

Taller 1b

David Alejandro Díaz Pineda

Eliminación Gaussiana con sustitución hacia atrás

1. El bug encontrado en el código era que el algoritmo de eliminación gaussiana opera con floats y había un error entre tipos de datos, por lo que el primer coeficiente tuvo que ser convertido de entero a flotante
2. Complejidad Gauss y Gauss-Jordan
Sumas y Restas

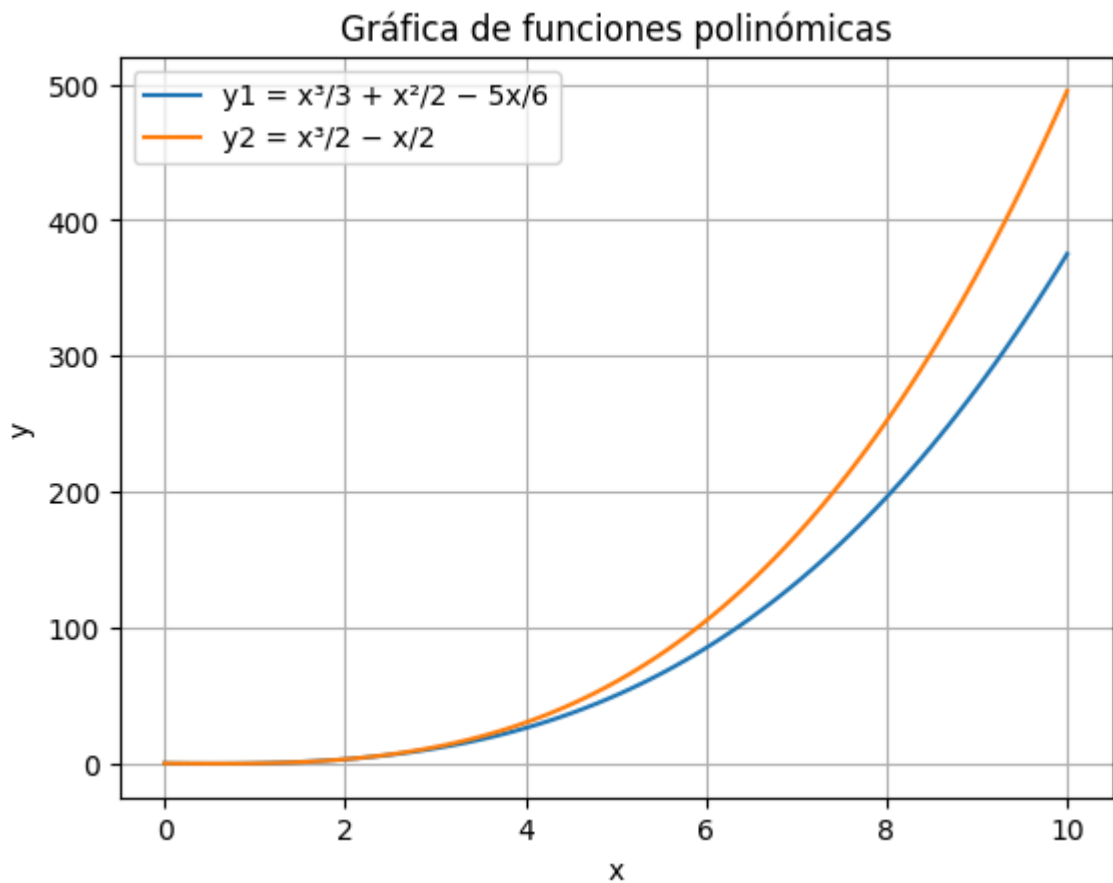
```
In [1]: import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)

y1 = (x**3)/3 + (x**2)/2 - (5*x)/6
y2 = (x**3)/2 - x/2

plt.plot(x, y1, label='y1 = x³/3 + x²/2 - 5x/6')
plt.plot(x, y2, label='y2 = x³/2 - x/2')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráfica de funciones polinómicas')
plt.legend()
plt.grid(True)
plt.show()
```



