Universidad Nacional de Colombia

Facultad de Ingeniería

Solución de $Ax = b$ por Gauss–Jordan, Gauss–Seidel y Cramer

Autor:

Nicolás Plata Molano

Profesor:

Daniel Felipe Rodríguez Ramírez

Bogotá, Colombia
Septiembre 2025

1. Descripción breve del código

Se implementó un script en Python que resuelve sistemas $Ax = b$ de tamaño $n \in [3, 10]$ por tres métodos: *Gauss–Jordan* (directo con pivoteo parcial), *Gauss–Seidel* (iterativo) y *Regla de Cramer*. La interfaz es por menús:

- Selección del método y del orden n.
- Ingreso de A y b por *pegado rápido* (permitiendo estilo MATLAB con corchetes y ';'), *paso a paso*, o *generando un ejemplo aleatorio* (diagonalmente dominante para favorecer la convergencia de Gauss–Seidel).
- Opción **modo detallado** que imprime en consola los **pasos**: en Gauss–Jordan, pivotes, normalizaciones y operaciones por filas; en Gauss–Seidel, las fórmulas de actualización variable por variable con la norma del cambio; en Cramer, $\det(A)$, matrices A_i y $\det(A_i)$ para cada i.

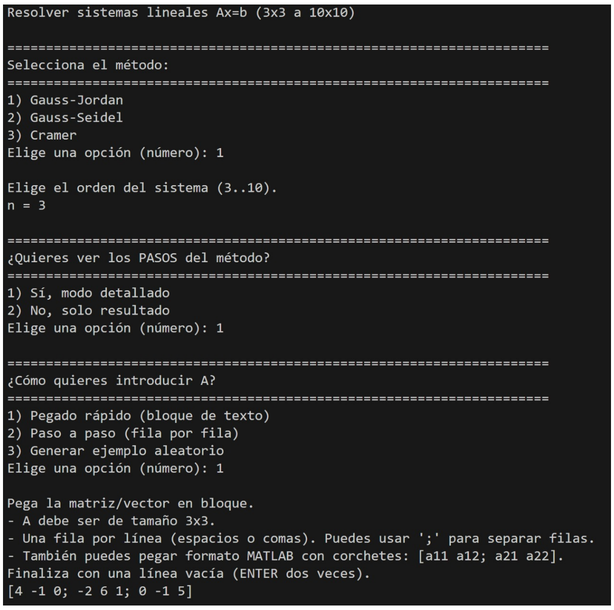
Nota de uso: las "capturas de consola" que se muestran se han escrito con `verbatim` para evitar errores de compilación por caracteres especiales como `^` o `_`.

2. Sistema de prueba

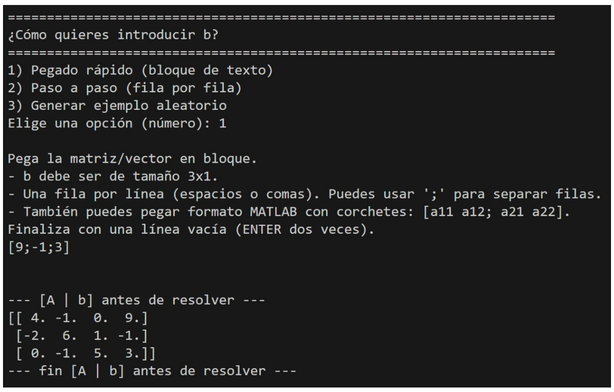
Se utiliza un sistema 3×3 con dominancia diagonal y solución entera sencilla. Por facilidad, emplearemos este mismo sistema de prueba en los tres métodos. :

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -2 & 6 & 1 \\ 0 & -1 & 5 \end{bmatrix}, \quad b = \begin{bmatrix} 9 \\ -7 \\ 16 \end{bmatrix}.$$

La solución verdadera (para validación) es $x_1 = 2,\ x_2 = -1,\ x_3 = 3$.



(a) Selección del método, orden y matriz A.



(b) Selección de matriz b y vista previa de $[A|b]$.

Figura 1: Menú del script y vista previa del sistema antes de resolver.

3. Gauss–Jordan (directo con pivoteo parcial)

Formar la aumentada $[A|b]$ y llevarla por operaciones elementales a $[I|x]$. Se normaliza cada pivote a 1 y se *anula su columna* arriba y abajo.

