The Tiny Project 1

Full Name: Nguyễn Lâm Minh Hòa

Student ID: 10422030

Class: CSE2023 - Group 2

Course: Programming 2

Instructor: Prof.Huynh Trung Hieu

Binh Duong, May 2025

Table of contents

I. Introduction

This project is developed as part of the Programming 2 course, with the objective of implementing a basic linear algebra library in C++ and applying it to real-world data analysis. The project is divided into two main parts:

- Part A – Development of a C++ library supporting vectors, matrices, and linear systems.

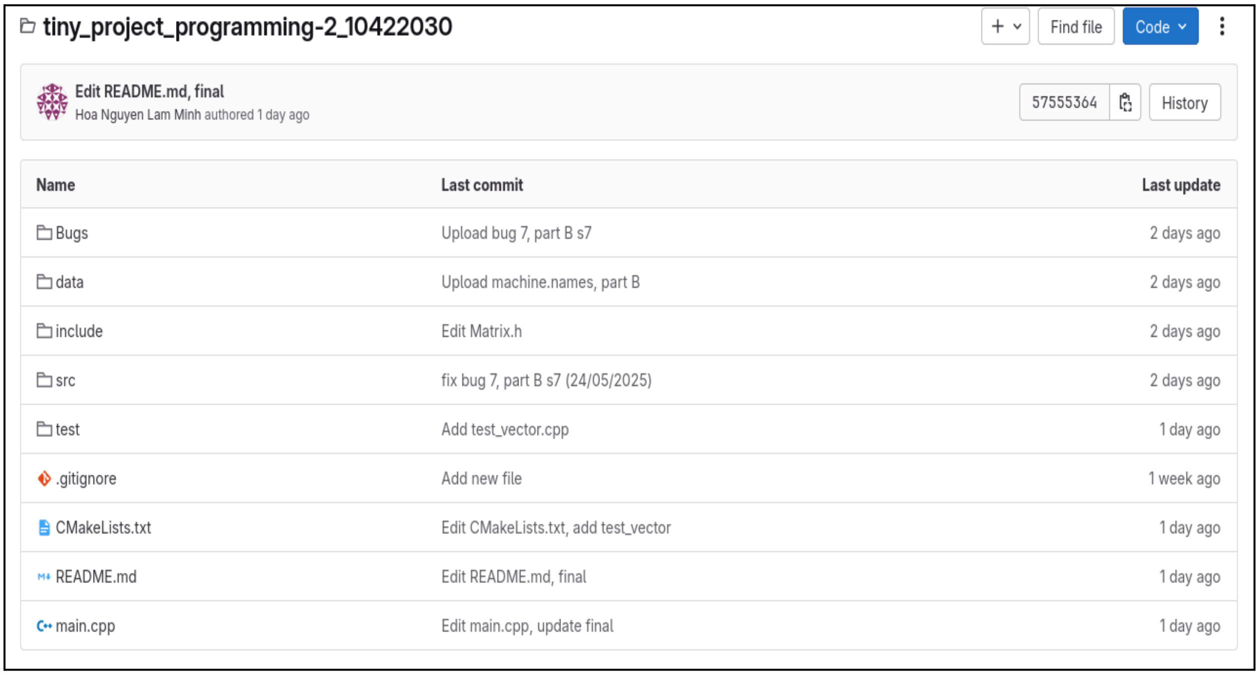
- Part B – Applying the library to perform linear regression using real-world data.

The implementation follows a structured process divided into 8 stages, with each stage tracked via a dedicated Git branch. Proper modular design, documentation, and unit tests were applied throughout.

II. Part A

1. Project Structure & CMake

- Created directory structure: include/, src/, test/, data/.
- Configured CMake for building with cmake .. && make.
- Implemented a basic main.cpp for demonstration purposes.



tiny_project_programming-2_10422030

Edit README.md, final
Hoa Nguyen Lam Minh authored 1 day ago

Name	Last commit	Last update
Bugs	Upload bug 7, part B s7	2 days ago
data	Upload machine.names, part B	2 days ago
include	Edit Matrix.h	2 days ago
src	fix bug 7, part B s7 (24/05/2025)	2 days ago
test	Add test_vector.cpp	1 day ago
.gitignore	Add new file	1 week ago
CMakeLists.txt	Edit CMakeLists.txt, add test_vector	1 day ago
README.md	Edit README.md, final	1 day ago
main.cpp	Edit main.cpp, update final	1 day ago

2. Vector Class

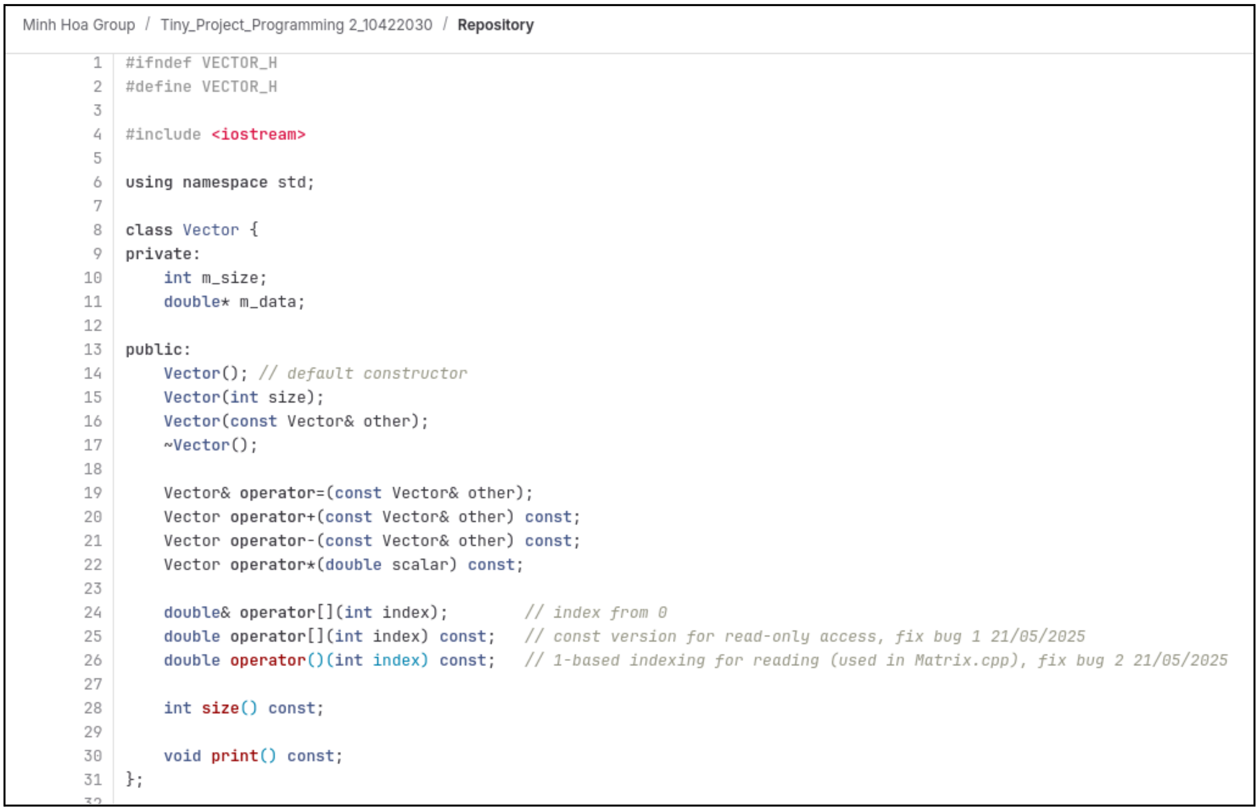
2.1: Vector.h

- Introduction

Defines the Vector class used to represent a dynamic one-dimensional array of floating-point numbers.

- Functional Overview

- Constructors (default, size-based, copy)
- Destructor for memory management
- Operator overloads:
    - +, -, * (scalar), =, [], ()
- size() – returns the vector length
- print() – outputs the vector values



Minh Hoa Group / Tiny_Project_Programming 2_10422030 / **Repository**

```
#ifndef VECTOR_H
#define VECTOR_H

#include <iostream>

using namespace std;

class Vector {
private:
    int m_size;
    double* m_data;

public:
    Vector(); // default constructor
    Vector(int size);
    Vector(const Vector& other);
    ~Vector();

    Vector& operator=(const Vector& other);
    Vector operator+(const Vector& other) const;
    Vector operator-(const Vector& other) const;
    Vector operator*(double scalar) const;

    double& operator[](int index);        // index from 0
    double operator[](int index) const;   // const version for read-only access, fix bug 1 21/05/2025
    double operator()(int index) const;   // 1-based indexing for reading (used in Matrix.cpp), fix bug 2 21/05/2025

    int size() const;

    void print() const;
};
```

(Overview of Vector.h file)

- Conclusion

Encapsulates all necessary operations for vector algebra. Supports both 0-based and 1-based element access, improving flexibility for use in matrix computations.

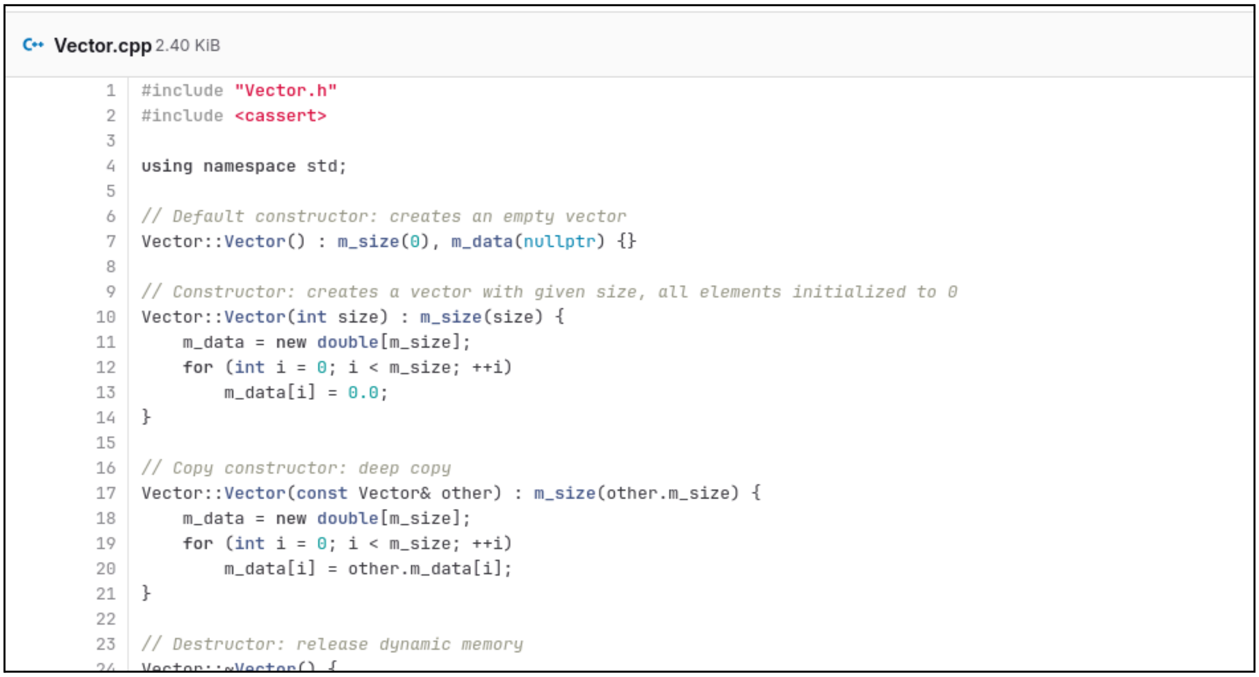
2.2: Vector.cpp

- Introduction

Implements the logic declared in Vector.h.

