Solución de sistemas de ecuaciones lineales

Condicionamiento. Métodos directos: eliimnación gaussiana, factorización LU. Pivoteo.

Manuel Carlevaro

Departamento de Ingeniería Mecánica

Grupo de Materiales Granulares - UTN FRLP

manuel.carlevaro@gmail.com

Cálculo Avanzado • 2023

Resolver:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

Resolver:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

O, en forma matricial:

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Resolver:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

O, en forma matricial:

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Matriz de coeficientes aumentada:

$$
\left[\begin{array}{cccc|c}
a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\
a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{N1} & a_{N2} & \ldots & a_{nn} & b_n
\end{array}\right]
$$

Resolver:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

O, en forma matricial:

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Matriz de coeficientes aumentada:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{nn} & b_n \end{array}\right]$$

Si $\mathbb{A} \in \mathbb{R}^{n\times n}$ y $\boldsymbol{b} \in \mathbb{R}$, la existencia y unicidad de la solución está asegurada si una de las siguientes condiciones se cumple:

- $\mathbb{A}$ es invertible (no singular)
- $\mathrm{rg}(A) = n$
- El sistema homogéneo $\mathbb{A}\boldsymbol{x} = \mathbf{0}$ admite solo la solución nula.

Resolver:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

O, en forma matricial:

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Matriz de coeficientes aumentada:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{nn} & b_n \end{array}\right]$$

Si $\mathbb{A} \in \mathbb{R}^{n\times n}$ y $\boldsymbol{b} \in \mathbb{R}$, la existencia y unicidad de la solución está asegurada si una de las siguientes condiciones se cumple:

- $\mathbb{A}$ es invertible (no singular)
- $\mathrm{rg}(A) = n$
- El sistema homogéneo $\mathbb{A}\boldsymbol{x} = \mathbf{0}$ admite solo la solución nula.

Solución: regla de Cramer

$$x_j = \frac{\Delta_j}{\det \mathbb{A}}$$

Esfuerzo computacional: $\mathcal{O}((n+1)!)$.
$n = 50$, Intel i7: 200 Gflops $\approx 5 \times 10^{45}$ años.

Métodos:

- Directos: alcanzan la solución en un número finito de pasos. $\mathcal{O}(2/3N^3)$.
  - Eliminación gaussiana
  - Factorización LU
  - LDM^T
  - Factorización de Cholesky
  - Factorización QR
- Iterativos: más eficientes en casos particulares. $\mathcal{O}(N^2)$.
  - Jacobi
  - Gauss-Seidel
  - Subespacios de Krylov
  - GMRES

Métodos:

- Directos: alcanzan la solución en un número finito de pasos. $\mathcal{O}(2/3N^3)$.
  - Eliminación gaussiana
  - Factorización LU
  - LDM^T
  - Factorización de Cholesky
  - Factorización QR
- Iterativos: más eficientes en casos particulares. $\mathcal{O}(N^2)$.
  - Jacobi
  - Gauss-Seidel
  - Subespacios de Krylov
  - GMRES

Estabilidad de la solución:

$$(\mathbb{A} + \delta\mathbb{A})(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{b} + \delta\boldsymbol{b}$$

Unicidad de la solución: $\mathbb{A}$ es no singular: $|\mathbb{A}| \neq 0$.
Número de condición: $\operatorname{cond}(\mathbb{A}) = \|\mathbb{A}\|\|\mathbb{A}^{-1}\|$.

Se puede demostrar[a], que si

$$\operatorname{cond}(\mathbb{A})\frac{\|\delta\mathbb{A}\|}{\|\mathbb{A}\|} < 1$$

se cumple:

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \frac{\operatorname{cond}(\mathbb{A})}{1 - \operatorname{cond}(\mathbb{A})\frac{\|\delta\mathbb{A}\|}{\|\mathbb{A}\|}}\left(\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} + \frac{\|\delta\mathbb{A}\|}{\|\mathbb{A}\|}\right)$$

En general es muy costoso evaluar $\|\mathbb{A}\|$. Usualmente se compara $|\mathbb{A}|$ con a_{ij}.

Ejemplo:

$$\begin{cases} 2x + y = 3 \\ 2x + 1.001y = 0 \end{cases}, \quad |\mathbb{A}| = 0.002$$

Solución: $x = 1501.5, y = -3000$.

