Gauss-Jacobi y seidel

January 21, 2026

1 Escuela Politécnica Nacional

1.1 Métodos Numéricos

Nombre: Lenin Amangandi

Tema: Métodos iterativos

Link al repositorio Taller N3B

1.2 1) Ejercicio 1

[75]:
```
%load_ext autoreload
%autoreload 2
from src import eliminacion_gaussiana, gauss_jacobi, gauss_seidel,gauss_jacobi2
import numpy as np
```

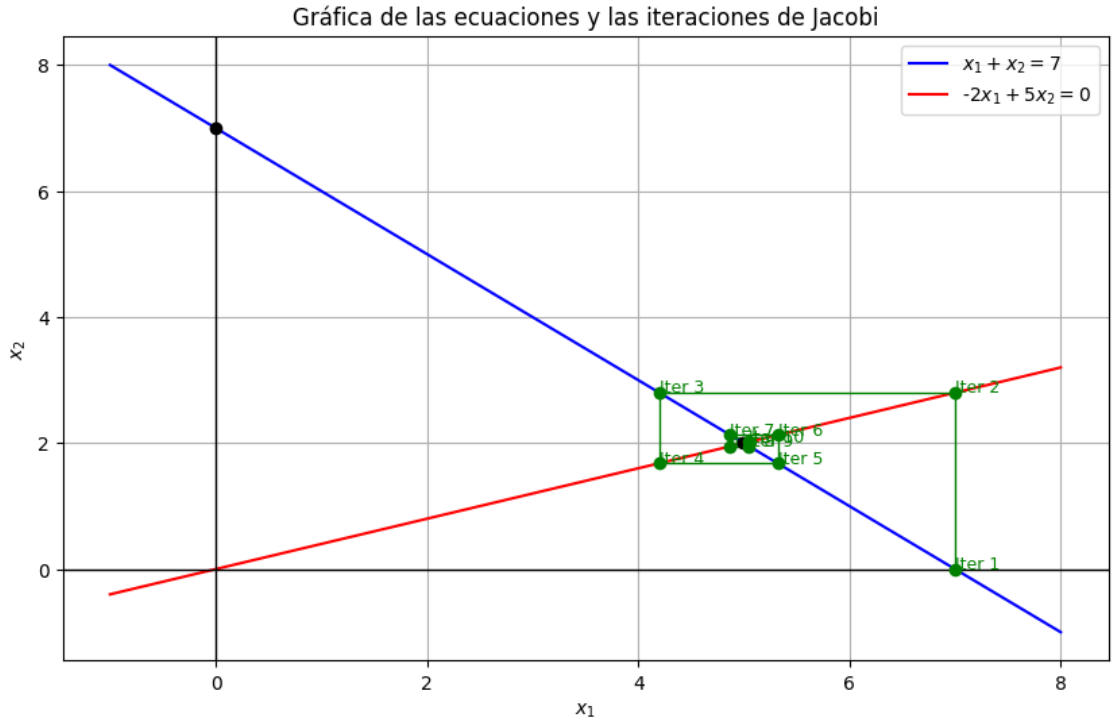
```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[76]:
```
A = [[1, 1],[-2, 5]]

