Generic Programming in C++ Project
LA-Library

Team Shadow

Amogh Bharne - PES1UG21CS072

a. Introduction

This project "LA-library" aims to provide functionality for calculations relating to linear algebra concepts. This includes operations on Vectors and Matrices. Additionally, there is functionality to auto differentiate a few number of essential functions, which further help in these calculations.

The library is a header-only library, which consists of custom implementation of Matrix, Vector and ADVariable(for auto differentiation) classes. The library supports various operations on these classes.

There is extensive use of templates and other generic programming features throughout, which are listed below.

b. Generic Programming features used:

1. **Matrix Class**:
    - **Templates**:
        - The `Matrix` class is implemented using templates for the element type `T` and the dimensions `Rows` and `Cols`.
    - **Concepts and Constraints**:
        - The `Numeric` concept is used to constrain the element type `T` to be a numeric type.
        - The `is_square_matrix` trait is used as a constraint for the `solve_linear_equations` member function.
    - **Template Friendship**:
        - The `operator*` function for multiplying a matrix and a vector is a friend function of the `Matrix` class.
    - **Type Traits**:
        - Type traits are used with `static_assert` to check for conditions during compile time.
2. **Vector Class**:
    - **Templates**:
        - The `Vector` class is implemented using templates for the element type `T` and the size `N`.
    - **Template Specialization**:
        - The `cross` member function is specialized for 3D vectors.
3. **ADVariable Class**:
    - **Templates**:
        - The `ADVariable` class is implemented using templates for the element type `T`.
    - **Template Specialization**:
        - The `ADVariable` class has a specialization for constant types (`T*`).

- **Variadic Templates**:
  - The `sum` function for ADVariable objects uses a variadic template to handle an arbitrary number of arguments.
- **Lambda Templates**:
  - The `transform` function for ADVariable objects takes a lambda function as a template parameter.
- **Fold Expression**:
  - The `sum` function for ADVariable objects uses a fold expression to perform the summation.
- **Concepts and Constraints**:
  - The `Numeric` concept is used to constrain the element type `T` of the `ADVariable` class.

c. Individual Contributions

As the only member of this team, I have worked on the entire code, which consists of :

- matrix.hpp
- vector.hpp
- auto_differentiation.hpp
- Testcases for the various features

d. Link to Git Repository

https://github.com/amogh-bharne/LA-library

e. List of known bugs

The ADVariable class contains only elementary functions for basic usecases, which might create bugs when tried for more complex tasks.

The matrix and vector methods have not been tested for large matrices. There can be potential bugs for such scenarios.