📊 Tiny Project Report

Name: Doan Le Gia Bao
Student ID: 10423008
Course: Programming 2

---

1. Introduction

This project is divided into two main parts:

✅ **Part A** focuses on building classes in C++, including `Vector`, `Matrix`, `LinearSystem`, and `GeneralLinearSystem` classes, with support for solving:

- Square systems
- Underdetermined systems
- Overdetermined systems

✅ **Part B** applies the developed tools to a **real-world machine learning task**: predicting **CPU performance** using **linear regression** on the **UCI Computer Hardware dataset**.

---

2. Part A – Building Classes

2.1 Vector Class

Implements **1D dynamic arrays** with custom memory management.

◇ **Constructors and Destructor:**

- Default, parameterized, and copy constructors

- Destructor

◇ **Overloaded Operators:**

- Assignment (`=`), negation (`-`), addition (`+`), subtraction (`-`)
- Scalar multiplication (`*`) and dot product (`·`)
- Indexing with both `[]` and `()` (1-based)

◇ **Additional methods:**

- Get the size of vector (`size`)
- Print all elements of vector (`print`)

◇ Designed for use as **vectors of unknowns** and **right-hand sides** in linear systems.

2.2 Matrix Class

2D dynamic matrix with safe memory management.

◇ **Constructors and Destructor:**

- Default, parameterized, and copy constructors
- Destructor

◇ **Supports:**

- Matrix addition (`+`), subtraction (`-`), multiplication (`*`) of suitably sized matrices, vectors, and scalars
- Transpose (`tranpose`), determinant (`determinant`), inverse (`inverse`), **Moore-Penrose pseudoinverse** (`pseudoInverse`)
- **Assert-based dimension checks** for safety

◇ **Additional methods:**

- Get the number of rows of matrix (`getNumRows`)
- Get the number of columns of matrix (`getNumCols`)
- Print all elements of vector (`print`)

2.3 LinearSystem, PosSymLinSystem

◇ `LinearSystem`

- Solves square systems $\boldsymbol{Ax} = \boldsymbol{b}$ using **Gaussian elimination with partial pivoting**.

◇ `PosSymLinSystem`

- Inherits from `LinearSystem`, uses **Conjugate Gradient method** for solving **symmetric positive-definite systems**.
- Automatically checks **symmetry**.

2.4 GeneralLinearSystem

Solves **non-square systems**:

- **Overdetermined** → Least-squares solution using **Moore-Penrose** or **Tikhonov regularization**.
- **Underdetermined** → Least-squares solution using **Moore-Penrose** or **Tikhonov regularization**..

◇ Forms the **computational core** for Part B’s regression task.

---

3. Part B – Linear Regression on CPU Performance

3.1 Objective

Build a **linear regression model** for predicting **PRP** (Published Relative Performance):

PRP=x1·MYCT+x2·MMIN+x3·MMAX+x4·CACH+x5·CHMIN+x6·CHMAX

3.2 Dataset

📂 **Source:** [UCI Computer Hardware Dataset]

- **209 instances** with **10 attributes** (6 features used for modeling).

3.3 Methodology

1. **Preprocessed dataset** to extract relevant numeric fields.
2. **Randomized 80/20 train-test split** using C++ `shuffle`.
3. Used `GeneralLinearSystem` to solve for weights via **Moore-Penrose pseudoinverse**.
4. Evaluated predictions using **Root Mean Square Error (RMSE)**.

3.4 Sample Output

Learned model coefficients:

```
Model coefficients (x1 to x6):
-0.0374181 0.0135913 0.00476773 0.565409 -0.505904 1.27356
Root mean square error: 41.8398
```

---

4. Conclusion

This project provided **hands-on experience** with:

- Core **linear algebra techniques**
- **Machine learning** applications
- **Numerical stability** and optimization

🎯 Final Thoughts

This project bridges **theory & practice**, demonstrating how **linear algebra powers machine learning**. Future work could extend this to **deep learning optimizations**.

📌 Appendix

◇ **Code Repository:** tinyProject