b = [7, 0]
n=2
gauss_jacobi(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)
```

[76]:
```
[array([[7.],
        [0.]]),
 array([[7. ],
        [2.8]]),
 array([[4.2],
        [2.8]]),
 array([[4.2 ],
        [1.68]]),
 array([[5.32],
        [1.68]]),
 array([[5.32 ],
        [2.128]]),
 array([[4.872],
        [2.128]]),
```

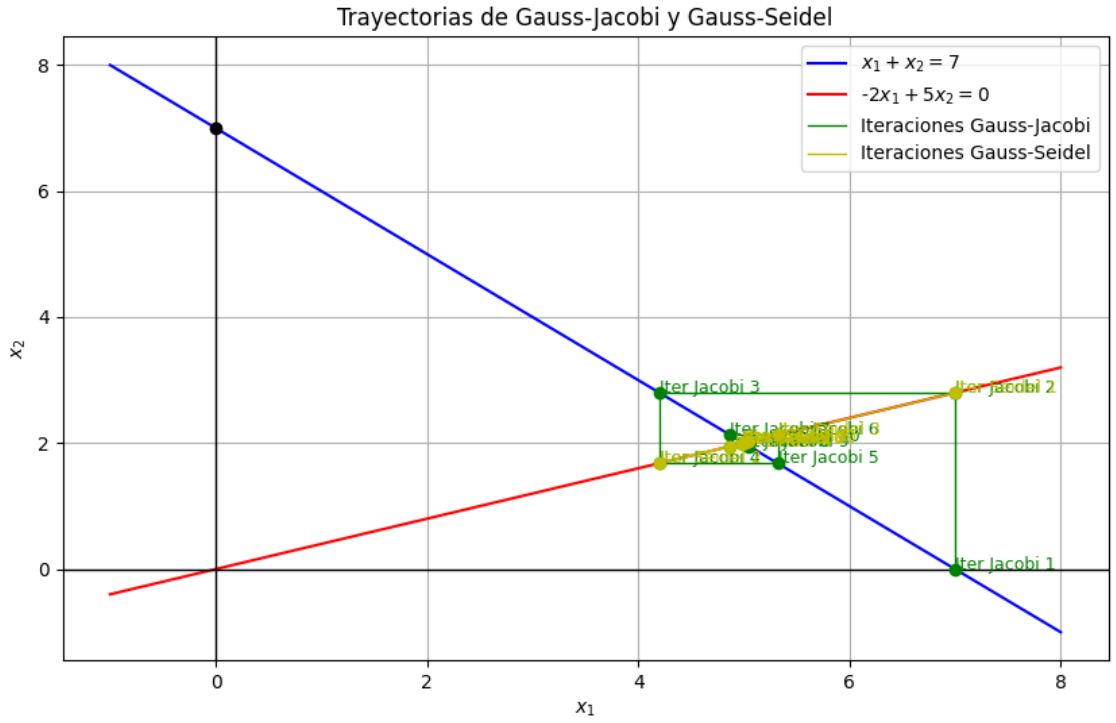
```
 array([[4.872 ],
        [1.9488]]),
 array([[5.0512],
        [1.9488]]),
 array([[5.0512 ],
        [2.02048]])]
```

[77]:
```python
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(-1, 8, 400)
x2_1 = 7 - x1
x2_2 = (2 * x1) / 5

A = [[1, 1],[-2, 5]]
b = [7, 0]
n=2
gauss_jacobi(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)
iterations = gauss_jacobi(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)

plt.figure(figsize=(10, 6))
plt.plot(x1, x2_1, label='$x_1 + x_2 = 7$', color='b')
plt.plot(x1, x2_2, label='-$2x_1 + 5x_2 = 0$', color='r')

plt.scatter([0, 5], [7, 2], color='black', zorder=5)

x_values = []
y_values = []

for i, iteration in enumerate(iterations):
    x_values.append(iteration[0][0])
    y_values.append(iteration[1][0])
    plt.scatter(iteration[0], iteration[1], color='g', zorder=10)
    plt.text(iteration[0], iteration[1], f'Iter {i+1}', fontsize=9, color='g')

plt.plot(x_values, y_values, color='g', linestyle='-', linewidth=1)

plt.title('Gráfica de las ecuaciones y las iteraciones de Jacobi')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')

plt.legend()
plt.grid(True)
plt.axhline(0, color='black',linewidth=1)
plt.axvline(0, color='black',linewidth=1)

plt.show()
```



[78]:
```
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(-1, 8, 400)
x2_1 = 7 - x1
x2_2 = (2 * x1) / 5

A = np.array([[1, 1], [-2, 5]])
b = np.array([7, 0])
n = 2

iterations_jacobi = gauss_jacobi(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)
iterations_seidel = gauss_seidel(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)

plt.figure(figsize=(10, 6))

plt.plot(x1, x2_1, label='$x_1 + x_2 = 7$', color='b')
plt.plot(x1, x2_2, label='-$2x_1 + 5x_2 = 0$', color='r')

plt.scatter([0, 5], [7, 2], color='black', zorder=5)

x_values_jacobi = []
y_values_jacobi = []
```

```
for i, iteration in enumerate(iterations_jacobi):
    x_values_jacobi.append(iteration[0][0])
    y_values_jacobi.append(iteration[1][0])
    plt.scatter(iteration[0], iteration[1], color='g', zorder=10)
    plt.text(iteration[0], iteration[1], f'Iter Jacobi {i+1}', fontsize=9,␣
 ↪color='g')

x_values_seidel = []
y_values_seidel = []

for i, iteration in enumerate(iterations_seidel):
    x_values_seidel.append(iteration[0][0])
    y_values_seidel.append(iteration[1][0])
    plt.scatter(iteration[0], iteration[1], color='y', zorder=10)
    plt.text(iteration[0], iteration[1], f'Iter Seidel {i+1}', fontsize=9,␣
 ↪color='y')

plt.plot(x_values_jacobi, y_values_jacobi, color='g', linestyle='-',␣
 ↪linewidth=1, label='Iteraciones Gauss-Jacobi')
plt.plot(x_values_seidel, y_values_seidel, color='y', linestyle='-',␣
 ↪linewidth=1, label='Iteraciones Gauss-Seidel')

