Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Метод Жордана-Гаусса для решения систем линейных алгебраических уравнений»**

**Выполнила**:

студентка группы 3823Б1ПМ1

Абракова Виктория Вячеславовна

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**СОДЕРЖАНИЕ**

[Постановка задачи 3](#_Toc167745784)

[Метод решения 4](#_Toc167745785)

[Руководство пользователя 5](#_Toc167745786)

[Описание программной реализации 6](#_Toc167745787)

[Подтверждение корректности 7](#_Toc167745788)

[Результаты экспериментов 8](#_Toc167745789)

[Заключение 9](#_Toc167745790)

[Список литературы 10](#_Toc167745791)

[Приложение 11](#_Toc167745792)

# Постановка задачи

В данной лабораторной работе требовалось реализовать метод Жордана-Гаусса для решения систем линейных уравнений, использовать при написании кода шаблоны, исключения и наследование.

# Метод решения

Для решения поставленной задачи, необходимо реализовать вектор, матрицу как вектор векторов и сам метод Жордана-Гаусса.

Метод Жордана-Гаусса – это метод решения систем линейных уравнений путём полного исключения неизвестных. Данный метод является модификацией метода Гаусса, только в случае метода Жордана-Гаусса элементарные преобразования проводятся дальше. Метод Жордана-Гаусса используется для решения систем линейных уравнений, а также для получения обратных матриц и нахождения ранга матрицы. Также этот метод весьма полезен и часто применяем для решения технических задач со множеством неизвестных.

Для решения получаемых на основе технических задач систем уравнений выделяют наибольшие по модулю переменные для уменьшения ошибки погрешности, а затем производят поочередное удаление лишних переменных из строчек матрицы.

# Руководство пользователя

При запуске программы пользователю предлагается ввести количество строк и столбцов матрицы, а также исходные элементы каждой строки через пробел. Далее на консоль выводится исходная матрица, матрица в ступенчатом виде и решение системы линейных уравнений, т.е. найденные переменные.

# Описание программной реализации

Программа начинает работу с функции main().

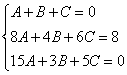
В программе реализованы классы Vector (вектор), Matrix (матрица), реализованный как вектор векторов и класс JordanGaussMatrix, наследующий методы класса Matrix.

Класс JordanGaussMatrix содержит функцию solve, которая решает системы линейных уравнений методом Жордана-Гаусса.

Помимо функции solve в программе есть функция printMatrix, которая выводит матрицу на консоль

# Подтверждение корректности

Для подтверждения корректности найденные значения переменных подставила в исходную систему линейных уравнений и оценила результат.

**Пример.**

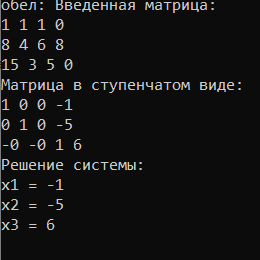


Рисунок 1. Решение примера методом Жордана-Гаусса

Проверка: подставим найденные значения переменных в левую часть каждого уравнения системы:

1. -1-5+6 = 0
2. 8\*(-1)+4\*(-5)+6\*6 = 8
3. 15\*(-1)+3\*(-5)+5\*6 = 0

Система решена верно. Можно сделать вывод, что для присоединённых квадратных матриц (со столбцом неизвестных) полного ранга метод работает корректно.

# Результаты экспериментов

В данной главе рассмотрим точность вычислений в числах с плавающей точкой.

В эксперименте генерируется квадратная матрица полного ранга случайным образом. Полученная матрица проходит 1000 итераций на каждой из них ее диагональные элементы умножаются на определенную константу строго большую единицы, но достаточно маленькую, чтобы тип данных double не переполнился.

В результате эксперимента был сделан вывод о том, что с ростом диагональных элементов ошибка уменьшается, а время работы практически не изменяется.

# Заключение

В ходе лабораторной работы бал реализован метод Жордана-Гаусса, используемый для решения систем линейных уравнеий.

В результате эксперимента было выявлено, что для чисел с плавающей точкой с ростом диагональных элементов ошибка уменьшается, а время работы практически не изменяется.