[a] Ver González, *Introducción al cálculo numérico* (2014), sección 2.3., y Quarteroni, Sacco y Saleri, *Numerical Mathematics* (2000), sección 3.1.

$$\begin{cases} 2x + y = 3 \\ 2x + 1.002y = 0 \end{cases}, \quad |\mathbb{A}| = 0.004$$

Solución: $x = 751.5, y = -1500$.

$\boxed{0.1\,\%\text{ de cambio en } a_{ij} \mapsto 100\,\%\text{ de cambio en } \boldsymbol{x}.}$

Mal condicionamiento

Si la solución de un sistema lineal cambia mucho cuando el problema cambia muy poco, la matriz está mal condicionada.

$$\begin{cases} 2x + y = 3 \\ 2x + 1.002y = 0 \end{cases}, \quad |\mathbb{A}| = 0.004$$

Solución: $x = 751.5, y = -1500$.

$\boxed{0.1\,\% \text{ de cambio en } a_{ij} \mapsto 100\,\% \text{ de cambio en } \boldsymbol{x}.}$

Mal condicionamiento

Si la solución de un sistema lineal cambia mucho cuando el problema cambia muy poco, la matriz está mal condicionada.

Otro ejemplo:

$$\mathbb{A} = \begin{bmatrix} 1 & 100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}| = 1$$

$$\mathbb{A}^{-1} = \begin{bmatrix} 1 & -100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}^{-1}| = 1$$

$$\begin{cases} 2x + y = 3 \\ 2x + 1.002y = 0 \end{cases}, \quad |\mathbb{A}| = 0.004$$

Solución: $x = 751.5, y = -1500$.

$\boxed{0.1\,\% \text{ de cambio en } a_{ij} \mapsto 100\,\% \text{ de cambio en } \boldsymbol{x}.}$

Mal condicionamiento

Si la solución de un sistema lineal cambia mucho cuando el problema cambia muy poco, la matriz está mal condicionada.

Otro ejemplo:

$$\mathbb{A} = \begin{bmatrix} 1 & 100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}| = 1$$

$$\mathbb{A}^{-1} = \begin{bmatrix} 1 & -100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}^{-1}| = 1$$

Soluciones:

$$\boldsymbol{b} = \begin{bmatrix} 100 \\ 1 \end{bmatrix} \to \boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; \boldsymbol{b} = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \to \boldsymbol{x} = \begin{bmatrix} 100 \\ 0 \end{bmatrix}$$

$\boxed{1\,\% \text{ de cambio en } \boldsymbol{b} \mapsto 100\,\% \text{ de cambio en } \boldsymbol{x}.}$

$$\begin{cases} 2x + y = 3 \\ 2x + 1.002y = 0 \end{cases}, \quad |\mathbb{A}| = 0.004$$

Solución: $x = 751.5, y = -1500$.

$\boxed{0.1\,\% \text{ de cambio en } a_{ij} \mapsto 100\,\% \text{ de cambio en } \boldsymbol{x}.}$

Mal condicionamiento

Si la solución de un sistema lineal cambia mucho cuando el problema cambia muy poco, la matriz está mal condicionada.

Otro ejemplo:

$$\mathbb{A} = \begin{bmatrix} 1 & 100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}| = 1$$

$$\mathbb{A}^{-1} = \begin{bmatrix} 1 & -100 \\ 0 & 1 \end{bmatrix}, \quad |\mathbb{A}^{-1}| = 1$$

Soluciones:

$$\boldsymbol{b} = \begin{bmatrix} 100 \\ 1 \end{bmatrix} \to \boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \to \boldsymbol{x} = \begin{bmatrix} 100 \\ 0 \end{bmatrix}$$

$\boxed{1\,\% \text{ de cambio en } \boldsymbol{b} \mapsto 100\,\% \text{ de cambio en } \boldsymbol{x}.}$

```python
#!/usr/bin/env python3

import numpy as np
from numpy import linalg as la

a = np.array([[1, 100],[0, 1]])
ai = np.array([[1, -100],[0, 1]])
a_norm = la.norm(a, ord=2)
ai_norm = la.norm(ai, ord=2)
print(f"||a|| = {a_norm}, ||ai|| = {ai_norm}")
print(f"cond(a) = {a_norm * ai_norm}")
print(f"cond(a) = {la.cond(a)}")
```

```
$ ./cond.py
||a|| = 100.00999900019995, ||ai|| = 100.00999900019995
cond(a) = 10001.999900019995
cond(a) = 10001.999900019995
```

Métodos directos:

- $a_{ik} \rightleftarrows a_{jk}, |\mathbb{A}| \to -|\mathbb{A}|$
- $k{\times}a_{ij}, |\mathbb{A}| \to k{\times}|\mathbb{A}|$
- $a_{ik} - k\, a_{ik}, |\mathbb{A}|$ no cambia

Métodos directos:

- $a_{ik} \rightleftarrows a_{jk}, |\mathbb{A}| \to -|\mathbb{A}|$
- $k{\times}a_{ij}, |\mathbb{A}| \to k{\times}|\mathbb{A}|$
- $a_{ik} - k\,a_{ik}, |\mathbb{A}|$ no cambia

Método	Forma inicial	Forma final
Eliminación gaussiana	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{U}\mathbf{x} = \mathbf{c}$
Descomposición LU	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{L}\mathbb{U}\mathbf{x} = \mathbf{b}$
Eliminación de Gauss-Jordan	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{I}\mathbf{x} = \mathbf{c}$

Métodos directos:

- $a_{ik} \rightleftarrows a_{jk}, |\mathbb{A}| \to -|\mathbb{A}|$
- $k{\times}a_{ij}, |\mathbb{A}| \to k{\times}|\mathbb{A}|$
- $a_{ik} - k\, a_{ik}, |\mathbb{A}|$ no cambia

Método	Forma inicial	Forma final
Eliminación gaussiana	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{U}\mathbf{x} = \mathbf{c}$
Descomposición LU	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{L}\mathbb{U}\mathbf{x} = \mathbf{b}$
Eliminación de Gauss-Jordan	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{I}\mathbf{x} = \mathbf{c}$

Matriz triangular superior 3x3

$$\mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Matriz triangular inferior 3x3

$$\mathbb{L} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix}$$

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$\mathrm{Ec}(i) \leftarrow \mathrm{Ec}(i) - \lambda \times \mathrm{Ec}(j)$$

donde $\mathrm{Ec}(j)$ se denomina *ecuación pivote*.

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$\mathrm{Ec}(i) \leftarrow \mathrm{Ec}(i) - \lambda \times \mathrm{Ec}(j)$$

donde $\mathrm{Ec}(j)$ se denomina *ecuación pivote*.

Ejemplo:

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 \\ -2x_1 + 4x_2 - 2x_3 = -16 \\ x_1 - 2x_2 + 4x_3 = 17 \end{cases}$$

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right]$$

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$\mathrm{Ec}(i) \leftarrow \mathrm{Ec}(i) - \lambda \times \mathrm{Ec}(j)$$

donde Ec(j) se denomina *ecuación pivote*.

Ejemplo:

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 \\ -2x_1 + 4x_2 - 2x_3 = -16 \\ x_1 - 2x_2 + 4x_3 = 17 \end{cases}$$

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right]$$

Ec(1): pivote;

Ec(2) ← Ec(2) $-(-0.5)\times$ Ec(1);

Ec(3) ← Ec(3) $-0.25\times$ Ec(1)

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & -1.5 & 3.75 & 14.25 \end{array}\right]$$

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$\text{Ec}(i) \leftarrow \text{Ec}(i) - \lambda \times \text{Ec}(j)$$

donde Ec(j) se denomina *ecuación pivote*.

Ejemplo:

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 \\ -2x_1 + 4x_2 - 2x_3 = -16 \\ x_1 - 2x_2 + 4x_3 = 17 \end{cases}$$

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right]$$

Ec(1): pivote;

Ec(2) ← Ec(2) $-(-0.5)\times$ Ec(1);

Ec(3) ← Ec(3) $-0.25\times$ Ec(1)

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & -1.5 & 3.75 & 14.25 \end{array}\right]$$

Ec(2): pivote;

Ec(3) ← Ec(3) $-(-0.5)\times$ Ec(2)

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & 0 & 3 & 9 \end{array}\right]$$

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$\text{Ec}(i) \leftarrow \text{Ec}(i) - \lambda \times \text{Ec}(j)$$

donde Ec(j) se denomina *ecuación pivote*.

Ejemplo:

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 \\ -2x_1 + 4x_2 - 2x_3 = -16 \\ x_1 - 2x_2 + 4x_3 = 17 \end{cases}$$

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right]$$

Ec(1): pivote;

Ec(2) $\leftarrow$ Ec(2) $-(-0.5)\times$ Ec(1);

Ec(3) $\leftarrow$ Ec(3) $-0.25\times$ Ec(1)

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & -1.5 & 3.75 & 14.25 \end{array}\right]$$

Ec(2): pivote;

Ec(3) $\leftarrow$ Ec(3) $-(-0.5)\times$ Ec(2)

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & 0 & 3 & 9 \end{array}\right]$$

Bonus: no se altera $|\mathbb{A}|$:

$$|\mathbb{A}| = |\mathbb{U}| = U_{11} \times U_{22} \times \cdots U_{NN}$$

Fase de solución: sustitución hacia atrás.

