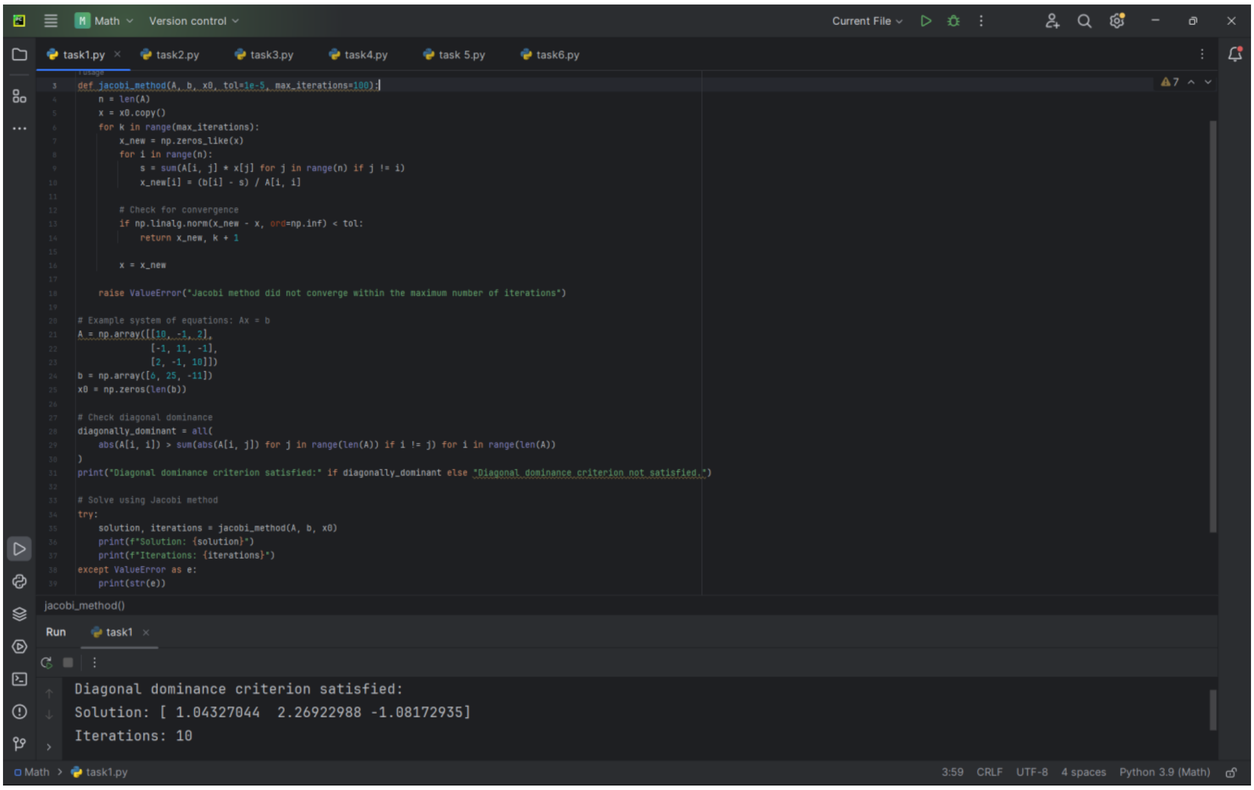
Task 1: Jacobi Method and Convergence Analysis



Description:

Objective: Solve a linear system using the Jacobi method and analyze convergence.

Implementation:

The matrix A and vector b define the system Ax=b.

The Jacobi method iteratively updates each variable using only the values from the previous iteration.

