МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа

«Решение системы линейных уравнений итерационным методом и

методом Гаусса-Зейделя»

по дисциплине

Вычислительная математика

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Суркова А.С.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

[Цель работы 3](#_Toc66356797)

[Вариант задания на лабораторную работу 3](#_Toc66356798)

[Краткие теоретические сведения и описание алгоритма работы программы в виде блок схем 4](#_Toc66356799)

[Метод Гаусса 4](#_Toc66356800)

[Метод простой итерации 7](#_Toc66356801)

[Метод Гаусса-Зейделя 8](#_Toc66356802)

[Расчётные данные (таблицы) 10](#_Toc66356803)

[Метод Гаусса (матрица, приведённая к треугольному виду) 10](#_Toc66356804)

[Метод простой итерации 10](#_Toc66356805)

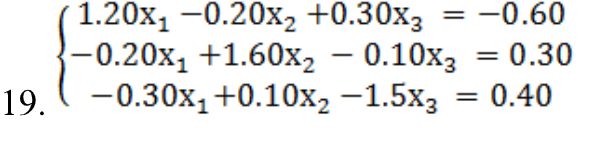
[Метод Гаусса-Зейделя 10](#_Toc66356806)

[Листинг программы 11](#_Toc66356807)

[Результаты работы программы 16](#_Toc66356808)

Цель работы: Закрепление знаний и умений по нахождению решений систем линейных уравнений различными способами.

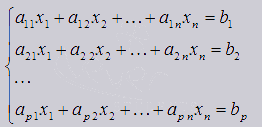
# Вариант задания на лабораторную работу



# Краткие теоретические сведения и описание алгоритма работы программы в виде блок схем

## Метод Гаусса

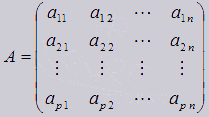
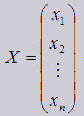
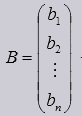
Пусть исходная система выглядит следующим образом:



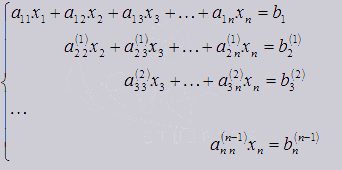
Её можно записать в матричном виде:



Где

Тогда, согласно свойству элементарных преобразований над строками, основную матрицу этой системы можно привести к ступенчатому виду (эти же преобразования нужно применять к столбцу свободных членов):



С этого момента начинаем обратный ход метода Гаусса: вычисляем xn из последнего уравнения как

,

с помощью полученного значения xn находим xn-1 из предпоследнего уравнения, и так далее, находим x1 из первого уравнения.

/\*Метод Гаусса

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f, b\_f, c\_f, d\_f - параметры первого уравнения

2)a\_s, b\_s, c\_s, d\_s - параметры второго уравнения

3)a\_t, b\_t, c\_t, d\_t - параметры третьего уравнения

Принцип работы:

1)Заполение вектора b свободными членами

2)Заполнение матрицы A коэффициентов системы переданными значениями

3)Поиск масимального коэффициента

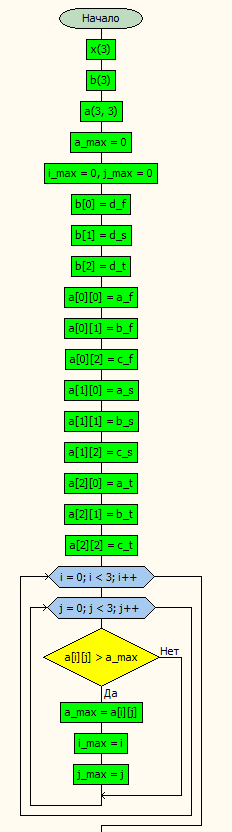
4)Перестановка первой строки и строки с максимальным коэффициентом в матрице A

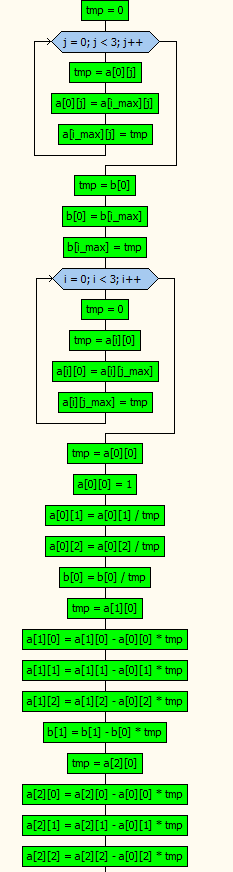
5)Перестановка первой строки и строки с максимальным коэффициентом в векторе b

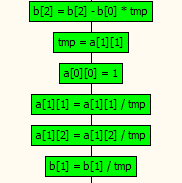
6)Перестановка первого столбца и столбца с максимальным коэффициентом в матрице A

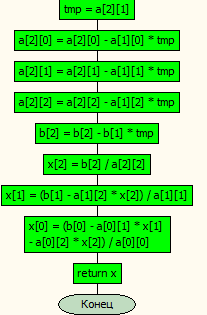
7)Приведение матрицы к треугольному виду