plt.title('Trayectorias de Gauss-Jacobi y Gauss-Seidel')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')

plt.legend()
plt.grid(True)
plt.axhline(0, color='black', linewidth=1)
plt.axvline(0, color='black', linewidth=1)

plt.show()
```



1.3 Ejercicio 2

[79]:
```
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(-1, 8, 400)

x2_1 = 6 - x1
x2_2 = 2 * x1

A = [[1, 1], [-2, 1]]
b = [6, 0]
n = 2

iterations = gauss_jacobi(A=A, b=b, x=[0]*n, tol=1e-5, max_iter=10)

plt.figure(figsize=(8, 6))

plt.plot(x1, x2_1, label='$x_1 + x_2 = 6$', color='b')
plt.plot(x1, x2_2, label='-$2x_1 + x_2 = 0$', color='r')

plt.scatter([0, 3], [6, 0], color='black', zorder=5)
```

```
x_values = []
y_values = []

for i, iteration in enumerate(iterations):
    x_values.append(iteration[0][0])
    y_values.append(iteration[1][0])
    plt.scatter(iteration[0], iteration[1], color='g', zorder=10)
    plt.text(iteration[0], iteration[1], f'Iter {i+1}', fontsize=9, color='g')

plt.plot(x_values, y_values, color='g', linestyle='-', linewidth=1)

plt.title('Gráfica de las ecuaciones y las iteraciones de Jacobi')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')

plt.legend()
plt.grid(True)
plt.axhline(0, color='black',linewidth=1)
plt.axvline(0, color='black',linewidth=1)

# Mostrar la gráfica
plt.show()
```

[80]:
```
A = [[1, 1], [-2, 1]]
b = [6, 0]
n = 2

gauss_jacobi2(A=A, b=b, x=[0,0], tol=1e-5, max_iter=10)
```

```
[01-21 11:41:16][INFO][User] i= 0 x: [[0. 0.]]
[01-21 11:41:16][INFO][User] i= 1 x: [[6. 0.]]
[01-21 11:41:16][INFO][User] i= 2 x: [[ 6. 12.]]
[01-21 11:41:16][INFO][User] i= 3 x: [[-6. 12.]]
[01-21 11:41:16][INFO][User] i= 4 x: [[ -6. -12.]]
[01-21 11:41:16][INFO][User] i= 5 x: [[ 18. -12.]]
[01-21 11:41:16][INFO][User] i= 6 x: [[18. 36.]]
[01-21 11:41:16][INFO][User] i= 7 x: [[-30.  36.]]
[01-21 11:41:16][INFO][User] i= 8 x: [[-30. -60.]]
[01-21 11:41:16][INFO][User] i= 9 x: [[ 66. -60.]]
[01-21 11:41:16][INFO][User] i= 10 x: [[ 66. 132.]]
```

[80]:
```
array([[ 66.],
       [132.]])
```

1.4 3) Ejercicio 3

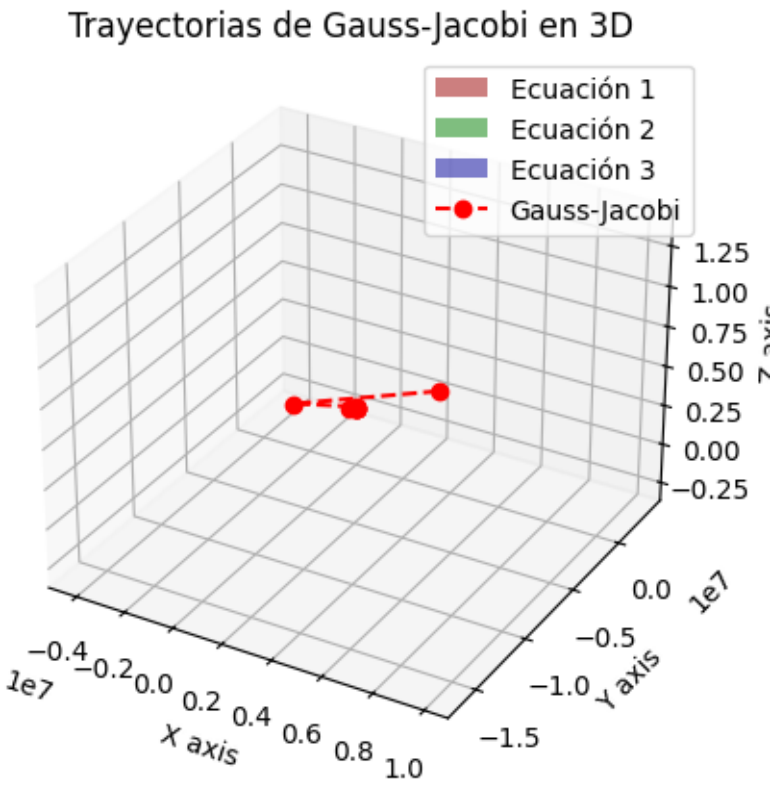
[81]:
```
import numpy as np