Aumentada inicial

$$\left[\begin{array}{ccc|c} 4 & -1 & 0 & 9 \\ -2 & 6 & 1 & -7 \\ 0 & -1 & 5 & 16 \end{array}\right].$$

Paso 1 (columna 1): Normalizar F1 con 4 y anular en F2.

$$\text{F1} \leftarrow \frac{1}{4}\text{F1} = \left[\begin{array}{ccc|c} 1 & -\frac{1}{4} & 0 & \frac{9}{4} \end{array}\right], \quad \text{F2} \leftarrow \text{F2} - (-2)\,\text{F1} = \text{F2} + 2\text{F1} = \left[\begin{array}{ccc|c} 0 & \frac{11}{2} & 1 & -\frac{5}{2} \end{array}\right].$$

La fila 3 no cambia (ya tiene 0 en la primera columna).

Paso 2 (columna 2): Normalizar F2 y anular en F1 y F3.

$$\text{F2} \leftarrow \tfrac{2}{11}\text{F2} = \left[\begin{array}{ccc|c} 0 & 1 & \frac{2}{11} & -\frac{5}{11} \end{array}\right].$$

$$\text{F1} \leftarrow \text{F1} + \tfrac{1}{4}\,\text{F2} \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{22} & \frac{47}{22} \end{array}\right], \qquad \text{F3} \leftarrow \text{F3} + \text{F2} \Rightarrow \left[\begin{array}{ccc|c} 0 & 0 & \frac{57}{11} & \frac{171}{11} \end{array}\right].$$

Paso 3 (columna 3): Normalizar F3 y anular en F1 y F2.

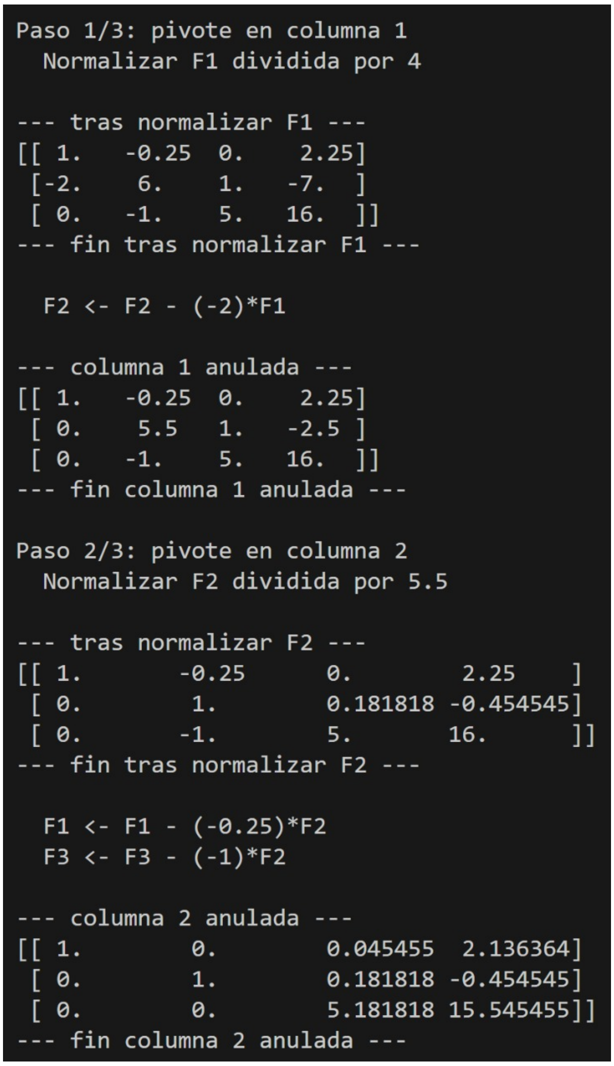
$$\text{F3} \leftarrow \tfrac{11}{57}\text{F3} = \left[\begin{array}{ccc|c} 0 & 0 & 1 & 3 \end{array}\right].$$

$$\text{F1} \leftarrow \text{F1} - \tfrac{1}{22}\text{F3} \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \end{array}\right], \quad \text{F2} \leftarrow \text{F2} - \tfrac{2}{11}\text{F3} \Rightarrow \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \end{array}\right].$$

$$\Rightarrow \quad [I|x] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 3 \end{array}\right], \quad \boxed{x = (2, -1, 3)}.$$