- Functional Overview

- Allocates and deallocates memory for dynamic arrays.
- Enforces bounds checking with assert.
- Implements arithmetic and indexing operations with clean and readable logic.
- operator() provides 1-based access (for matrix-style compatibility).



Vector.cpp 2.40 KiB

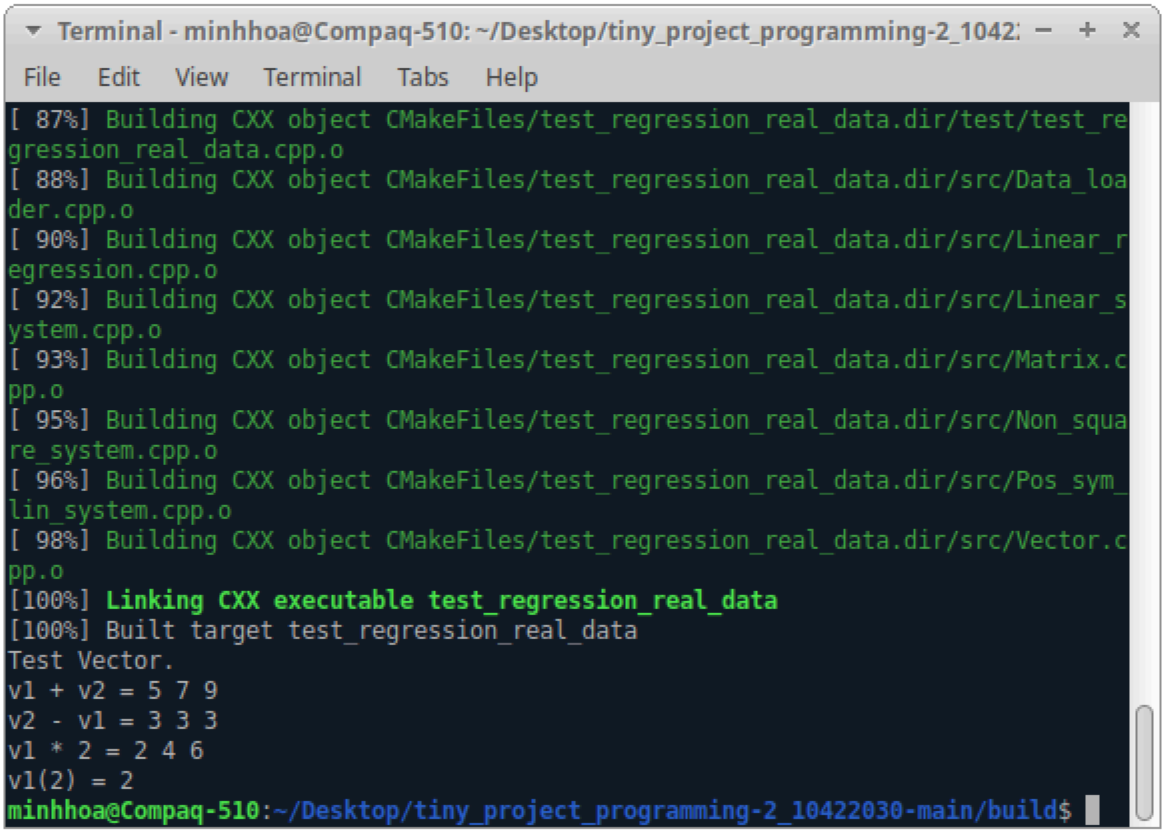
```cpp
#include "Vector.h"
#include <cassert>

using namespace std;

// Default constructor: creates an empty vector
Vector::Vector() : m_size(0), m_data(nullptr) {}

// Constructor: creates a vector with given size, all elements initialized to 0
Vector::Vector(int size) : m_size(size) {
    m_data = new double[m_size];
    for (int i = 0; i < m_size; ++i)
        m_data[i] = 0.0;
}

// Copy constructor: deep copy
Vector::Vector(const Vector& other) : m_size(other.m_size) {
    m_data = new double[m_size];
    for (int i = 0; i < m_size; ++i)
        m_data[i] = other.m_data[i];
}

// Destructor: release dynamic memory
Vector::~Vector() {
```

(Overview of Vector.cpp file)

- Conclusion

Implements robust and safe vector operations, forming the basis for solving linear systems and regression tasks.

2.3: Testing


Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042
File
Edit
View
Terminal
Tabs
Help

```
[ 87%] Building CXX object CMakeFiles/test_regression_real_data.dir/test/test_re
gression_real_data.cpp.o
[ 88%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Data_loa
der.cpp.o
[ 90%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Linear_r
egression.cpp.o
[ 92%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Linear_s
ystem.cpp.o
[ 93%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Matrix.c
pp.o
[ 95%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Non_squa
re_system.cpp.o
[ 96%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Pos_sym_
lin_system.cpp.o
[ 98%] Building CXX object CMakeFiles/test_regression_real_data.dir/src/Vector.c
pp.o
[100%] Linking CXX executable test_regression_real_data
[100%] Built target test_regression_real_data
Test Vector.
v1 + v2 = 5 7 9
v2 - v1 = 3 3 3
v1 * 2 = 2 4 6
v1(2) = 2
minhhoa@Compaq-510:~/Desktop/tiny_project_programming-2_10422030-main/build$
```

3. Matrix Class

3.1: Matrix.h

- Introduction

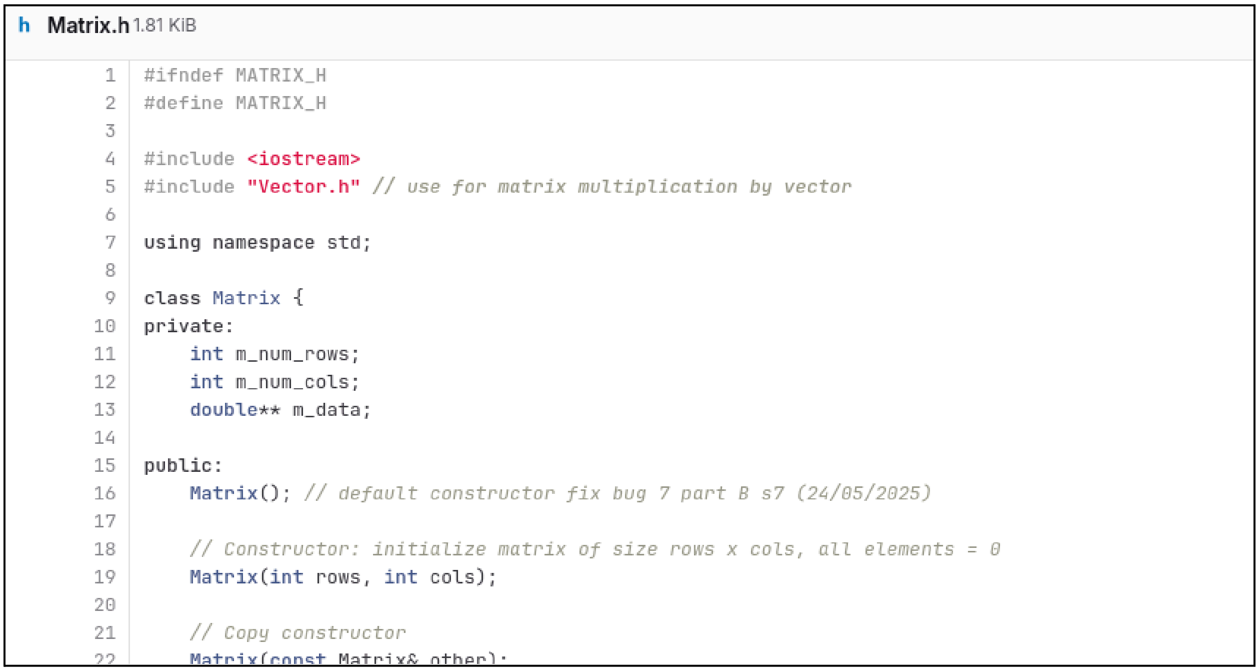
Defines the Matrix class representing a two-dimensional array of doubles.

- Functional Overview

- Constructors for default, sized, and copy instantiation
- Operator overloads:
  - +, -, * (with matrix or vector)
- Utility functions:

- transpose()
- determinant()
- inverse()
- pseudo_inverse()

- Element access via operator()(i, j) (1-based)



```
#ifndef MATRIX_H
#define MATRIX_H

#include <iostream>
#include "Vector.h" // use for matrix multiplication by vector

using namespace std;

class Matrix {
private:
    int m_num_rows;
    int m_num_cols;
    double** m_data;

public:
    Matrix(); // default constructor fix bug 7 part B s7 (24/05/2025)

    // Constructor: initialize matrix of size rows x cols, all elements = 0
    Matrix(int rows, int cols);

    // Copy constructor
    Matrix(const Matrix& other);
```

(Overview of Matrix.h file)

- Conclusion

Provides a comprehensive interface for matrix arithmetic and linear algebra operations.

