

# Resolución de Ejercicios - Método Gauss-Jordan

Eddy Kennedy Mamani Hallasi

17 de enero de 2025

## Introducción

En este informe se resuelven cinco ejercicios de sistemas de ecuaciones lineales utilizando el método de Gauss-Jordan. Para cada ejercicio, se presenta el concepto, la matriz aumentada inicial, el paso a paso con las matrices intermedias, y el resultado final.

## Repositorio en GitHub

El código Python utilizado para realizar estos cálculos, junto con las librerías empleadas, está disponible en el siguiente enlace: <https://github.com/MetodosDeOptimizacion/GaussJordan.git>

## Ejercicio 1

**Concepto:** Este ejercicio resuelve un sistema de ecuaciones lineales para determinar los valores de los pesos de un modelo de regresión lineal.

**Sistema de ecuaciones:**

$$\begin{aligned}2w_1 + 3w_2 - w_3 &= 5, \\ -w_1 + 2w_2 + 4w_3 &= 6, \\ 3w_1 - w_2 + 2w_3 &= 7.\end{aligned}$$

**Matriz aumentada inicial:**

$$\left[ \begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ -1 & 2 & 4 & 6 \\ 3 & -1 & 2 & 7 \end{array} \right]$$

**Paso a paso:**

1. Dividir la primera fila entre 2 para hacer el pivote en (1, 1) igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 1,5 & -0,5 & 2,5 \\ -1 & 2 & 4 & 6 \\ 3 & -1 & 2 & 7 \end{array} \right]$$

2. Sustraer  $-1$  veces la primera fila de la segunda para hacer 0 en (2, 1):

$$\left[ \begin{array}{ccc|c} 1 & 1,5 & -0,5 & 2,5 \\ 0 & 3,5 & 3,5 & 8,5 \\ 3 & -1 & 2 & 7 \end{array} \right]$$

3. Sustraer 3 veces la primera fila de la tercera para hacer 0 en (3,1):

$$\left[ \begin{array}{ccc|c} 1 & 1,5 & -0,5 & 2,5 \\ 0 & 3,5 & 3,5 & 8,5 \\ 0 & -5,5 & 3,5 & -0,5 \end{array} \right]$$

4. Dividir la segunda fila entre 3.5 para hacer el pivote (2,2) igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 1,5 & -0,5 & 2,5 \\ 0 & 1 & 1 & 2,429 \\ 0 & -5,5 & 3,5 & -0,5 \end{array} \right]$$

5. Eliminar los valores en (1,2) y (3,2) usando la segunda fila:

$$\left[ \begin{array}{ccc|c} 1 & 0 & -2 & 0,857 \\ 0 & 1 & 1 & 2,429 \\ 0 & 0 & 9 & 12,857 \end{array} \right]$$

6. Dividir la tercera fila para hacer el pivote (3,3) igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 0 & -2 & 0,857 \\ 0 & 1 & 1 & 2,429 \\ 0 & 0 & 1 & 1,429 \end{array} \right]$$

7. Eliminar los valores en (1,3) y (2,3) usando la tercera fila:

$$\left[ \begin{array}{ccc|c} 1 & 0 & 0 & 1,714 \\ 0 & 1 & 0 & 1,000 \\ 0 & 0 & 1 & 1,429 \end{array} \right]$$

## Soluciones:

- $w_1 = 1,714$
- $w_2 = 1,000$
- $w_3 = 1,429$

```

1 import numpy as np
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4
5 def gauss_jordan_paso_a_paso(aumentada):
6     pasos = []
7     filas, columnas = aumentada.shape
8
9     for i in range(filas):
10        pasos.append(f"Paso {i+1}: Hacer el pivote en la fila {i+1} igual a 1")
11        pivote = aumentada[i, i]
12        if pivote != 0:
13            aumentada[i] = aumentada[i] / pivote
14            pasos.append(aumentada.copy())
15
16        pasos.append(f"Hacer ceros en la columna {i+1} para las demás filas:")
17        for j in range(filas):
18            if j != i:
19                factor = aumentada[j, i]
20                aumentada[j] = aumentada[j] - factor * aumentada[i]
21                pasos.append(f"Fila {j+1} - ({factor}) * Fila {i+1}")
22                pasos.append(aumentada.copy())
23
24        pasos.append("Matriz reducida en forma escalonada:")
25        pasos.append(aumentada.copy())
26
27    soluciones = []
28    for i in range(len(aumentada)):
29        soluciones.append(f"x{i+1} = {aumentada[i, -1]}")
30
31    return pasos, soluciones
32
33 def mostrar_resultados(pasos, soluciones, inicial):
34     resultados = tk.Toplevel()
35     resultados.title("Resultados del Método Gauss-Jordan")