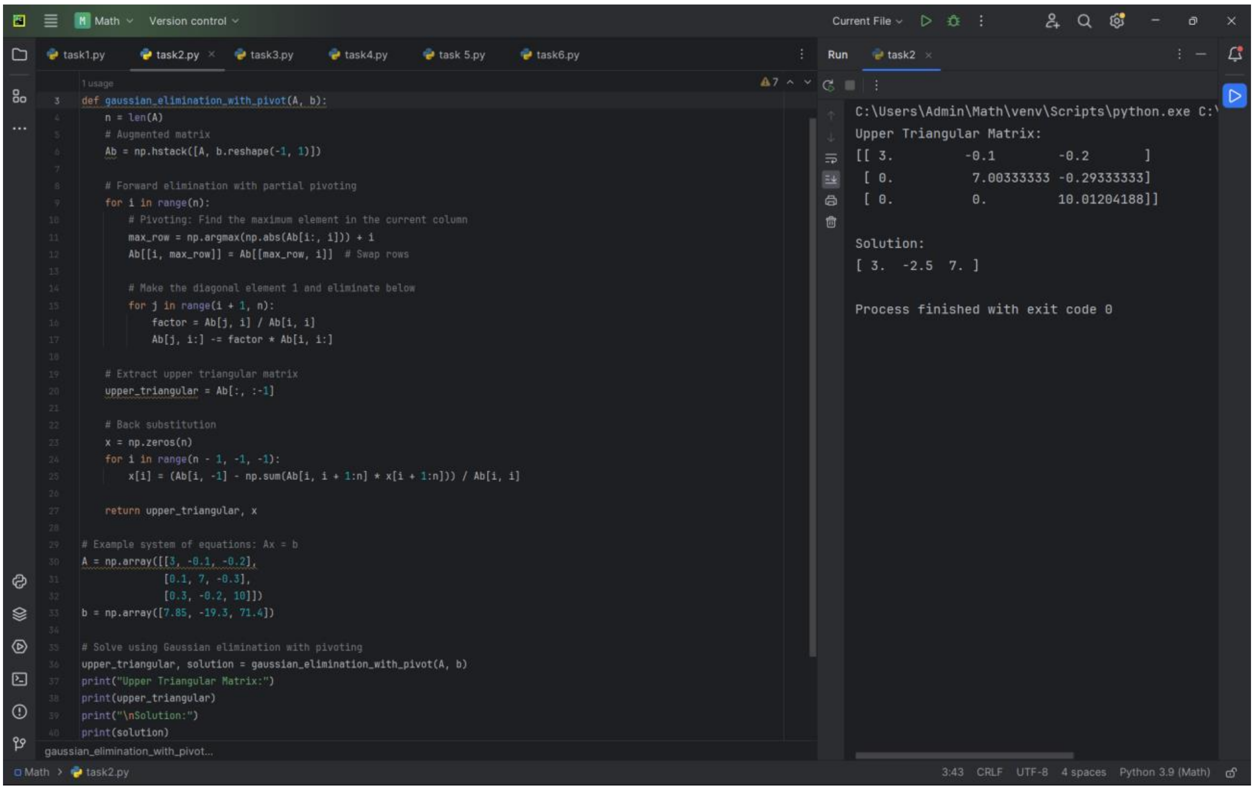
The diagonal dominance of the matrix is checked using the criterion $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$, which is critical for ensuring convergence.

Convergence is determined by comparing the change in the solution vector across iterations against a defined tolerance 10^{-5}.

Explaining Convergence:

Convergence depends on the matrix structure. Diagonal dominance or a spectral radius of the iteration matrix less than 1 ensures that errors diminish over iterations

Task 2: Gaussian Method with Leading Element Selection



Description:

Objective: Solve a linear system using Gaussian elimination with partial pivoting to improve numerical stability.

Implementation:

Pivot selection identifies the row with the largest absolute value in the current column, minimizing numerical errors during division.

Forward elimination transforms the matrix into an upper triangular form by eliminating variables step-by-step.

Back substitution starts from the last variable and moves upward, solving for each variable sequentially.

