

SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

CONDICIONAMIENTO. MÉTODOS DIRECTOS: ELIMINACIÓN GAUSSIANA, FACTORIZACIÓN LU. PIVOTEO. MÉTODOS INDIRECTOS. GAUSS-SEIDEL.

Manuel Carlevaro

Departamento de Ingeniería Mecánica

Grupo de Materiales Granulares - UTN FRLP

manuel.carlevaro@gmail.com

Resolver:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

Resolver:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

O, en forma matricial:

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

Resolver:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

O, en forma matricial:

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

Matriz de coeficientes aumentada:

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

Resolver:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

O, en forma matricial:

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

Matriz de coeficientes aumentada:

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

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

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

Resolver:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

O, en forma matricial:

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

Matriz de coeficientes aumentada:

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

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

- ▶ \mathbb{A} es invertible (no singular)
- ▶ $\text{rg}(A) = n$
- ▶ El sistema homogéneo $\mathbb{A}\mathbf{x} = \mathbf{0}$ 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.

Estabilidad de la solución:

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

Unicidad de la solución: \mathbb{A} es **no singular**: $|\mathbb{A}| \neq 0$.

Número de condición: $\text{cond}(\mathbb{A}) = \|\mathbb{A}\| \|\mathbb{A}^{-1}\|$.

Se puede demostrar¹, que si

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

se cumple:

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

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: $x = 1501.5, y = -3000$.

¹Ver González, *Introducción al cálculo numérico* (2014), sección 2.3., y Quarteroni, Sacco y Saleri, *Numerical Mathematics* (2000), sección 3.1.

Estabilidad de la solución:

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

Unicidad de la solución: \mathbb{A} es **no singular**: $|\mathbb{A}| \neq 0$.

Número de condición: $\text{cond}(\mathbb{A}) = \|\mathbb{A}\| \|\mathbb{A}^{-1}\|$.

Se puede demostrar¹, que si

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

se cumple:

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

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: $x = 1501.5, y = -3000$.

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

Solución: $x = 751.5, y = -1500$.

0.1% de cambio en $a_{ij} \mapsto 100\%$ de cambio en \mathbf{x} .

Mal condicionamiento

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

¹Ver González, *Introducción al cálculo numérico* (2014), sección 2.3., y Quarteroni, Sacco y Saleri, *Numerical Mathematics* (2000), sección 3.1.

Otro ejemplo:

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

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

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:

$$\mathbf{b} = \begin{bmatrix} 100 \\ 1 \end{bmatrix} \rightarrow \mathbf{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \rightarrow \mathbf{x} = \begin{bmatrix} 100 \\ 0 \end{bmatrix}$$

1% de cambio en $\mathbf{b} \mapsto 100\%$ de cambio en \mathbf{x} .
--

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:

$$\mathbf{b} = \begin{bmatrix} 100 \\ 1 \end{bmatrix} \rightarrow \mathbf{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \rightarrow \mathbf{x} = \begin{bmatrix} 100 \\ 0 \end{bmatrix}$$

