

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Кубанский государственный университет»

Факультет математики и компьютерных наук

## **Семестровая работа**

### **Метод Гаусса**

Специальность 02.03.01 Математика и компьютерные науки

Выполнил:

Студент 1 курса, 13А группы  
математического факультета  
КубГУ

специальности

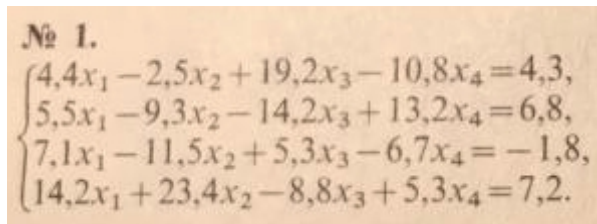
«математика и компьютерные  
науки»

очной формы обучения

Акимов Вадим

## ЗАДАЧА

Используя метод Гаусса решить систему линейных уравнений:



№ 1.

$$\begin{cases} 4,4x_1 - 2,5x_2 + 19,2x_3 - 10,8x_4 = 4,3, \\ 5,5x_1 - 9,3x_2 - 14,2x_3 + 13,2x_4 = 6,8, \\ 7,1x_1 - 11,5x_2 + 5,3x_3 - 6,7x_4 = -1,8, \\ 14,2x_1 + 23,4x_2 - 8,8x_3 + 5,3x_4 = 7,2. \end{cases}$$

## КРАТКОЕ ОПИСАНИЕ МЕТОДА

Метод Гаусса – это метод последовательного исключения неизвестных. С помощью элементарных преобразований, основную матрицу, состоящую из коэффициентов уравнений и свободных членов, можно привести к треугольному виду. Суть второго этапа(обратного хода) заключается в том, чтобы выразить все получившиеся переменные.

## СПИСОК ИДЕНТИФИКАТОРОВ

*in* — поток считывания данных из файла, *std::ifstream*

*w, h*- количество строчек и столбцов матрицы, *unsigned int*

*mat* — матрица коэффициентов со свободными членами, *double\*\**

*ans* — массив решения матрицы, *double\**

## ОПИСАНИЕ АЛГОРИТМА

Пользователь вводит матрицу коэффициентов, после чего она выводится на консоль. Далее применяется алгоритм Гаусса с заменой строк таким образом, чтобы наибольший элемент оказался на главной диагонали. Матрица приводится к треугольному виду, после чего находятся и выводятся на консоль все неизвестные.

## ТЕКСТ ПРОГРАММЫ

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <iomanip>
```

```
#include <cmath>
```

```
using std::cin;
```

```
using std::cout;
```

```
double ** create_mat(unsigned _w, unsigned _h) {  
    double ** tmp = new double*[_h];  
    for (unsigned iter1 = 0; iter1 < _h; ++iter1) {  
        tmp[iter1] = new double[_w];  
        for (unsigned iter2 = 0; iter2 < _w; ++iter2) {  
            tmp[iter1][iter2] = 0;  
        }  
    }  
    return tmp;  
}
```

```
void delete_mat(double ** mat, unsigned _w, unsigned _h) {  
    for (unsigned iter = 0; iter < _h; ++iter)  
        delete [] mat[iter];  
    delete [] mat;  
}
```

```
void generate_mat(double ** mat, unsigned _w, unsigned _h, std::ifstream & in) {  
    for (unsigned iter1 = 0; iter1 < _h; ++iter1)  
        for (unsigned iter2 = 0; iter2 < _w; ++iter2)
```

```

        in >> mat[iter1][iter2];
    }

void print_mat(double ** mat, unsigned _w, unsigned _h) {
    for (unsigned iter1 = 0; iter1 < _h; ++iter1) {
        for (unsigned iter2 = 0; iter2 < _w; ++iter2) {
            if (iter2 == _w - 1)
                cout << "| ";
            cout << std::fixed << std::setprecision(1) <<
                std::setw(5) << mat[iter1][iter2] << ' ';
        }
        cout << '\n';
    }
    cout << "\n\n";
}

```