```

**Resultados del Método Gauss-Jordan**

Fila 1 - (1.5) \* Fila 2

[[ 1.	0.	-2.	-1.14285714]
[ 0.	1.	1.	2.42857143]
[ 0.	-5.5	3.5	-0.5 ]]

Fila 3 - (-5.5) \* Fila 2

[[ 1.	0.	-2.	-1.14285714]
[ 0.	1.	1.	2.42857143]
[ 0.	0.	9.	12.85714286]]

Paso 3: Hacer el pivote en la fila 3 igual a 1

[[ 1.	0.	-2.	-1.14285714]
[ 0.	1.	1.	2.42857143]
[ 0.	0.	1.	1.42857143]]

Hacer ceros en la columna 3 para las demás filas:

Fila 1 - (-2.0) \* Fila 3

[[1.	0.	0.	1.71428571]
[0.	1.	1.	2.42857143]
[0.	0.	1.	1.42857143]]

Fila 2 - (1.0) \* Fila 3

[[1.	0.	0.	1.71428571]
[0.	1.	0.	1.]
[0.	0.	1.	1.42857143]]

Matriz reducida en forma escalonada:

[[1.	0.	0.	1.71428571]
[0.	1.	0.	1.]
[0.	0.	1.	1.42857143]]

Soluciones:

x1 = 1.7142857142857144  
x2 = 1.0  
x3 = 1.4285714285714284

## Ejercicio 2

**Concepto:** Resolver un sistema de ecuaciones para calibrar hiperparámetros en un modelo matemático.

**Sistema de ecuaciones:**

$$\begin{aligned}x + 2y + 3z &= 12, \\2x - y + z &= 4, \\-x + 2y - 2z &= 0.\end{aligned}$$

**Matriz aumentada inicial:**

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 2 & -1 & 1 & 4 \\ -1 & 2 & -2 & 0 \end{array} \right]$$

**Paso a paso:**

1. Dejar el pivote  $(1, 1)$  como 1 (ya está).
2. Sustraer 2 veces la primera fila de la segunda para hacer 0 en  $(2, 1)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 0 & -5 & -5 & -20 \\ -1 & 2 & -2 & 0 \end{array} \right]$$

3. Sumar la primera fila a la tercera para hacer 0 en  $(3, 1)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 0 & -5 & -5 & -20 \\ 0 & 4 & 1 & 12 \end{array} \right]$$

4. Dividir la segunda fila entre  $-5$  para hacer el pivote  $(2, 2)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 0 & 1 & 1 & 4 \\ 0 & 4 & 1 & 12 \end{array} \right]$$

5. Sustraer 4 veces la segunda fila de la tercera para hacer 0 en  $(3, 2)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & -3 & -4 \end{array} \right]$$

6. Dividir la tercera fila entre  $-3$  para hacer el pivote  $(3, 3)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 12 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 1 & 1,333 \end{array} \right]$$

7. Eliminar los valores en (1, 3) y (2, 3) usando la tercera fila:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 0 & 8 \\ 0 & 1 & 0 & 2,667 \\ 0 & 0 & 1 & 1,333 \end{array} \right]$$

**Soluciones:**

- $x = 2,667$
- $y = 2,667$
- $z = 1,333$

```

Actividad04 > GaussJordan.py > ...
1 import numpy as np
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4
5 def gauss_jordan_paso_a_paso(aumentada):
6     pasos = []
7     filas, columnas = aumentada.shape
8
9     for i in range(filas):
10        pasos.append(f"Paso {i+1}: Hacer el pivote en la fila {i+1} igual a 1")
11        pivote = aumentada[i, i]
12        if pivote != 0:
13            aumentada[i] = aumentada[i] / pivote
14            pasos.append(aumentada.copy())
15
16        pasos.append(f"Hacer ceros en la columna {i+1} para las demás filas:")
17        for j in range(filas):
18            if j != i:
19                factor = aumentada[j, i]
20                aumentada[j] = aumentada[j] - factor * aumentada[i]
21                pasos.append(f"Fila {j+1} - ({factor}) * Fila {i+1}")
22                pasos.append(aumentada.copy())
23
24        pasos.append("Matriz reducida en forma escalonada:")
25        pasos.append(aumentada.copy())
26
27        soluciones = []
28        for i in range(len(aumentada)):
29            soluciones.append(f"x{i+1} = {aumentada[i, -1]}")
30
31        return pasos, soluciones
32
33 def mostrar_resultados(pasos, soluciones, inicial):
34     resultados = tk.Toplevel()
35     resultados.title("Resultados del Método Gauss-Jordan")