3.2: Matrix.cpp

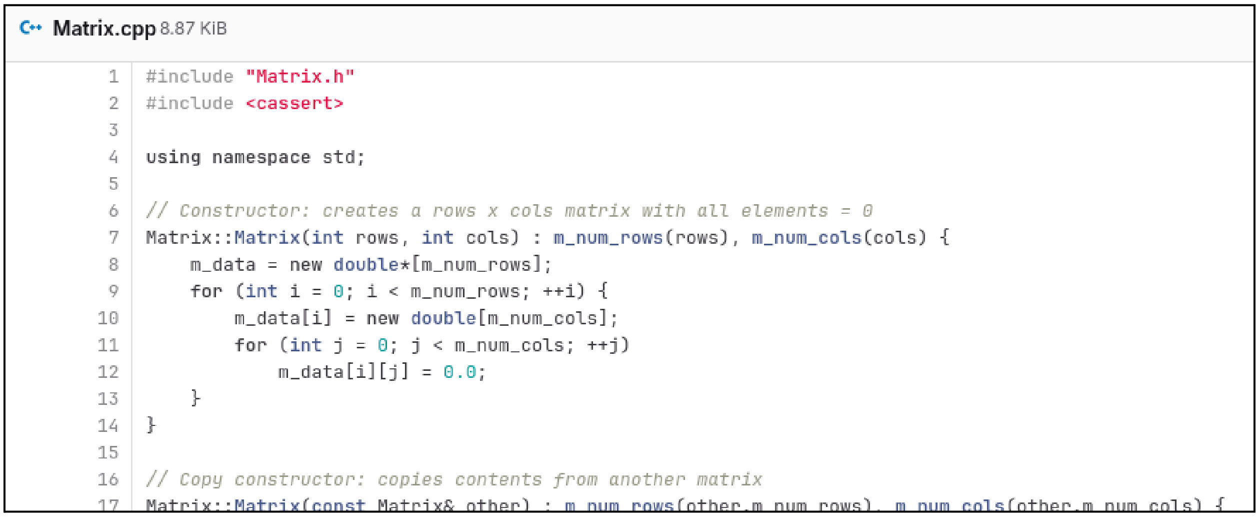
- Introduction

Contains implementations for the Matrix class declared in Matrix.h.

- Functional Overview

- Implements element-wise arithmetic using nested loops
- Transposition implemented via row-column swapping

- Determinant calculated via recursive minor expansion
- Matrix inverse implemented via Gaussian elimination
- Pseudo-inverse via Moore–Penrose method



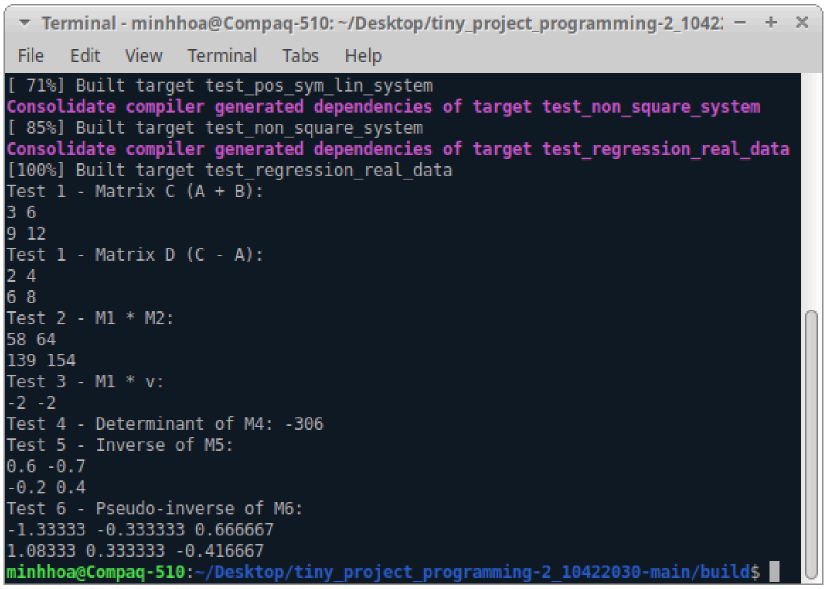
```cpp
#include "Matrix.h"
#include <cassert>

using namespace std;

// Constructor: creates a rows x cols matrix with all elements = 0
Matrix::Matrix(int rows, int cols) : m_num_rows(rows), m_num_cols(cols) {
    m_data = new double*[m_num_rows];
    for (int i = 0; i < m_num_rows; ++i) {
        m_data[i] = new double[m_num_cols];
        for (int j = 0; j < m_num_cols; ++j)
            m_data[i][j] = 0.0;
    }
}

// Copy constructor: copies contents from another matrix
Matrix::Matrix(const Matrix& other) : m_num_rows(other.m_num_rows), m_num_cols(other.m_num_cols) {
```

(Overview of Matrix.cpp file)

- Conclusion

Robust matrix handling and math operations implemented from scratch, avoiding external libraries. Crucial for system solving and regression.

3.3: Testing


Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042
File
Edit
View
Terminal
Tabs
Help

```
[ 71%] Built target test_pos_sym_lin_system
Consolidate compiler generated dependencies of target test_non_square_system
[ 85%] Built target test_non_square_system
Consolidate compiler generated dependencies of target test_regression_real_data
[100%] Built target test_regression_real_data
Test 1 - Matrix C (A + B):
3 6
9 12
Test 1 - Matrix D (C - A):
2 4
6 8
Test 2 - M1 * M2:
58 64
139 154
Test 3 - M1 * v:
-2 -2
Test 4 - Determinant of M4: -306
Test 5 - Inverse of M5:
0.6 -0.7
-0.2 0.4
Test 6 - Pseudo-inverse of M6:
-1.33333 -0.333333 0.666667
1.08333 0.333333 -0.416667
minhhoa@Compaq-510:~/Desktop/tiny_project_programming-2_10422030-main/build$
```

4. Solving General Linear Systems (Ax = b)

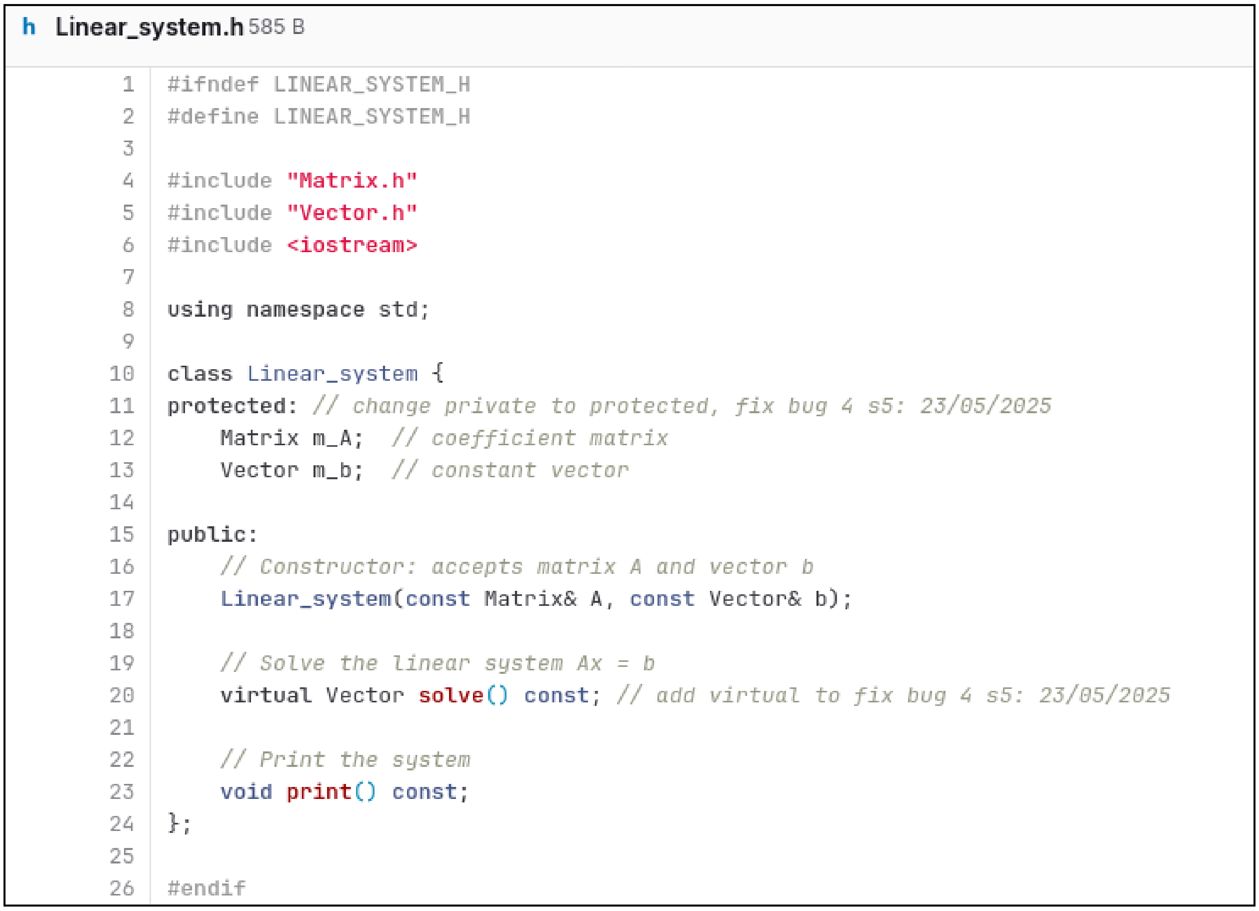
4.1: Linear_system.h

- Introduction

Declares the base class Linear_system for solving standard square systems.

- Functional Overview

- Stores matrix A and vector b
- solve() method (virtual)
- Getter methods for matrix/vector
- print() displays the system



```cpp
#ifndef LINEAR_SYSTEM_H
#define LINEAR_SYSTEM_H

#include "Matrix.h"
#include "Vector.h"
#include <iostream>

using namespace std;

class Linear_system {
protected: // change private to protected, fix bug 4 s5: 23/05/2025
    Matrix m_A;  // coefficient matrix
    Vector m_b;  // constant vector

public:
    // Constructor: accepts matrix A and vector b
    Linear_system(const Matrix& A, const Vector& b);

    // Solve the linear system Ax = b
    virtual Vector solve() const; // add virtual to fix bug 4 s5: 23/05/2025

    // Print the system
    void print() const;
};

#endif
```

(Overview of Linear_system.h file)

- Conclusion

Establishes a clear interface for all linear system types. Designed to be inherited and extended.