Explaining Pivot Importance:

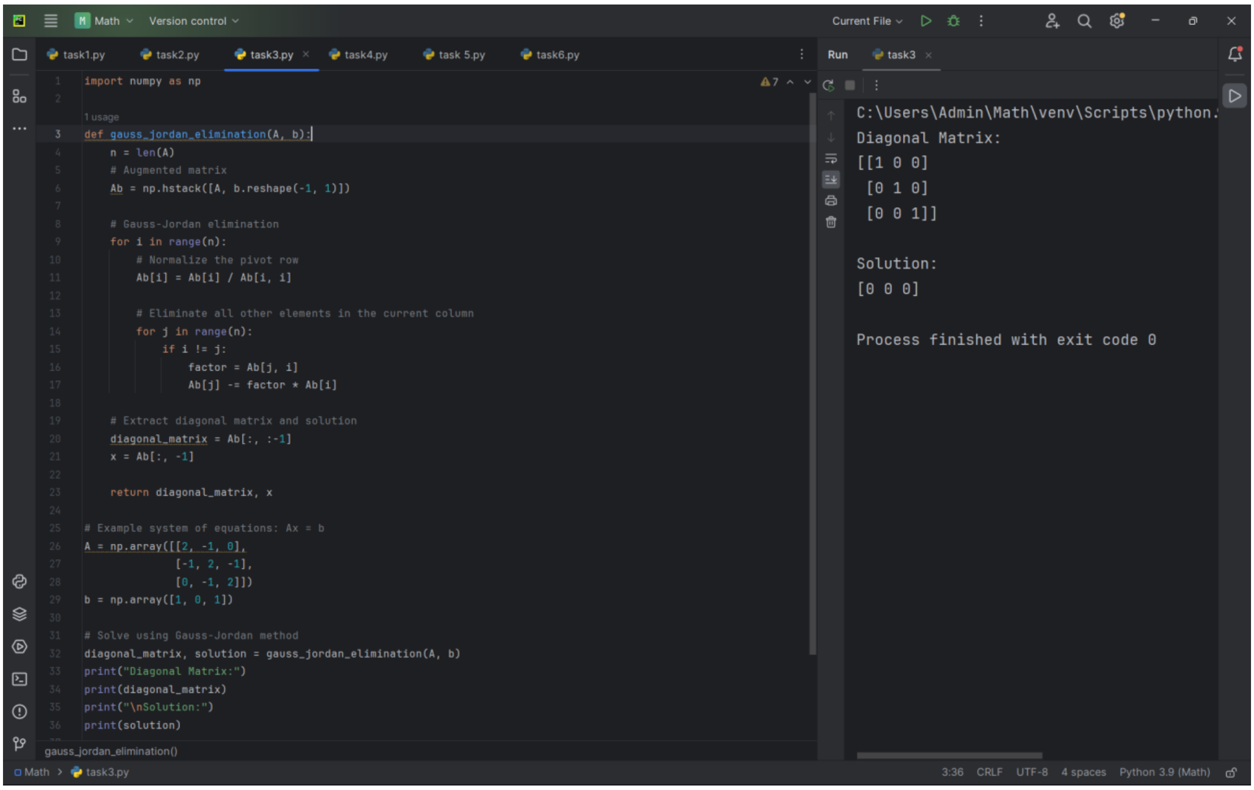
Proper pivoting avoids division by small numbers, which can cause rounding errors and instability in floating-point arithmetic. It ensures a more accurate and stable solution.

Output:

Upper triangular matrix after forward elimination.

Final solution vector x.

Task 3: Gauss-Jordan Method



Description:

Objective: Solve a linear system using the Gauss-Jordan method, transforming the matrix to diagonal form.

Implementation:

Each pivot row is normalized by dividing it by the pivot element, converting the pivot element to 1.

Other rows are adjusted to zero out elements in the pivot column, leading to a diagonal matrix.

Advantages Over Gauss Method:

The Gauss-Jordan method eliminates the need for back substitution, providing the solution directly.