8)Обратный ход метода Гаусса для поиска корней системы\*/



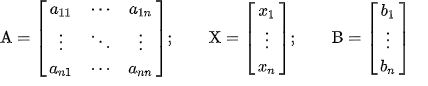




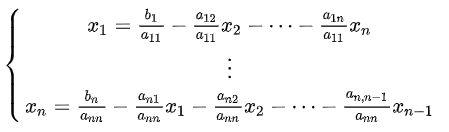


## Метод простой итерации

Пусть дана СЛАУ вида: {\displaystyle AX=B} AX=B, где



Предполагая, что aii не равно 0, выразим х1 через первое уравнение, х2 через второе и т.д.



На первом этапе вектор Х принимают равным нулевому вектору (х1=0, х2=0, …, хn=0). Вычисление корней продолжается

, где *Е* допустимая погрешность.

/\*Метод простой итерации

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f,b\_f,c\_f,d\_f - параметры первого уравнения

2)a\_s,b\_s,c\_s,d\_s - параметры второго уравнения

3)a\_t,b\_t,c\_t,d\_t - параметры третьего уравнения

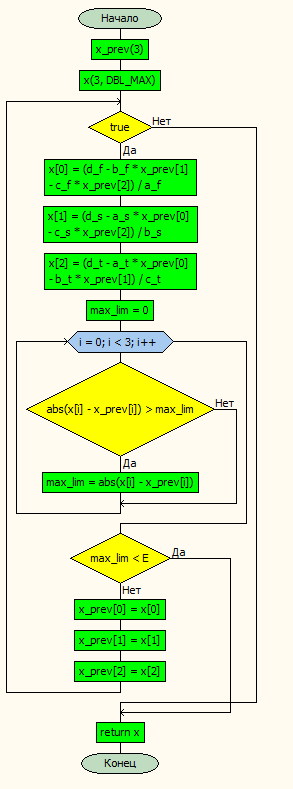
Принцип работы:

1)Формируются начальные решения системы уравнений - нули

2)Из каждого уравнения выражается корень(из первого-первый, из второго-второй, из третьего-третий)

3)Используя корни, найденные на предыдущем шаге, ищутся новые корни до тех пор, пока точность корней не достигнет заданной

\*/



## Метод Гаусса-Зейделя

Метод Гаусса-Зейделя –это модификация метода простой итерации. При вычислении корней системы используются значения, полученные на текущем шагу, что снижает количество итераций для нахождения корней.

/\*Метод Гауса-Зейделя

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f,b\_f,c\_f,d\_f - параметры первого уравнения

2)a\_s,b\_s,c\_s,d\_s - параметры второго уравнения

3)a\_t,b\_t,c\_t,d\_t - параметры третьего уравнения

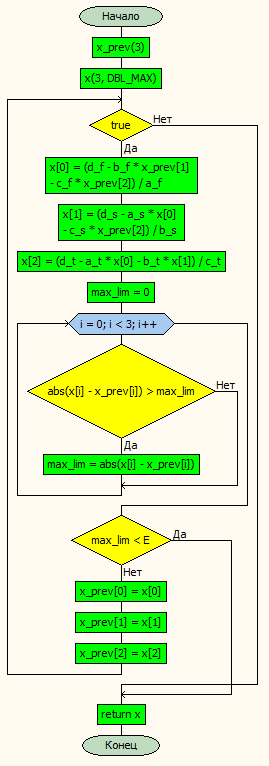
Принцип работы:

1)Формируются начальные решения системы уравнений - нули

2)Из каждого уравнения выражается корень(из первого-первый, из второго-второй, из третьего-третий)

3)Используя корни, найденные на предыдущем и текущем шаге, ищутся новые корни до тех пор, пока точность корней не достигнет заданной

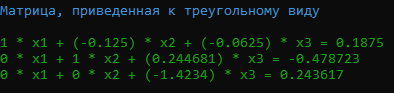
\*/



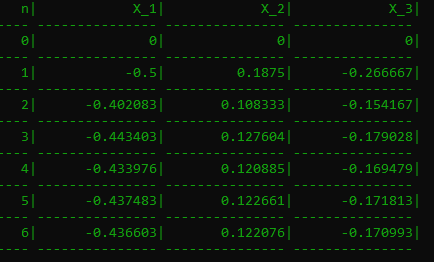
# Расчётные данные (таблицы)

Таблицы построены в результате работы созданной программы.