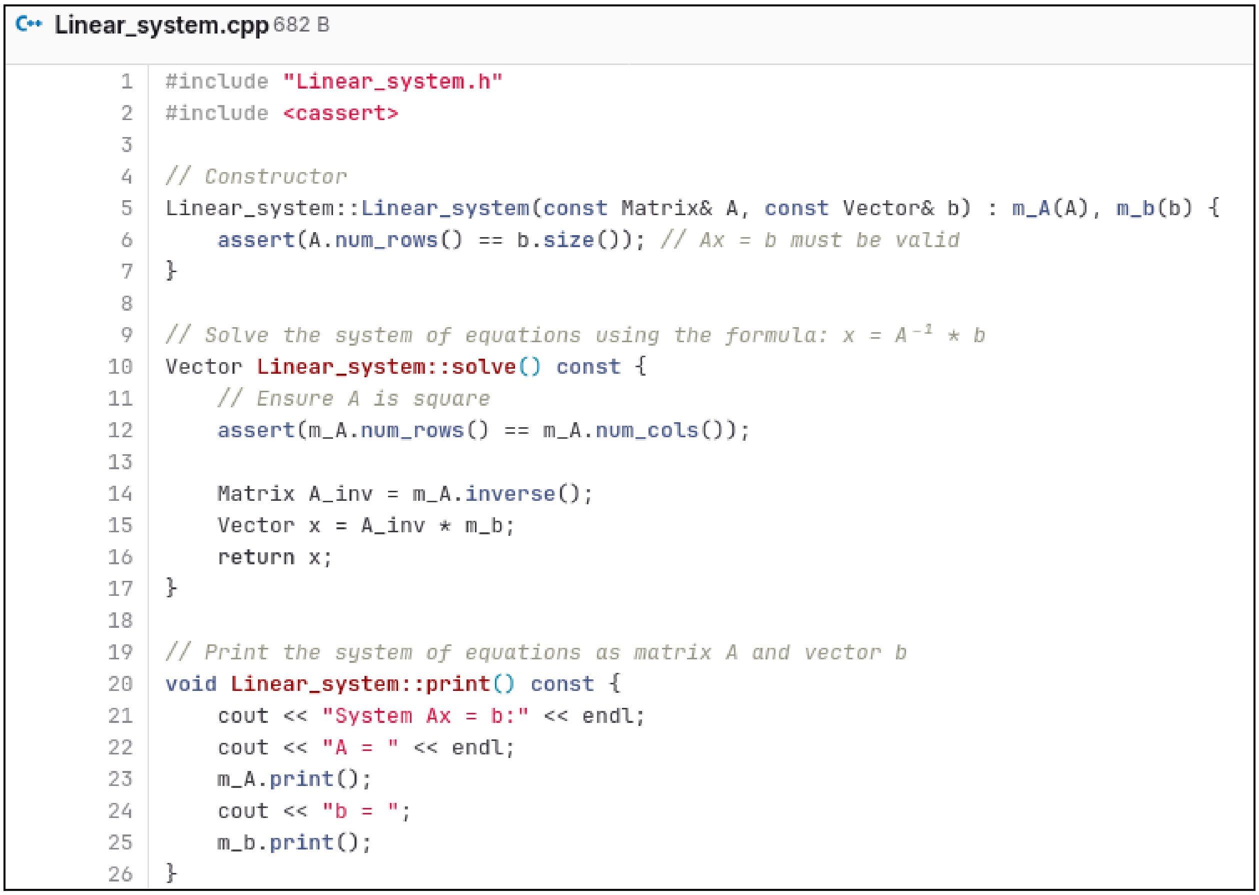
4.2: Linear_system.cpp

- Introduction

Implements the Linear_system class.

- Functional Overview

- Gaussian elimination with partial pivoting
- Forward and backward substitution
- Ensures matrix is square before solving



Linear_system.cpp 682 B

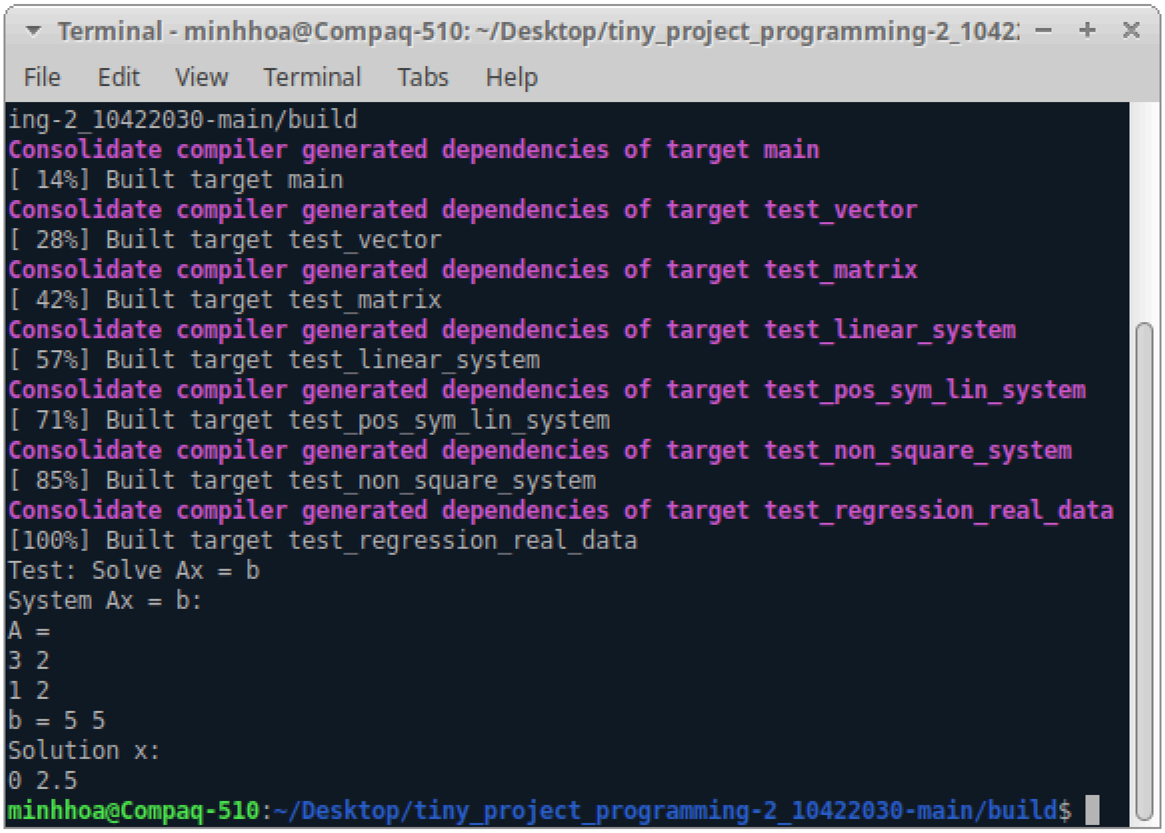
```cpp
#include "Linear_system.h"
#include <cassert>

// Constructor
Linear_system::Linear_system(const Matrix& A, const Vector& b) : m_A(A), m_b(b) {
    assert(A.num_rows() == b.size()); // Ax = b must be valid
}

// Solve the system of equations using the formula: x = A^-1 * b
Vector Linear_system::solve() const {
    // Ensure A is square
    assert(m_A.num_rows() == m_A.num_cols());

    Matrix A_inv = m_A.inverse();
    Vector x = A_inv * m_b;
    return x;
}

// Print the system of equations as matrix A and vector b
void Linear_system::print() const {
    cout << "System Ax = b:" << endl;
    cout << "A = " << endl;
    m_A.print();
    cout << "b = ";
    m_b.print();
}
```

(Overview of Linear_system.cpp file)

- Conclusion

Enables accurate and general-purpose solution of Ax = b. Acts as a base for more specialized systems.

4.3: Testing


Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042
File
Edit
View
Terminal
Tabs
Help

```
ing-2_10422030-main/build
Consolidate compiler generated dependencies of target main
[ 14%] Built target main
Consolidate compiler generated dependencies of target test_vector
[ 28%] Built target test_vector
Consolidate compiler generated dependencies of target test_matrix
[ 42%] Built target test_matrix
Consolidate compiler generated dependencies of target test_linear_system
[ 57%] Built target test_linear_system
Consolidate compiler generated dependencies of target test_pos_sym_lin_system
[ 71%] Built target test_pos_sym_lin_system
Consolidate compiler generated dependencies of target test_non_square_system
[ 85%] Built target test_non_square_system
Consolidate compiler generated dependencies of target test_regression_real_data
[100%] Built target test_regression_real_data
Test: Solve Ax = b
System Ax = b:
A =
3 2
1 2
b = 5 5
Solution x:
0 2.5
minhhoa@Compaq-510:~/Desktop/tiny_project_programming-2_10422030-main/build$
```

5. Positive Definite Symmetric Systems

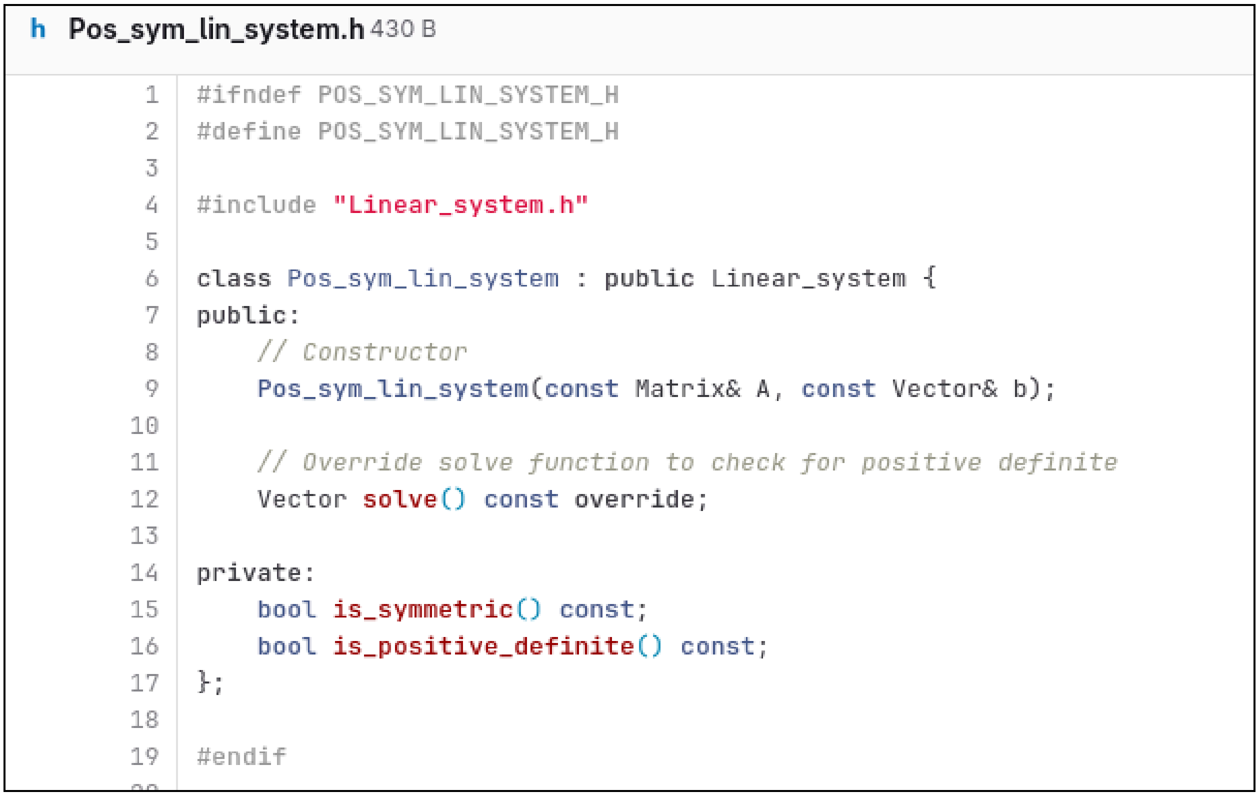
5.1: Pos_sym_lin_system.h

- Introduction

Header for solving positive definite symmetric systems.