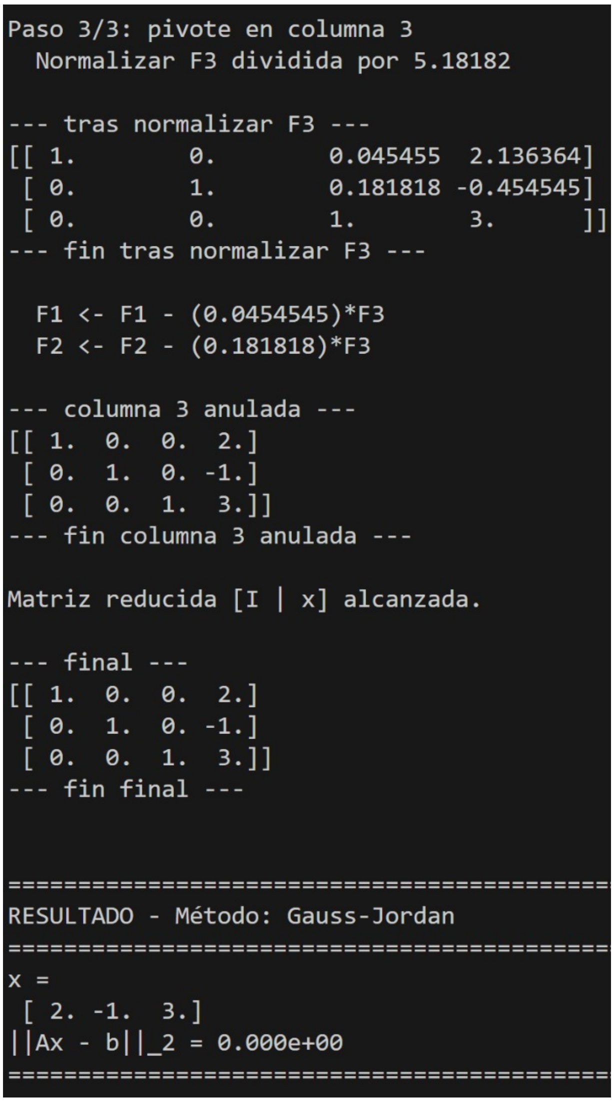
Traza típica en consola (modo detallado):

```
Paso 1/3: pivote en columna 1
  Normalizar F1 dividida por 4
  F2 <- F2 - (-2)*F1
[A|b] columna 1 anulada ...

Paso 2/3: pivote en columna 2
  Normalizar F2 dividida por 5.5
  F1 <- F1 - (-0.25)*F2
  F3 <- F3 - (-1)*F2
[A|b] columna 2 anulada ...

Paso 3/3: pivote en columna 3
  Normalizar F3 dividida por 5.1818
  F1 <- F1 - (0.0455)*F3
  F2 <- F2 - (0.1818)*F3
Matriz reducida [I | x] alcanzada.
```



```
Paso 1/3: pivote en columna 1
  Normalizar F1 dividida por 4

--- tras normalizar F1 ---
[[ 1.   -0.25  0.    2.25]
 [-2.    6.    1.   -7.  ]
 [ 0.   -1.    5.   16.  ]]
--- fin tras normalizar F1 ---

  F2 <- F2 - (-2)*F1

--- columna 1 anulada ---
[[ 1.   -0.25  0.    2.25]
 [ 0.    5.5   1.   -2.5 ]
 [ 0.   -1.    5.   16.  ]]
--- fin columna 1 anulada ---

Paso 2/3: pivote en columna 2
  Normalizar F2 dividida por 5.5

--- tras normalizar F2 ---
[[ 1.        -0.25       0.         2.25    ]
 [ 0.         1.         0.181818 -0.454545]
 [ 0.        -1.         5.        16.      ]]
--- fin tras normalizar F2 ---

  F1 <- F1 - (-0.25)*F2
  F3 <- F3 - (-1)*F2

--- columna 2 anulada ---
[[ 1.         0.         0.045455  2.136364]
 [ 0.         1.         0.181818 -0.454545]
 [ 0.         0.         5.181818 15.545455]]
--- fin columna 2 anulada ---
```