```

**Resultados del Método Gauss-Jordan**

```

[ [ 0.  4.  1. 12.] ]
Hacer ceros en la columna 2 para las demás filas:
Fila 1 - (2.0) * Fila 2
[[ [ 1.  0.  1.  4.]
   [-0.  1.  1.  4.]
   [ 0.  4.  1. 12.] ]
Fila 3 - (4.0) * Fila 2
[[ [ 1.  0.  1.  4.]
   [-0.  1.  1.  4.]
   [ 0.  0. -3. -4.] ]
Paso 3: Hacer el pivote en la fila 3 igual a 1
[[ [ 1.  0.  1.  4.]
   [-0.  1.  1.  4.]
   [-0.  1.  1.  4.] ]
Hacer ceros en la columna 3 para las demás filas:
Fila 1 - (1.0) * Fila 3
[[ [ 1.  0.  0.  2.66666667]
   [-0.  1.  1.  4.]
   [-0.  1.  1.  4.] ]
Fila 2 - (1.0) * Fila 3
[[ [ 1.  0.  0.  2.66666667]
   [ 0.  1.  0.  2.66666667]
   [-0.  0.  1.  1.33333333] ]
Matriz reducida en forma escalonada:
[[ [ 1.  0.  0.  2.66666667]
   [ 0.  1.  0.  2.66666667]
   [-0.  0.  1.  1.33333333] ]
Soluciones:
x1 = 2.666666666666667
x2 = 2.666666666666667
x3 = 1.3333333333333333

```

## Ejercicio 3

**Concepto:** Este sistema busca la asignación óptima de recursos entre módulos.  
**Sistema de ecuaciones:**

$$\begin{aligned} a + b + c &= 6, \\ 2a - b + 3c &= 13, \\ -a + 2b - c &= 2. \end{aligned}$$

**Matriz aumentada inicial:**

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 2 & -1 & 3 & 13 \\ -1 & 2 & -1 & 2 \end{array} \right]$$

**Paso a paso:**

1. Dejar el pivote (1, 1) como 1 (ya está).
2. Sustraer 2 veces la primera fila de la segunda para hacer 0 en (2, 1):

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & -3 & 1 & 1 \\ -1 & 2 & -1 & 2 \end{array} \right]$$

3. Sumar la primera fila a la tercera para hacer 0 en (3,1):

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & -3 & 1 & 1 \\ 0 & 3 & 0 & 8 \end{array} \right]$$

4. Dividir la segunda fila entre  $-3$  para hacer el pivote (2,2) igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & -0,333 & -0,333 \\ 0 & 3 & 0 & 8 \end{array} \right]$$

5. Sustraer 3 veces la segunda fila de la tercera para hacer 0 en (3,2):

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & -0,333 & -0,333 \\ 0 & 0 & 1 & 9 \end{array} \right]$$

6. Eliminar los valores en (1,2) y (1,3) usando las filas correspondientes:

$$\left[ \begin{array}{ccc|c} 1 & 0 & 0 & -5,667 \\ 0 & 1 & 0 & 2,667 \\ 0 & 0 & 1 & 9 \end{array} \right]$$

