

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

«Программная реализация решения системы линейных уравнений методом Гаусса»

Выполнил:

студент группы 3824Б1ПМ1
Яковлев Р.О.

Проверила:

Бусько П.В.

Содержание.

Введение	3
Постановка задачи.....	4
Руководство пользователя	5
Описание программной реализации	7
Результаты экспериментов.....	10
Литература	14
Приложение	15

Введение

В далеком 19 веке начала свое существование теория матриц, основоположниками которой принято считать Уильяма Гамильтона и Артура Кэли. С тех пор матрицы стали неотъемлемой частью жизни человечества. Так как матрица является более компактной записью системы линейных уравнений, то матрицы используются во многих разделах математики. Для решения СЛУ за несколько веков придумали множество разных способов, в том числе классический метод решения, который был назван в честь немецкого математика Карла Фридриха Гаусса.

Технологии развивались, поэтому сейчас руками преобразовывать матрицы крайне неудобно, гораздо легче написать программу, которая будет это делать за тебя. В этом и заключается моя лабораторная работа — написать программу на языке C++, которая будет решать систему линейных уравнений методом Гаусса.

Постановка задачи

Основная задача — реализовать программу, которая будет решать системы линейных уравнений (матрицы) методом Гаусса.

Подзадачи:

1. Реализовать шаблонный класс вектора, затем класса матрицы, который является шаблоном класса вектор от вектора.
2. Реализовать класс СЛУ (Система линейных уравнений), который является наследником класса матрицы
3. Написать решение СЛУ методом Гаусса.
4. Провести исследование, в котором проверяется правильность работы программы для разных заданных расширенных матриц.

Руководство пользователя

Первое, что видит пользователь при запуске программы, просьбу ввести количество строк в матрице.

```
C:\Users\Volrum\source\repos\Lab3>
Enter matrix rows:
3
```

Рис. 1. Ввод количества строк в матрице

После ввода строк программа просит пользователя ввести саму матрицу.

```
C:\Users\Volrum\source\repos\Lab3\x64\Debug>
Enter matrix rows:
3
Enter your matrix:
3 4 5
6 7 8
9 10 10
```

Рис. 2. Ввод матрицы

Затем пользователю необходимо ввести вторую часть расширенной матрицы, то есть ответ.

```
C:\Users\Volrum\source\repos\Lab3\x64\Debug>
Enter matrix rows:
3
Enter your matrix:
3 4 5
6 7 8
9 10 10
Enter your answer:
14 7 3
```

Рис. 3. Ввод ответа (части b матрицы)

После этого программа последовательно выводит в консоль преобразования матрицы, и в конце концов финальное решение СЛУ.

```
C:\Users\Volrum\source\repos\Lab3\x64\Debug>
Enter matrix rows:
3
Enter your matrix:
3 4 5
6 7 8
9 10 10
Enter your answer:
14 7 3
3 4 5
0 -1 -2
0 -2 -5
3 4 5
0 -1 -2
0 0 -1
3 4 5
0 -1 -2
0 0 -1
X: -26.3333 27 -3
Do you want to enter a new answer? (1 - yes, 0 - no)
```

Рис. 4. Вывод пошагового решения

После вывода ответа программа спрашивает пользователя, хочет ли он ввести еще один ответ, если пользователь вводит 0, то программа завершает работу, если вводит 1, то программа повторяет все действия, начиная с третьего шага.

Описание программной реализации

Проект состоит из одного файла “Lab3.cpp”.

Подключенные в файле библиотеки:

- `<iostream>`, используется для ввода и вывода данных в консольном приложении;
- `<iomanip>`, используется для более красивого вывода матриц в консоль;

Реализованные классы:

1. Шаблонный класс `Vector` (см. Приложение, рис. 5)

Поля:

- `T* arr`, массив `T`, где `T` это тип данных
- `int size`, размер вектора

Методы:

- `Vector()`, конструктор по умолчанию;
- `Vector(int _size)`, конструктор с параметром размера массива;
- `~Vector()`, деструктор;
- `T& operator[](int index)`, метод (перегрузка оператора `[]`), возвращающий `index`-овый элемент массива;
- `friend std::ostream& operator << (std::ostream& out, Vector vector)`, метод (перегрузка оператора `<<`), позволяющий выводить данный класс `Vector` через библиотеку `<iostream>`;
- `friend std::istream& operator >> (std::istream& in, Vector vector)`, метод (перегрузка оператора `>>`), позволяющий вводить данный класс `Vector` через библиотеку `<iostream>`;

2. Шаблонный класс `Matrix` (см. Приложение, рис. 6)

Данный класс наследуется от класса `Vector`, и является шаблоном класса `Vector` от `Vector`.