(a) Gauss–Jordan: Paso 1/3 y Paso 2/3.



```
Paso 3/3: pivote en columna 3
  Normalizar F3 dividida por 5.18182

--- tras normalizar F3 ---
[[ 1.         0.         0.045455  2.136364]
 [ 0.         1.         0.181818 -0.454545]
 [ 0.         0.         1.         3.      ]]
--- fin tras normalizar F3 ---

  F1 <- F1 - (0.0454545)*F3
  F2 <- F2 - (0.181818)*F3

--- columna 3 anulada ---
[[ 1.  0.  0.  2.]
 [ 0.  1.  0. -1.]
 [ 0.  0.  1.  3.]]
--- fin columna 3 anulada ---

Matriz reducida [I | x] alcanzada.

--- final ---
[[ 1.  0.  0.  2.]
 [ 0.  1.  0. -1.]
 [ 0.  0.  1.  3.]]
--- fin final ---


==============================================
RESULTADO - Método: Gauss-Jordan
==============================================
x =
 [ 2. -1.  3.]
||Ax - b||_2 = 0.000e+00
==============================================
```

(b) Gauss–Jordan: Paso 3/3 y matriz $[I|x]$.

Figura 2: Trazas de consola del método de Gauss–Jordan: evolución de $[A|b]$ hasta $[I|x]$.

4. Gauss–Seidel (iterativo)

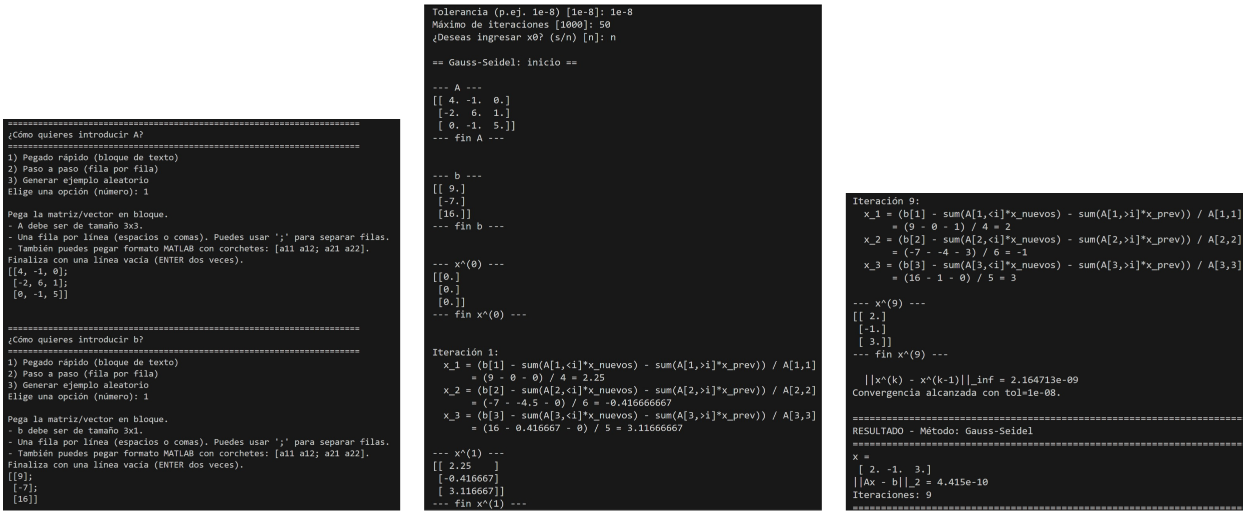
Se despeja cada ecuación por la variable diagonal y se actualiza en orden usando los valores más recientes. Para nuestro sistema:

$$x_1 = \tfrac{9+x_2}{4}, \qquad x_2 = \tfrac{-7+2x_1-x_3}{6}, \qquad x_3 = \tfrac{16+x_2}{5}.$$