## Soluciones:

- $a = -5,667$
- $b = 2,667$
- $c = 9,000$

The image shows a Python script on the left and its output on the right. The script implements the Gauss-Jordan elimination method for a 3x4 augmented matrix. It uses NumPy for matrix operations and Tkinter for a simple GUI to display the results. The output window, titled 'Resultados del Método Gauss-Jordan', shows the step-by-step row operations and the final reduced row echelon form (RREF) of the matrix. The final solution is displayed as x1 = -5.666666666666667, x2 = 2.666666666666667, and x3 = 9.0.

```

Actividad04 > GaussJordan.py > gauss_jordan_paso_a_paso
1 import numpy as np
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4
5 def gauss_jordan_paso_a_paso(aumentada):
6     pasos = []
7     filas, columnas = aumentada.shape
8
9     for i in range(filas):
10        pasos.append(f"Paso {i+1}: Hacer el pivote en la fila {i+1} igual a 1")
11        pivote = aumentada[i, i]
12        if pivote != 0:
13            aumentada[i] = aumentada[i] / pivote
14            pasos.append(aumentada.copy())
15
16        pasos.append(f"Hacer ceros en la columna {i+1} para las demás filas:")
17        for j in range(filas):
18            if j != i:
19                factor = aumentada[j, i]
20                aumentada[j] = aumentada[j] - factor * aumentada[i]
21                pasos.append(f"Fila {j+1} - ({factor}) * Fila {i+1}")
22                pasos.append(aumentada.copy())
23
24        pasos.append("Matriz reducida en forma escalonada:")
25        pasos.append(aumentada.copy())
26
27        soluciones = []
28        for i in range(len(aumentada)):
29            soluciones.append(f"x{i+1} = {aumentada[i, -1]}")
30
31        return pasos, soluciones
32
33 def mostrar_resultados(pasos, soluciones, inicial):
34     resultados = tk.Toplevel()
35     resultados.title("Resultados del Método Gauss-Jordan")

```

**Resultados del Método Gauss-Jordan**

```

===== en forma escalonada =====
Fila 1 - (1.0) * Fila 2
[[ 1.  0.  1.33333333  6.33333333]
 [-0.  1. -0.33333333 -0.33333333]
 [ 0.  3.  0.  8. ]]
Fila 3 - (3.0) * Fila 2
[[ 1.  0.  1.33333333  6.33333333]
 [-0.  1. -0.33333333 -0.33333333]
 [ 0.  0.  1.  9. ]]
Paso 3: Hacer el pivote en la fila 3 igual a 1
[[ 1.  0.  1.33333333  6.33333333]
 [-0.  1. -0.33333333 -0.33333333]
 [ 0.  0.  1.  9. ]]
Hacer ceros en la columna 3 para las demás filas:
Fila 1 - (1.3333333333333333) * Fila 3
[[ 1.  0.  0. -5.66666667]
 [-0.  1. -0.33333333 -0.33333333]
 [ 0.  0.  1.  9. ]]
Fila 2 - (-0.3333333333333333) * Fila 3
[[ 1.  0.  0. -5.66666667]
 [ 0.  1.  0.  2.66666667]
 [ 0.  0.  1.  9. ]]
Matriz reducida en forma escalonada:
[[ 1.  0.  0. -5.66666667]
 [ 0.  1.  0.  2.66666667]
 [ 0.  0.  1.  9. ]]

Soluciones:
x1 = -5.666666666666667
x2 = 2.666666666666667
x3 = 9.0

```

## Ejercicio 4

**Concepto:** Este ejercicio optimiza los parámetros en un modelo matemático.

**Sistema de ecuaciones:**

$$p + 2q + 3r = 10,$$

$$2p - q + 4r = 12,$$

$$3p + 3q - r = 6.$$

**Matriz aumentada inicial:**

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 2 & -1 & 4 & 12 \\ 3 & 3 & -1 & 6 \end{array} \right]$$

**Paso a paso:**

1. Dejar el pivote  $(1, 1)$  como 1 (ya está).
2. Sustraer 2 veces la primera fila de la segunda para hacer 0 en  $(2, 1)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0 & -5 & -2 & -8 \\ 3 & 3 & -1 & 6 \end{array} \right]$$

3. Sustraer 3 veces la primera fila de la tercera para hacer 0 en  $(3, 1)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0 & -5 & -2 & -8 \\ 0 & -3 & -10 & -24 \end{array} \right]$$

4. Dividir la segunda fila entre  $-5$  para hacer el pivote  $(2, 2)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0 & 1 & 0,4 & 1,6 \\ 0 & -3 & -10 & -24 \end{array} \right]$$

5. Sustraer  $-3$  veces la segunda fila de la tercera para hacer 0 en  $(3, 2)$ :

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0 & 1 & 0,4 & 1,6 \\ 0 & 0 & -8,8 & -19,2 \end{array} \right]$$

6. Dividir la tercera fila entre  $-8,8$  para hacer el pivote  $(3, 3)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0 & 1 & 0,4 & 1,6 \\ 0 & 0 & 1 & 2,182 \end{array} \right]$$

7. Eliminar los valores en  $(1, 3)$  y  $(2, 3)$  usando la tercera fila:

$$\left[ \begin{array}{ccc|c} 1 & 2 & 0 & 3,454 \\ 0 & 1 & 0 & 0,727 \\ 0 & 0 & 1 & 2,182 \end{array} \right]$$