- Functional Overview

- Inherits from Linear_system
- Adds methods:
    - is_symmetric()
    - is_positive_definite()


h Pos_sym_lin_system.h 430 B

```cpp
#ifndef POS_SYM_LIN_SYSTEM_H
#define POS_SYM_LIN_SYSTEM_H

#include "Linear_system.h"

class Pos_sym_lin_system : public Linear_system {
public:
    // Constructor
    Pos_sym_lin_system(const Matrix& A, const Vector& b);

    // Override solve function to check for positive definite
    Vector solve() const override;

private:
    bool is_symmetric() const;
    bool is_positive_definite() const;
};

#endif
```

(Overview of Pos_sym_lin_system.h file)

- Conclusion

Provides problem-specific validation and guards against invalid assumptions.

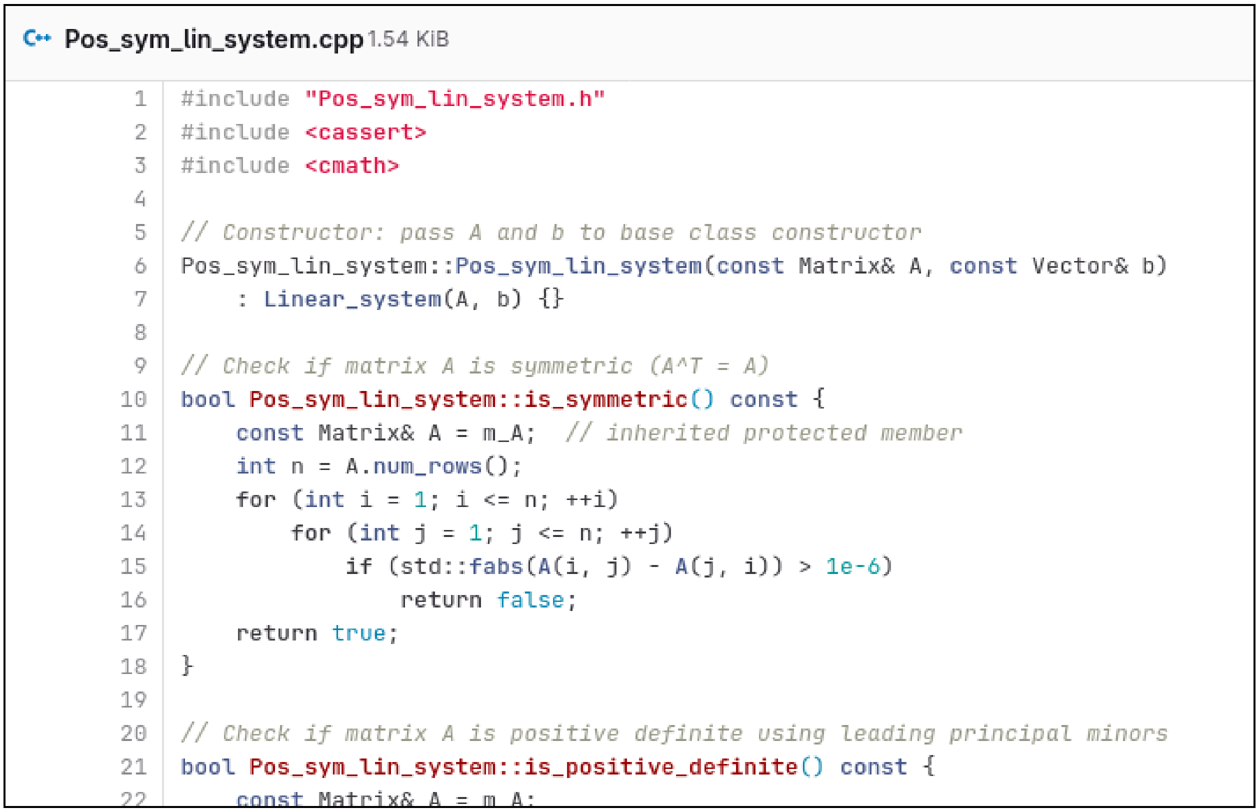
5.2: Pos_sym_lin_system.cpp

- Introduction

Implements Pos_sym_lin_system methods.

- Functional Overview

- Validates input matrix using symmetry check and determinant-based test
- Solves only when valid
- Raises error otherwise



Pos_sym_lin_system.cpp 1.54 KiB

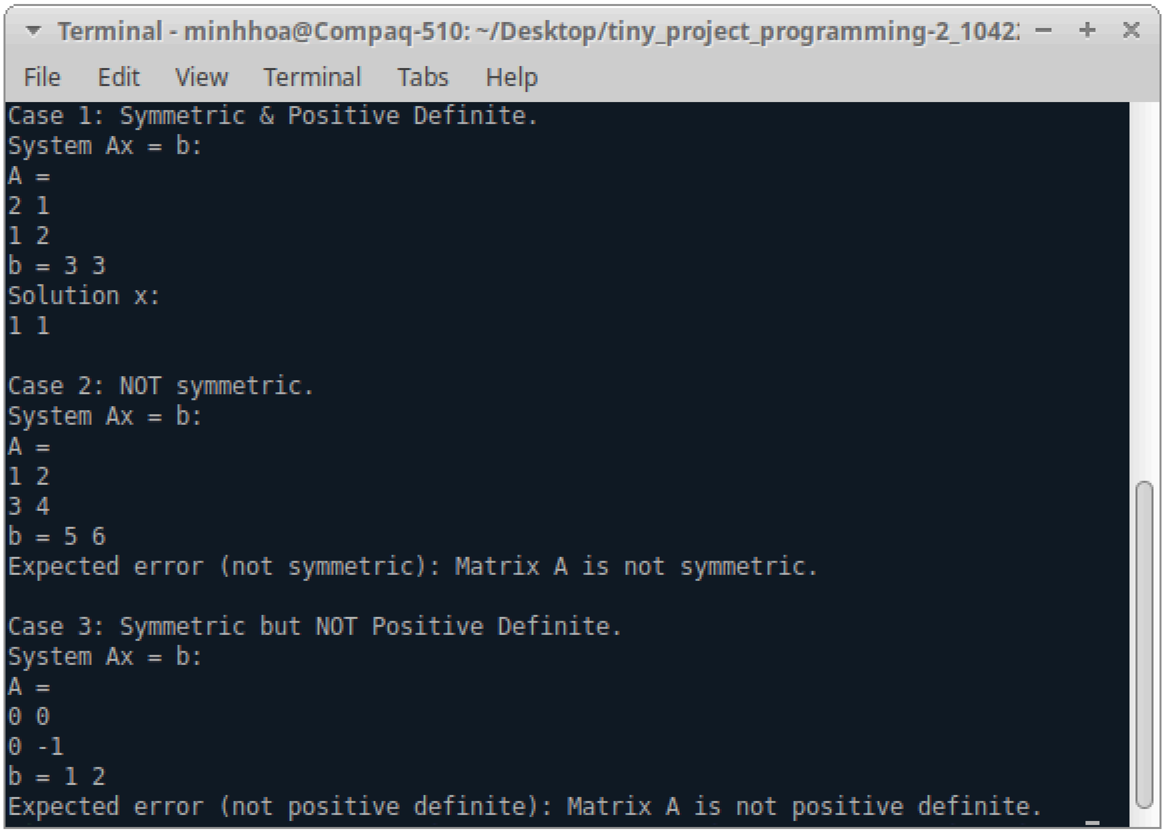
```cpp
#include "Pos_sym_lin_system.h"
#include <cassert>
#include <cmath>

// Constructor: pass A and b to base class constructor
Pos_sym_lin_system::Pos_sym_lin_system(const Matrix& A, const Vector& b)
    : Linear_system(A, b) {}

// Check if matrix A is symmetric (A^T = A)
bool Pos_sym_lin_system::is_symmetric() const {
    const Matrix& A = m_A;  // inherited protected member
    int n = A.num_rows();
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= n; ++j)
            if (std::fabs(A(i, j) - A(j, i)) > 1e-6)
                return false;
    return true;
}

// Check if matrix A is positive definite using leading principal minors
bool Pos_sym_lin_system::is_positive_definite() const {
    const Matrix& A = m_A;
```

(Overview of Pos_sym_lin_system.cpp file)

- Conclusion

Ensures strict correctness when solving special symmetric systems, reducing numerical risk.