Con $x^{(0)} = (0,0,0)$:
Iteración 1: $x_1^{(1)} = 2{,}25$, $x_2^{(1)} = -0{,}4166667$, $x_3^{(1)} = 3{,}1166667$.
Iteración 2: $x_1^{(2)} = 2{,}1458333$, $x_2^{(2)} = -0{,}9708333$, $x_3^{(2)} = 3{,}0058333$.
Iteración 3: $x^{(3)} \approx (2{,}0072917, -0{,}9985417, 3{,}0002917)$.
Iteración 4: $x^{(4)} \approx (2{,}0003646, -0{,}9999271, 3{,}0000146)$.
Con tolerancia 10^{-8}, converge rápidamente a $x^\star = (2,-1,3)^\top$ (A es dominante).
Traza típica en consola (modo detallado):

```
== Gauss-Seidel: inicio ==
x^(0) = [0, 0, 0]^T
```

```
Iteración 1:
  x1 = (9 - 0 - 0)/4 = 2.25
  x2 = (-7 + 2*2.25 - 0)/6 = -0.4166667
  x3 = (16 + (-0.4166667))/5 = 3.1166667
  ||x^(1)-x^(0)||_inf = 3.1166667

Iteración 2: ...  x = [2.1458333, -0.9708333, 3.0058333]
Iteración 3: ...  x = [2.0072917, -0.9985417, 3.0002917]
Iteración 4: ...  x = [2.0003646, -0.9999271, 3.0000146]
Iteración 5: ...  x = [2.0000182, -0.9999964, 3.0000007]
Iteración 6: ...  x = [2.0000009, -0.9999998, 3.0000000]
Iteración 7: ...  x = [2.0000000, -1.0000000, 3.0000000]
Iteración 8: ...  x = [2.0000000, -1.0000000, 3.0000000]
Iteración 9: ...  x = [2.0000000, -1.0000000, 3.0000000]

||x^(k) - x^(k-1)||_inf = 2.164713e-09
Convergencia alcanzada con tol=1e-08.
```



```
======================================================================
¿Cómo quieres introducir A?
======================================================================
1) Pegado rápido (bloque de texto)
2) Paso a paso (fila por fila)
3) Generar ejemplo aleatorio
Elige una opción (número): 1

Pega la matriz/vector en bloque.
- A debe ser de tamaño 3x3.
- Una fila por línea (espacios o comas). Puedes usar ';' para separar filas.
- También puedes pegar formato MATLAB con corchetes: [a11 a12; a21 a22].
Finaliza con una línea vacía (ENTER dos veces).
[[4, -1, 0];
 [-2, 6, 1];
 [0, -1, 5]]


======================================================================
¿Cómo quieres introducir b?
======================================================================
1) Pegado rápido (bloque de texto)
2) Paso a paso (fila por fila)
3) Generar ejemplo aleatorio
Elige una opción (número): 1

Pega la matriz/vector en bloque.
- b debe ser de tamaño 3x1.
- Una fila por línea (espacios o comas). Puedes usar ';' para separar filas.
- También puedes pegar formato MATLAB con corchetes: [a11 a12; a21 a22].
Finaliza con una línea vacía (ENTER dos veces).
[[9];
 [-7];
 [16]]
```

```
Tolerancia (p.ej. 1e-8) [1e-8]: 1e-8
Máximo de iteraciones [1000]: 50
¿Deseas ingresar x0? (s/n) [n]: n

== Gauss-Seidel: inicio ==

--- A ---
[[ 4. -1.  0.]
 [-2.  6.  1.]
 [ 0. -1.  5.]]
--- fin A ---


--- b ---
[[ 9.]
 [-7.]
 [16.]]
--- fin b ---


--- x^(0) ---
[[0.]
 [0.]
 [0.]]
--- fin x^(0) ---


Iteración 1:
  x_1 = (b[1] - sum(A[1,<i]*x_nuevos) - sum(A[1,>i]*x_prev)) / A[1,1]
       = (9 - 0 - 0) / 4 = 2.25
  x_2 = (b[2] - sum(A[2,<i]*x_nuevos) - sum(A[2,>i]*x_prev)) / A[2,2]
       = (-7 - -4.5 - 0) / 6 = -0.416666667
  x_3 = (b[3] - sum(A[3,<i]*x_nuevos) - sum(A[3,>i]*x_prev)) / A[3,3]
       = (16 - 0.416667 - 0) / 5 = 3.11666667

--- x^(1) ---
[[ 2.25    ]
 [-0.416667]
 [ 3.116667]]
--- fin x^(1) ---
```

```
Iteración 9:
  x_1 = (b[1] - sum(A[1,<i]*x_nuevos) - sum(A[1,>i]*x_prev)) / A[1,1]
       = (9 - 0 - 1) / 4 = 2
  x_2 = (b[2] - sum(A[2,<i]*x_nuevos) - sum(A[2,>i]*x_prev)) / A[2,2]
       = (-7 - -4 - 3) / 6 = -1
  x_3 = (b[3] - sum(A[3,<i]*x_nuevos) - sum(A[3,>i]*x_prev)) / A[3,3]
       = (16 - 1 - 0) / 5 = 3

--- x^(9) ---
[[ 2.]
 [-1.]
 [ 3.]]
--- fin x^(9) ---

  ||x^(k) - x^(k-1)||_inf = 2.164713e-09
Convergencia alcanzada con tol=1e-08.

======================================================================
RESULTADO - Método: Gauss-Seidel
======================================================================
x =
 [ 2. -1.  3.]
||Ax - b||_2 = 4.415e-10
Iteraciones: 9
======================================================================
```

(a) Inicio: entrada de A, b, parámetros y $x^{(0)}$.

