# MAT-468: Sesión 2, Solución de sistemas lineales

**Felipe Osorio**

http://fosorios.mat.utfsm.cl

Departamento de Matemática, UTFSM

# Solución de sistemas lineales

El problema de resolver el sistema lineal

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

es central en cálculo científico. Es bien sabido que:

- Si existe solución el sistema, entonces se dice consistente.
- Una solución $\boldsymbol{x}$ puede ser escrita como $\boldsymbol{A}^{-}\boldsymbol{b}$, para $\boldsymbol{A}^{-}$ alguna inversa de $\boldsymbol{A}$.
- Si $\boldsymbol{A}$ es cuadrada entonces la solución es dada por $\boldsymbol{A}^{-1}\boldsymbol{b}$.

Nuestro objetivo es:

- Describir algunos métodos para calcular una solución.
- Nunca se obtendrá $\boldsymbol{A}^{-1}$.

# Solución de sistemas lineales

El problema de resolver el sistema lineal

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

es central en cálculo científico. Es bien sabido que:

- Si existe solución el sistema, entonces se dice consistente.
- Una solución $\boldsymbol{x}$ puede ser escrita como $\boldsymbol{A}^{-}\boldsymbol{b}$, para $\boldsymbol{A}^{-}$ alguna inversa de $\boldsymbol{A}$.
- Si $\boldsymbol{A}$ es cuadrada entonces la solución es dada por $\boldsymbol{A}^{-1}\boldsymbol{b}$.

Nuestro objetivo es:

- Describir algunos métodos para calcular una solución.
- Nunca se obtendrá $\boldsymbol{A}^{-1}$.

# Sistemas triangulares

El sistema más simple para resolver es del tipo

$$\boldsymbol{T}\boldsymbol{x} = \boldsymbol{b},$$

donde $\boldsymbol{T}$ es matriz triangular $n \times n$, $\boldsymbol{x} \in \mathbb{R}^n$ y $\boldsymbol{b} \in \mathbb{R}^n$.

Suponga que $\boldsymbol{T}$ es matriz triangular superior. En este caso debemos resolver el sistema:

$$\begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & & t_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

utilizando sustitución hacia atrás (Asumiremos que $t_{ii} \neq 0$, $i = 1, \dots, n$).

# Sustitución hacia atrás

Note que en el sistema de ecuaciones anterior podemos resolver la última ecuación de manera trivial:

$$x_n = b_n / t_{nn},$$

Reemplazando esta ecuación en la anterior, tenemos

$$x_{n-1} = \frac{1}{t_{n-1,n-1}} \left( b_{n-1} - t_{n-1,n} \, x_n \right)$$

Procediendo de ese modo, obtenemos la solución,

$$x_i = \frac{1}{t_{ii}} \Big( b_i - \sum_{j=i+1}^{n} t_{ij} x_j \Big),$$

para $i = 1, \dots, n$.

Observación: Note que, es posible sobreescribir la solución $\boldsymbol{x}$ en el vector $\boldsymbol{b}$.

## Sistemas triangulares

- El "sistema triangular" más simple surge cuando $\boldsymbol{T}$ es diagonal
- Cuando $\boldsymbol{T}$ es triangular inferior, el sistema es resuelto por sustitución adelante.
- El que $\boldsymbol{T}$ sea triangular unitaria $(t_{ii} = 1)$ no añade dificultad al problema.
- La inversa de una matriz triangular inferior (superior) también es una matriz triangular inferior (superior).
- Lo anterior permite obtener la inversa de una matriz triangular "in-place".

### Resultado 1 (Factorización LU)

Sea $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ matriz cuadrada tal que todos sus cofactores principales son no nulos, es decir

$$a_{11} \neq 0, \quad \det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \neq 0, \quad \det\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \neq 0, \quad \dots \quad \det(\boldsymbol{A}) \neq 0.$$

Entonces, existe una única matriz triangular inferior unitaria $\boldsymbol{L}$ y una matriz triangular superior $\boldsymbol{U}$, tal que $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$, y

$$\det(\boldsymbol{A}) = u_{11}\, u_{22} \cdots u_{nn}.$$

# Factorización LU

**Algoritmo 1:** Factorización LU

**Entrada:** Matriz $\boldsymbol{A} \in \mathbb{R}^{n \times n}$.
**Salida :** Factores $\boldsymbol{L}$ y $\boldsymbol{U}$, matrices triangulares inferior y superior, respectivamente.

