrow & \beta_n = b_n - \alpha_n\gamma_{n-1}
\end{array}\right\} \longrightarrow$$

$$\boxed{\begin{array}{l}
\beta_1 = b_1 \\
\text{Para } i = 2, \ldots, n \\
\quad \left| \begin{array}{l}
\gamma_{i-1} = c_{i-1} \\
\alpha_i = a_i/\beta_{i-1} \\
\beta_i = b_i - \alpha_i\gamma_{i-1}
\end{array}\right.
\end{array}}$$

Costo operacional:

$$\sum_{i=2}^{n} 3 = 3(n-1) \text{ flop}$$

## Solución de sistemas con matrices tridiagonales

- Al resolver un sistema $\boldsymbol{Ax} = \boldsymbol{d}$ con matriz $\boldsymbol{A}$ tridiagonal, a partir de su factorización LU,

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

los sistemas triangulares también pueden resolverse muy fácilmente:

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha_2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix} \rightarrow$$

$y_1 = d_1$

Para $i = 2, \ldots, n$

$\quad y_i = d_i - \alpha_i y_{i-1}$

Costo operacional:

$$\sum_{i=2}^{n} 2 = 2(n-1) \text{ flop}$$

# Solución de sistemas con matrices tridiagonales (cont.)

$$\begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \beta_{n-1} & \gamma_{n-1} \\ 0 & \cdots & \cdots & 0 & \beta_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} \rightarrow$$

$x_n = y_n / \beta_n$

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

$\qquad x_i = \left(y_i - \gamma_i x_{i+1}\right) / \beta_i$

Costo operacional:

$$1 + \sum_{i=1}^{n-1} 3 = 1 + 3(n-1)\ \mathrm{flop}$$

- El costo total de resolver un sistema de ecuaciomes con matriz tridiagonal mediante el algortimo de Thomas es de $3(n-1) + 2(n-1) + 1 + 3n - 2 = 8n - 7\ \mathrm{flop}$.
- Compárese este costo con el del método de eliminación gaussiana aplicado a ciegas sin sacar provecho de la estructura tridiagonal de la matriz: $\frac{2}{3}n^3$.

  Por ejemplo, un sistema $1000 \times 1000$ cuesta aproximadamente $666\,666\,666\ \mathrm{flop}$ por M.E.G. y aproximadamente $8\,000\ \mathrm{flop}$ mediante este algoritmo.

# Matrices definidas positivas

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

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

- 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 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{t}}$.

- 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 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{t}}$.
- 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:

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

$$\approx \frac{1}{3} n^3 \text{ flop},$$

$+\, n$ raíces cuadradas.

# Método de Cholesky (cont.)

- 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:

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

- 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 $\approx 2n^2$.

- Por lo tanto 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, el método de Cholesky es **estable respecto a la propagación de errores de redondeo**.

# Factorización de Cholesky en MATLAB

- **Comando:** $\boxed{\texttt{R=chol(A)}}$
- R es una matriz triangular superior
- $\mathrm{R}^t$*R = A.

```
A =
      2     -1      0
     -1      2     -1
      0     -1      2
>> R=chol(A)
R =
    1.4142   -0.7071         0
         0    1.2247   -0.8165
         0         0    1.1547
>> B=R'*R
B =

    2.0000   -1.0000         0
   -1.0000    2.0000   -1.0000
         0   -1.0000    2.0000
```