

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Инженерно-физический факультет
Кафедра автоматизированных систем обработки информации и управления

Отчёт по практике

Решение системы линейных алгебраических уравнений методом Гаусса.

2 курс, группа 2ИВТ

Выполнил:

_____ Д. Э. Чич
«___» _____ 2025 г.

Руководитель:

_____ С. В. Теплоухов
«___» _____ 2025 г.

Майкоп, 2025 г.

Оглавление

Теория.....	3
Ход выполнения работы.....	5
1. Подключение библиотек.....	5
2. Константа EPS.....	6
3. Функция printSystem().....	6
4. Основная функция main().....	7
4.2 Прямой ход метода Гаусса.....	8
4.3 Проверка на совместность.....	9
4.4 Обратный ход.....	9
4.5 Вывод решения.....	9
Код программы.....	10
Пример работы программы.....	15

Теория

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ). Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна. А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, домножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие

уравнения, и так далее, поднимаясь по «ступенькам» вверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

В простейшем случае алгоритм выглядит так:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \dots + a_{1n} \cdot x_n & = b_1 & (1) \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \dots + a_{2n} \cdot x_n & = b_2 & (2) \\ \dots & & \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \dots + a_{mn} \cdot x_n & = b_m & (m) \end{cases}$$

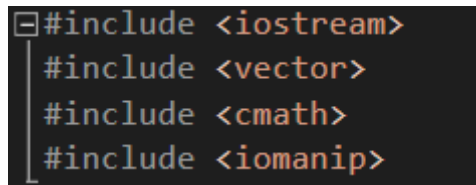
Прямой ход:

$$\begin{aligned} (2) &\rightarrow (2) - (1) \cdot \left(\frac{a_{21}}{a_{11}}\right) & : & \quad a'_{22} \cdot x_2 + a'_{23} \cdot x_3 + \dots + a'_{2n} \cdot x_n = b'_2 \\ (3) &\rightarrow (3) - (1) \cdot \left(\frac{a_{31}}{a_{11}}\right) & : & \quad a'_{32} \cdot x_2 + a'_{33} \cdot x_3 + \dots + a'_{3n} \cdot x_n = b'_3 \\ &\dots & & \\ (m) &\rightarrow (m) - (1) \cdot \left(\frac{a_{m1}}{a_{11}}\right) & : & \quad a'_{m2} \cdot x_2 + a'_{m3} \cdot x_3 + \dots + a'_{mn} \cdot x_n = b'_m \\ (3) &\rightarrow (3) - (2) \cdot \left(\frac{a'_{32}}{a'_{22}}\right) & : & \quad a''_{33} \cdot x_3 + \dots + a''_{3n} \cdot x_n = b''_3 \\ &\dots & & \\ (m) &\rightarrow (m) - (m-1) \cdot \left(\frac{a^{(m-2)}_{m,n-1}}{a^{(m-2)}_{m-1,n-1}}\right) & : & \quad a^{(m-1)}_{mm} \cdot x_m + \dots + a^{(m-1)}_{mn} \cdot x_n = b^{(m-1)}_m \end{aligned}$$

Обратный ход. Из последнего ненулевого уравнения выражаем базисную переменную через небазисные и подставляем в предыдущие уравнения. Повторяя эту процедуру для всех базисных переменных, получаем фундаментальное решение.

Ход выполнения работы

1. Подключение библиотек



```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>
```

Рисунок 1

- **iostream** - обеспечивает базовые возможности ввода/вывода. Мы используем:
 - `cin` для чтения данных
 - `cout` для вывода результатов
 - `endl` для перевода строки
- **vector** - предоставляет шаблонный класс `vector`, который мы используем для:
 - Хранения матрицы системы (`vector<vector<double>>`)
 - Хранения решения (`vector<double>`)
- **cmath** - содержит математические функции:
 - `abs()` - для вычисления абсолютного значения (модуля) числа
 - Другие функции (хотя в этой программе используются только `abs`)
- **iomanip** - предоставляет средства для форматирования вывода:
 - `setw()` - устанавливает ширину поля вывода
 - `fixed` - фиксирует вывод чисел с плавающей точкой
 - `setprecision()` - задает количество знаков после запятой

2. Константа EPS

```
const double EPS = 1e-9;
```

Рисунок 2

- **Назначение:** Используется для корректного сравнения чисел с плавающей точкой с нулем

3. Функция printSystem()

```
void printSystem(const vector<vector<double>>& matrix) {  
    int n = matrix.size();  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            cout << setw(8) << fixed << setprecision(3) << matrix[i][j];  
        }  
        cout << " |" << setw(8) << matrix[i][n] << endl;  
    }  
    cout << endl;  
}
```