Divisiones y Multiplicaciones

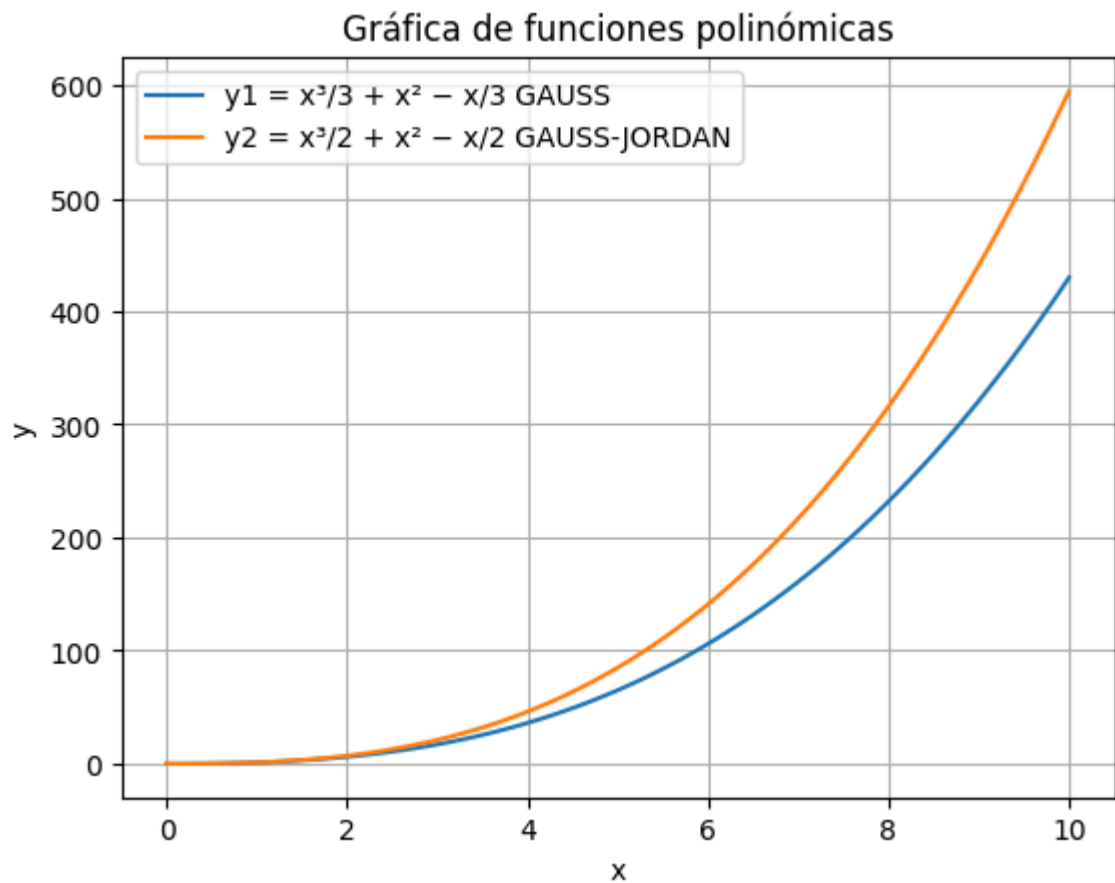
```
In [2]: import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)

y1 = (x**3)/3 + (x**2) - (x)/3
y2 = (x**3)/2 + (x**2) - x/2

plt.plot(x, y1, label='y1 = x³/3 + x² - x/3 GAUSS')
plt.plot(x, y2, label='y2 = x³/2 + x² - x/2 GAUSS-JORDAN')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráfica de funciones polinómicas')
plt.legend()
plt.grid(True)
plt.show()
```



