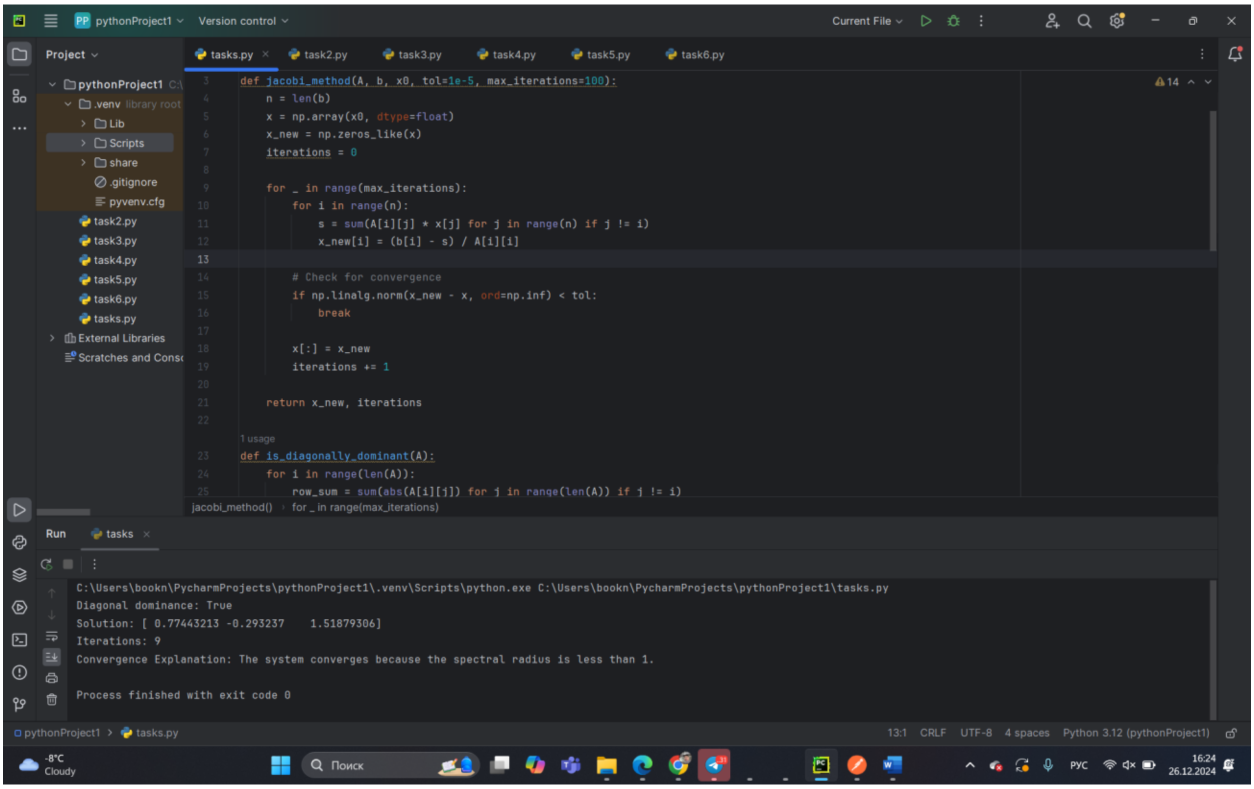
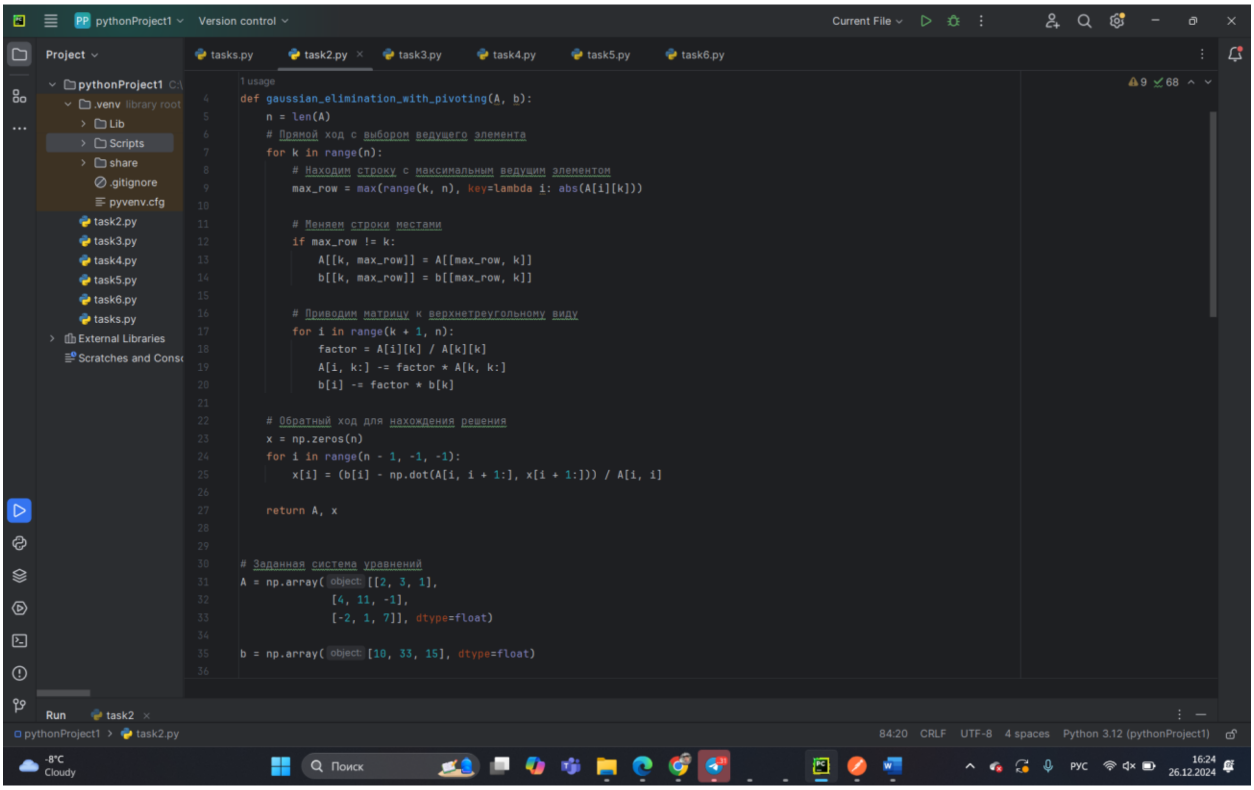
Task1:



pythonProject1
Version control
Current File
Project
tasks.py
task2.py
task3.py
task4.py
task5.py
task6.py
def jacobi_method(A, b, x0, tol=1e-5, max_iterations=100):
    n = len(b)
    x = np.array(x0, dtype=float)
    x_new = np.zeros_like(x)
    iterations = 0
    for _ in range(max_iterations):
        for i in range(n):
            s = sum(A[i][j] * x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i][i]
        # Check for convergence
        if np.linalg.norm(x_new - x, ord=np.inf) < tol:
            break
        x[:] = x_new
        iterations += 1
    return x_new, iterations
1 usage
def is_diagonally_dominant(A):
    for i in range(len(A)):
        row_sum = sum(abs(A[i][j]) for j in range(len(A)) if j != i)
jacobi_method() › for _ in range(max_iterations)
Run
tasks
C:\Users\bookn\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\bookn\PycharmProjects\pythonProject1\tasks.py
Diagonal dominance: True
Solution: [ 0.77443213 -0.293237    1.51879306]
Iterations: 9
Convergence Explanation: The system converges because the spectral radius is less than 1.
Process finished with exit code 0
pythonProject1 › tasks.py
13:1 CRLF UTF-8 4 spaces Python 3.12 (pythonProject1)
-8°C
Cloudy
Поиск
РУС
16:24
26.12.2024


Task2:



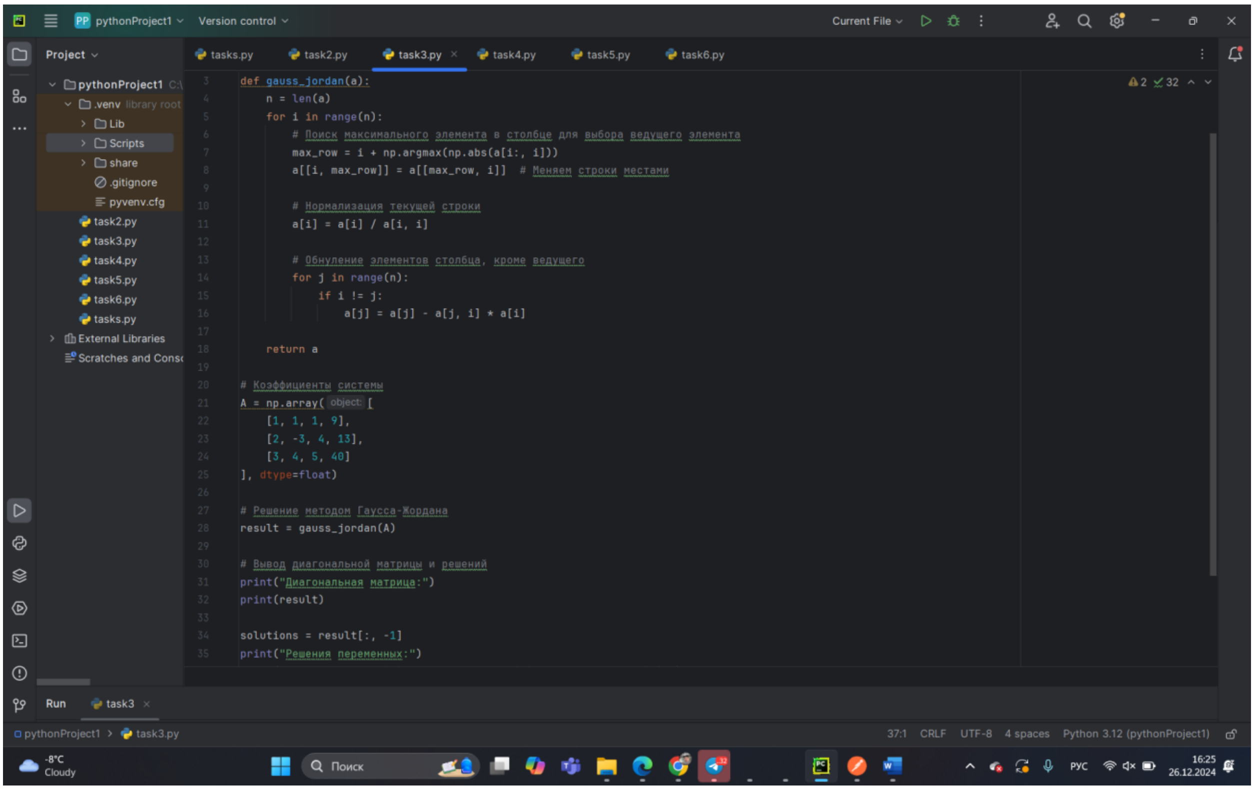
pythonProject1
Version control
Current File
Project
tasks.py
task2.py
task3.py
task4.py
task5.py
task6.py
1 usage
def gaussian_elimination_with_pivoting(A, b):
    n = len(A)
    # Прямой ход с выбором ведущего элемента
    for k in range(n):
        # Находим строку с максимальным ведущим элементом
        max_row = max(range(k, n), key=lambda i: abs(A[i][k]))
        # Меняем строки местами
        if max_row != k:
            A[[k, max_row]] = A[[max_row, k]]
            b[[k, max_row]] = b[[max_row, k]]
        # Приводим матрицу к верхнетреугольному виду
        for i in range(k + 1, n):
            factor = A[i][k] / A[k][k]
            A[i, k:] -= factor * A[k, k:]
            b[i] -= factor * b[k]
    # Обратный ход для нахождения решения
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i + 1:], x[i + 1:])) / A[i, i]
    return A, x
# Заданная система уравнений
A = np.array( object: [[2, 3, 1],
                 [4, 11, -1],
                 [-2, 1, 7]], dtype=float)
b = np.array( object: [10, 33, 15], dtype=float)
Run
task2
pythonProject1 › task2.py
84:20 CRLF UTF-8 4 spaces Python 3.12 (pythonProject1)
-8°C
Cloudy
Поиск
РУС
16:24
26.12.2024

```
C:\Users\bookn\PycharmProjects\pythonPr
Верхнетреугольная матрица:
[[ 4.   11.   -1. ]
 [ 0.    6.5   6.5]
 [ 0.    0.    4. ]]

Решение системы:
[-0.86538462  3.44230769  1.40384615]
Верхнетреугольная матрица:
[[ 4.   11.   -1. ]
 [ 0.    6.5   6.5]
 [ 0.    0.    4. ]]

Решение системы:
[-0.86538462  3.44230769  1.40384615]
```