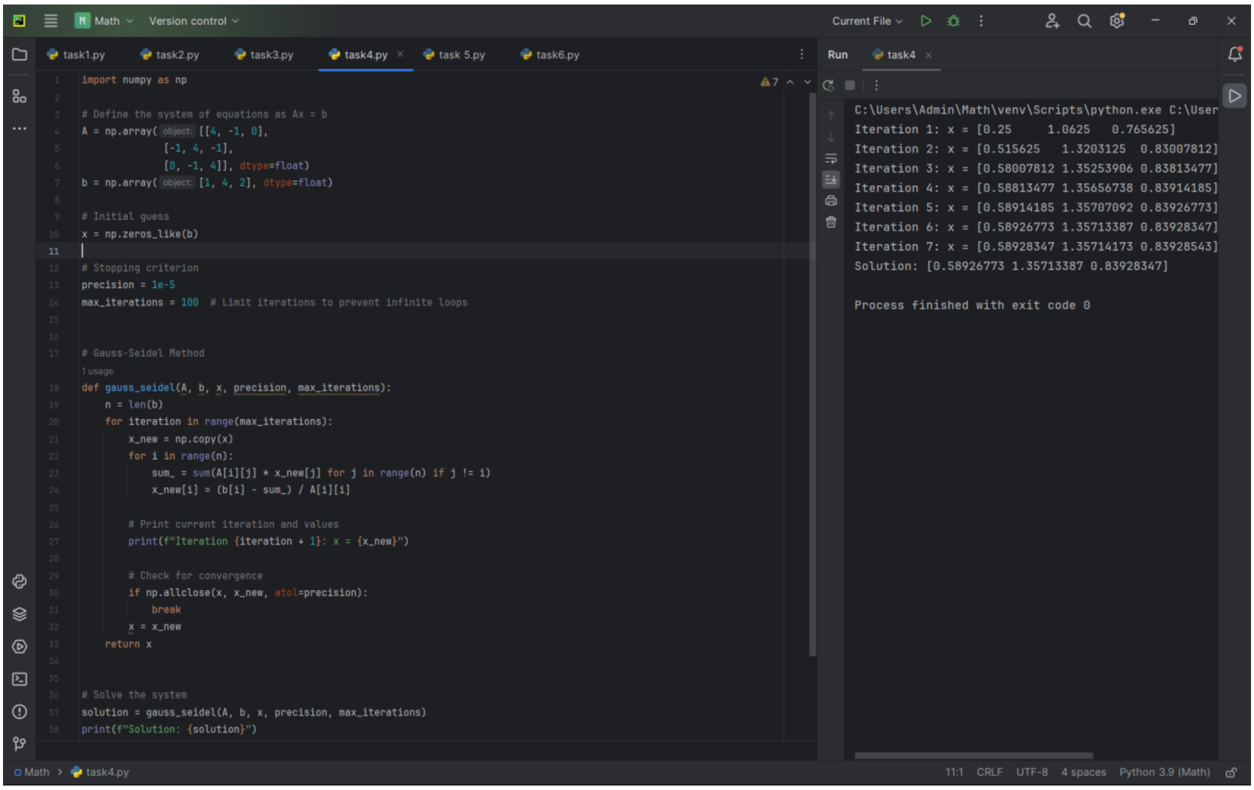
It simultaneously calculates the inverse matrix (if required) and is more efficient for certain systems, especially when solving multiple systems with the same coefficient matrix.

Output:

Diagonal matrix with the solution vector directly visible.

Final solution vector x.

Task 4: Gauss-Seidel Method



Description:

Objective: Solve a linear system using the Gauss-Seidel method with a specified stopping criterion.

Implementation:

Each variable is updated immediately after it is computed, using the most recent values to accelerate convergence.

Convergence is assessed using the infinity norm of the difference between successive iterations, stopping when it is below 10−510^{-5}10−5.

Explaining Stopping Criterion:

A stricter stopping criterion (smaller tolerance) ensures higher accuracy but requires more iterations and increases execution time.