# Список литературы

1. Как решить систему линейных уравнений?: [Электронный ресурс] // Mathprofi. URL: http://mathprofi.ru/kak\_reshit\_sistemu\_uravnenii.html (Дата обращения: 22.05.2024)
2. Метод Жордана-Гаусса: [Электронный ресурс] // Справочник от автора 24. URL: <https://spravochnick.ru/matematika/metod_zhordana-gaussa/> (Дата обращения: 20.05.2024)

# Приложение

﻿#include <iostream>

#include <cmath>

#include <stdexcept>

#include <algorithm>

// Класс вектор

template <class T>

class Vector {

private:

T\* data;

int size;

public:

Vector() {

data = nullptr;

size = 0;

}

Vector(int size) : size(size) {

data = new T[size];

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

data[i] = T();//Инициализация элементов

}

}

~Vector() {

delete[] data;

}

T& operator[](int index) {

if (index < 0 || index >= size) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

const T& operator[](int index) const {

if (index < 0 || index >= size) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

int getSize() const {

return size;

}

void setSize(int size) {

delete[] data;

this->size = size;

data = new T[size];

for (int i = 0; i < size; i++)

{

data[i] = T();

}

return;

}

};

// Класс матрица, реализованный как вектор векторов

template <class T>

class Matrix {

protected:

Vector<Vector<T>> data;

public:

Matrix(int rows, int cols) {

data.setSize(rows);

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

data[i].setSize(cols);

}

}

~Matrix()

{

}

Vector<T>& operator[](int index) {

if (index < 0 || index >= data.getSize()) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

const Vector<T>& operator[](int index) const {

if (index < 0 || index >= data.getSize()) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

int rows() const {

return data.getSize();

}

int cols() const {

return data[0].getSize();

}

};

// Функция для вывода матрицы

template <typename T>

void printMatrix(const Matrix<T>& matrix) {

for (int i = 0; i < matrix.rows(); ++i) {

for (int j = 0; j < matrix.cols(); ++j) {

if (abs(matrix[i][j]) > 1e-14) {

std::cout << matrix[i][j] << " ";

}

else {

std::cout << 0 << " ";

}

}

std::cout << std::endl;

}

}

template <typename T>

class JordanGaussMatrix : public Matrix<T> {

public:

JordanGaussMatrix(int rows, int cols) : Matrix<T>(rows, cols) {}

void solve() {

int n = this->rows();

for (int i = 0; i < std::min(this->rows(), this->cols()); ++i) {

// Поиск максимального по модулю элемента в столбце

int maxRow = i;

for (int k = i + 1; k < n; ++k) {

if (std::abs(this->data[k][i]) > std::abs(this->data[maxRow][i])) {

maxRow = k;

}

}

//меняем строки местами

if (maxRow != i) {

for (int j = 0; j < this->cols(); ++j) {

std::swap(this->data[i][j], this->data[maxRow][j]);

}

}

//нулевой столбец

T scale = this->data[i][i];

if (abs(scale) < 1e-14) {

continue;

}

// Приводим диагональный элемент к 1

for (int j = 0; j < this->cols(); ++j) {

this->data[i][j] /= scale;

}

// Зануляем оставшиеся элементы в столбце

for (int k = 0; k < n; ++k) {

if (k != i) {

T factor = this->data[k][i];

for (int j = 0; j < this->cols(); ++j) {

this->data[k][j] -= factor \* this->data[i][j];

}

}

}

}

}

Vector<T> getSolution() const {

Vector<T> solution(this->rows());

for (int i = 0; i < this->rows(); ++i) {

solution[i] = this->data[i][this->cols() - 1];

}

return solution;

}

};

int main() {

setlocale(LC\_ALL, "rus");

int n, m;

std::cout << "Введите количество строк матрицы: ";

std::cin >> n;

std::cout << "Введите количество столбцов матрицы: ";

std::cin >> m;

JordanGaussMatrix<double> matrix(n, m);

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

std::cout << "Введите элементы " << i + 1 << "-й строки через пробел: ";

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

std::cin >> matrix[i][j];

}

}

std::cout << "Введенная матрица:" << std::endl;

printMatrix(matrix);

try {

matrix.solve();

std::cout << "Матрица в ступенчатом виде:" << std::endl;

printMatrix(matrix);

Vector<double> solution = matrix.getSolution();

std::cout << "Решение системы:" << std::endl;

for (int i = 0; i < solution.getSize(); ++i) {

std::cout << "x" << i + 1 << " = " << solution[i] << std::endl;

}

}

catch (const std::exception& e) {

std::cerr << "Error: " << e.what() << std::endl;

}

return 0;

}