Поля:

- `int rows`, размер вектора

А также в `Matrix` входят те же поля, что и в класс `Vector`.

Методы:

- `Matrix()`, конструктор по умолчанию;
- `Matrix(int _rows)`, конструктор с параметром размера матрицы;

- `Matrix(Matrix<T> &other)`, конструктор копирования;
- `~Matrix()`, деструктор;
- `Vector<T>& operator[](int index)`, метод (перегрузка оператора `[]`), возвращающий `index`-овый вектор матрицы;
- `friend std::ostream& operator << (std::ostream& out, Matrix matrix)`, метод (перегрузка оператора `<<`), позволяющий выводить данный класс `Matrix` через библиотеку `<iostream>`;
- `friend std::istream& operator >> (std::istream& in, Matrix matrix)`, метод (перегрузка оператора `>>`), позволяющий вводить данный класс `Matrix` через библиотеку `<iostream>`;

3. Шаблонный класс `SLU` (см. Приложение, рис. 7)

Данный класс является наследником класса `Matrix`.

Поля:

- `Vector<T> ans`, вектор, передающийся как часть `b` в расширенную матрицу;
- `Vector<T> x`, вектор, являющийся решением СЛУ;

А также в поля `SLU` входят те же поля, что и в класс `Matrix`.

Методы:

- `SLU(Matrix<T> a)`, конструктор, принимающий в качестве параметра матрицу;
- `~SLU()`, деструктор;
- `Vector<T> Solve(Vector<T> b)`, метод, решающий СЛУ относительно вектора `b`, который передают в метод. Возвращает вектор (решение СЛУ);

Описание функции `Main` (см. Приложение, рис. 8):

Переменные:

- `int rows`, количество строк и столбцов в матрице;
- `bool isContinue`, параметр цикла, который отвечает за то, будет ли пользователь вводить ответ еще один раз;
- `Vector<double> b`, вектор `b` для расширенной матрицы;
- `Vector<double> x`, вектор, являющийся решением СЛУ;
- `Matrix<double> matrix`, матрица;
- `SLU<double> slu`, СЛУ, которую будет решать программа;

Цикл `while`, принимающий параметр `isContinue`, в котором заново вводится ответ для СЛУ, решается система и выводится ответ.

Описание функции `Solve` (см. Приложение, рис. 7):

Имеет одну переменную типа `Matrix<T> newMatrix`, необходимую для того, чтобы при вводе нового ответа в консоль, СЛУ решалась с неизменной основной матрицей. Соответственно при помощи конструктора копирования, куда мы передаем основную матрицу, инициализируется `newMatrix`, с которой будут проходить дальнейшие преобразования.

Первым циклом `for` функция проходит по главной диагонали матрицы. Вторым циклом проверяет на какой коэффициент нужно домножить i -ую строку, что при сложении с j -ой элемент, находящийся на j -ой строке и i -ом столбце занулился. И третьим циклом складывает k -ый элемент j -ой строки с k -ым элементом i -ой строки, а также j -ый с i -ый элементом правой части, умноженными на этот самый коэффициент.

После выполнения 2-ого и 3-его цикла `for` программа выводит в консоль матрицу, для того, чтобы пользователь мог отследить, что поменялось, скажем так промежуточный результат.

Наконец, после выполнения первого цикла, программа начинает высчитывать итоговый ответ. Из i -ого элемента правой части поочередно вычитаются j -ые элементы i -ой строки, умноженные на соответственно значение j -ого элемента правой части. После чего i -ый элемент правой части делится на значение элемента i -ой строки i -ого столбца.

В конечном итоге функция возвращает вектор x , который является решением СЛУ.

Результаты экспериментов

Для того чтобы проверить, насколько правильно работает моя программа, я протестировал ее на трех системах, которым соответствовали матрицы 2x2, 3x3, 4x4 соответственно. Вот результаты.