**begin**
    Hacer $\boldsymbol{L} = \boldsymbol{I}_n$ y $\boldsymbol{U} = \boldsymbol{0}$.
    **for** $i = 1$ **to** $n$ **do**
        **for** $j = 1$ **to** $i$ **do**
            $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$
        **end**
        **for** $j = i + 1$ **to** $n$ **do**
            $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}\right) / u_{jj}$
        **end**
    **end**
**end**

# Factorización LU

La principal utilidad de la factorización LU es resolver sistemas lineales del tipo

$$\boldsymbol{Ax} = \boldsymbol{LUx} = \boldsymbol{b},$$

mediante resolver los sistemas triangulares

$$\boldsymbol{Lz} = \boldsymbol{b}, \qquad \boldsymbol{Ux} = \boldsymbol{z},$$

que deben ser desarrollados via sustitución adelante y hacia atrás, respectivamente.

Observaciones:

- Hallar la factorización LU es equivalente a resolver un sistema lineal mediante eliminación gaussiana.
- Existen generalizaciones de LU para matrices rectangulares y singulares.

EX UMBRA IN SOLEM

# Factorización LU

La principal utilidad de la factorización LU es resolver sistemas lineales del tipo

$$\boldsymbol{Ax} = \boldsymbol{LUx} = \boldsymbol{b},$$

mediante resolver los sistemas triangulares

$$\boldsymbol{Lz} = \boldsymbol{b}, \qquad \boldsymbol{Ux} = \boldsymbol{z},$$

que deben ser desarrollados via sustitución adelante y hacia atrás, respectivamente.

Observaciones:

- Hallar la factorización LU es equivalente a resolver un sistema lineal mediante eliminación gaussiana.
- Existen generalizaciones de LU para matrices rectangulares y singulares.

### Resultado 2 (Factorización Cholesky)

Sea $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ es matriz simétrica y definida positiva, entonces existe una única matriz triangular superior $\boldsymbol{G} \in \mathbb{R}^{n\times n}$ con elementos diagonales positivos tal que

$$\boldsymbol{A} = \boldsymbol{G}^\top \boldsymbol{G}$$

*Observación*

Note que si usamos la factorización Cholesky para resolver el sistema $\boldsymbol{Ax} = \boldsymbol{b}$. Entonces debemos resolver los sistemas triangulares

$$\boldsymbol{G}^\top \boldsymbol{y} = \boldsymbol{b}, \qquad \text{y} \qquad \boldsymbol{Gx} = \boldsymbol{y}.$$

En efecto,

$$\boldsymbol{Ax} = (\boldsymbol{G}^\top \boldsymbol{G})\boldsymbol{x} = \boldsymbol{G}^\top(\boldsymbol{Gx}) = \boldsymbol{G}^\top \boldsymbol{y} = \boldsymbol{b}.$$

**Resultado 2 (Factorización Cholesky)**

Sea $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ es matriz simétrica y definida positiva, entonces existe una única matriz triangular superior $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ con elementos diagonales positivos tal que

$$\boldsymbol{A} = \boldsymbol{G}^\top \boldsymbol{G}$$

*Observación*

Note que si usamos la factorización Cholesky para resolver el sistema $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Entonces debemos resolver los sistemas triangulares

$$\boldsymbol{G}^\top \boldsymbol{y} = \boldsymbol{b}, \qquad \text{y} \qquad \boldsymbol{G}\boldsymbol{x} = \boldsymbol{y}.$$

En efecto,

$$\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{G}^\top \boldsymbol{G})\boldsymbol{x} = \boldsymbol{G}^\top (\boldsymbol{G}\boldsymbol{x}) = \boldsymbol{G}^\top \boldsymbol{y} = \boldsymbol{b}.$$

# Factorización Cholesky

**Algoritmo 2:** Factorización Cholesky

```
Entrada: Matriz A ∈ ℝ^{n×n}.
Salida : Factor Cholesky T ∈ ℝ^{n×n}.
begin
    t_11 = sqrt(a_11).
    for j = 2 to n do
        t_1j = a_1j / t_11.
    end
    for i = 2 to n do
        t_ii = sqrt(a_ii − Σ_{k=1}^{i−1} t_ki^2),
        for j = i + 1 to n do
            t_ij = (a_ij − Σ_{k=1}^{i−1} t_ki t_kj) / t_ii
        end
    end
end
```

Where:

$$t_{11} = \sqrt{a_{11}}, \quad t_{1j} = a_{1j}/t_{11}, \quad t_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} t_{ki}^2}, \quad t_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} t_{ki}t_{kj}\right)/t_{ii}$$

# Operador Sweep

El operador Sweep es un método alternativo para invertir matrices y resolver sistemas lineales.