(b) Iteración 1: actualización de x_1, x_2, x_3.

(c) Iteración 9 y convergencia con tol$=10^{-8}$.

Figura 3: Ejecución del método de Gauss–Seidel. Se muestran la configuración inicial, la primera iteración y la iteración final donde se alcanza la convergencia.

Nota: por brevedad no se incluyen todas las capturas de cada iteración, ya que serían numerosas; aquí se presentan solo los momentos clave (inicio, primera iteración y convergencia final).

5. Regla de Cramer

Idea. Si $\det(A) \neq 0$, entonces $x_i = \dfrac{\det(A_i)}{\det(A)}$, con A_i igual a A reemplazando su columna i por b.

Determinante de A.

$$\det(A) = 4\begin{vmatrix} 6 & 1 \\ -1 & 5 \end{vmatrix} - (-1)\begin{vmatrix} -2 & 1 \\ 0 & 5 \end{vmatrix} = 4(30+1) - (+1)(-10) = \boxed{114}.$$

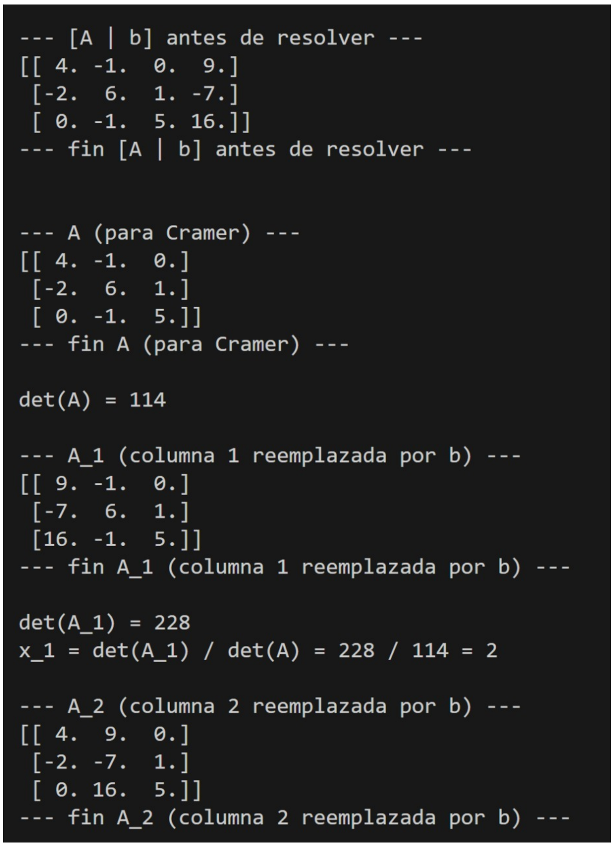
A_1 y $\det(A_1)$: $A_1 = \begin{bmatrix} 9 & -1 & 0 \\ -7 & 6 & 1 \\ 16 & -1 & 5 \end{bmatrix}$, $\det(A_1) = 9(31) - (-1)(-51) = 279 - 51 = \boxed{228} \Rightarrow$ $x_1 = 228/114 = \boxed{2}$.

A_2 y $\det(A_2)$: $A_2 = \begin{bmatrix} 4 & 9 & 0 \\ -2 & -7 & 1 \\ 0 & 16 & 5 \end{bmatrix}$, $\det(A_2) = 4(-51) - 9(-10) = -204 + 90 = \boxed{-114} \Rightarrow$ $x_2 = -114/114 = \boxed{-1}$.

A_3 y $\det(A_3)$: $A_3 = \begin{bmatrix} 4 & -1 & 9 \\ -2 & 6 & -7 \\ 0 & -1 & 16 \end{bmatrix}$, $\det(A_3) = 4(89) - (-1)(-32) + 9(2) = 356 - 32 + 18 =$ $\boxed{342} \Rightarrow x_3 = 342/114 = \boxed{3}$.

