1) # Jacobi Method

def jacobi\_method(A, b, tolerance=1e-4, max\_iterations=1000):

n = len(b)

x\_old = [0 for \_ in range(n)] # Initial guess, x\_0 = 0

x\_new = x\_old.copy()

for k in range(max\_iterations):

for i in range(n):

sigma = sum(A[i][j] \* x\_old[j] for j in range(n) if i != j)

x\_new[i] = (b[i] - sigma) / A[i][i]

# Check convergence

if all(abs(x\_new[i] - x\_old[i]) < tolerance for i in range(n)):

return x\_new, k + 1 # Return the solution and number of iterations

x\_old = x\_new.copy()

return x\_new, max\_iterations   
   
# Gauss-Seidel Method

def gauss\_seidel\_method(A, b, tolerance=1e-4, max\_iterations=1000):

n = len(b)

x = [0 for \_ in range(n)] # Initial guess, x\_0 = 0

for k in range(max\_iterations):

x\_old = x.copy()

for i in range(n):

sigma = sum(A[i][j] \* x[j] for j in range(n) if i != j)

x[i] = (b[i] - sigma) / A[i][i]

# Check convergence

if all(abs(x[i] - x\_old[i]) < tolerance for i in range(n)):

return x, k + 1 # Return the solution and number of iterations

return x, max\_iterations   
  
A = [

[4, -1, 0, 0],

[-1, 4, -1, 0],

[0, -1, 4, -1],

[0, 0, -1, 3]

]

b = [15, 10, 10, 10]  
  
# Assuming gaussian\_elimination is available in the imported lab2\_gauss\_method module

from lab2\_gauss\_method import gaussian\_elimination

# Solve using Gaussian Elimination

gaussian\_solution = gaussian\_elimination([row[:] for row in A], b[:])

print("Solution using Gaussian Elimination:", gaussian\_solution)

# Solve using Jacobi and Gauss-Seidel methods

x\_jacobi, iter\_jacobi = jacobi\_method(A, b)

x\_gauss\_seidel, iter\_gauss\_seidel = gauss\_seidel\_method(A, b)

print(f"Solution using Jacobi Method (Iterations: {iter\_jacobi}): {x\_jacobi}")

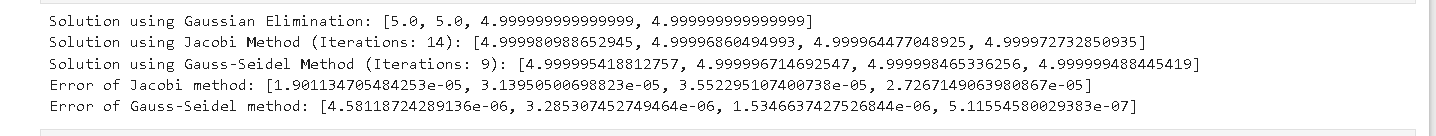
print(f"Solution using Gauss-Seidel Method (Iterations: {iter\_gauss\_seidel}): {x\_gauss\_seidel}")

# Calculate errors (absolute difference) compared to Gaussian Elimination

error\_jacobi = [abs(x\_jacobi[i] - gaussian\_solution[i]) for i in range(len(b))]

error\_gauss\_seidel = [abs(x\_gauss\_seidel[i] - gaussian\_solution[i]) for i in range(len(b))]

print(f"Error of Jacobi method: {error\_jacobi}")

print(f"Error of Gauss-Seidel method: {error\_gauss\_seidel}")  
  
  
2)   
3) 