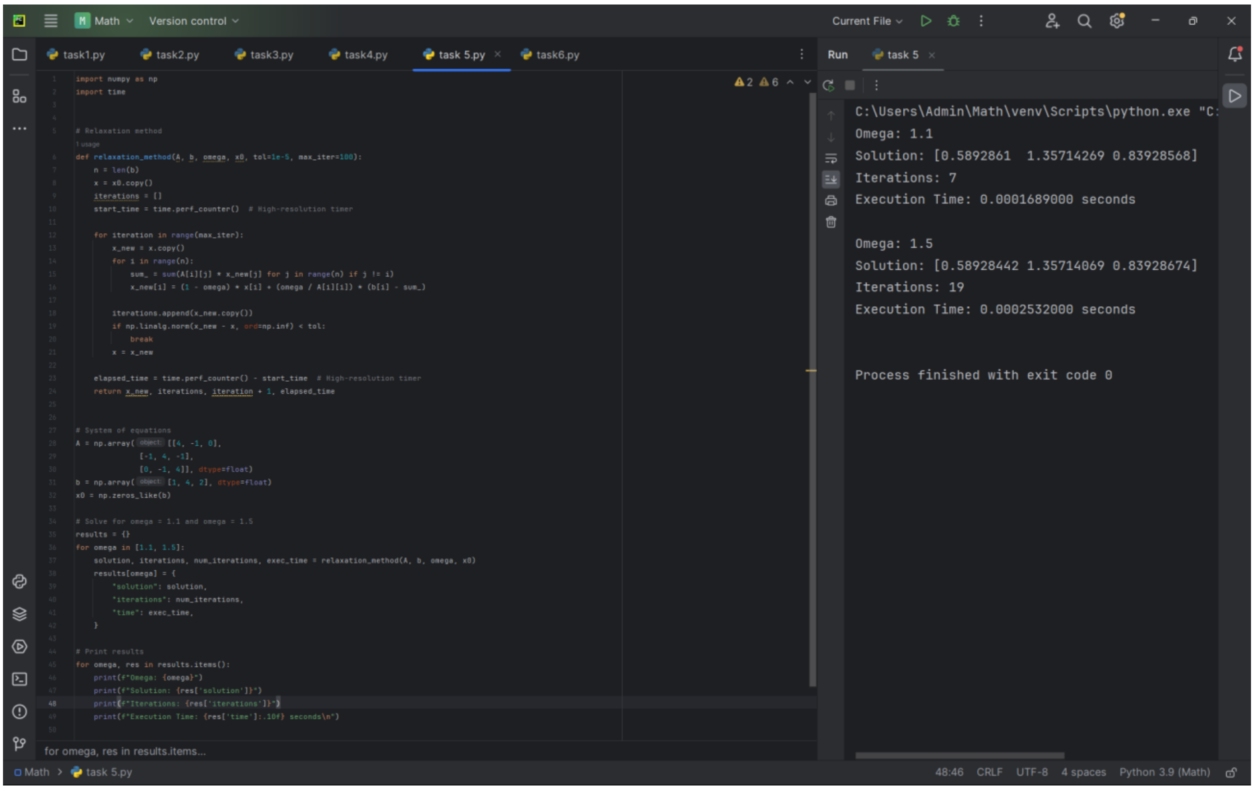
A looser criterion reduces iterations and execution time but may compromise accuracy.

Output:

Iterative results with the current values of variables.

Total number of iterations to achieve the specified accuracy.

Task 5: Relaxation Method



Description:

Objective: Solve a linear system using the relaxation method with different relaxation parameters ω.

Implementation:

The parameter ω\omegaω modifies the update formula to speed up or slow down convergence.

For ω>1 (over-relaxation), the method accelerates updates, reducing iterations if stable.

For ω<1 (under-relaxation), the method converges more slowly but is more stable.