Рисунок 3

1. **Параметр:** Принимает константную ссылку на матрицу (избегаем копирования)
2. **Размер матрицы:** matrix.size() - количество строк (уравнений)
3. **Вложенные циклы:**
 - Внешний цикл по строкам (i)
 - Внутренний цикл по столбцам (j)
4. **Форматирование вывода:**
 - setw(8) - каждое число занимает 8 символов
 - fixed - фиксированная точка (не экспоненциальная форма)
 - setprecision(3) - 3 знака после запятой
5. **Разделитель:** | отделяет коэффициенты от свободного члена

6. **Перевод строки:** endl в конце каждой строки и пустая строка после всей матрицы

4. Основная функция main()

```
int n;
cout << "Введите количество уравнений: ";
cin >> n;

vector<vector<double>> matrix(n, vector<double>(n + 1));

cout << "Введите коэффициенты системы (по строкам):\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j <= n; j++) {
        cin >> matrix[i][j];
    }
}
```

Рисунок 4

- **n** - количество уравнений (и неизвестных)
- **matrix** - двумерный вектор размером $n \times (n+1)$:
 - n строк
 - n+1 столбцов (последний столбец - свободные члены)
- **Ввод данных:** построчное заполнение матрицы

4.2 Прямой ход метода Гаусса

```
for (int col = 0; col < n; col++) {
    int max_row = col;
    for (int i = col + 1; i < n; i++) {
        if (abs(matrix[i][col]) > abs(matrix[max_row][col])) {
            max_row = i;
        }
    }

    if (max_row != col) {
        swap(matrix[col], matrix[max_row]);
        cout << "Меняем строки " << col + 1 << " и " << max_row + 1 << ":\n";
        printSystem(matrix);
    }

    if (abs(matrix[col][col]) < EPS) {
        cout << "Столбец " << col + 1 << " содержит только нули!\n";
        continue;
    }

    double pivot = matrix[col][col];
    for (int j = col; j <= n; j++) {
        matrix[col][j] /= pivot;
    }

    cout << "Нормализуем строку " << col + 1 << ":\n";
    printSystem(matrix);

    for (int i = col + 1; i < n; i++) {
        double factor = matrix[i][col];
        for (int j = col; j <= n; j++) {
            matrix[i][j] -= factor * matrix[col][j];
        }
        cout << "Обнуляем элемент в строке " << i + 1 << ":\n";
        printSystem(matrix);
    }
}
```

Рисунок 5

1. **Выбор ведущего элемента:** поиск максимального по модулю элемента в столбце (устойчивость алгоритма)
2. **Перестановка строк:** если максимальный элемент не на диагонали
3. **Нормализация строки:** деление всей строки на диагональный элемент
4. **Обнуление элементов ниже:** вычитание текущей строки из нижележащих

4.3 Проверка на совместность

```
for (int i = 0; i < n; i++) {
    bool all_zero = true;
    for (int j = 0; j < n; j++) {
        if (abs(matrix[i][j]) > EPS) {
            all_zero = false;
            break;
        }
    }
    if (all_zero && abs(matrix[i][n]) > EPS) {
        cout << "Система не имеет решений!\n";
    }
}
```

Рисунок 6

- **Анализ строк:** если все коэффициенты нулевые, а свободный член нет → система несовместна
- EPS используется для корректного сравнения с нулем

4.4 Обратный ход

```
vector<double> solution(n);
for (int i = n - 1; i >= 0; i--) {
    solution[i] = matrix[i][n];
    for (int j = i + 1; j < n; j++) {
        solution[i] -= matrix[i][j] * solution[j];
    }
}
```

Рисунок 7

- **Нахождение решения:** начиная с последнего уравнения
- **Принцип работы:**
 1. Берем свободный член (matrix[i][n])
 2. Вычитаем известные переменные (уже найденные)
 3. Получаем значение текущей переменной

4.5 Вывод решения

```
cout << "Решение системы:\n";
for (int i = 0; i < n; i++) {
    cout << "x" << i + 1 << " = " << solution[i] << endl;
}
```

Рисунок 8

- **Формат вывода:** x1 = значение, x2 = значение и т.д.