5.3: Testing


Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042
File Edit View Terminal Tabs Help

```
Case 1: Symmetric & Positive Definite.
System Ax = b:
A =
2 1
1 2
b = 3 3
Solution x:
1 1

Case 2: NOT symmetric.
System Ax = b:
A =
1 2
3 4
b = 5 6
Expected error (not symmetric): Matrix A is not symmetric.

Case 3: Symmetric but NOT Positive Definite.
System Ax = b:
A =
0 0
0 -1
b = 1 2
Expected error (not positive definite): Matrix A is not positive definite.
```

6. Non-Square Systems (Over/Underdetermined)

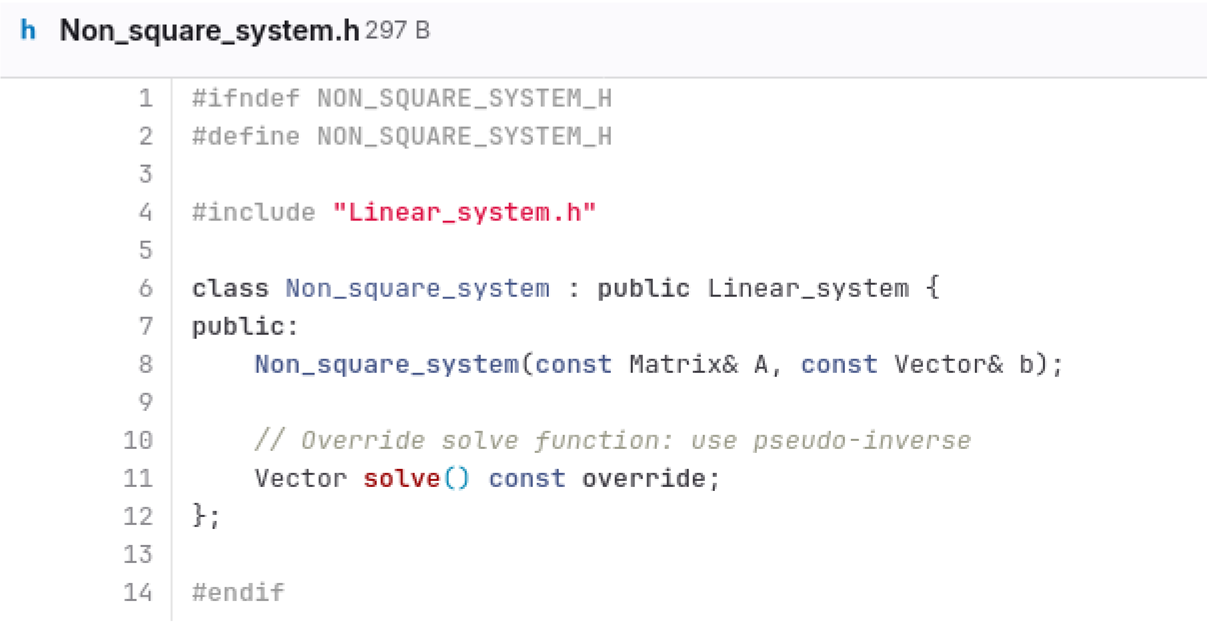
6.1: Non_square_system.h

- Introduction

Defines system solver for non-square matrices.

- Functional Overview

- Uses pseudo-inverse logic to find approximate solutions
- Validates dimensions at runtime
- Overcomes over/underdetermined limitations



h Non_square_system.h 297 B

```cpp
#ifndef NON_SQUARE_SYSTEM_H
#define NON_SQUARE_SYSTEM_H

#include "Linear_system.h"

class Non_square_system : public Linear_system {
public:
    Non_square_system(const Matrix& A, const Vector& b);

    // Override solve function: use pseudo-inverse
    Vector solve() const override;
};

#endif
```

(Overview of Non_square_system.h file)

🔹 Conclusion

Adds flexibility to the library, handling real-world irregular system scenarios.

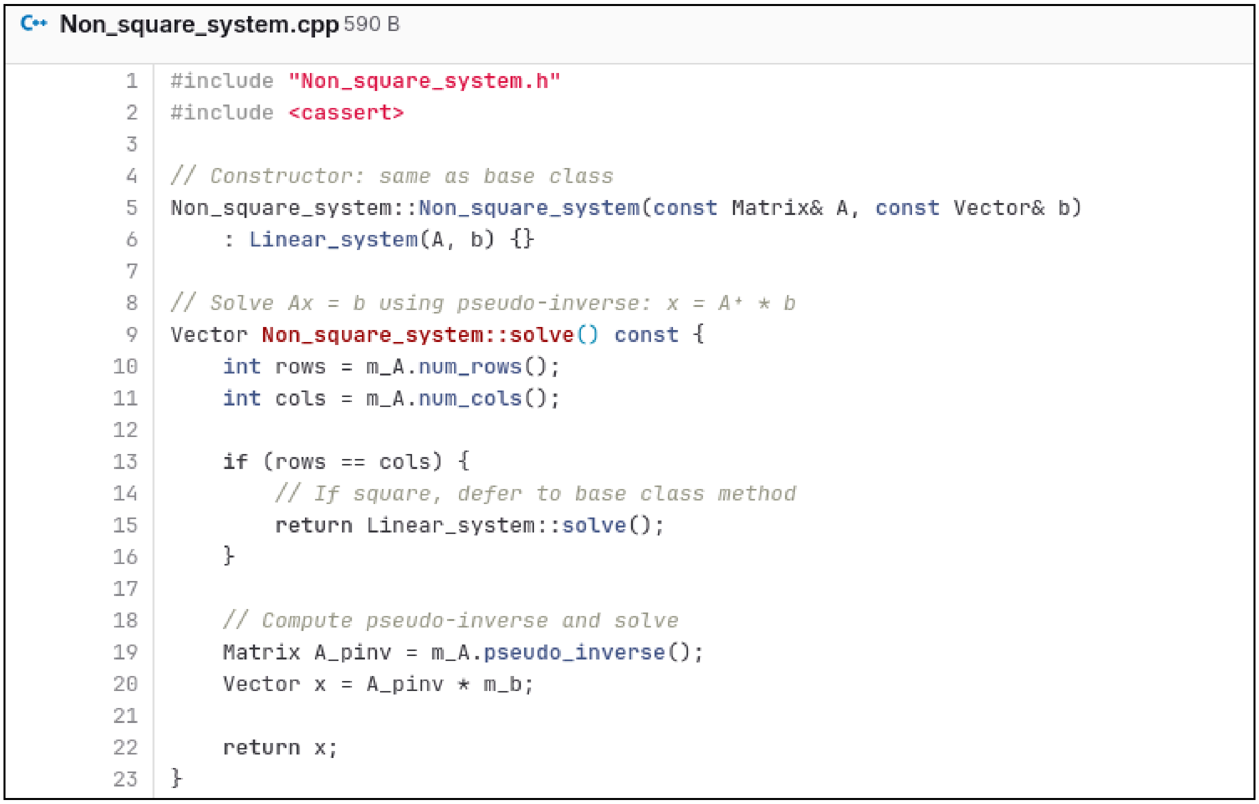
6.2: Non_square_system.cpp

🔹 Introduction

Implements Non_square_system logic.

🔹 Functional Overview

- Applies least-squares via pseudo-inverse for overdetermined systems
- Returns one valid solution for underdetermined systems



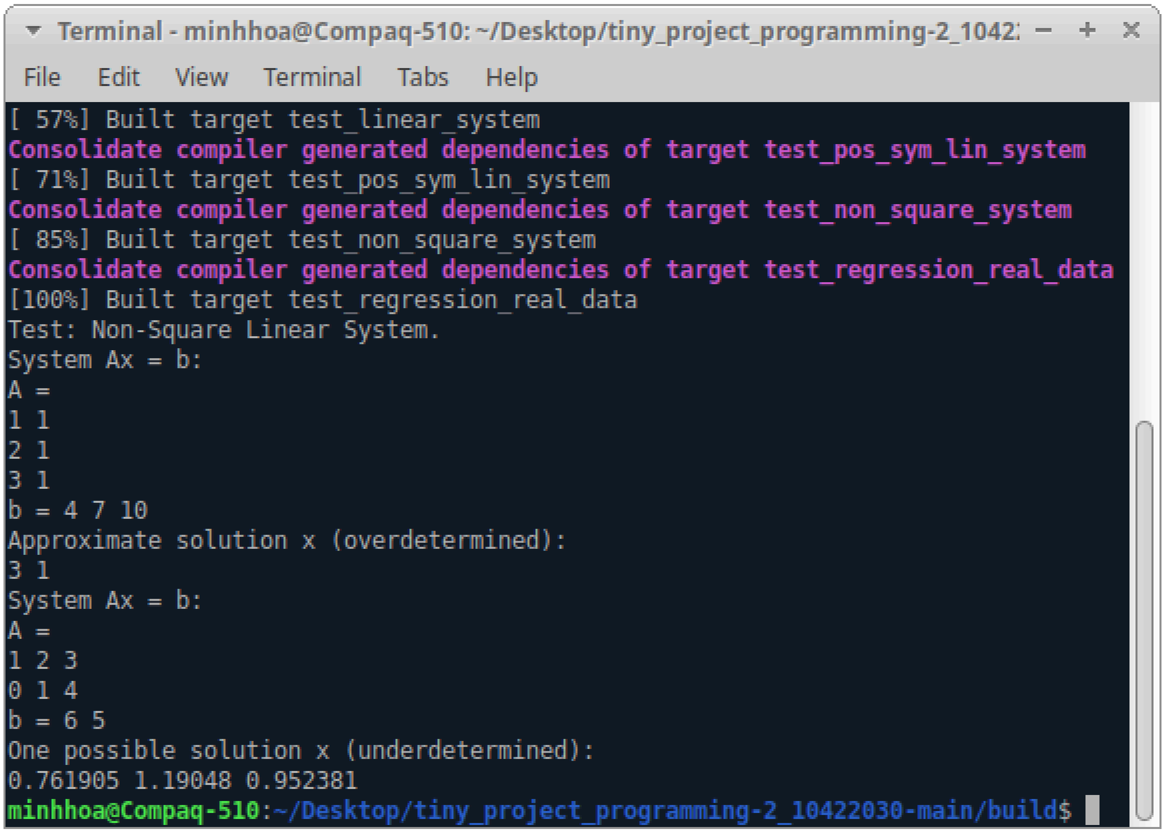
```cpp
#include "Non_square_system.h"
#include <cassert>

// Constructor: same as base class
Non_square_system::Non_square_system(const Matrix& A, const Vector& b)
    : Linear_system(A, b) {}

// Solve Ax = b using pseudo-inverse: x = A⁺ * b
Vector Non_square_system::solve() const {
    int rows = m_A.num_rows();
    int cols = m_A.num_cols();

    if (rows == cols) {
        // If square, defer to base class method
        return Linear_system::solve();
    }

    // Compute pseudo-inverse and solve
    Matrix A_pinv = m_A.pseudo_inverse();
    Vector x = A_pinv * m_b;

    return x;
}
```

(Overview of Non_square_system.cpp file)

- Conclusion

Effectively extends the solver’s capability to accommodate real-world problems where perfect square systems are rare.