## Soluciones:

- $p = 3,454$
- $q = 0,727$
- $r = 2,182$

The image shows a Python script named 'GaussJordan.py' and its output. The script implements the Gauss-Jordan elimination method for a system of linear equations. It takes an augmented matrix as input and returns the steps of the process and the final solution.

```

1 import numpy as np
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4
5 def gauss_jordan_paso_a_paso(aumentada):
6     pasos = []
7     filas, columnas = aumentada.shape
8
9     for i in range(filas):
10        pasos.append(f"Paso {i+1}: Hacer el pivote en la fila {i+1} igual a 1")
11        pivote = aumentada[i, i]
12        if pivote != 0:
13            aumentada[i] = aumentada[i] / pivote
14            pasos.append(aumentada.copy())
15
16        pasos.append(f"Hacer ceros en la columna {i+1} para las demás filas:")
17        for j in range(filas):
18            if j != i:
19                factor = aumentada[j, i]
20                aumentada[j] = aumentada[j] - factor * aumentada[i]
21                pasos.append(f"Fila {j+1} - ({factor}) * Fila {i+1}")
22                pasos.append(aumentada.copy())
23
24    pasos.append("Matriz reducida en forma escalonada:")
25    pasos.append(aumentada.copy())
26
27    soluciones = []
28    for i in range(len(aumentada)):
29        soluciones.append(f"x{i+1} = {aumentada[i, -1]}")
30
31    return pasos, soluciones
32
33 def mostrar_resultados(pasos, soluciones, inicial):
34     resultados = tk.Toplevel()
35     resultados.title("Resultados del Método Gauss-Jordan")

```

The output window, titled "Resultados del Método Gauss-Jordan", displays the following results:

```

Paso 1: Hacer el pivote en la fila 1 igual a 1
[[ 1.  2.  3. 10.]
 [-0.  1.  0.4 1.6]
 [ 0. -3. -10. -24.]]
Hacer ceros en la columna 2 para las demás filas:
Fila 1 - (2.0) * Fila 2
[[ 1.  0.  2.2 6.8]
 [-0.  1.  0.4 1.6]
 [ 0. -3. -10. -24.]]
Fila 3 - (-3.0) * Fila 2
[[ 1.  0.  2.2 6.8]
 [-0.  1.  0.4 1.6]
 [ 0.  0. -8.8 -19.2]]
Paso 3: Hacer el pivote en la fila 3 igual a 1
[[ 1.  0.  2.2 6.8]
 [-0.  1.  0.4 1.6]
 [-0.  0. -1. 2.18181818]]
Hacer ceros en la columna 3 para las demás filas:
Fila 1 - (2.2) * Fila 3
[[ 1.  0.  0. 2. ]
 [-0.  1.  0.4 1.6]
 [-0.  0. -1. 2.18181818]]
Fila 2 - (0.4) * Fila 3
[[ 1.  0.  0. 2. ]
 [-0.  1.  0. 0.72727273]
 [-0.  0. -1. 2.18181818]]
Matriz reducida en forma escalonada:
[[ 1.  0.  0. 2. ]
 [ 0.  1.  0. 0.72727273]
 [-0.  0.  1. 2.18181818]]
Soluciones:
x1 = 2.0
x2 = 0.7272727272727273
x3 = 2.1818181818181818

```

## Ejercicio 5

**Concepto:** Este ejercicio estima la demanda de inventario utilizando un modelo lineal.  
**Sistema de ecuaciones:**

$$\begin{aligned} u + v + 2w &= 9, \\ 2u - 3v + 4w &= 5, \\ u - 2v + w &= 1. \end{aligned}$$

**Matriz aumentada inicial:**

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 2 & -3 & 4 & 5 \\ 1 & -2 & 1 & 1 \end{array} \right]$$

**Paso a paso:**

1. Dejar el pivote (1,1) como 1 (ya está).
2. Sustraer 2 veces la primera fila de la segunda para hacer 0 en (2,1):

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & -5 & 0 & -13 \\ 1 & -2 & 1 & 1 \end{array} \right]$$

3. Sustraer la primera fila de la tercera para hacer 0 en (3,1):

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & -5 & 0 & -13 \\ 0 & -3 & -1 & -8 \end{array} \right]$$