1% de cambio en $\mathbf{b} \mapsto 100\%$ de cambio en \mathbf{x} .

```
1 #!/usr/bin/env python3
2
3 import numpy as np
4 from numpy import linalg as la
5
6 a = np.array([[1, 100],[0, 1]])
7 ai = np.array([[1, -100],[0, 1]])
8 a_norm = la.norm(a, ord=2)
9 ai_norm = la.norm(ai, ord=2)
10 print(f"||a|| = {a_norm}, ||ai|| = {ai_norm}")
11 print(f"cond(a) = {a_norm * ai_norm}")
12 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: 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
-

Métodos directos:

- ▶ $a_{ik} \leftrightarrow a_{jk}, |\mathbb{A}| \rightarrow -|\mathbb{A}|$
- ▶ $k \times a_{ij}, |\mathbb{A}| \rightarrow k \times |\mathbb{A}|$
- ▶ $a_{ik} - k a_{ik}, |\mathbb{A}|$ no cambia

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:

- ▶ $a_{ik} \leftrightarrow a_{jk}, |\mathbb{A}| \rightarrow -|\mathbb{A}|$
- ▶ $k \times a_{ij}, |\mathbb{A}| \rightarrow k \times |\mathbb{A}|$
- ▶ $a_{ik} - k a_{ik}, |\mathbb{A}|$ 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	$\mathbb{A}\mathbf{x} = \mathbf{b}$	$\mathbb{I}\mathbf{x} = \mathbf{c}$

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$E_c(i) \leftarrow E_c(i) - \lambda \times E_c(j)$$

donde $E_c(j)$ se denomina *ecuación pivote*.

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$Ec(i) \leftarrow Ec(i) - \lambda \times Ec(j)$$

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

Ejemplo:

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

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

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$Ec(i) \leftarrow Ec(i) - \lambda \times Ec(j)$$

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;

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

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

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

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$Ec(i) \leftarrow Ec(i) - \lambda \times Ec(j)$$

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;

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

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

Eliminación gaussiana

Dos fases: eliminación y solución.

Fase de eliminación:

Utiliza solo una operación elemental:

$$Ec(i) \leftarrow Ec(i) - \lambda \times Ec(j)$$

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 $|\mathbb{A}|$:

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

Fase de solución: sustitución hacia atrás.

Algoritmo:

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

Eliminación:

- 1: **for** $i \leftarrow k + 1, k + 2, \dots, n$ **do**
- 2: $\lambda \leftarrow a_{ik}/a_{kk}$
- 3: **for** $j \leftarrow k, n$ **do**
- 4: $a_{ij} \leftarrow a_{ij} - \lambda a_{kj}$
- 5: **end for**
- 6: $b_i \leftarrow b_i - \lambda b_k$
- 7: **end for**

Solución:

$$x_n = b_n/a_{nn}$$

$$x_k = \left(b_k - \sum_{j=k+1}^n a_{kj}x_j \right) \frac{1}{a_{kk}}$$

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

Factorización o descomposición LU:

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

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

Luego de la factorización:

$$\mathbb{L}\mathbb{U}\mathbf{x} = \mathbf{b}$$

La solución consiste en:

$$\mathbb{L}\mathbf{y} = \mathbf{b}$$

resuelta por sustitución hacia adelante,
seguida de:

$$\mathbb{U}\mathbf{x} = \mathbf{y}$$

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

Método de Doolittle:

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

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:

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

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:

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

fila 3 \leftarrow fila 3 $- L_{31} \times$ fila 1 (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:

$$2x_1 - x_2 = 1$$

$$-x_1 + 2x_2 - x_3 = 0$$

$$-x_2 + x_3 = 0$$

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:

$$2x_1 - x_2 = 1$$

$$-x_1 + 2x_2 - x_3 = 0$$

$$-x_2 + x_3 = 0$$

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

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

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

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

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

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

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:

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

$$\left[\begin{array}{ccc} -2 & 4 & -1 \\ 1 & -1 & 3 \\ 4 & -2 & 1 \end{array} \right]$$

NO diagonalmente dominante

$$\left[\begin{array}{ccc} 4 & -2 & 1 \\ -2 & 4 & -1 \\ 1 & -1 & 3 \end{array} \right]$$

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 $i \leftrightarrow k$.

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 $i \leftrightarrow k$.

► **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 $i \leftrightarrow k$.

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 $i \leftrightarrow k$.

► **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 $i \leftrightarrow k$.

► **Pivoteo parcial escalado:**

► Calcular $s_i = \max_{1 \leq j \leq N} |a_{ij}|$, $i = 1, \dots, N$

► $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\frac{|a_{ki}|}{s_k} = \max_{j>i} \frac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrow k$ y $s_i \leftrightarrow s_k$.

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 $i \leftrightarrow k$.

► **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 $i \leftrightarrow k$.

► **Pivoteo parcial escalado:**

► Calcular $s_i = \max_{1 \leq j \leq N} |a_{ij}|$, $i = 1, \dots, N$

► $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\frac{|a_{ki}|}{s_k} = \max_{j>i} \frac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrow k$ y $s_i \leftrightarrow s_k$.

► **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 $i \leftrightarrow j$ y columnas $i \leftrightarrow k$.

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 $i \leftrightarrow k$.

► **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 $i \leftrightarrow k$.

► **Pivoteo parcial escalado:**

► Calcular $s_i = \max_{1 \leq j \leq N} |a_{ij}|$, $i = 1, \dots, N$

► $a_{ii} = 0 \mapsto$ buscar la fila k tal que $\frac{|a_{ki}|}{s_k} = \max_{j>i} \frac{|a_{ji}|}{s_j} \wedge a_{ji} \neq 0$ e intercambiar filas $i \leftrightarrow k$ y $s_i \leftrightarrow s_k$.

► **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 $i \leftrightarrow j$ y columnas $i \leftrightarrow k$.

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

Ejemplo con Python:

```
1 #!/usr/bin/env python3
2
3 import numpy as np
4 from scipy import linalg
5
6 a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
7 b = np.array([2, 4, -1])
8 x = linalg.solve(a, b)
9 print('x = ', x)
10
11 # Verificación
12 print('a @ x ?', (np.dot(a, x) == b))
13 print('a @ x ?', np.allclose(a @ x, b))
14 print('a @ x =', a @ x)
15 # Descomposición LU
16 p, l, u = linalg.lu(a)
17 print(p)
18 print(l)
19 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]]
```

Ventajas:

- ▶ Es posible almacenar solo los elementos no nulos de la matriz
- ▶ Las iteraciones son auto-correctivas

Ventajas:

- ▶ Es posible almacenar solo los elementos no nulos de la matriz
- ▶ Las iteraciones son auto-correctivas

Desventajas:

- ▶ Más lentos que los métodos directos
- ▶ No siempre convergen (garantizado cuando la matriz es diagonalmente dominante)

Ventajas:

- ▶ Es posible almacenar solo los elementos no nulos de la matriz
- ▶ Las iteraciones son auto-correctivas

Desventajas:

- ▶ Más lentos que los métodos directos
- ▶ No siempre convergen (garantizado cuando la matriz es diagonalmente dominante)

Método de Jacobi

Escribimos las ecuaciones $\mathbf{Ax} = \mathbf{b}$ en notación escalar:

$$\sum_{j=1}^N a_{ij}x_j = b_i, \quad i = 1, 2, \dots, n$$

Extraemos el término que contiene x_i :

$$a_{ii}x_i + \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij}x_j = b_i, \quad i = 1, 2, \dots, n$$

Resolviendo para x_i tenemos:

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij}x_j \right), \quad i = 1, 2, \dots, n$$

Esquema iterativo: para cada $k \geq 1$, generamos $x_i^{(k)}$ de $\mathbf{x}^{(k)}$ a partir de $\mathbf{x}^{(k-1)}$ mediante

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k-1)} \right), \quad i = 1, 2, \dots, n$$

Esquema iterativo: para cada $k \geq 1$, generamos $x_i^{(k)}$ de $\mathbf{x}^{(k)}$ a partir de $\mathbf{x}^{(k-1)}$ mediante

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k-1)} \right), \quad i = 1, 2, \dots, n$$

Método de Gauss-Seidel:

Para $i > 1$ tenemos $x_1^{(k)}, x_2^{(k)}, \dots, x_{i-1}^{(k)}$ de $\mathbf{x}^{(k)}$:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^N a_{ij} x_j^{(k-1)} \right), \quad i = 1, 2, \dots, n$$

Esquema iterativo: para cada $k \geq 1$, generamos $x_i^{(k)}$ de $\mathbf{x}^{(k)}$ a partir de $\mathbf{x}^{(k-1)}$ mediante

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k-1)} \right), \quad i = 1, 2, \dots, n$$

Método de Gauss-Seidel:

Para $i > 1$ tenemos $x_1^{(k)}, x_2^{(k)}, \dots, x_{i-1}^{(k)}$ de $\mathbf{x}^{(k)}$:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^N a_{ij} x_j^{(k-1)} \right), \quad i = 1, 2, \dots, n$$