Первая расширенная матрица была:

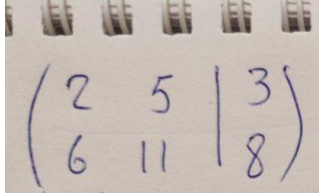

$$\left(\begin{array}{cc|c} 2 & 5 & 3 \\ 6 & 11 & 8 \end{array} \right)$$

Рис. 9. Матрица 2x2 для проверки

Вот мое решение:

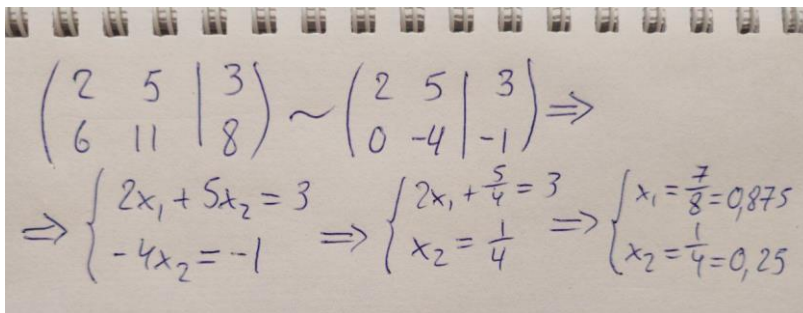
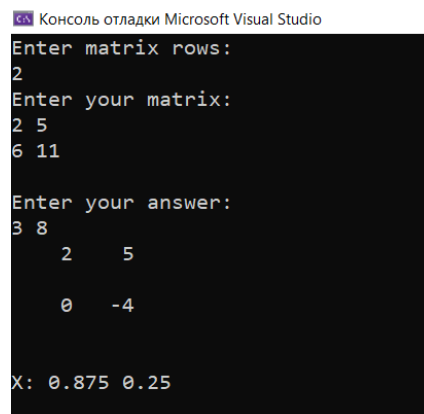

$$\begin{aligned} \left(\begin{array}{cc|c} 2 & 5 & 3 \\ 6 & 11 & 8 \end{array} \right) &\sim \left(\begin{array}{cc|c} 2 & 5 & 3 \\ 0 & -4 & -1 \end{array} \right) \Rightarrow \\ \Rightarrow \begin{cases} 2x_1 + 5x_2 = 3 \\ -4x_2 = -1 \end{cases} &\Rightarrow \begin{cases} 2x_1 + \frac{5}{4} = 3 \\ x_2 = \frac{1}{4} \end{cases} \Rightarrow \begin{cases} x_1 = \frac{7}{8} = 0,875 \\ x_2 = \frac{1}{4} = 0,25 \end{cases} \end{aligned}$$

Рис. 10. Ручное решение СЛУ для первой матрицы

Ответ — (0,875 0,25). А здесь решение моей программы:



```
Консоль отладки Microsoft Visual Studio
Enter matrix rows:
2
Enter your matrix:
2 5
6 11

Enter your answer:
3 8
    2    5
    0   -4

X: 0.875 0.25
```

Рис. 11. Программное решение СЛУ для первой матрицы

Ответ в точности такой же. Следующая матрица:

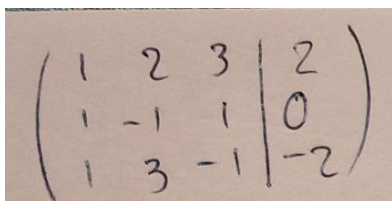

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 2 \\ 1 & -1 & 1 & 0 \\ 1 & 3 & -1 & -2 \end{array} \right)$$

Рис. 12. Матрица 3x3 для проверки

Мое решение:

$$\begin{pmatrix} 1 & 2 & 3 & | & 2 \\ 1 & -1 & 1 & | & 0 \\ 1 & 3 & -1 & | & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 & | & 2 \\ 0 & -3 & -2 & | & -2 \\ 0 & 1 & -4 & | & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 & | & 2 \\ 0 & -3 & -2 & | & -2 \\ 0 & 0 & -\frac{14}{3} & | & -\frac{14}{3} \end{pmatrix}$$

$$\Rightarrow \begin{cases} x_1 + 2x_2 + 3x_3 = 2 \\ -3x_2 - 2x_3 = -2 \\ -\frac{14}{3}x_3 = -\frac{14}{3} \end{cases} \Rightarrow \begin{cases} x_1 + 2x_2 = -1 \\ -3x_2 = 0 \\ x_3 = 1 \end{cases} \Rightarrow \begin{cases} x_1 = -1 \\ x_2 = 0 \\ x_3 = 1 \end{cases}$$