In [3]: `%load_ext autoreload`

```
In [4]: %autoreload 2
from src import eliminacion_gaussiana

Ab = [[1., 1, 0, 3, 4], [2, 1, -1, 1, 1], [3, -1, -1, 2, -3], [-1, 2, 3, -1, 4]]

sol, suma_count, resta_count, mult_count, div_count, intercambio_count = elimina
print('Solución:', sol)
print('Sumas realizadas:', suma_count)
print('Restas realizadas:', resta_count)
print('Multiplicaciones realizadas:', mult_count)
print('Divisiones realizadas:', div_count)
print('Intercambios de filas:', intercambio_count)
```

```
[12-25 13:20:08][INFO] 2025-12-25 13:20:08.596701
[12-25 13:20:08][INFO]
[[ 1.  1.  0.  3.  4.]
 [ 0. -1. -1. -5. -7.]
 [ 0. -4. -1. -7. -15.]
 [ 0.  3.  3.  2.  8.]]
[12-25 13:20:08][INFO]
[[ 1.  1.  0.  3.  4.]
 [ 0. -1. -1. -5. -7.]
 [ 0.  0.  3. 13. 13.]
 [ 0.  0.  0. -13. -13.]]
[12-25 13:20:08][INFO]
[[ 1.  1.  0.  3.  4.]
 [ 0. -1. -1. -5. -7.]
 [ 0.  0.  3. 13. 13.]
 [ 0.  0.  0. -13. -13.]]
Solución: [-1.  2.  0.  1.]
Sumas realizadas: 6
Restas realizadas: 29
Multiplicaciones realizadas: 32
Divisiones realizadas: 10
Intercambios de filas: 0
```

```
In [5]: %autoreload 2
        from src import eliminacion_gaussiana

        Ab = [[1, -1, 2, -1, -8], [2, -2, 3, -3, -20], [1, 1, 1, 0, -2], [1, -1, 4, 3, 4]

        sol, suma_count, resta_count, mult_count, div_count, intercambio_count = elimina
        print('Solución:', sol)
        print('Sumas realizadas:', suma_count)
        print('Restas realizadas:', resta_count)
        print('Multiplicaciones realizadas:', mult_count)
        print('Divisiones realizadas:', div_count)
        print('Intercambios de filas:', intercambio_count)
```

```
[12-25 13:20:08][INFO]
[[ 1 -1  2 -1 -8]
 [ 0  0 -1 -1 -4]
 [ 0  2 -1  1  6]
 [ 0  0  2  4 12]]
[12-25 13:20:08][INFO]
[[ 1 -1  2 -1 -8]
 [ 0  2 -1  1  6]
 [ 0  0 -1 -1 -4]
 [ 0  0  2  4 12]]
[12-25 13:20:08][INFO]
[[ 1 -1  2 -1 -8]
 [ 0  2 -1  1  6]
 [ 0  0 -1 -1 -4]
 [ 0  0  0  2  4]]
Solución: [-7.  3.  2.  2.]
Sumas realizadas: 6
Restas realizadas: 29
Multiplicaciones realizadas: 32
Divisiones realizadas: 10
Intercambios de filas: 1
```

Soluciones infinitas

El siguiente sistema de ecuaciones tiene soluciones infinitas:

$$\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 2 & 2 & 1 & 6 \\ 1 & 1 & 2 & 6 \end{array}$$

```
In [6]: # %load_ext autoreload
%autoreload 2
from src import eliminacion_gaussiana

Ab = [[1., 1, 1, 4], [2, 2, 1, 6], [1, 1, 2, 6]]

try:
    sol, suma_count, resta_count, mult_count, div_count, intercambio_count = eli
    print('Solución:', sol)
    print('Sumas realizadas:', suma_count)
    print('Restas realizadas:', resta_count)
    print('Multiplicaciones realizadas:', mult_count)
    print('Divisiones realizadas:', div_count)
    print('Intercambios de filas:', intercambio_count)
except Exception as e:
    print('Error:', e)
```

```
[12-25 13:20:08][INFO]
[[ 1.  1.  1.  4.]
 [ 0.  0. -1. -2.]
 [ 0.  0.  1.  2.]]
Error: No existe solución única.
```

Sin solución

El siguiente sistema de ecuaciones no tiene solución:

$$\begin{array}{cccc} 1 & 1 & 1 & 4 \\ 2 & 2 & 1 & 4 \\ 1 & 1 & 2 & 6 \end{array}$$

```
In [7]: # %load_ext autoreload
%autoreload 2
from src import eliminacion_gaussiana

Ab = [[1., 1, 1, 4], [2, 2, 1, 4], [1, 1, 2, 6]]

try:
    sol, suma_count, resta_count, mult_count, div_count, intercambio_count = eli
    print('Solución:', sol)
    print('Sumas realizadas:', suma_count)
    print('Restas realizadas:', resta_count)
    print('Multiplicaciones realizadas:', mult_count)
    print('Divisiones realizadas:', div_count)
    print('Intercambios de filas:', intercambio_count)
```

```
except Exception as e:  
    print('Error:', e)
```

[12-25 13:20:08][INFO]

```
[[ 1.  1.  1.  4.]
```

```
 [ 0.  0. -1. -4.]
```

```
 [ 0.  0.  1.  2.]]
```

Error: No existe solución única.