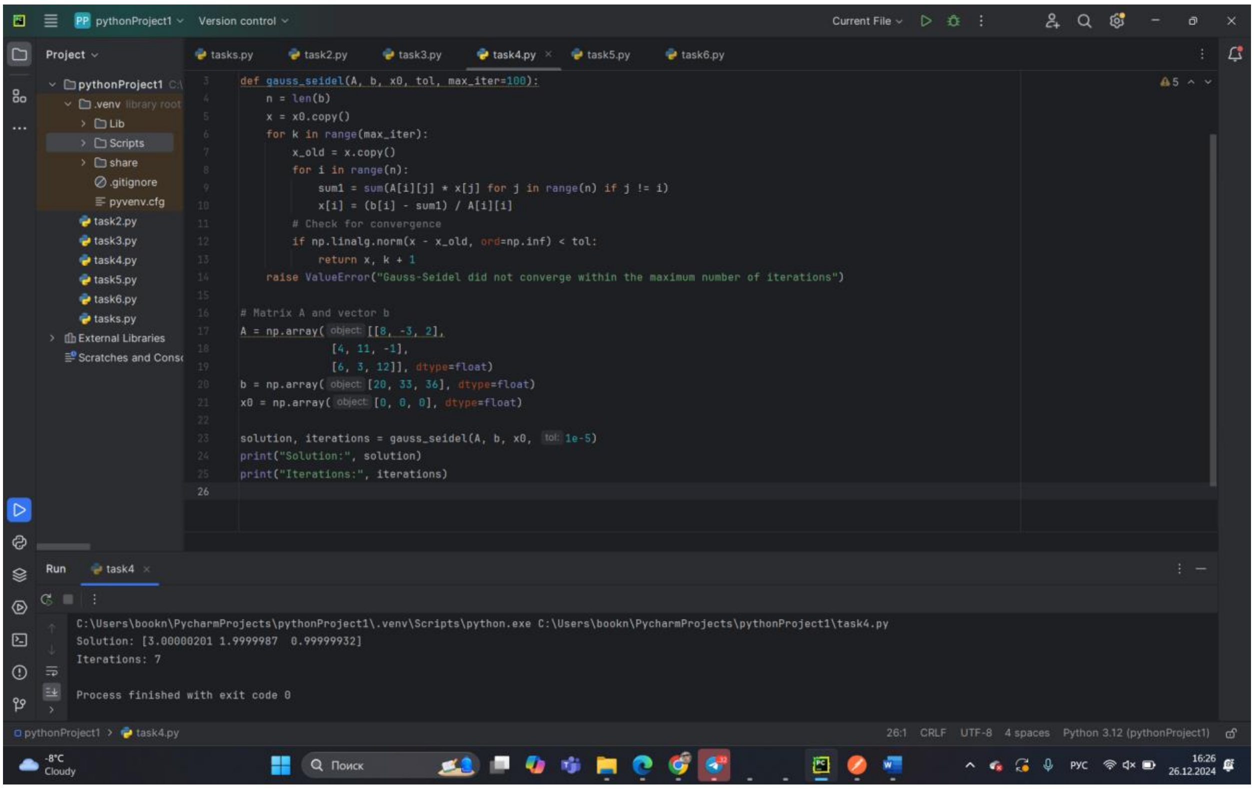
Task3:



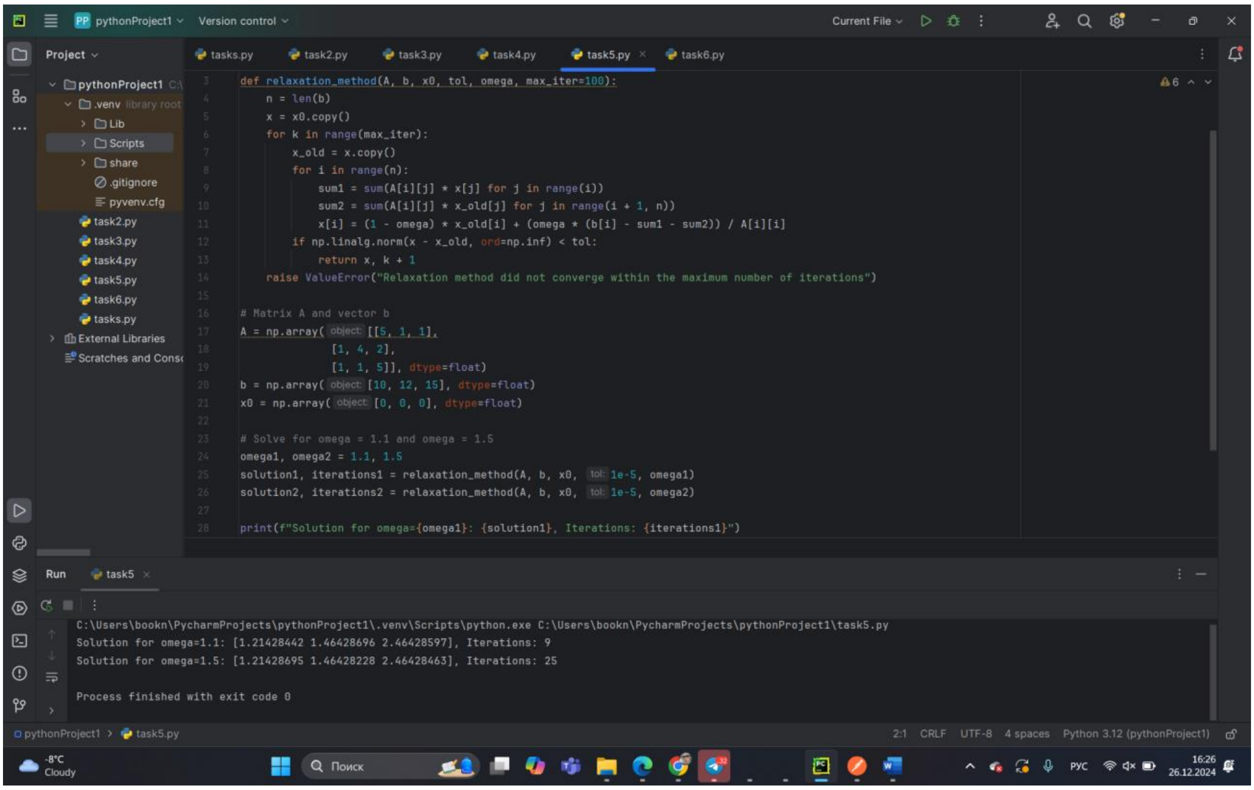

```
C:\Users\bookn\PycharmProjects\pythonProject1\.venv\Scripts\python.exe
Диагональная матрица:
[[ 1.  0.  0.  1.]
 [-0.  1.  0.  3.]
 [-0. -0.  1.  5.]]
Решения переменных:
x = 0.9999999999999982, y = 3.0000000000000004, z = 5.000000000000001

Process finished with exit code 0
```