Técnica de relajación (Jacobi):

$$x_i^{(k)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k-1)} \right) + (1 - \omega) x_i^{(k)}$$

Determinación de ω :

Si $\Delta x^{(k)} = |x^{(k-1)} - x^{(k)}|$ calculado con $\omega = 1$:

$$\omega_{\text{opt}} \approx \frac{2}{1 + \sqrt{1 - (\Delta x^{(k+p)}) / \Delta x^{(k)}}^{1/p}}$$

donde p es un entero positivo.

Esquema general con relajación:

- ▶ Realizar k iteraciones con $\omega = 1$ ($k = 10$). Luego de la iteración k almacenar $\Delta x^{(k)}$.
- ▶ Realizar p iteraciones adicionales y almacenar $\Delta x^{(k+p)}$ para la última iteración.
- ▶ Realizar las iteraciones siguientes con $\omega = \omega_{\text{opt}}$.

Técnica de relajación (Jacobi):

$$x_i^{(k)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k-1)} \right) + (1 - \omega) x_i^{(k)}$$

Determinación de ω :

Si $\Delta x^{(k)} = |x^{(k-1)} - x^{(k)}|$ calculado con $\omega = 1$:

$$\omega_{\text{opt}} \approx \frac{2}{1 + \sqrt{1 - (\Delta x^{(k+p)}) / \Delta x^{(k)}}^{1/p}}$$

donde p es un entero positivo.

Esquema general con relajación:

- ▶ Realizar k iteraciones con $\omega = 1$ ($k = 10$). Luego de la iteración k almacenar $\Delta x^{(k)}$.
- ▶ Realizar p iteraciones adicionales y almacenar $\Delta x^{(k+p)}$ para la última iteración.
- ▶ Realizar las iteraciones siguientes con $\omega = \omega_{\text{opt}}$.

Algoritmo:

Requiere \mathbb{A} , \mathbf{b} y ω

▷ inputs

Devuelve \mathbf{x}

▷ output

1: $\mathbf{x} \leftarrow$ valores aleatorios

2: **repeat**

3: **for** $i \leftarrow 1, \dots, N$ **do**

4: $\sigma \leftarrow 0$

5: **for** $j \leftarrow 1, \dots, N$ **do**

6: **if** $j \neq i$ **then**

7: $\sigma \leftarrow \sigma + a_{ij} x_j$

8: **end if**

9: **end for**

10: $x_i \leftarrow x_i + \omega \left(\frac{b_i - \sigma}{a_{ii}} - x_i \right)$

11: **end for**

▷ Fin j -loop

12: Verificar convergencia

13: **until** Convergencia alcanzada

▷ Fin i -loop

```
1 #!/usr/bin/env python3
2
3 import numpy as np
4 from scipy import linalg
5
6 a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
7 b = np.array([2, 4, -1])
8 x = linalg.solve(a, b)
9 print('x = ', x)
10
11 # Verificación
12 print('a @ x ?', (np.dot(a, x) == b))
13 print('a @ x ?', np.allclose(a @ x, b))
14 print('a @ x =', a @ x)
15 # Descomposición LU
16 p, l, u = linalg.lu(a)
17 print(p)
18 print(l)
19 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]]
```

- ▶ 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](https://doi.org/10.1017/9781108942607). Capítulo 3.
- ▶ A. Quarteroni, R. Sacco y F. Saleri. **Numerical Mathematics**. New York, United States: Springer-Verlag, 2000. Capítulo 3.