Algoritmo:

$$
\left[\begin{array}{ccccccccc|c}
a_{11} & a_{12} & a_{13} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} & b_1 \\
0 & a_{22} & a_{23} & \cdots & a_{2k} & \cdots & a_{2j} & \cdots & a_{2n} & b_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} & b_k \\
\hline
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & a_{ik} & \cdots & a_{ij} & \cdots & a_{in} & b_i \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & a_{nk} & \cdots & a_{nj} & \cdots & a_{nn} & b_N
\end{array}\right]
$$

Eliminación:

for $i \leftarrow k+1, k+2, \ldots n$ **do**
    $\lambda \leftarrow a_{ik}/a_{kk}$
    for $j \leftarrow k, n$ **do**
        $a_{ij} \leftarrow a_{ij} - \lambda a_{kj}$
    end for
    $b_i \leftarrow b_i - \lambda b_k$
end for

Solución:

$$
\begin{aligned}
x_n &= b_n/a_{nn} \\
x_k &= \left(b_k - \sum_{j=k+1}^{n} a_{kj}x_j\right)\frac{1}{a_{kk}}
\end{aligned}
$$

$$k = n-1, n-2, \ldots, 1$$

Factorización o descomposición LU:

$$\mathbb{A} = \mathbb{L}\mathbb{U}$$

Nombre	Restricción
Descomposición de Doolittle	$L_{ii} = 1, \ i = 1, 2, \cdots, n$
Descomposición de Crout	$U_{ii} = 1, \ i = 1, 2, \cdots, n$
Descomposición de Choleski ($\mathbb{A}$ debe ser simétrica y definida positiva)	$\mathbb{L} = \mathbb{U}^T$

Luego de la factorización:

$$\mathbb{L}\mathbb{U}\,\boldsymbol{x} = \boldsymbol{b}$$

La solución consiste en:

$$\mathbb{L}\,\boldsymbol{y} = \boldsymbol{b}$$

resuelta por sustitución hacia adelante, seguida de:

$$\mathbb{U}\,\boldsymbol{x} = \boldsymbol{y}$$

que da el resultado $\boldsymbol{x}$ obtenido por sustitución hacia atrás.

Método de Doolittle:

$$\mathbb{L} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \qquad \mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Encontramos $\mathbb{A}$:

$$\mathbb{A} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21} + U_{22} & U_{13}L_{21} + U_{23} \\ U_{11}L_{31} & U_{12}L_{31} + U_{22}L_{32} & U_{13}L_{31} + U_{23}L_{32} + U_{33} \end{bmatrix}$$

Método de Doolittle:

$$\mathbb{L} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \qquad \mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Encontramos $\mathbb{A}$:

$$\mathbb{A} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21} + U_{22} & U_{13}L_{21} + U_{23} \\ U_{11}L_{31} & U_{12}L_{31} + U_{22}L_{32} & U_{13}L_{31} + U_{23}L_{32} + U_{33} \end{bmatrix}$$

Aplicamos ahora eliminación gaussiana con las siguientes operaciones elementales:

$$\text{fila } 2 \leftarrow \text{fila } 2 - L_{21} \times \text{ fila } 1 \text{ (elimina } a_{21})$$
$$\text{fila } 3 \leftarrow \text{fila } 3 - L_{31} \times \text{ fila } 1 \text{ (elimina } a_{31})$$

$$\mathbb{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

Método de Doolittle (cont.): tomamos ahora la segunda fila como pivote:

$$\mathbb{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

fila 3 $\leftarrow$ fila 3 $-L_{32}\times$ fila 2 (elimina a_{32})

$$\mathbb{A}'' = \mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Método de Doolittle (cont.): tomamos ahora la segunda fila como pivote:

$$\mathbb{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

fila 3 $\leftarrow$ fila 3 $-L_{32}\times$ fila 2 (elimina a_{32})

$$\mathbb{A}'' = \mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Características del método de Doolittle:

- La matriz $\mathbb{U}$ es idéntica a la matriz triangular superior que resulta de la eliminación gaussiana
- Los elementos no diagonales de $\mathbb{L}$ son los factores que multiplican a la ecuación pivote durante la eliminación gaussiana: L_{ij} es el multiplicador que elimina a_{ij}

Método de Doolittle (cont.): tomamos ahora la segunda fila como pivote:

$$\mathbb{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

fila 3 $\leftarrow$ fila 3 $-L_{32}\times$ fila 2 (elimina a_{32})

$$\mathbb{A}'' = \mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Características del método de Doolittle:

- La matriz $\mathbb{U}$ es idéntica a la matriz triangular superior que resulta de la eliminación gaussiana
- Los elementos no diagonales de $\mathbb{L}$ son los factores que multiplican a la ecuación pivote durante la eliminación gaussiana: L_{ij} es el multiplicador que elimina a_{ij}

Almacenamiento:

$$\mathbb{L}/\mathbb{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ L_{21} & U_{22} & U_{23} \\ L_{31} & L_{32} & U_{33} \end{bmatrix}$$

Sistema:

$$\begin{aligned} 2x_1 - x_2 \phantom{{}-x_3} &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ - x_2 + x_3 &= 0 \end{aligned}$$

Solución: $x_1 = x_2 = x_3 = 1$

$$\left[\begin{array}{ccc|c} 2 & -1 & 0 & 1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{array}\right]$$

OK

Sistema:

$$\begin{aligned} 2x_1 - x_2 \phantom{{}-x_3} &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ - x_2 + x_3 &= 0 \end{aligned}$$

Solución: $x_1 = x_2 = x_3 = 1$

$$\left[\begin{array}{ccc|c} 2 & -1 & 0 & 1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{array}\right]$$

OK

$$\left[\begin{array}{ccc|c} 0 & -1 & 1 & 0 \\ -1 & 2 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array}\right]$$

NOK

Pivoteo

$$[\mathbb{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ -1 & 2 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array}\right] \rightarrow [\mathbb{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & 2-1/\varepsilon & -1+1/\varepsilon & 0 \\ 0 & -1+2/\varepsilon & -2/\varepsilon & 1 \end{array}\right]$$

Pivoteo

$$[\mathbb{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ -1 & 2 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array}\right] \to [\mathbb{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & 2-1/\varepsilon & -1+1/\varepsilon & 0 \\ 0 & -1+2/\varepsilon & -2/\varepsilon & 1 \end{array}\right]$$

Almacenamiento en la memoria ($\varepsilon \ll 1$):

$$[\mathbb{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & -1/\varepsilon & 1/\varepsilon & 0 \\ 0 & 2/\varepsilon & -2/\varepsilon & 1 \end{array}\right]$$

Pivoteo

$$[\mathbb{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ -1 & 2 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array}\right] \to [\mathbb{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & 2-1/\varepsilon & -1+1/\varepsilon & 0 \\ 0 & -1+2/\varepsilon & -2/\varepsilon & 1 \end{array}\right]$$

Almacenamiento en la memoria ($\varepsilon \ll 1$):

$$[\mathbb{A}'|\mathbf{b}'] = \left[\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ 0 & -1/\varepsilon & 1/\varepsilon & 0 \\ 0 & 2/\varepsilon & -2/\varepsilon & 1 \end{array}\right]$$

$\mathbb{A}$ es *diagonalmente dominante* si:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{ij}|; \; (i = 1, 2, \ldots, N)$$

$$\begin{bmatrix} -2 & 4 & -1 \\ 1 & -1 & 3 \\ 4 & -2 & 1 \end{bmatrix}$$

NO diagonalmente dominante

$$\begin{bmatrix} 4 & -2 & 1 \\ -2 & 4 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

Diagonalmente dominante

En todos los casos: si $a_{ii} \neq 0 \mapsto$ no intercambiar filas.

- **Pivoteo trivial**:
  - $a_{ii} = 0 \mapsto$ buscar el primer $a_{ki} \neq 0 (k > i)$ e intercambiar filas $i \leftrightarrows k$.

Pivoteo: estrategias

En todos los casos: si $a_{ii} \neq 0 \mapsto$ no intercambiar filas.

- **Pivoteo trivial**:
  - $a_{ii} = 0 \mapsto$ buscar el primer $a_{ki} \neq 0 (k > i)$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial**:
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $|a_{ki}| = \max\limits_{j>i} |a_{ji}| \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$.

Pivoteo: estrategias

En todos los casos: si $a_{ii} \neq 0 \mapsto$ no intercambiar filas.

