Report6_D1265154

Implementation Steps

1. Class Design and Inheritance:

I initiated the project by designing a base class Matrix, which would serve as the foundation for matrix-related operations and memory management. Subsequently, I derived the Vector class to handle column vectors, integrating vector-specific functionalities. Additionally, I developed the SMatrix class to manage square matrices and execute determinant computations.

2. Matrix Operations:

I proceeded to implement various matrix operations, including addition, subtraction, multiplication, and scalar operations. To facilitate intuitive mathematical expressions involving matrices, I employed operator overloading extensively.

3. Input and Output:

To ensure seamless integration with standard I/O streams, I implemented friend functions to manage the input and output of matrices and vectors.

4. Linear Equation Solver:

As a key component of the project, I crafted the linear_equation_system_solver.cpp program. This program facilitated tasks such as reading the system size, generating random coefficients, and solving equations using the determinant method.

Challenges and Solutions

1. Determinant Calculation

The computation of determinants for matrices posed a significant challenge, particularly in terms of ensuring accuracy and performance for matrices of varying sizes.

Solution:

To tackle this challenge, I developed the `determinant` method within the SMatrix class, employing recursion and cofactor expansion. Furthermore, I implemented optimized solutions for base cases, such as 1x1 and 2x2 matrices, to enhance performance.

2. Operator Overloading

Challenge:

Overloading operators for matrix operations necessitated meticulous attention to detail to ensure correct functionality and code readability.

Solution:

To overcome this challenge, I meticulously defined operator overloads within the Matrix class and as friend functions. This approach facilitated intuitive usage while maintaining consistency and efficiency in operations.

3. Linear Equation Solver Verification

Ensuring the correctness of solutions for linear equation systems was imperative. However, verifying solutions accurately while accounting for potential numerical errors presented a considerable challenge.

Solution:

To address this challenge, I implemented a verification step within the linear equation solver program. This step involved subsequently checking if the result approximated a zero vector within a specified error tolerance. Careful consideration of double precision and meticulous comparison techniques were employed to handle floating-point inaccuracies effectively.