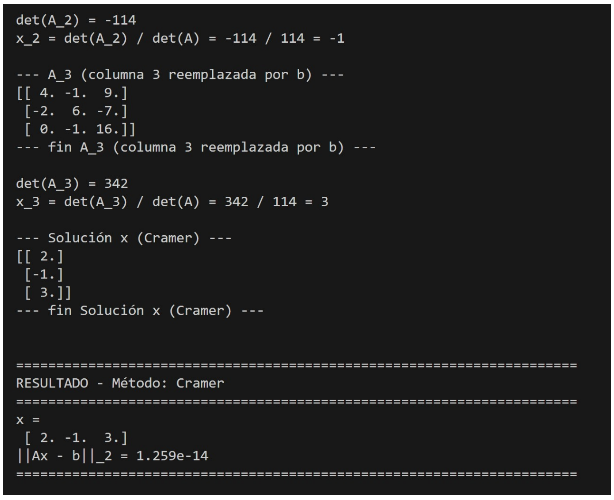
Traza típica en consola (modo detallado):

```
A (para Cramer)
det(A) = 114
A_1 (columna 1 <- b), det(A_1) = 228  => x_1 = 2
A_2 (columna 2 <- b), det(A_2) = -114 => x_2 = -1
A_3 (columna 3 <- b), det(A_3) = 342  => x_3 = 3
x = [2, -1, 3]^T,   ||Ax-b||_2 = 0.0
```



```
--- [A | b] antes de resolver ---
[[ 4. -1.  0.  9.]
 [-2.  6.  1. -7.]
 [ 0. -1.  5. 16.]]
--- fin [A | b] antes de resolver ---


--- A (para Cramer) ---
[[ 4. -1.  0.]
 [-2.  6.  1.]
 [ 0. -1.  5.]]
--- fin A (para Cramer) ---

det(A) = 114

--- A_1 (columna 1 reemplazada por b) ---
[[ 9. -1.  0.]
 [-7.  6.  1.]
 [16. -1.  5.]]
--- fin A_1 (columna 1 reemplazada por b) ---

det(A_1) = 228
x_1 = det(A_1) / det(A) = 228 / 114 = 2

--- A_2 (columna 2 reemplazada por b) ---
[[ 4.  9.  0.]
 [-2. -7.  1.]
 [ 0. 16.  5.]]
--- fin A_2 (columna 2 reemplazada por b) ---
```

(a) Previo y primer reemplazo: vista de $[A\,|\,b]$ antes de resolver, A para Cramer, $\det(A) = 114$; construcción de A_1 (col. $1 \leftarrow b$), $\det(A_1) = 228$ y cociente $x_1 = \det(A_1)/\det(A) = 2$.



```
det(A_2) = -114
x_2 = det(A_2) / det(A) = -114 / 114 = -1

--- A_3 (columna 3 reemplazada por b) ---
[[ 4. -1.  9.]
 [-2.  6. -7.]
 [ 0. -1. 16.]]
--- fin A_3 (columna 3 reemplazada por b) ---

det(A_3) = 342
x_3 = det(A_3) / det(A) = 342 / 114 = 3

--- Solución x (Cramer) ---
[[ 2.]
 [-1.]
 [ 3.]]
--- fin Solución x (Cramer) ---


======================================================================
RESULTADO - Método: Cramer
======================================================================
x =
 [ 2. -1.  3.]
||Ax - b||_2 = 1.259e-14
======================================================================
```

(b) Siguientes reemplazos y cierre: A_2 (col. $2 \leftarrow b$) con $\det(A_2) = -114 \Rightarrow x_2 = -1$; A_3 (col. $3 \leftarrow b$) con $\det(A_3) = 342 \Rightarrow x_3 = 3$; vector solución $x = [2, -1, 3]^T$ y residuo $\|Ax - b\|_2$.

Figura 4: Regla de Cramer: reemplazo de columnas por b, cálculo de determinantes y obtención de x.

6. Conclusiones breves

Los tres métodos recuperan $x = (2, -1, 3)$. Gauss–Jordan es directo y sistemático; Cramer es didáctico en tamaños pequeños pero costoso para n grande; Gauss–Seidel converge en pocas iteraciones gracias a la dominancia diagonal de A. Las *trazas* impresas por el script permiten *auditar* cada paso del proceso y alinean el desarrollo manual con la ejecución computacional.