Рис. 13. Ручное решение СЛУ для второй матрицы

Решение программы:

```

Консоль отладки Microsoft Visual Studio
Enter matrix rows:
3
Enter your matrix:
1 2 3
1 -1 1
1 3 -1

Enter your answer:
2 0 -2
  1   2   3
    0  -3  -2
    0   1  -4

  1   2   3
  0  -3  -2
  0   0 -4.66667

x:  -1  -0  1

```

Рис. 14. Программное решение СЛУ для второй матрицы

Ответ (-1 0 1) одинаковый.

Последняя матрица:

$$\begin{pmatrix} 8 & -2 & 0 & 0 & | & 6 \\ -1 & 6 & -2 & 0 & | & 3 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix}$$

Рис. 15. Матрица 4x4 для проверки

Мое решение:

$$\begin{pmatrix} 8 & -2 & 0 & 0 & | & 6 \\ -1 & 6 & -2 & 0 & | & 3 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix} \sim \begin{pmatrix} -1 & 6 & -2 & 0 & | & 3 \\ 8 & -2 & 0 & 0 & | & 6 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix} \sim \begin{pmatrix} -1 & 6 & -2 & 0 & | & 3 \\ 0 & 46 & -16 & 0 & | & 30 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix} \sim \\
 \sim \begin{pmatrix} -1 & 6 & -2 & 0 & | & 3 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 46 & -16 & 0 & | & 30 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix} \sim \begin{pmatrix} -1 & 6 & -2 & 0 & | & 3 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -246 & 92 & | & -154 \\ 0 & 0 & -1 & 6 & | & 5 \end{pmatrix} \sim \begin{pmatrix} -1 & 6 & -2 & 0 & | & 3 \\ 0 & 2 & 10 & -4 & | & 8 \\ 0 & 0 & -246 & 92 & | & -154 \\ 0 & 0 & 0 & \frac{622}{123} & | & \frac{622}{123} \end{pmatrix} \Rightarrow \\
 \Rightarrow \begin{cases} -x_1 + 6x_2 - 2x_3 = 3 \\ 2x_2 + 10x_3 = 8 \\ -246x_3 = -246 \\ x_4 = 1 \end{cases} \Rightarrow \begin{cases} -x_1 + 6x_2 = 5 \\ 2x_2 = 2 \\ x_3 = 1 \\ x_4 = 1 \end{cases} \Rightarrow \begin{cases} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1 \\ x_4 = 1 \end{cases}$$

Рис. 16. Ручное решение СЛУ для третьей матрицы

Решение программы:

```

C:\Users\Volnum\source\repos\Lab3\Lab3.exe
Enter matrix rows:
4
Enter your matrix:
8 -2 0 0
-1 6 -2 0
0 2 10 -4
0 0 -1 6
Enter your answer:
6 3 8 5
8 -2 0 0
0 5.75 -2 0
0 2 10 -4
0 0 -1 6
8 -2 0 0
0 5.75 -2 0
0 0 10.6957 -4
0 0 -1 6
8 -2 0 0
0 5.75 -2 0
0 0 10.6957 -4
0 0 0 5.62602
X: 1 1 1 1

```

Рис. 17. Программное решение СЛУ для третьей матрицы

Ответ (1 1 1 1) вновь совпадает в моем решении и решении программы, что говорит о том, что моя программа написана правильно, и с ее помощью можно без труда решать системы линейных уравнений.

Заключение

В заключение хочется сказать, что цель моей лабораторной достигнута. Написанная мной программа действительно помогает быстро решать разные системы линейных уравнений, и ее можно использовать при решении каких-либо математических задач, где используются матрицы.

Литература

1. Вирт Н. Алгоритмы и структуры данных.
2. Дж. Клейнберг, Э. Тардос. Алгоритмы: разработка и применение.
3. Р. Мартин. Чистый код.

