Vietnamese – German University

DEPARTMENT OF COMPUTER SCIENCE

The tiny Project's Report:

Lecturer: Huynh Trung Hieu

Student – ID:

Nguyen Duc Trung – 10423116

Nguyen Ngoc Binh Son – 10423100

Tram Dinh Khoa – 10423059

Vo Minh Luan – 10423074

Ha Thien Phuc – 10423163

Tran Duc Dung – 10423134

Binh Duong – 06/2025

Abstract

This project focuses on building a simple yet powerful C++ library to solve linear systems, with a real-world application in predicting CPU performance using linear regression. The work is split into two parts. In the first part, we designed and implemented three main classes: Vector, Matrix, and LinearSystem. These classes handle tasks like memory management, arithmetic operations, and computing things like the determinant or inverse of a matrix. We also extended the system to handle special types of problems, such as symmetric matrices or systems that don't have a unique solution. In the second part, we applied our code to a real dataset from the UCI Machine Learning Repository. Using linear regression, we predicted CPU performance based on hardware specs, then tested how accurate our model was using RMSE (root mean square error). Overall, this project combines core programming skills with mathematical problem-solving and practical data analysis.

Vietnamese-German University

Table of Contents

1. Project’s Objective

The program is divided into two main parts:

Part A: Develop reusable and robust Vector and Matrix classes with proper memory management and operator overloading. Implement a LinearSystem class to solve equations of the form Ax = b using methods such as Gaussian elimination and the Conjugate Gradient method for special cases. Extend the solution to handle under- and over-determined systems using the Moore-Penrose inverse and Tikhonov regularization.

Part B: Apply the developed tools to a real-world problem by building a linear regression model that predicts computer CPU performance based on hardware specifications. Evaluate the accuracy of the regression model using standard metrics like Root Mean Square Error (RMSE).

2. Overall Architecture

The program is divided into two main parts:

Part A – Implementing and Solving Linear Systems:

- **Vector class**: Represents a vector, supports operator overloading, handles memory management, and allows access using one-based indexing.
- **Matrix class**: Represents a matrix with full functionality such as addition, subtraction, multiplication with vectors/matrices/scalars, computing determinant, inverse, and Moore Penrose pseudoinverse.
- **LinearSystem class**: Solves linear systems Ax=bAx = bAx=b using Gaussian elimination with pivoting.
- **PosSymLinSystem class** (derived from LinearSystem): Designed for symmetric positive definite systems, uses the conjugate gradient method for solving.
- **Support for under-determined or ill-posed systems**: Uses Moore-Penrose pseudoinverse or Tikhonov regularization.

Part B – CPU Performance Prediction Using Linear Regression:

- Dataset obtained from the UCI Machine Learning Repository (209 instances).
- Data is split into training set (80%) and testing set (20%).
- Build a linear regression model to predict PRP (Published Relative Performance).
- Use the LinearSystem or PosSymLinSystem class to solve for model parameters.
- Evaluate the model using RMSE (Root Mean Square Error).

3. Technologies Used

Programming Language: C++

Standard Libraries: <iostream>, <cassert>, <cmath>, <fstream>, <vector>, <stdexcept>,etc.

Numerical Libraries: Self-implemented.

Source Control: Git/Gitlab

Input Data: Data loaded from a CSV file (from UCI) or entered manually via CLI.

4. Details of the Main components.

a. Main files and their function

Part A: Linear Algebra & Solvers

Vector.h/Vector.cpp: Declares and implements the Vector class for 1D arrays (vectors).

Matrix.h/ Matrix.cpp: Declares and implements the Matrix class for 2D arrays (matrices).

LinearSystem.h/ LinearSystem.cpp: Generate structure to solve a system of linear equations Ax=b.

PosSymLinSystem.h/PosSymLinSystem.cpp: Solves systems where the matrix A is positive semidefinite and symmetric.

Tikhonov.h/Tikhonov.cpp : Implements Tikhonov regularization, also known as ridge regression.

Test.cpp: Entry point to test the linear algebra solvers.

main.cpp: Entry point. Reads data, runs regression, test vector/matrix classes.

data.csv: UCI dataset containing computer hardware performance.

Part B: Machine Learning (Regression)

AI.cpp: Main file for linear regression using matrix algebra.

AI_without_4_new_val.cpp: Trains and evaluates on existing data.

machine.data/ machine.names: Dataset and metadata for the machine learning part.

b. Class Descriptions

Part A: Linear Algebra Classes

1. Vector Class

- **Attributes**:

    - int mSize: size of the vector.
    - double* mData: pointer to the dynamic array of data.
- **Key Methods**:
    - Constructor & destructor (handles memory).
    - Overloaded operators: =, +, -, * for scalars/vectors.
    - operator[] (with bounds checking).
    - operator() (one-based indexing).
- **Role**: Core class for representing and manipulating 1D data. Used in system solving and regression models.

2. Matrix Class

- **Attributes**:
    - int mNumRows, mNumCols: number of rows and columns.
    - double** mData: pointer to a 2D dynamic array.
- **Key Methods**:
    - Constructor & destructor (allocate and deallocate memory).
    - Copy constructor (deep copy of matrix).
    - Public accessors for dimensions.
    - Overloaded operators: =, +, -, * (matrix/vector/scalar).
    - (i, j) operator for one-based entry access.
    - Determinant(), Inverse(), PseudoInverse().
- **Role**: Represents 2D matrix data. Foundation for solving systems of equations.

3. LinearSystem Class

- **Attributes**:
    - int mSize: size of system.
    - Matrix* mpA: pointer to coefficient matrix.
    - Vector* mpb: pointer to right-hand-side vector.
- **Key Methods**:

- Constructor that initializes matrix and vector.
- Vector Solve(): solves system using Gaussian elimination.

- **Role**: Encapsulates logic for solving systems Ax=bAx = bAx=b, used in regression and testing.

4. PosSymLinSystem Class

- **Methods**:
  - Overrides Solve() using **conjugate gradient** method.
  - Validates symmetry of matrix A.
- **Role**: Specialized solver for symmetric positive definite systems.

5. Tikhonov Class.

- **Methods:**
  - Computes the regularized solution to the system using the formula

$$\mathbf{x = (A^TA + \lambda I)^{-1} A^Tb}$$

- **Role:** Solves ill – posed systems using Tikhonov regularization.

6. Linear Regression Module

- **Goal**: Estimate CPU performance via linear model:

$$\mathbf{PRP = x_1 . MYCT + x_2 . MMIN + ... + x_6 . CHMAX}$$

- **Implementation**:
  - Use matrix AAA of size 209x6 for features.
  - Use vector bbb of size 209 for target PRP.
  - Solve Ax=bAx = bAx=b using least squares (Moore-Penrose inverse).
  - Evaluate with RMSE on test data (20%).

Part B: AI Linear Regression

No class defined – structured as a procedural program.

Role: Implements Linear Regression using the Normal Equation.

5. Conclusion

This project successfully integrates fundamental concepts of C++ programming with practical applications in numerical linear algebra and machine learning. By developing custom Vector, Matrix, and LinearSystem classes from scratch, we deepened our understanding of memory management, operator overloading, and algorithm design. The application of these tools to solve real-world problems—such as predicting CPU performance through linear regression—demonstrates the versatility and reliability of our implementation. Furthermore, the inclusion of advanced techniques like the Moore-Penrose pseudoinverse and Tikhonov regularization showcases our ability to handle ill-posed or complex systems. Overall, this project not only strengthened our programming and mathematical skills but also highlighted the importance of clean, modular code for solving real-world engineering challenges