In [8]: `import numpy as np`

```
def eliminacion_gaussiana_contador(A):  
    if not isinstance(A, np.ndarray):  
        A = np.array(A, dtype=float)  
  
    n = A.shape[0]  
  
    suma_count = 0  
    resta_count = 0  
    mult_count = 0  
    div_count = 0  
    intercambio_count = 0  
  
    for i in range(n - 1):  
  
        p = i  
        for k in range(i + 1, n):  
            if abs(A[k, i]) > abs(A[p, i]):  
                p = k  
  
        if A[p, i] == 0:  
            return None, suma_count, resta_count, mult_count, div_count, interca  
  
        if p != i:  
            A[[i, p]] = A[[p, i]]  
            intercambio_count += 1  
  
        for j in range(i + 1, n):  
            m = A[j, i] / A[i, i]  
            div_count += 1  
  
            for k in range(i, n + 1):  
                A[j, k] = A[j, k] - m * A[i, k]  
                mult_count += 1  
                resta_count += 1  
  
    x = np.zeros(n)  
  
    x[-1] = A[-1, -1] / A[-1, -2]  
    div_count += 1  
  
    for i in range(n - 2, -1, -1):  
        suma = 0  
        for j in range(i + 1, n):  
            suma += A[i, j] * x[j]  
            mult_count += 1  
            suma_count += 1  
  
        x[i] = (A[i, -1] - suma) / A[i, i]  
        resta_count += 1
```

```

        div_count += 1

    return x, suma_count, resta_count, mult_count, div_count, intercambio_count

```

```

In [9]: def gauss_jordan_contador(A):
        if not isinstance(A, np.ndarray):
            A = np.array(A, dtype=float)

        n = A.shape[0]

        suma_count = 0
        resta_count = 0
        mult_count = 0
        div_count = 0
        intercambio_count = 0

        for i in range(n):

            p = i
            for k in range(i + 1, n):
                if abs(A[k, i]) > abs(A[p, i]):
                    p = k

            if A[p, i] == 0:
                return None, suma_count, resta_count, mult_count, div_count, intercambio_count

            if p != i:
                A[[i, p]] = A[[p, i]]
                intercambio_count += 1

            pivote = A[i, i]
            for k in range(i, n + 1):
                A[i, k] = A[i, k] / pivote
                div_count += 1

            for j in range(n):
                if j != i:
                    factor = A[j, i]
                    for k in range(i, n + 1):
                        A[j, k] = A[j, k] - factor * A[i, k]
                        mult_count += 1
                        resta_count += 1

        return A[:, -1], suma_count, resta_count, mult_count, div_count, intercambio_count

```

```

In [10]: Ab = [[1, -1, 2, -1, -8], [2, -2, 3, -3, -20], [1, 1, 1, 0, -2], [1, -1, 4, 3, 4], [2, 1, 1, 1, 1]]

sol_g, s_g, r_g, m_g, d_g, i_g = eliminacion_gaussiana_contador(Ab.copy())
sol_gj, s_gj, r_gj, m_gj, d_gj, i_gj = gauss_jordan_contador(Ab.copy())

print("=== Eliminación Gaussiana ===")
print("Solución:", sol_g)
print("Sumas:", s_g, "Restas:", r_g)
print("Mult:", m_g, "Div:", d_g)
print("Intercambios:", i_g)

print("\n=== Gauss-Jordan ===")

```

```
print("Solución:", sol_gj)
print("Sumas:", s_gj, "Restas:", r_gj)
print("Mult:", m_gj, "Div:", d_gj)
print("Intercambios:", i_gj)
```

=== Eliminación Gaussiana ===

Solución: [-7. 3. 2. 2.]

Sumas: 6 Restas: 29

Mult: 32 Div: 10

Intercambios: 3

=== Gauss-Jordan ===

Solución: [-7. 3. 2. 2.]

Sumas: 0 Restas: 42

Mult: 42 Div: 14

Intercambios: 3