4. Dividir la segunda fila entre  $-5$  para hacer el pivote  $(2, 2)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & 1 & 0 & 2,6 \\ 0 & -3 & -1 & -8 \end{array} \right]$$

5. Sumar 3 veces la segunda fila a la tercera para hacer 0 en  $(3, 2)$ :

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & 1 & 0 & 2,6 \\ 0 & 0 & -1 & 0,2 \end{array} \right]$$

6. Dividir la tercera fila entre  $-1$  para hacer el pivote  $(3, 3)$  igual a 1:

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & 1 & 0 & 2,6 \\ 0 & 0 & 1 & 0,2 \end{array} \right]$$

7. Eliminar los valores en  $(1, 3)$  y  $(2, 3)$  usando la tercera fila:

$$\left[ \begin{array}{ccc|c} 1 & 1 & 0 & 8,6 \\ 0 & 1 & 0 & 2,6 \\ 0 & 0 & 1 & 0,2 \end{array} \right]$$

## Soluciones:

- $u = 8,6$
- $v = 2,6$
- $w = 0,2$

The image shows a Python script in a code editor and its output in a terminal window. The script implements the Gauss-Jordan method for solving a system of linear equations. It starts with an augmented matrix and performs row operations to reach row echelon form. The output displays the steps of the algorithm, including pivoting and row reduction, and finally shows the solutions for the variables x1, x2, and x3.

```

Restricciones.py GaussJordan.py x
Actividad04 > GaussJordan.py > ...
1 import numpy as np
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4
5 def gauss_jordan_paso_a_paso(aumentada):
6     pasos = []
7     filas, columnas = aumentada.shape
8
9     for i in range(filas):
10        pasos.append(f"Paso {i+1}: Hacer el pivote en la fila {i+1} igual a 1")
11        pivote = aumentada[i, i]
12        if pivote != 0:
13            aumentada[i] = aumentada[i] / pivote
14            pasos.append(aumentada.copy())
15
16        pasos.append(f"Hacer ceros en la columna {i+1} para las demás filas:")
17        for j in range(filas):
18            if j != i:
19                factor = aumentada[j, i]
20                aumentada[j] = aumentada[j] - factor * aumentada[i]
21                pasos.append(f"File {j+1} - ({factor}) * File {i+1}")
22                pasos.append(aumentada.copy())
23
24        pasos.append("Matriz reducida en forma escalonada:")
25        pasos.append(aumentada.copy())
26
27        soluciones = []
28        for i in range(len(aumentada)):
29            soluciones.append(f"x{i+1} = {aumentada[i, -1]}")
30
31        return pasos, soluciones
32
33 def mostrar_resultados(pasos, soluciones, inicial):
34     resultados = tk.Toplevel()
35     resultados.title("Resultados del Método Gauss-Jordan")

```

**Resultados del Método Gauss-Jordan**

```

[[ 1.  1.  2.  9.]
 [-0.  1. -0.  2.6]
 [ 0. -3. -1. -8.]]
Hacer ceros en la columna 2 para las demás filas:
File 1 - (1.0) * File 2
[[ 1.  0.  2.  6.4]
 [-0.  1. -0.  2.6]
 [ 0. -3. -1. -8.]]
File 3 - (-3.0) * File 2
[[ 1.  0.  2.  6.4]
 [-0.  1. -0.  2.6]
 [ 0.  0. -1. -0.2]]
Paso 3: Hacer el pivote en la fila 3 igual a 1
[[ 1.  0.  2.  6.4]
 [-0.  1. -0.  2.6]
 [-0. -0.  1.  0.2]]
Hacer ceros en la columna 3 para las demás filas:
File 1 - (2.0) * File 3
[[ 1.  0.  0.  6.]
 [-0.  1. -0.  2.6]
 [-0. -0.  1.  0.2]]
File 2 - (-0.0) * File 3
[[ 1.  0.  0.  6.]
 [-0.  1.  0.  2.6]
 [-0. -0.  1.  0.2]]
Matriz reducida en forma escalonada:
[[ 1.  0.  0.  6.]
 [-0.  1.  0.  2.6]
 [-0. -0.  1.  0.2]]
Soluciones:
x1 = 6.0000000000000002
x2 = 2.6
x3 = 0.19999999999999993

```

## Conclusión

El método de Gauss-Jordan fue aplicado para resolver los sistemas de ecuaciones de manera detallada, mostrando paso a paso las transformaciones en las matrices aumentadas. Este enfoque garantiza claridad en la comprensión de cada proceso algebraico.