Explaining the Effect of ω:

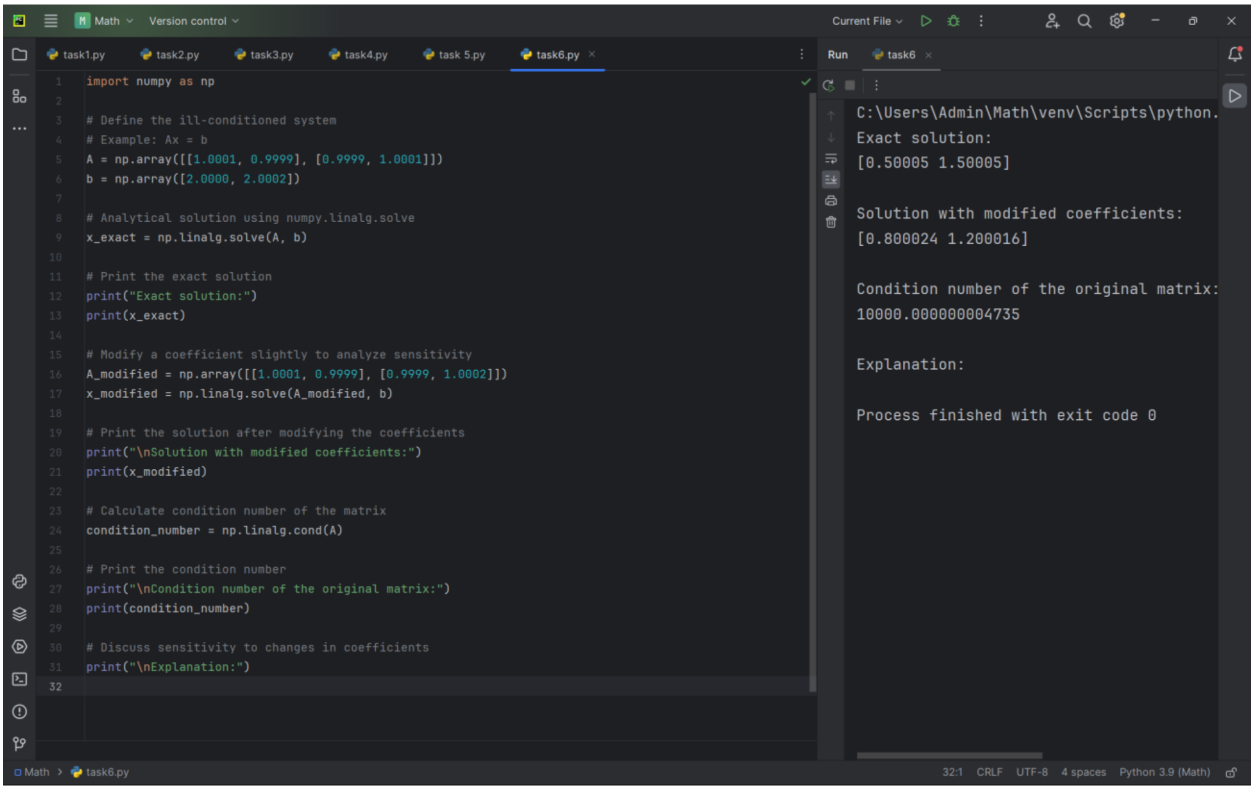
The relaxation parameter significantly impacts the speed of convergence. Over-relaxation is efficient but can lead to divergence if ω is too large.

Output:

Solutions for both ω=1.1 and ω=1.5.

Comparison of iterations and execution times for each parameter.

Task 6: Ill-Conditioned Systems



Description:

Objective: Solve an ill-conditioned system analytically and numerically, demonstrating sensitivity to coefficient changes.

Implementation:

The original system is solved to find the baseline solution.

A small perturbation is introduced to one coefficient in the matrix, and the system is resolved to observe the change in the solution.

Explaining Sensitivity:

Ill-conditioned systems have high condition numbers, meaning small changes in the coefficients lead to large changes in the solution.

This occurs because the rows or columns of the matrix are nearly linearly dependent, amplifying errors.

Output:

Analytical solution for the unperturbed system.

Numerical solution for the unperturbed and perturbed systems.

Comparison of the solutions to demonstrate sensitivity.