МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ УПРАВЛЕНИЯ

ОТЧЕТ

по учебной вычислительной (ознакомительной) практике

Дрожжи Кирилла Витальевича

студента 1 курса, 11 группы

специальность

«Прикладная информатика»

Руководитель практики:

ассистент А.В. Дрепакова

Минск, 2024

# Оглавление

[Оглавление 2](#_Toc178104977)

[Введение 3](#_Toc178104978)

[Описание программы 4](#_Toc178104979)

[Выполнение примера с заданной матрицей 6](#_Toc178104980)

[Полученные знания и навыки 7](#_Toc178104981)

[Заключение 8](#_Toc178104982)

[Приложение А 9](#_Toc178104983)

# Введение

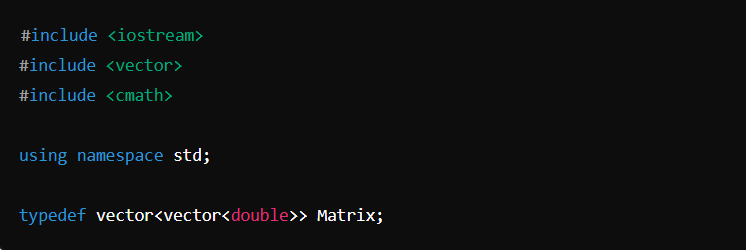
В данной лабораторной работе была реализована программа на языке C++, выполняющая разложение Холецкого заданной симметричной положительно определенной матрицы. Разложение Холецкого является важным алгоритмом в численных методах линейной алгебры и применяется для решения систем линейных уравнений, вычисления определителей и обратных матриц.

### Цель работы

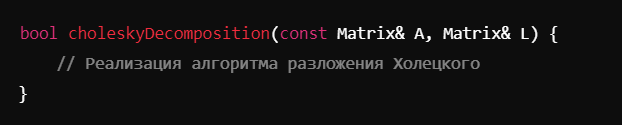
* Изучить и реализовать алгоритм разложения Холецкого.
* Применить алгоритм к конкретной матрице и проанализировать результаты.
* Закрепить навыки работы с матрицами и динамическими структурами данных в C++.

# Описание программы

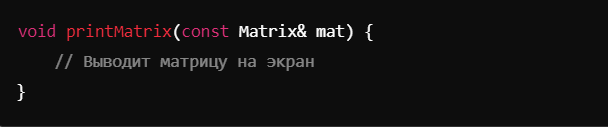
Программа состоит из следующих основных компонентов:

1. Подключение библиотек и определение типов

* Подключены стандартные библиотеки для ввода-вывода, работы с векторами и математическими операциями.
* Определен тип Matrix как вектор векторов типа double для представления матриц.

1. Функция choleskyDecomposition

* Принимает на вход симметричную положительно определенную матрицу A.
* Вычисляет нижнетреугольную матрицу L такую, что.
* Проверяет положительную определенность матрицы; если условие не выполняется, возвращает false.

1. Функция printMatrix

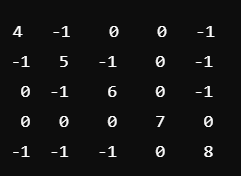
Форматирует и выводит матрицу в удобочитаемом виде.

1. Функция main

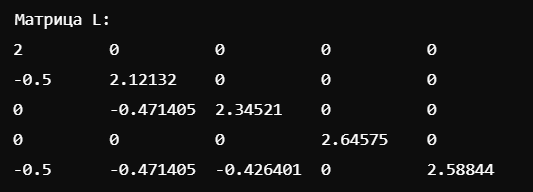
* Инициализирует матрицу A с заданными элементами.
* Вызывает функцию choleskyDecomposition для вычисления матрицы L.
* Выводит результат или сообщение об ошибке.

# Выполнение примера с заданной матрицей

### Заданная матрица

Матрица A задана следующим образом:

### Результаты выполнения

После запуска программы получаем следующий вывод:

### Проверка корректности

* Убедимся, что L является нижнетреугольной матрицей.
* Проверим, что с помощью вычислительных средств или вручную на небольших элементах.

Так и есть, в результате умножения получаем матрицу .

# Полученные знания и навыки

В ходе выполнения лабораторной работы я:

* Изучил алгоритм разложения Холецкого: понял его теоретические основы и области применения.
* Реализовал алгоритм на языке C++: закрепил навыки программирования, работы с вложенными циклами и обработкой ошибок.
* Научился работать с матрицами в C++: использовал динамические структуры данных для хранения и обработки матриц.
* Понял важность проверки положительной определенности: осознал, что разложение Холецкого применимо только к симметричным положительно определенным матрицам.
* Отладил и протестировал программу: улучшил навыки поиска и исправления ошибок в коде.

# Заключение

Лабораторная работа позволила на практике применить знания из курса линейной алгебры и численных методов. Реализация разложения Холецкого расширила понимание алгоритмов разложения матриц и их реализации в программировании. Полученные навыки будут полезны в дальнейших исследованиях и разработках в области вычислительной математики и программной инженерии.

# Приложение А

Исходный код алгоритма (файл main.cpp)

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

typedef vector<vector<double>> Matrix;

bool choleskyDecomposition(const Matrix& A, Matrix& L) {

    int n = A.size();

    L = Matrix(n, vector<double>(n, 0.0));

    for (int i = 0; i < n; i++) {

        for (int j = 0; j <= i; j++) {

            double sum = 0.0;

            if (j == i) { // Диагональный элемент

                for (int k = 0; k < j; k++)

                    sum += L[j][k] \* L[j][k];

                double value = A[j][j] - sum;

                if (value <= 0.0)

                    return false; // Матрица не положительно определена

                L[j][j] = sqrt(value);

            } else {

                for (int k = 0; k < j; k++)

                    sum += L[i][k] \* L[j][k];

                L[i][j] = (A[i][j] - sum) / L[j][j];

            }

        }

    }

    return true;

}

void printMatrix(const Matrix& mat) {

    std::vector<double>::size\_type n = mat.size();

    for (std::vector<double>::size\_type i = 0; i < n; i++) {

        for (std::vector<double>::size\_type j = 0; j < mat[i].size(); j++)

            cout << mat[i][j] << "\t";

        cout << endl;

    }

}

int main() {

    Matrix A = {

        {4, -1, 0, 0, -1},

        {-1, 5, -1, 0, -1},

        {0, -1, 6, 0, -1},

        {0, 0, 0, 7, 0},

        {-1, -1, -1, 0, 8}

    };

    Matrix L;

    if (choleskyDecomposition(A, L)) {

        cout << "Матрица L:" << endl;

        printMatrix(L);

    } else {

        cout << "Матрица не является положительно определенной." << endl;

    }

    return 0;

}