```

void print_ans(double * ans, double _h) {
    cout << "Answer: \n";
    for (unsigned iter = 1; iter <= _h; ++iter)
        cout << " x" << iter << " = "
            << std::fixed << std::setprecision(6) << ans[iter - 1] << '\n';
    cout << "\n\n";
}

```

```

void my_swap(double & v1, double & v2) {
    if (v1 != v2) {
        double tmp = v1;
        v1 = v2;
        v2 = tmp;
    }
}

```

```

    }
}

void swap_str(double ** mat, unsigned str1, unsigned str2, unsigned _w) {
    if (str1 != str2) {
        for (unsigned iter = 0; iter < _w; ++iter)
            my_swap(mat[str1][iter], mat[str2][iter]);
    }
}

```

```

void sum_strings(double ** mat, unsigned _w, unsigned str1, unsigned str2, unsigned
start_index) {
    double tmp = -mat[str2][start_index] / mat[str1][start_index];
    for (unsigned iter = start_index; iter < _w; ++iter)
        mat[str2][iter] += (mat[str1][iter] * tmp);
    mat[str2][start_index] = 0;
}

```

```

void move_max(double ** mat, unsigned _w, unsigned _h, unsigned start_str) {
    unsigned max_str = start_str,
        max_value = abs(mat[start_str][start_str]);
    for (unsigned iter1 = start_str + 1; iter1 < _h; ++iter1) {
        if (std::abs(mat[iter1][start_str]) > max_value) {
            max_str = iter1;
            max_value = std::abs(mat[iter1][start_str]);
        }
    }
    swap_str(mat, start_str, max_str, _w);
}

```

```

void triangle_view(double ** mat, unsigned _w, unsigned _h) {
    for (unsigned iter1 = 0; iter1 < _h; ++iter1) {
        move_max(mat, _w, _h, iter1);
        for (unsigned iter2 = iter1 + 1; iter2 < _h; ++iter2)
            sum_strings(mat, _w, iter1, iter2, iter1);
    }
    print_mat(mat, _w, _h);
}

```

```

void calc_root(double ** mat, unsigned _w, unsigned str, double * ans) {
    double sum = mat[str][_w - 1];
    for (unsigned iter = str + 1; iter < _w - 1; ++iter)
        sum += ( -mat[str][iter] * ans[iter] );
    ans[str] = sum / mat[str][str];
}

```

```

void get_answer(double ** mat, unsigned _w, unsigned _h, double * ans) {
    for (int iter = _h - 1; iter >= 0; --iter)
        calc_root(mat, _w, iter, ans);
}

```

```

int main() {
    std::ifstream in("mat_2");
    unsigned w, h;
    in >> w >> h;
    double ** mat = create_mat(w, h);
    generate_mat(mat, w, h, in);
    double * ans = new double[h];
    for (unsigned iter = 0; iter < h; ++iter)

```

```
        ans[iter] = 0;
    cout << "Input mat:\n";
    print_mat(mat, w, h);
    cout << "Triangle view:\n";
    triangle_view(mat, w, h);
    get_answer(mat, w, h, ans);
    print_ans(ans, h);
    delete_mat(mat, w, h);
    delete [] ans;
    return 0;
}
```

## РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

```
Input mat:
  4.4  -2.5  19.2 -10.8 |  4.3
  5.5  -9.3 -14.2  13.2 |  6.8
  7.1 -11.5   5.3  -6.7 | -1.8
 14.2  23.4  -8.8   5.3 |  7.2

Triangle view:
 14.2  23.4  -8.8   5.3 |  7.2
  0.0 -23.2   9.7  -9.3 | -5.4
  0.0   0.0 -18.5  18.5 |  8.3
  0.0   0.0   0.0   9.4 | 12.3

Answer:
x1 = 0.444837
x2 = 0.067314
x3 = 0.868535
x4 = 1.311562
```