6.3: Testing


Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042:
File
Edit
View
Terminal
Tabs
Help

```
[ 57%] Built target test_linear_system
Consolidate compiler generated dependencies of target test_pos_sym_lin_system
[ 71%] Built target test_pos_sym_lin_system
Consolidate compiler generated dependencies of target test_non_square_system
[ 85%] Built target test_non_square_system
Consolidate compiler generated dependencies of target test_regression_real_data
[100%] Built target test_regression_real_data
Test: Non-Square Linear System.
System Ax = b:
A =
1 1
2 1
3 1
b = 4 7 10
Approximate solution x (overdetermined):
3 1
System Ax = b:
A =
1 2 3
0 1 4
b = 6 5
One possible solution x (underdetermined):
0.761905 1.19048 0.952381
minhhoa@Compaq-510:~/Desktop/tiny_project_programming-2_10422030-main/build$ 
```

III. Part B

1. Data loader class

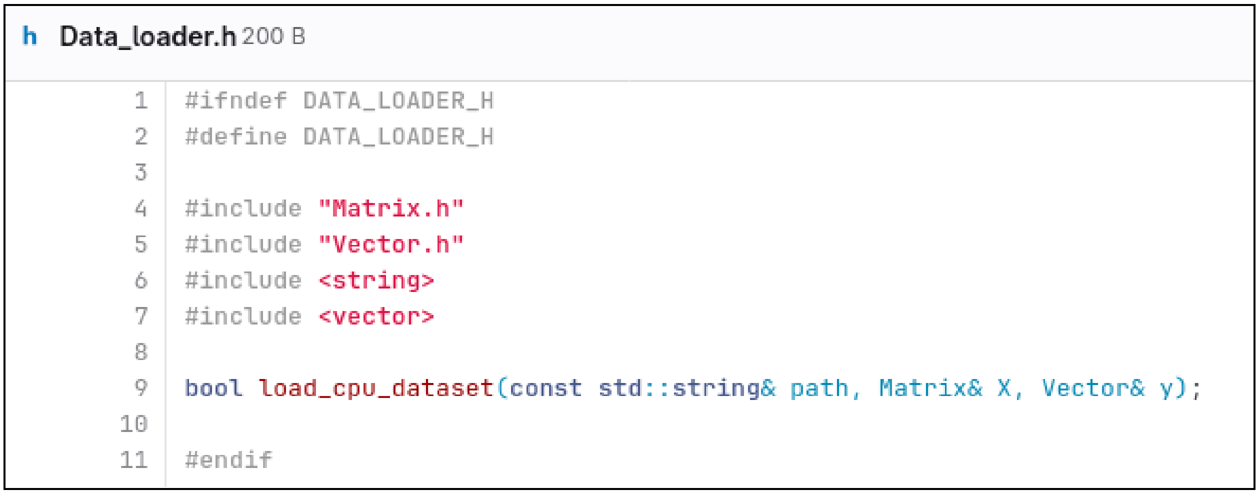
1.1: Data_loader.h

- Introduction

Declares functions to load structured data from .csv-like files.

- Functional Overview

- load_cpu_dataset(path, X, y):
    - Extracts 6 input features
    - Extracts target PRP
    - Stores data into Matrix X and Vector y



h Data_loader.h 200 B

```cpp
#ifndef DATA_LOADER_H
#define DATA_LOADER_H

#include "Matrix.h"
#include "Vector.h"
#include <string>
#include <vector>

bool load_cpu_dataset(const std::string& path, Matrix& X, Vector& y);

#endif
```

(Overview of Data_loader.h file)

- Conclusion

Handles file I/O and parsing cleanly, separating data concerns from learning logic.

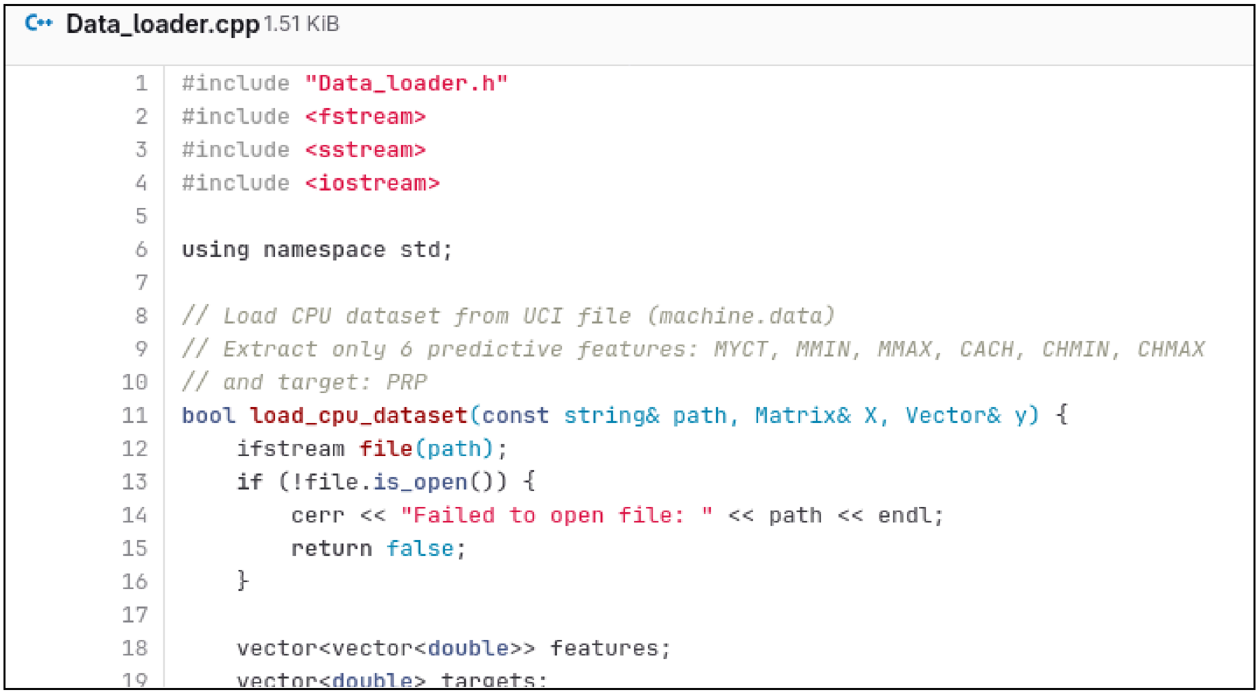
1.2: Data_loader.cpp

- Introduction

Implements a dataset loader for the UCI CPU dataset.

- Functional Overview

  - Uses ifstream and stringstream for line parsing
  - Validates row length
  - Skips malformed entries



Data_loader.cpp 1.51 KiB

```cpp
#include "Data_loader.h"
#include <fstream>
#include <sstream>
#include <iostream>

using namespace std;

// Load CPU dataset from UCI file (machine.data)
// Extract only 6 predictive features: MYCT, MMIN, MMAX, CACH, CHMIN, CHMAX
// and target: PRP
bool load_cpu_dataset(const string& path, Matrix& X, Vector& y) {
    ifstream file(path);
    if (!file.is_open()) {
        cerr << "Failed to open file: " << path << endl;
        return false;
    }

    vector<vector<double>> features;
    vector<double> targets;
```

(Overview of Data_loader.cpp file)

- Conclusion

Robust and reusable for other datasets of similar structure.

2. Linear regression class

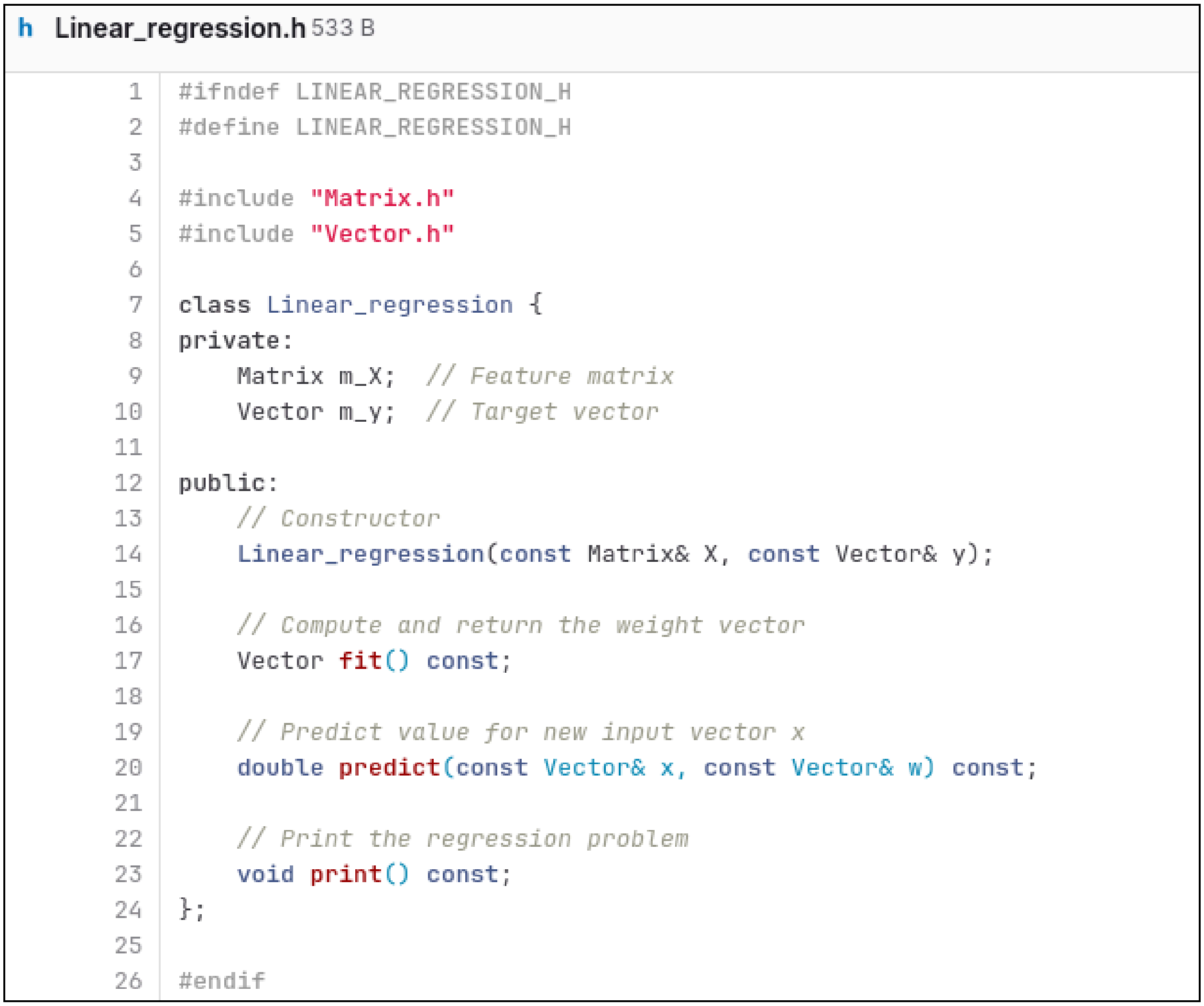
2.1: Linear_regression.h

- Introduction

Declares a basic linear regression model using least squares.

