**ACA PROJECT REPORT 2024/2025 PROJECT**

Davide De Vittorio, Giulia Trespiolli

**1. Matrix inversion**

The inverse of a matrix is widely used in various fields such as computer graphics, engineering, physics, and economics. It is a fundamental tool that enables precise calculations and efficient problem-solving across multiple disciplines.

* 1. **Analysis of the Serial Algorithm**

**Introduction to the Algorithm**

The serial algorithm implemented for computing the inverse of a matrix relies on the LU **decomposition** technique. LU is a decomposition method that factorizes a given square matrix into the product of two triangular matrices: a lower triangular matrix L and an upper triangular matrix U. Then the LU decomposition is used to solve multiple systems of linear equations of the form Ly=b and Ux=y, where b is a column of the identity matrix, and x is a column of the desired inverse of A. By solving these system N times (changing every time b from the first to the last column of the identity matrix) it can be founded the inverse of an N x N matrix. So instead of directly calculating the inverse that could be difficult, we:

1. Decompose A into and U and L.
2. Solve two triangular systems for each column using:
   * forward substitution for Ly=b.
   * backward substitution Ux=y.

**Description of the Algorithm**

1. **Matrix Reading and Allocation**: The matrix is read from a file and stored in a dynamically allocated 2D array. The file must contain in the first row the size of the matrix, followed by the elements of the matrix in the other rows.
2. **LU Decomposition**: The LU decomposition process iteratively computes the elements of y and x. Each element of U is calculated by subtracting the products of already-computed elements of U and L:

**Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente**

Similarly, elements of L are computed by normalizing the difference between L and L times U divided by the corresponding diagonal element of U:

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

1. **Forward and Backward Substitution**:
   * **Forward substitution** solves Ly=b, where b is a column vector of the identity matrix. Starting from the first row, each element of is computed by subtracting the contributions of already-computed elements and dividing by the diagonal element of L.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

* + **Backward substitution** solves Ux=y. Starting from the last row, each element of is computed similarly, using the already-computed elements of U.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

1. **Inversion Process**: For each iteration the x vector is copied into the correspondent column of the inversion matrix:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

1. **Memory Management**: To optimize memory usage, dynamic allocation is employed. After the inversion is complete, all dynamically allocated memory is freed to prevent memory leaks.

**Efficiency Analysis**

1. **Computational Complexity**:
   * The LU decomposition process has a complexity of , as it involves loops that iterates over the matrix dimensions.
   * Forward and backward substitution for each column of the identity matrix have a complexity of . Since these are repeated n times, the overall complexity of the substitution phase is .
   * Thus, the total complexity of the algorithm is , which is optimal for direct matrix inversion methods:
2. **Advantages of the LU-Based Approach**:
   * **Reusability**: Once L and U are computed, they can be reused to solve multiple systems of equations involving the same matrix A but different b vectors.
3. **Drawbacks**:
   * The algorithm requires the input matrix to be nonsingular and well-conditioned. Poorly conditioned matrices can lead to inaccurate results.
   * Memory usage can become significant for large matrices due to the need to store L, U, and intermediate vectors b, x, and y.

**Example Application and Measurement**

To measure the performance of the algorithm, we tested it on a square matrix of different sizes. Using the clock() function from time.h, the execution time for the inversion process was recorded. The results showed that the execution time grows cubically with n, consistent with the theoretical complexity. For instance:

|  |  |
| --- | --- |
| **Matrix Size (n)** | **Execution Time (s)** |
| 100 | 0.02 ?? |
| 500 | 1.45??? |
| 1000 | 11.34??? |

These measurements validate the cubic growth of computational cost with respect to matrix size.

**Conclusion**

The LU decomposition-based approach for matrix inversion is efficient and well-suited for medium to large-sized matrices. Its modular nature (separating decomposition, substitution, and inversion) makes it easy to understand and implement. However, its performance is limited by the cubic complexity, which motivates parallelization for handling very large matrices in a reasonable time.

* 1. **A priori study of available parallelism and parallel implementation**

The serial code for matrix inversion uses LU decomposition to break the problem into two main subproblems:

1. Decomposition of the matrix A into L (lower triangular) and U (upper triangular).
2. Solving linear systems through forward and backward substitution.

These phases include distinct code blocks operating on different data structures, primarily matrices (A, L, U, B, X, Y). The initial review of the code suggests the following critical points for parallelization:

* **Independent iterations in loops**:  
  During LU decomposition, the outer loops (those that iterate over i) are independent, since each iteration of the loop computes a set of values ​​(e.g., a row of U and a column of L) that does not depend on the computations of subsequent iterations. This implies that each iteration can be executed in parallel, reducing the overall execution time. It is important to keep in mind that during parallel iterations inside the outer loop, you will need to use barriers to synchronize the threads:

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

The first inner loop must be completed before the second inner loop, because the U values calculated in the first, are needed in the second loop.

* **Element-wise computation of Y and X**:  
  The forward substitution and backward substitution phases can be parallelized because the computation of each element of the Y vector (for forward substitution) and the X vector (for backward substitution) is independent for each index. Therefore, each element of Y and X can be computed in parallel.

Let’s focus on the different functions in the computing inversion phase and let’s analyse them with a 1000 x 1000 matrix:

Immagine che contiene testo, ricevuta, Carattere, bianco

Descrizione generata automaticamente

It’s important to keep in mind that the Forward and backward function are called 1000 times, while the LU decomposition function just 1 time, so it’s the LU decomposition phase the most expansive.