# Project Documentation:
# **Matrix Handling Library**


Project author:
**Sylwester Dąbrowski**
Supervisor:
**M.Sc. Eng. Jakub Zimnol**
Course:
Object-Oriented Programming Languages

9 stycznia 2026

# 1 Project Description

The project presents an implementation of a simple mathematical matrix handling library written in C++, using generic programming mechanisms (templates). The library enables creation and execution of operations on rectangular, square, and triangular matrices, while preserving correct mathematical properties and ensuring runtime error control.

The main goal of the project was to design a clear and safe interface that allows performing basic linear algebra operations on matrices, such as addition, subtraction, multiplication, transposition, determinant calculation, and matrix inversion.

The library may be used in simple numerical computations, educational tasks related to linear algebra, and as a foundation for further extensions, for example by implementing more advanced numerical algorithms.

# 2 Class Structure and Description

The project consists of three main classes organized hierarchically:

- `Matrix<T>` – a base class representing a general rectangular matrix of arbitrary dimensions. It provides basic matrix operations and mechanisms for accessing matrix elements.

- `SquareMatrix<T>` – a class derived from `Matrix<T>`, intended for handling square matrices. It extends the base functionality with operations specific to square matrices, such as determinant computation, cofactor matrix calculation, and matrix inversion.

- `TriangularMatrix<T>` – a class derived from `SquareMatrix<T>` that represents triangular matrices (lower or upper). The class enforces the triangular structure through controlled write access to elements and offers an optimized determinant calculation.

The inheritance relationships reflect the mathematical hierarchy between different matrix types, allowing code reuse and maintaining interface consistency.

# 3 Compilation and Execution

The project uses the `CMake` build system. To compile the application, a compiler supporting the `C++20` standard is required.

The following sequence of commands builds the project correctly:

```
mkdir build
cd build
cmake ..
make
```

As a result of the compilation process, an executable file named `matrixSD` is generated. It contains a demonstration program showcasing the capabilities of the implemented library.

# 4 Additional Information

The project code is divided into header and source files, and its technical documentation has been prepared using the Doxygen format directly within the header files.

The `main.cpp` file serves as a test and demonstration program, presenting the correctness of the implemented classes, exception handling, and example computations performed using the library.