

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

ОТЧЕТ ПО ПРАКТИКЕ

Вариант 3. *Решение системы линейных  
алгебраических уравнений методом Гаусса.*

2 курс, группа 2ИВТ

Выполнил:

\_\_\_\_\_ Е. Е. Боцман  
«\_\_\_» \_\_\_\_\_ 2021 г.

Руководитель:

\_\_\_\_\_ С. В. Теплоухов  
«\_\_\_» \_\_\_\_\_ 2021 г.

Майкоп, 2021 г.

# 1. Введение

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), находят 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:

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

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.

## 2. Ход работы

### 2.1. Код приложения

```

#include <iostream>
#include <string>
#include <math.h>
#include <algorithm>
#include <conio.h>
#include <cstdlib>
#include <stdio.h>
using namespace std;
const double epsilon = 0.00001;
double *GaussianElimination(double **Matrix, double *y, int n){
double *x, max;
int k, index;
x = new double[n];
k = 0;
while (k < n){
max = abs(Matrix[k][k]);
index = k;
for (int i = (k + 1); i < n; i++){
if (abs(Matrix[i][k]) > max){
max = abs(Matrix[i][k]);
index = i;
}
}
if (max < epsilon){
cout << "В матрице присутствует нулевой столбец, решение невозможно";
return 0;
}
}
}

```

```

}
for (int j = 0; j < n; j++){
swap(Matrix[k][j], Matrix[index][j]);
}
swap(y[k], y[index]);
for (int i = k; i < n; i++){
double dud = Matrix[i][k];
if (abs(dud) < epsilon){
continue;
}
for(int j = 0; j < n; j++){
Matrix[i][j] = Matrix[i][j] - Matrix[k][j];
}
y[i] = y[i] - y[k];
}
k++;
}
for (k = n - 1; k >= 0; k--){
x[k] = y[k];
for (int i = 0; i < k; i++){
y[i] = y[i] - Matrix[i][k] * x[k];
}
}
return x;
}

#include <iostream>
#include <string>
#include <math.h>
#include <algorithm>
#include <conio.h>
#include <cstdlib>
#include <stdio.h>
using namespace std;
const double epsilon = 0.00001;
double *GaussianElimination(double **Matrix, double *y, int n){
double *x, max;
int k, index;
x = new double[n];
k = 0;
while (k < n){
max = abs(Matrix[k][k]);
index = k;
for (int i = (k + 1); i < n; i++){
if (abs(Matrix[i][k]) > max){

```

```

max = abs(Matrix[i][k]);
index = i;
}
}
if (max < epsilon){
cout << "В матрице присутствует нулевой столбец, решение невозможно";
return 0;
}
for (int j = 0; j < n; j++){
swap(Matrix[k][j], Matrix[index][j]);
}
swap(y[k], y[index]);
for (int i = k; i < n; i++){
double dud = Matrix[i][k];
if (abs(dud) < epsilon){
continue;
}
for(int j = 0; j < n; j++){
Matrix[i][j] = Matrix[i][j] - Matrix[k][j];
}
y[i] = y[i] - y[k];
}
k++;
}
for (k = n - 1; k >= 0; k--){
x[k] = y[k];
for (int i = 0; i < k; i++){
y[i] = y[i] - Matrix[i][k] * x[k];
}
}
return x;
}
int main(){
double **Matrix, *y, *x;
int n;
setlocale(LC_ALL, "Russian");
cout << "Введите ранг матрицы: ";
cin >> n;
Matrix = new double*[n];
y = new double[n];
for (int i = 0; i < n; i++){
Matrix[i] = new double[n];
for (int j = 0; j < n; j++){
cout << "Matrix[" << i << "][" << j << "] = ";

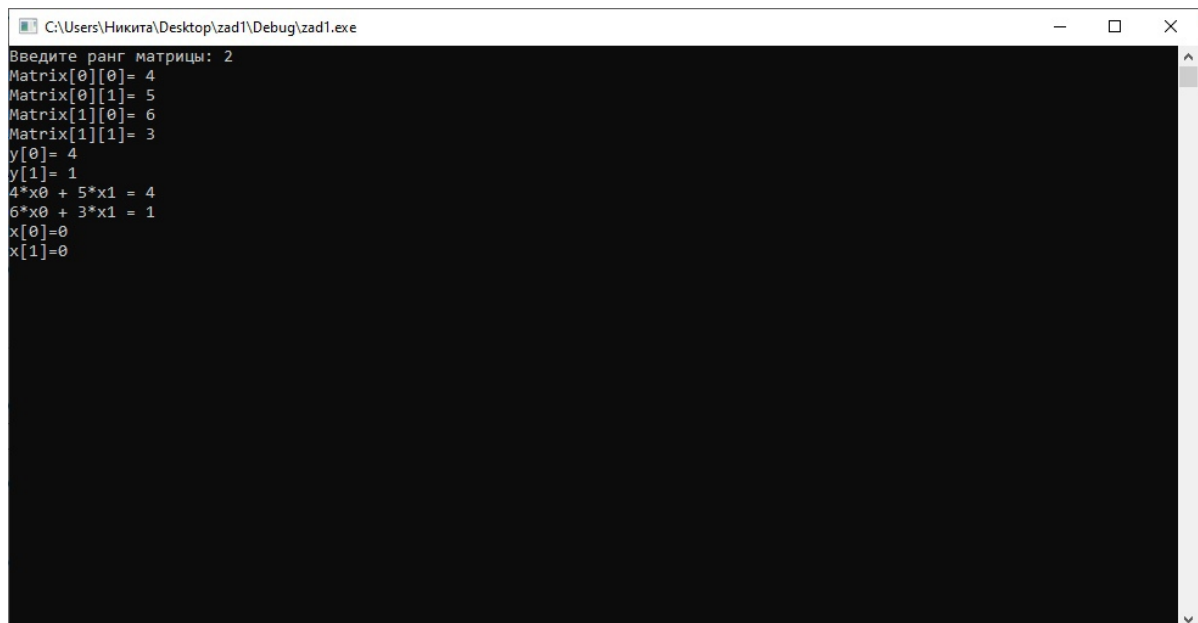
```

```

cin >> Matrix[i][j];
}
}
for (int i = 0; i < n; i++){
cout << "y[" << i << "]= ";
cin >> y[i];
}
for (int i = 0; i < n; i++){
for (int j = 0; j < n; j++){
cout << Matrix[i][j] << "*x" << j;
if (j < n - 1){
cout << " + ";
}
}
cout << " = " << y[i] << endl;
}
x = GaussianElimination (Matrix, y, n);
for (int i = 0; i < n; i++){
cout << "x[" << i << "]= " << x[i] << endl;
}
getch();
return 0;
}

```

### 3. Работа программы



```

C:\Users\Никита\Desktop\zad1\Debug\zad1.exe
Введите ранг матрицы: 2
Matrix[0][0]= 4
Matrix[0][1]= 5
Matrix[1][0]= 6
Matrix[1][1]= 3
y[0]= 4
y[1]= 1
4*x0 + 5*x1 = 4
6*x0 + 3*x1 = 1
x[0]=0
x[1]=0

```

## Список литературы

- [1] Кнут Д.Э. Всё про  $\text{T}_\text{E}\text{X}$ . — Москва: Изд. Вильямс, 2003 г. 550 с.
- [2] Львовский С.М. Набор и верстка в системе  $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ . — 3-е издание, исправленное и дополненное, 2003 г.
- [3] Воронцов К.В.  $\text{L}_\text{A}\text{T}_\text{E}\text{X}$  в примерах 2005 г.