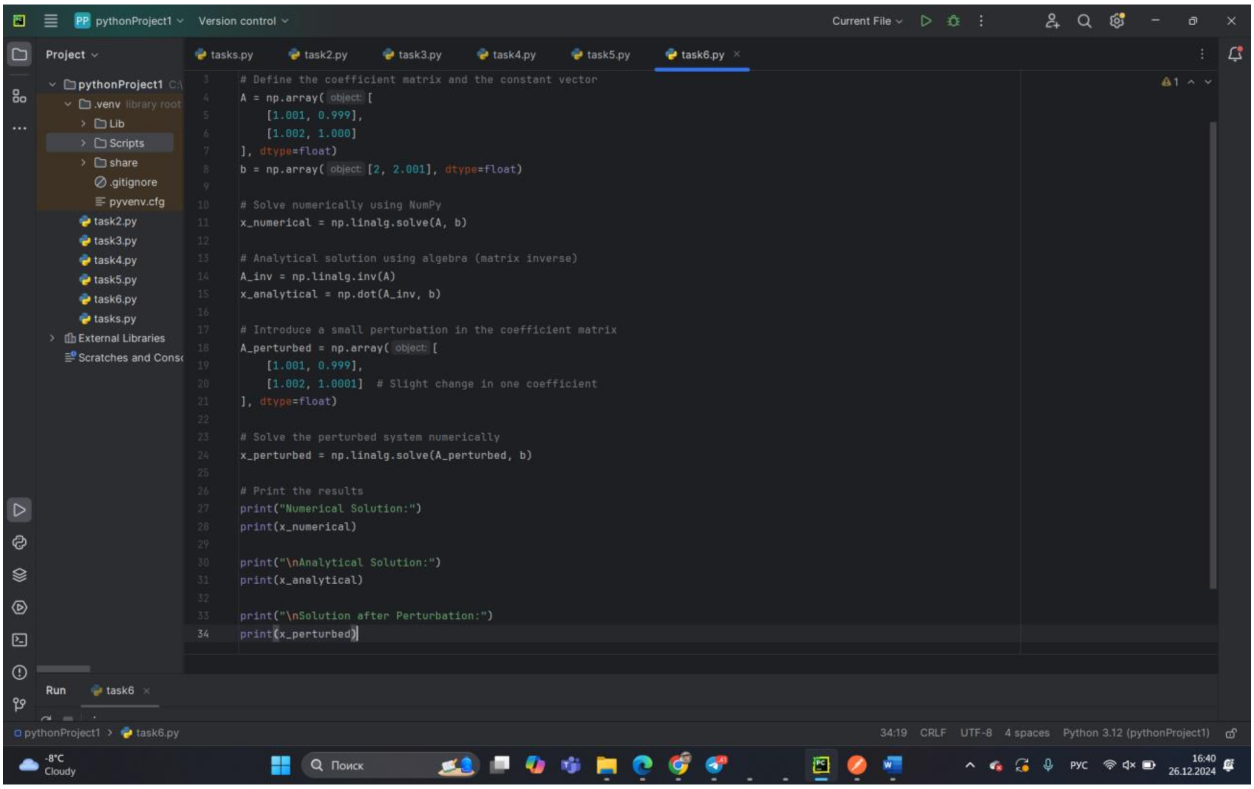
Task4:

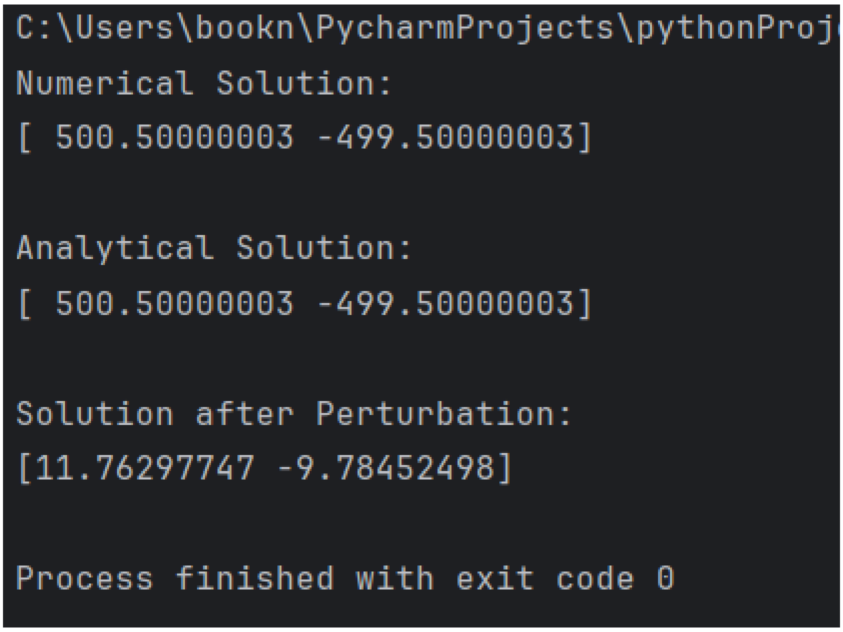

def gauss_seidel(A, b, x0, tol, max_iter=100):
    n = len(b)
    x = x0.copy()
    for k in range(max_iter):
        x_old = x.copy()
        for i in range(n):
            sum1 = sum(A[i][j] * x[j] for j in range(n) if j != i)
            x[i] = (b[i] - sum1) / A[i][i]
        # Check for convergence
        if np.linalg.norm(x - x_old, ord=np.inf) < tol:
            return x, k + 1
    raise ValueError("Gauss-Seidel did not converge within the maximum number of iterations")
# Matrix A and vector b
A = np.array( object: [[8, -3, 2],
              [4, 11, -1],
              [6, 3, 12]], dtype=float)
b = np.array( object: [20, 33, 36], dtype=float)
x0 = np.array( object: [0, 0, 0], dtype=float)
solution, iterations = gauss_seidel(A, b, x0, tol: 1e-5)
print("Solution:", solution)
print("Iterations:", iterations)
C:\Users\bookn\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\bookn\PycharmProjects\pythonProject1\task4.py
Solution: [3.00000201 1.9999987  0.99999932]
Iterations: 7
Process finished with exit code 0

Task5:


def relaxation_method(A, b, x0, tol, omega, max_iter=100):
    n = len(b)
    x = x0.copy()
    for k in range(max_iter):
        x_old = x.copy()
        for i in range(n):
            sum1 = sum(A[i][j] * x[j] for j in range(i))
            sum2 = sum(A[i][j] * x_old[j] for j in range(i + 1, n))
            x[i] = (1 - omega) * x_old[i] + (omega * (b[i] - sum1 - sum2)) / A[i][i]
        if np.linalg.norm(x - x_old, ord=np.inf) < tol:
            return x, k + 1
    raise ValueError("Relaxation method did not converge within the maximum number of iterations")
# Matrix A and vector b
A = np.array( object: [[5, 1, 1],
              [1, 4, 2],
              [1, 1, 5]], dtype=float)
b = np.array( object: [10, 12, 15], dtype=float)
x0 = np.array( object: [0, 0, 0], dtype=float)
# Solve for omega = 1.1 and omega = 1.5
omega1, omega2 = 1.1, 1.5
solution1, iterations1 = relaxation_method(A, b, x0, tol: 1e-5, omega1)
solution2, iterations2 = relaxation_method(A, b, x0, tol: 1e-5, omega2)
print(f"Solution for omega={omega1}: {solution1}, Iterations: {iterations1}")
C:\Users\bookn\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\bookn\PycharmProjects\pythonProject1\task5.py
Solution for omega=1.1: [1.21428442 1.46428696 2.46428597], Iterations: 9
Solution for omega=1.5: [1.21428695 1.46428228 2.46428463], Iterations: 25
Process finished with exit code 0

Task6:


pythonProject1
Version control
Current File
Project
tasks.py
task2.py
task3.py
task4.py
task5.py
task6.py
pythonProject1
.venv library root
Lib
Scripts
share
.gitignore
pyvenv.cfg
External Libraries
# Define the coefficient matrix and the constant vector
A = np.array( object: [
    [1.001, 0.999],
    [1.002, 1.000]
], dtype=float)
b = np.array( object: [2, 2.001], dtype=float)
# Solve numerically using NumPy
x_numerical = np.linalg.solve(A, b)
# Analytical solution using algebra (matrix inverse)
A_inv = np.linalg.inv(A)
x_analytical = np.dot(A_inv, b)
# Introduce a small perturbation in the coefficient matrix
A_perturbed = np.array( object: [
    [1.001, 0.999],
    [1.002, 1.0001]  # Slight change in one coefficient
], dtype=float)
# Solve the perturbed system numerically
x_perturbed = np.linalg.solve(A_perturbed, b)
# Print the results
print("Numerical Solution:")
print(x_numerical)
print("\nAnalytical Solution:")
print(x_analytical)
print("\nSolution after Perturbation:")
print(x_perturbed)
Run
task6
pythonProject1 > task6.py
34:19 CRLF UTF-8 4 spaces Python 3.12 (pythonProject1)
-8°C Cloudy
Поиск
РУС
16:40
26.12.2024


C:\Users\bookn\PycharmProjects\pythonProj
Numerical Solution:
[ 500.50000003 -499.50000003]
Analytical Solution:
[ 500.50000003 -499.50000003]
Solution after Perturbation:
[11.76297747 -9.78452498]
Process finished with exit code 0