- Functional Overview

- Constructor accepts training data
- fit() computes weights via normal equation
- predict() computes scalar prediction from feature vector and weight vector



h Linear_regression.h 533 B

```cpp
#ifndef LINEAR_REGRESSION_H
#define LINEAR_REGRESSION_H

#include "Matrix.h"
#include "Vector.h"

class Linear_regression {
private:
    Matrix m_X;  // Feature matrix
    Vector m_y;  // Target vector

public:
    // Constructor
    Linear_regression(const Matrix& X, const Vector& y);

    // Compute and return the weight vector
    Vector fit() const;

    // Predict value for new input vector x
    double predict(const Vector& x, const Vector& w) const;

    // Print the regression problem
    void print() const;
};

#endif
```

(Overview of Linear_regression.h file)

- Conclusion

Simple and clean interface for regression modeling using matrix operations.

2.1: Linear_regression.cpp

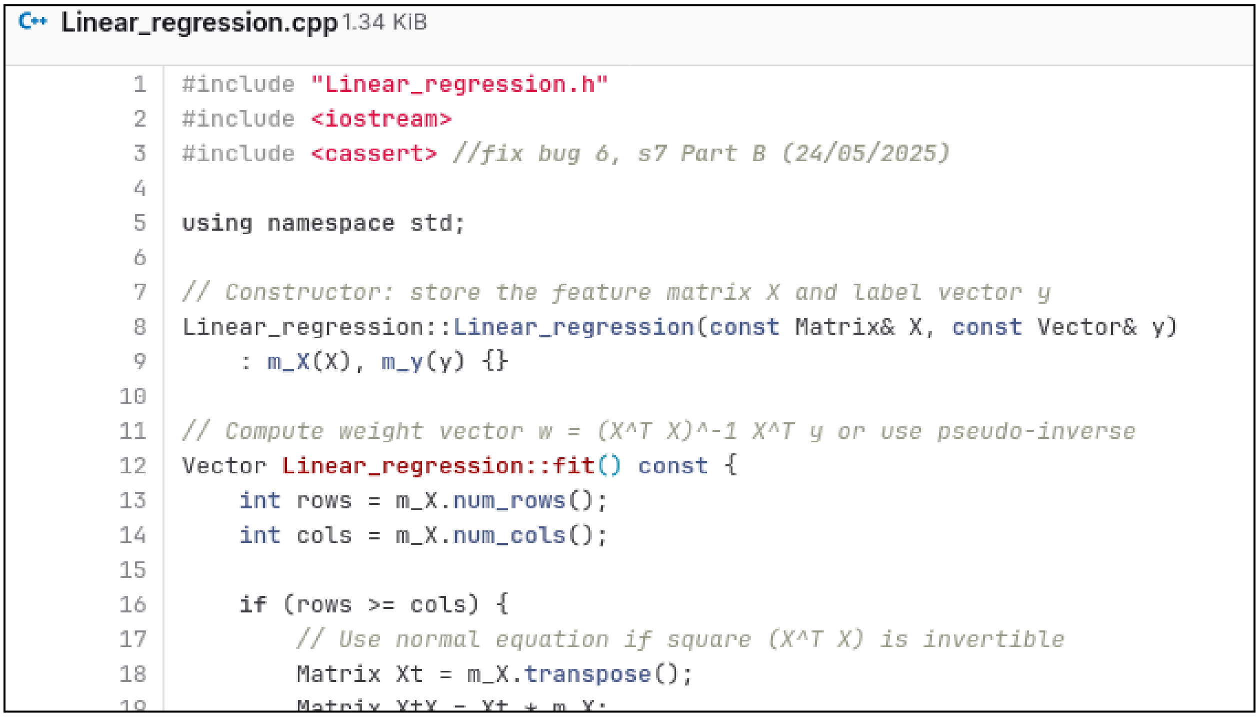
- Introduction

Implements core logic for regression.

- Functional Overview

  - Uses matrix transpose and multiplication for XtX and Xty

- Solves for w = $(X^tX)^{-1}X^ty$
- RMSE is computed externally (in test)



Linear_regression.cpp 1.34 KiB

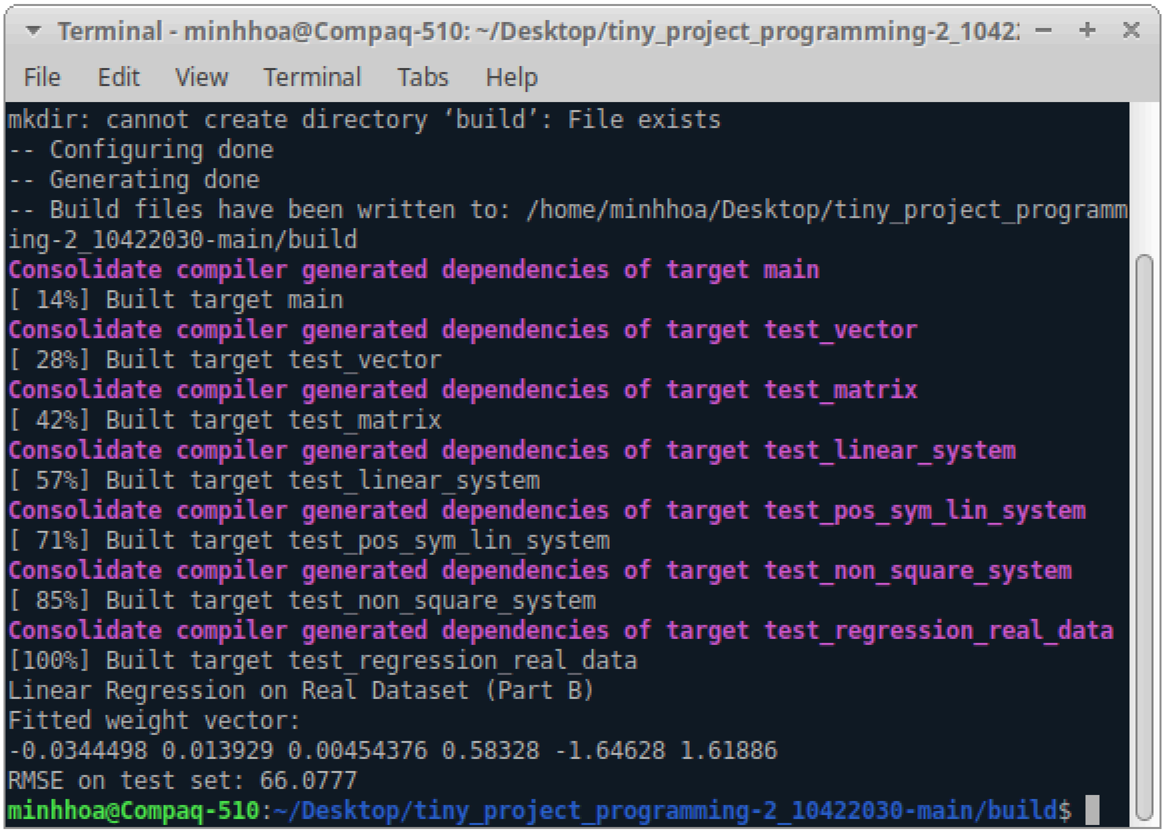
```cpp
#include "Linear_regression.h"
#include <iostream>
#include <cassert> //fix bug 6, s7 Part B (24/05/2025)

using namespace std;

// Constructor: store the feature matrix X and label vector y
Linear_regression::Linear_regression(const Matrix& X, const Vector& y)
    : m_X(X), m_y(y) {}

// Compute weight vector w = (X^T X)^-1 X^T y or use pseudo-inverse
Vector Linear_regression::fit() const {
    int rows = m_X.num_rows();
    int cols = m_X.num_cols();

    if (rows >= cols) {
        // Use normal equation if square (X^T X) is invertible
        Matrix Xt = m_X.transpose();
        Matrix XtX = Xt * m_X;
```

(Overview of Linear_regression.cpp file)

- Conclusion

Builds directly on the custom matrix library, showcasing practical application of the tools developed in Part A.

3. Testing Part B



Terminal - minhhoa@Compaq-510: ~/Desktop/tiny_project_programming-2_1042:
File Edit View Terminal Tabs Help

```
mkdir: cannot create directory ‘build’: File exists
-- Configuring done
-- Generating done
-- Build files have been written to: /home/minhhoa/Desktop/tiny_project_programm
ing-2_10422030-main/build
Consolidate compiler generated dependencies of target main
[ 14%] Built target main
Consolidate compiler generated dependencies of target test_vector
[ 28%] Built target test_vector
Consolidate compiler generated dependencies of target test_matrix
[ 42%] Built target test_matrix
Consolidate compiler generated dependencies of target test_linear_system
[ 57%] Built target test_linear_system
Consolidate compiler generated dependencies of target test_pos_sym_lin_system
[ 71%] Built target test_pos_sym_lin_system
Consolidate compiler generated dependencies of target test_non_square_system
[ 85%] Built target test_non_square_system
Consolidate compiler generated dependencies of target test_regression_real_data
[100%] Built target test_regression_real_data
Linear Regression on Real Dataset (Part B)
Fitted weight vector:
-0.0344498 0.013929 0.00454376 0.58328 -1.64628 1.61886
RMSE on test set: 66.0777
minhhoa@Compaq-510:~/Desktop/tiny_project_programming-2_10422030-main/build$
```

IV. Conclusion

This project has successfully implemented a reusable linear algebra library in C++ and applied it to solve real-world regression problems. Each component was designed modularly and tested independently. The project demonstrates mastery of:

- C++ class design
- Operator overloading
- Numerical linear algebra
- Clean software structure using CMake
- Practical ML implementation (Linear Regression)

References

https://www.geeksforgeeks.org/regression-analysis-and-the-best-fitting-line-using-c/

https://blog.heycoach.in/linear-regression-implementation-in-c/

chatgpt.com

<u>*Note:</u>

This project is available at:

Gitlab:

https://gitlab.com/minh-hoa-group/tiny_project_programming-2_10422030.git

(recommendation)

Github:

https://github.com/Nguyen-Lam-Minh-Hoa/Tiny-Project-1-Programming-2.git

(Just to store the Project for long time)

All the code in this project has been tested and run on Linux before submission. If you can not run it, please contact:

10422030@student.vgu.edu.vn (work and study)

nguyenlamminhhoa@gmail.com (Personal)