Suponga que $\boldsymbol{A} = (a_{ij})$ es matriz cuadrada $n \times n$. Aplicando el operador Sweep sobre el $k$-ésimo elemento diagonal de $\boldsymbol{A}$ ($a_{kk} \neq 0$) permite obtener la matriz $\boldsymbol{B}$, definida como:

$$\begin{aligned}
b_{kk} &= \frac{1}{a_{kk}}, \\
b_{ik} &= -\frac{a_{ik}}{a_{kk}}, \qquad i \neq k, \\
b_{kj} &= \frac{a_{kj}}{a_{kk}}, \qquad j \neq k, \\
b_{ij} &= a_{ij} - \frac{a_{ik}a_{kj}}{a_{kk}}, \qquad i, j \neq k,
\end{aligned}$$

y escribimos $\boldsymbol{B} = \mathrm{Sweep}(k)\boldsymbol{A}$.

# Operador Sweep

**Propiedades:**

- $\operatorname{Sweep}(k)\operatorname{Sweep}(k)\boldsymbol{A} = \boldsymbol{A}$.
- $\operatorname{Sweep}(k)\operatorname{Sweep}(r)\boldsymbol{A} = \operatorname{Sweep}(r)\operatorname{Sweep}(k)\boldsymbol{A}$.
- $\boldsymbol{A}^{-1} = \prod_{i=1}^{n} \operatorname{Sweep}(i)\boldsymbol{A}$.

Considere $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ matriz particionada como:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix},$$

donde $\boldsymbol{A}_{11} \in \mathbb{R}^{r \times r}$ $(r < n)$. Suponga que se aplica el operador Sweep sobre los elementos diagonales de $\boldsymbol{A}_{11}$. De este modo,

$$\boldsymbol{B} = \prod_{i=1}^{r} \mathrm{Sweep}(i)\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{pmatrix},$$

con

$$\begin{aligned} \boldsymbol{B}_{11} &= \boldsymbol{A}_{11}^{-1}, & \boldsymbol{B}_{12} &= \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}, \\ \boldsymbol{B}_{21} &= -\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}, & \boldsymbol{B}_{22} &= \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}. \end{aligned}$$

## Comentarios sobre el operador Sweep

- Si $\boldsymbol{A}$ es matriz simétrica, el operador Sweep preserva la simetría de $\boldsymbol{A}$.
- Existen varias definiciones ligeramente diferentes del operador Sweep.
- Problemas de inestabilidad pueden ocurrir cuando algún $a_{kk}$ es cercano a cero.

# Refinamiento iterativo

Ideas:

- Es sabido que los métodos directos son bastante sensibles a errores de redondeo.
- El refinamiento iterativo permite mejorar la solución obtenida por un método directo.

Suponga una solución aproximada $\boldsymbol{x}^{(0)}$ del sistema lineal $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. El refinamiento iterativo es un proceso que construye la secuencia $\{\boldsymbol{x}^{(k)}\}$, basada en el vector de residuos:

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}, \qquad r = 0, 1, \dots,$$

De este modo, resolviendo el sistema lineal $\boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{r}$, obtenemos que $\boldsymbol{x} + \boldsymbol{\delta}$ corresponde a una solución del sistema original.

El procedimiento es útil si somos capaces de calcular el vector de residuos con una muy alta precisión. Lo anterior motiva el siguiente algoritmo.

Ideas:

- Es sabido que los métodos directos son bastante sensibles a errores de redondeo.
- El refinamiento iterativo permite mejorar la solución obtenida por un método directo.

Suponga una solución aproximada $\boldsymbol{x}^{(0)}$ del sistema lineal $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. El refinamiento iterativo es un proceso que construye la secuencia $\{\boldsymbol{x}^{(k)}\}$, basada en el vector de residuos:

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}, \qquad r = 0, 1, \dots,$$

De este modo, resolviendo el sistema lineal $\boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{r}$, obtenemos que $\boldsymbol{x} + \boldsymbol{\delta}$ corresponde a una solución del sistema original.

El procedimiento es útil si somos capaces de calcular el vector de residuos con una muy alta precisión. Lo anterior motiva el siguiente algoritmo.

# Refinamiento iterativo

Ideas:

- Es sabido que los métodos directos son bastante sensibles a errores de redondeo.
- El refinamiento iterativo permite mejorar la solución obtenida por un método directo.