- **Pivoteo trivial**:
  - $a_{ii} = 0 \mapsto$ buscar el primer $a_{ki} \neq 0 (k > i)$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial**:
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $|a_{ki}| = \max\limits_{j>i} |a_{ji}| \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial escalado**:
  - Calcular $s_i = \max\limits_{1 \leq j \leq N} |a_{ij}|, \ i = 1, \ldots, N$
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\dfrac{|a_{ki}|}{s_k} = \max\limits_{j>i} \dfrac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$ y $s_i \leftrightarrows s_k$.

En todos los casos: si $a_{ii} \neq 0 \mapsto$ no intercambiar filas.

- **Pivoteo trivial**:
  - $a_{ii} = 0 \mapsto$ buscar el primer $a_{ki} \neq 0 (k > i)$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial**:
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $|a_{ki}| = \max\limits_{j>i} |a_{ji}| \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial escalado**:
  - Calcular $s_i = \max\limits_{1 \leq j \leq N} |a_{ij}|,\ i = 1, \ldots, N$
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\dfrac{|a_{ki}|}{s_k} = \max\limits_{j>i} \dfrac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$ y $s_i \leftrightarrows s_k$.
- **Pivoteo completo o maximal**:
  - $a_{ii} = 0 \mapsto$ buscar la fila $j > i$ y columna $k > i$ tal que $|a_{jk}| = \max\limits_{\substack{l>i \\ m>i}} |a_{lm}| \wedge a_{jk} \neq 0$ e intercambiar filas $i \leftrightarrows j$ y columnas $i \leftrightarrows k$.

Pivoteo: estrategias

En todos los casos: si $a_{ii} \neq 0 \mapsto$ no intercambiar filas.

- **Pivoteo trivial**:
  - $a_{ii} = 0 \mapsto$ buscar el primer $a_{ki} \neq 0 (k > i)$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial**:
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $|a_{ki}| = \max_{j>i} |a_{ji}| \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$.
- **Pivoteo parcial escalado**:
  - Calcular $s_i = \max_{1 \leq j \leq N} |a_{ij}|, \; i = 1, \ldots, N$
  - $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\dfrac{|a_{ki}|}{s_k} = \max_{j>i} \dfrac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrows k$ y $s_i \leftrightarrows s_k$.
- **Pivoteo completo o maximal**:
  - $a_{ii} = 0 \mapsto$ buscar la fila $j > i$ y columna $k > i$ tal que $|a_{jk}| = \max_{\substack{l>i \\ m>i}} |a_{lm}| \wedge a_{jk} \neq 0$ e intercambiar filas $i \leftrightarrows j$ y columnas $i \leftrightarrows k$.

Nota: en matemática "$x = 0$, $x \neq 0$", en *mundo real* "$|x| < \varepsilon$, $|x| > \varepsilon$".

Ejemplo con Python:

```python
#!/usr/bin/env python3

import numpy as np
from scipy import linalg

a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
b = np.array([2, 4, -1])
x = linalg.solve(a, b)
print('x = ', x)

# Verificación
print('a @ x ?', (np.dot(a, x) == b))
print('a @ x ?', np.allclose(a@x, b))
print('a @ x =', a@x)
# Descomposición LU
p, l, u = linalg.lu(a)
print(p)
print(l)
print(u)
```

```
$ ./ejemplo.py
x =  [ 2. -2.  9.]
a @ x ? [ True  True  True]
a @ x ? True
a @ x = [ 2.  4. -1.]
[[1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]]
[[ 1.          0.          0.        ]
 [ 0.          1.          0.        ]
 [ 0.33333333 -0.33333333  1.        ]]
[[3.         2.         0.        ]
 [0.         5.         1.        ]
 [0.         0.         0.33333333]]
```

Lecturas recomendadas I

- R.L. Burden, D.J. Faires y A.M. Burden. *Análisis numérico*. 10.ª ed. Mexico: Cengage Learning, 2017. Capítulo 6.
- Gilbert Strang. *Linear Algebra and Its Applications, 4th Edition*. 4th. Brooks Cole, 2006. Capítulo 7.
- A.J. Salgado y S.M. Wise. *Classical Numerical Analysis*. Cambridge, United Kingdom: Cambridge University Press, 2023. DOI: 10.1017/9781108942607. Capítulo 3.
- A. Quarteroni, R. Sacco y F. Saleri. *Numerical Mathematics*. New York, United States: Springer-Verlag, 2000. Capítulo 3.