def gauss_jacobi(A, b, x0, tol, max_iter):
    n = len(b)
    x = np.array(x0, dtype=float)
    iterations = []

    for k in range(max_iter):
        x_new = np.zeros(n)
        for i in range(n):
            suma = np.dot(A[i, :], x) - A[i, i] * x[i]
            x_new[i] = (b[i] - suma) / A[i, i]

        iterations.append(x_new.copy())

        if np.linalg.norm(x_new - x) < tol:
            break

        x = x_new

    return np.array(iterations)
```

```
A = np.array([[1, -1, 2, 0],
              [-1, 1, -1, 3],
              [2, -1, 1, -1],
              [0, 3, -1, 2]], dtype=float)
b = np.array([1360, 1130, 1350, 3650], dtype=float)

x0_1 = np.array([0, 0, 0, 0], dtype=float)

iterations = gauss_jacobi(A, b, x0_1, tol=1e-5, max_iter=10)

for i, iter_x in enumerate(iterations):
    print(f"Iter {i + 1}: {iter_x}")
```

```
Iter 1: [1360. 1130. 1350. 1825.]
Iter 2: [ -210. -1635.  1585.   805.]
Iter 3: [-3445.    90.   940.  5070.]
Iter 4: [  -430. -16585.  13400.   2160.]
Iter 5: [-42025.    7620.  -12215.   33402.5]
Iter 6: [  33410.  -153317.5  126422.5  -15712.5]
Iter 7: [-404802.5  208100.  -234500.   295012.5]
Iter 8: [  678460.  -1523210.   1314067.5  -427575. ]
Iter 9: [-4149985.    3276382.5  -3306355.    2943673.75]
Iter 10: [  9890452.5  -16286231.25  14521376.25  -6565926.25]
```

[82]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

A = np.array([[1, -1, 2, 0],
              [-1, 1, -1, 3],
              [2, -1, 1, -1],
              [0, 3, -1, 2]], dtype=float)
b = np.array([1360, 1130, 1350, 3650], dtype=float)

def gauss_jacobi(A, b, x0, tol, max_iter):
    n = len(b)
    x = np.array(x0, dtype=float)
    iterations = []

    for k in range(max_iter):
        x_new = np.zeros(n)
        for i in range(n):
            suma = np.dot(A[i, :], x) - A[i, i] * x[i]
            x_new[i] = (b[i] - suma) / A[i, i]

        iterations.append(x_new.copy())
```

```
        if np.linalg.norm(x_new - x) < tol:
            break

        x = x_new

    return np.array(iterations)

x0_1 = np.array([0, 0, 0, 0], dtype=float)

iterations = gauss_jacobi(A, b, x0_1, tol=1e-5, max_iter=10)


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')


x_vals = np.linspace(0, 1000, 100)
y_vals = np.linspace(0, 1000, 100)
X, Y = np.meshgrid(x_vals, y_vals)

Z1 = (1360 + Y - 2 * X)
Z2 = (1130 + X + Y) / 1
Z3 = (1350 - 2 * X + Y)

ax.plot_surface(X, Y, Z1, alpha=0.5, rstride=100, cstride=100, color='r',␣
 ↪label='Ecuación 1')
ax.plot_surface(X, Y, Z2, alpha=0.5, rstride=100, cstride=100, color='g',␣
 ↪label='Ecuación 2')
ax.plot_surface(X, Y, Z3, alpha=0.5, rstride=100, cstride=100, color='b',␣
 ↪label='Ecuación 3')


jacobi_points = iterations

jacobi_x, jacobi_y, jacobi_z = zip(*[(point[0], point[1], point[2]) for point␣
 ↪in jacobi_points])

ax.plot(jacobi_x, jacobi_y, jacobi_z, 'ro--', label='Gauss-Jacobi')

ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')

plt.title('Trayectorias de Gauss-Jacobi en 3D')

plt.legend()
plt.show()
```


Trayectorias de Gauss-Jacobi en 3D
Ecuación 1
Ecuación 2
Ecuación 3
Gauss-Jacobi
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0.0
1e7
−0.5
−1.0
Y axis
−1.5
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e7
X axis