Код программы

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>

using namespace std;

const double EPS = 1e-9;

void printSystem(const vector<vector<double>>& matrix) {
    int n = matrix.size();
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << setw(8) << fixed << setprecision(3) << matrix[i][j];
        }
        cout << " |" << setw(8) << matrix[i][n] << endl;
    }
    cout << endl;
}

int getPositiveInteger(const string& prompt) {
    int value;
    while (true) {
        cout << prompt;
        cin >> value;

        if (cin.fail() || value <= 0) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
    }
}
```

```

        cout << "Ошибка! Пожалуйста, введите положительное целое число.\n";
    } else {
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return value;
    }
}
}

```

```

double getDouble(const string& prompt) {
    double value;
    while (true) {
        cout << prompt;
        cin >> value;

        if (cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Ошибка! Пожалуйста, введите числовое значение.\n";
        } else {
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            return value;
        }
    }
}

int main() {
    setlocale(0, "ru");
    int n;
    cout << "Введите количество уравнений: ";
    cin >> n;
}

```

```

vector<vector<double>> matrix(n, vector<double>(n + 1));

cout << "Введите коэффициенты системы (по строкам):\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j <= n; j++) {
        cin >> matrix[i][j];
    }
}

cout << "\nИсходная система:\n";
printSystem(matrix);

for (int col = 0; col < n; col++) {
    int max_row = col;
    for (int i = col + 1; i < n; i++) {
        if (abs(matrix[i][col]) > abs(matrix[max_row][col])) {
            max_row = i;
        }
    }

    if (max_row != col) {
        swap(matrix[col], matrix[max_row]);
        cout << "Меняем строки " << col + 1 << " и " << max_row + 1 << ":\n";
        printSystem(matrix);
    }

    if (abs(matrix[col][col]) < EPS) {
        cout << "Столбец " << col + 1 << " содержит только нули!\n";
    }
}

```

```

        continue;
    }

    double pivot = matrix[col][col];
    for (int j = col; j <= n; j++) {
        matrix[col][j] /= pivot;
    }

    cout << "Нормализуем строку " << col + 1 << ":\n";
    printSystem(matrix);

    for (int i = col + 1; i < n; i++) {
        double factor = matrix[i][col];
        for (int j = col; j <= n; j++) {
            matrix[i][j] -= factor * matrix[col][j];
        }
        cout << "Обнуляем элемент в строке " << i + 1 << ":\n";
        printSystem(matrix);
    }
}

for (int i = 0; i < n; i++) {
    bool all_zero = true;
    for (int j = 0; j < n; j++) {
        if (abs(matrix[i][j]) > EPS) {
            all_zero = false;
            break;
        }
    }
}

```

```

    if (all_zero && abs(matrix[i][n]) > EPS) {
        cout << "Система не имеет решений!\n";

    }
}

vector<double> solution(n);
for (int i = n - 1; i >= 0; i--) {
    solution[i] = matrix[i][n];
    for (int j = i + 1; j < n; j++) {
        solution[i] -= matrix[i][j] * solution[j];
    }
}

cout << "Решение системы:\n";
for (int i = 0; i < n; i++) {
    cout << "x" << i + 1 << " = " << solution[i] << endl;
}
}

```

Пример работы программы

```
Введите количество уравнений: 3
Введите коэффициенты системы (по строкам):
4 3 2 5
4 1 3 4
3 4 2 1

Исходная система:
  4.000   3.000   2.000 |   5.000
  4.000   1.000   3.000 |   4.000
  3.000   4.000   2.000 |   1.000

Нормализуем строку 1:
  1.000   0.750   0.500 |   1.250
  4.000   1.000   3.000 |   4.000
  3.000   4.000   2.000 |   1.000

Обнуляем элемент в строке 2:
  1.000   0.750   0.500 |   1.250
  0.000  -2.000   1.000 |  -1.000
  3.000   4.000   2.000 |   1.000

Обнуляем элемент в строке 3:
  1.000   0.750   0.500 |   1.250
  0.000  -2.000   1.000 |  -1.000
  0.000   1.750   0.500 |  -2.750

Нормализуем строку 2:
  1.000   0.750   0.500 |   1.250
  0.000   1.000  -0.500 |   0.500
  0.000   1.750   0.500 |  -2.750

Обнуляем элемент в строке 3:
  1.000   0.750   0.500 |   1.250
  0.000   1.000  -0.500 |   0.500
  0.000   0.000   1.375 |  -3.625

Нормализуем строку 3:
  1.000   0.750   0.500 |   1.250
  0.000   1.000  -0.500 |   0.500
  0.000   0.000   1.000 |  -2.636

Решение системы:
x1 = 3.182
x2 = -0.818
x3 = -2.636
```

Рисунок 9