Suponga una solución aproximada $\boldsymbol{x}^{(0)}$ del sistema lineal $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. El refinamiento iterativo es un proceso que construye la secuencia $\{\boldsymbol{x}^{(k)}\}$, basada en el vector de residuos:

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}, \qquad r = 0, 1, \dots,$$

De este modo, resolviendo el sistema lineal $\boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{r}$, obtenemos que $\boldsymbol{x} + \boldsymbol{\delta}$ corresponde a una solución del sistema original.

El procedimiento es útil si somos capaces de calcular el vector de residuos con una muy alta precisión. Lo anterior motiva el siguiente algoritmo.

EX UMBRA IN SOLEM

# Refinamiento iterativo

**Algoritmo 3:** Refinamiento iterativo.

**Parámetros:** Tolerancia $\tau$, y número máximo de iteraciones $k_{\max}$.

```
1  begin
2      k ← 0
3      Calcular r^(k) = b − A x^(k) en alta precisión
4      Resolver el sistema A δ = r^(k) para obtener δ^(k)
5      Actualizar x^(k+1) = x^(k) + δ^(k)
6      if ‖δ^(k)‖_∞ ≤ τ ‖x^(k+1)‖_∞ then
7          return x = x^(k+1), y detener el algoritmo.
8      else if k < k_max then
9          Hacer k ← k + 1 y x^(k) ← x^(k+1).
10         Volver a Paso 3.
11     else
12         Indicar: El algoritmo no converge después de k_max iteraciones.
13     end
14 end
```

**Algoritmo 4:** Refinamiento iterativo.

**Parámetros:** Tolerancia $\tau$, y número máximo de iteraciones $k_{\max}$.

```
1  begin
2      k ← 0
3      Calcular r^(k) = b − A x^(k) en alta precisión
4      Resolver el sistema A δ = r^(k) para obtener δ^(k)
5      Actualizar x^(k+1) = x^(k) + δ^(k)
6      if ‖δ^(k)‖_∞ ≤ τ‖x^(k+1)‖_∞ then
7          return x = x^(k+1), y detener el algoritmo.
8      else if k < k_max then
9          Hacer k ← k + 1 y x^(k) ← x^(k+1).
10         Volver a Paso 3.
11     else
12         Indicar: El algoritmo no converge después de k_max iteraciones.
13     end
14 end
```

# Refinamiento iterativo

**Algoritmo 5:** Refinamiento iterativo.

**Parámetros:** Tolerancia $\tau$, y número máximo de iteraciones $k_{\max}$.

1 **begin**
2     $k \leftarrow 0$
3     Calcular $\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}$ en alta precisión
4     Resolver el sistema $\boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{r}^{(k)}$ para obtener $\boldsymbol{\delta}^{(k)}$
5     Actualizar $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{\delta}^{(k)}$
6     **if** $\|\boldsymbol{\delta}^{(k)}\|_\infty \leq \tau \|\boldsymbol{x}^{(k+1)}\|_\infty$ **then**
7         **return** $\boldsymbol{x} = \boldsymbol{x}^{(k+1)}$, y detener el algoritmo.
8     **else if** $k < k_{\max}$ **then**
9         Hacer $k \leftarrow k + 1$ y $\boldsymbol{x}^{(k)} \leftarrow \boldsymbol{x}^{(k+1)}$.
10         Volver a Paso **3**.
11     **else**
12         Indicar: El algoritmo no converge después de $k_{\max}$ iteraciones.
13     **end**
14 **end**

## Comentarios sobre el refinamiento iterativo

- Note que el algoritmo puede no alcanzar convergencia.
- El uso de refinamiento iterativo como un método general es limitado por la alta precisión en el paso 1.
- La factorización de $\boldsymbol{A}$ es disponible luego de calcular $\boldsymbol{x}^{(0)}$.
- El costo computacional del refinamiento iterativo suele ser pequeño.

# Tarea

1. Escriba una rutina para resolver el sistema de ecuaciones lineales $\boldsymbol{Ax} = \boldsymbol{b}$ con:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 1 & -1 & -1 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 4 \\ 9 \\ -1 \end{pmatrix}$$

   usando la descomposición LU y refinamiento iterativo.

2. Implementar el operador Sweep usando su lenguaje de programación favorito.

3. Pruebe su rutina para obtener la inversa de una matriz usando el operador Sweep con la siguiente matriz:

$$\boldsymbol{B} = \begin{pmatrix} 30 & 16 & 46 \\ 16 & 10 & 26 \\ 46 & 26 & 72 \end{pmatrix}.$$

4. Verifique que $\boldsymbol{B}$ es matriz semidefinida positiva usando la factorización Cholesky. ¿En cuál etapa del algoritmo el procedimiento falla?