Приложение

```
template<typename T>
class Vector {
public:
    T* arr;
    int size;

    Vector() {
        size = 0;
        arr = nullptr;
    }

    Vector(int _size) {
        size = _size;
        arr = new T[_size];
    }

    ~Vector() {
        size = 0;
        arr = nullptr;
    }

    T& operator[](int index) {
        return arr[index];
    }

    friend std::ostream& operator << (std::ostream& out, Vector vector) {
        for (int i = 0; i < vector.size; i++) {
            std::cout << std::setw(5) << vector.arr[i] << std::setw(1);
        }
        std::cout << "\n";
        return out;
    }

    friend std::istream& operator >> (std::istream& in, Vector vector) {
        for (int i = 0; i < vector.size; i++) {
            std::cin >> vector.arr[i];
        }
        return in;
    }
};
```

Рис. 5. Класс Vector

```
template <typename T>
class Matrix : public Vector<Vector<T>> {
public:
    int rows;

    Matrix() {
        rows = 0;
        this->arr = nullptr;
    }

    Matrix(int _rows) {
        rows = _rows;
        this->arr = new Vector<T>[rows];
        for (int i = 0; i < rows; i++) {
            this->arr[i] = Vector<T>(rows);
        }
    }

    Matrix(Matrix<T> &other) {
        rows = other.rows;
        this->arr = other.arr;
    }

    ~Matrix() {
        this->arr = nullptr;
        this->size = 0;
        rows = 0;
    }

    Vector<T>& operator[](int index) { return this->arr[index]; }

    friend std::ostream& operator << (std::ostream& out, Matrix matrix) {
        for (int i = 0; i < matrix.rows; i++) {
            std::cout << matrix.arr[i];
        }
        std::cout << "\n";
        return out;
    }

    friend std::istream& operator >> (std::istream& in, Matrix matrix) {
        for (int i = 0; i < matrix.rows; i++) {
            std::cin >> matrix.arr[i];
        }
        std::cout << "\n";
        return in;
    }
};
```

Рис. 6. Класс Matrix

```

template <typename T> <T> Укажите аргументы примера шаблона для IntelliSense
class SLU : public Matrix<T> {
public:
    Vector<T> ans, x;

    SLU(Matrix<T> a) : Matrix<T>(a){
        x = Vector<T>(a.rows);
    }

    ~SLU() {
        ans = NULL;
        x = NULL;
    }

    Vector<T> Solve(Vector<T>& _b) {
        Matrix<T> newMatrix(*this);
        ans = Vector<T>(_b);

        for (int i = 0; i < newMatrix.rows - 1; i++) {
            for (int j = i + 1; j < newMatrix.rows; j++) {
                T multiplicator = -1 * (newMatrix.arr[j][i] / newMatrix.arr[i][i]);
                for (int k = i; k < newMatrix.rows; k++) {
                    newMatrix.arr[j][k] += multiplicator * newMatrix.arr[i][k];
                }
                ans[j] += multiplicator * ans[i];
            }

            for (int j = 0; j < newMatrix.rows; j++) {
                std::cout << newMatrix.arr[j] << "\n";
            }
            std::cout << "\n";
        }

        for (int i = newMatrix.rows - 1; i >= 0; i--) {
            x[i] = ans[i];
            for (int j = i + 1; j < newMatrix.rows; j++) {
                x[i] -= newMatrix.arr[i][j] * x[j];
            }
            x[i] /= newMatrix.arr[i][i];
        }
        return x;
    }
};

```

Рис. 7. Класс SLU

```

int main()
{
    int rows;
    bool isContinue = true;

    std::cout << "Enter matrix rows:\n";
    std::cin >> rows;

    Vector<double> b(rows);
    Matrix<double> matrix(rows);
    SLU<double> slu(matrix);

    std::cout << "Enter your matrix:\n";
    std::cin >> matrix;

    while (isContinue) {
        int cont;

        std::cout << "Enter your answer:\n";
        std::cin >> b;

        Vector<double> x(rows);
        x = slu.Solve(b);
        std::cout << "X: " << x << "\n";

        std::cout << "Do you want to enter a new answer? (1 - yes, 0 - no)" << "\n";
        std::cin >> cont;
        if (cont == 0) {
            isContinue = false;
            break;
        }
    }
}

```

Рис. 8. Функция main