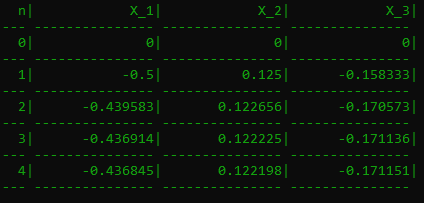
## Метод Гаусса (матрица, приведённая к треугольному виду)



## Метод простой итерации



## Метод Гаусса-Зейделя



# Листинг программы

#include <iostream>

#include <vector>

#include <iomanip>

double const E = 0.001;

using namespace std;

/\*Метод Гаусса

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f,b\_f,c\_f,d\_f - параметры первого уравнения

2)a\_s,b\_s,c\_s,d\_s - параметры второго уравнения

3)a\_t,b\_t,c\_t,d\_t - параметры третьего уравнения

Принцип работы:

1)Заполнение вектора b свободными членами

2)Заполнение матрицы A коэффициентов системы переданными значениями

3)Поиск максимального коэффициента

4)Перестановка первой строки и строки с максимальным коэффициентом в матрице A

5)Перестановка первой строки и строки с максимальным коэффициентом в векторе b

6)Перестановка первого столбца и столбца с максимальным коэффициентом в матрице A

7)Приведение матрицы к треугольному виду

8)Обратный ход метода Гаусса для поиска корней системы

\*/

vector<double> Gauss(double a\_f, double b\_f, double c\_f, double d\_f, double a\_s, double b\_s, double c\_s, double d\_s, double a\_t, double b\_t, double c\_t, double d\_t) {

vector<double>x(3);

vector<double>b(3);

vector<vector<double>>a(3);

double a\_max = 0;

int i\_max = 0, j\_max=0;

//Выделение памяти для матрицы коэффициентов

for (int i = 0; i < 3; i++) { a[i].resize(3); }

//Заполение вектора b свободными членами

b[0] = d\_f; b[1] = d\_s; b[2] = d\_t;

//Заполнение матрицы A коэффициентов системы переданными значениями

a[0][0] = a\_f; a[0][1] = b\_f; a[0][2] = c\_f;

a[1][0] = a\_s; a[1][1] = b\_s; a[1][2] = c\_s;

a[2][0] = a\_t; a[2][1] = b\_t; a[2][2] = c\_t;

//Поиск максимального коэффициента в матрице A

for (int i = 0; i < 3; i++){

for (int j = 0; j < 3; j++){

if (a[i][j] > a\_max) {

a\_max = a[i][j];

i\_max = i;

j\_max = j;

}

}

}

//Перестановка первой строки и строки с максимальным коэффициентом в матрице a

double tmp = 0;

for (int j = 0; j < 3; j++){

tmp = a[0][j];

a[0][j] = a[i\_max][j];

a[i\_max][j] = tmp;

}

//Перестановка первой строки и строки с максимальным коэффициентом в векторе b

tmp = b[0];

b[0] = b[i\_max];

b[i\_max] = tmp;

//Перестановка первого столбца и столбца с максимальным коэффициентом в матрице a

for (int i = 0; i < 3; i++) {

double tmp = 0;

tmp = a[i][0];

a[i][0] = a[i][j\_max];

a[i][j\_max] = tmp;

}

//Приведение матрицы к треугольному виду

tmp = a[0][0];

a[0][0] = 1;

a[0][1] = a[0][1] / tmp;

a[0][2] = a[0][2] / tmp;

b[0] = b[0] / tmp;

tmp = a[1][0];

a[1][0] = a[1][0] - a[0][0] \* tmp;

a[1][1] = a[1][1] - a[0][1] \* tmp;

a[1][2] = a[1][2] - a[0][2] \* tmp;

b[1] = b[1] - b[0] \* tmp;

tmp = a[2][0];

a[2][0] = a[2][0] - a[0][0] \* tmp;

a[2][1] = a[2][1] - a[0][1] \* tmp;

a[2][2] = a[2][2] - a[0][2] \* tmp;

b[2] = b[2] - b[0] \* tmp;

tmp = a[1][1];

a[0][0] = 1;

a[1][1] = a[1][1] / tmp;

a[1][2] = a[1][2] / tmp;

b[1] = b[1] / tmp;

tmp = a[2][1];

a[2][0] = a[2][0] - a[1][0] \* tmp;

a[2][1] = a[2][1] - a[1][1] \* tmp;

a[2][2] = a[2][2] - a[1][2] \* tmp;

b[2] = b[2] - b[1] \* tmp;

//Вывод матрицы на консоль

cout<<"Матрица, приведенная к треугольному виду\n"<<"\n\u001B[32m"

<< a[0][0] << " \* x1 + (" << a[0][1] << ") \* x2 + (" << a[0][2]

<< ") \* x3 = " << b[0] << "\n"

<< a[1][0] << " \* x1 + " << a[1][1] << " \* x2 + (" << a[1][2]

<< ") \* x3 = " << b[1] << "\n"

<< a[2][0] << " \* x1 + " << a[2][1]<< " \* x2 + (" << a[2][2]

<< ") \* x3 = " << b[2] << "\n";

//Обратный ход метода Гаусса для поиска корней системы

x[2] = b[2] / a[2][2];

x[1] = (b[1] - a[1][2] \* x[2]) / a[1][1];

x[0] = (b[0] - a[0][1] \* x[1] - a[0][2] \* x[2]) / a[0][0];

return x;

}

/\*Метод простой итерации

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f,b\_f,c\_f,d\_f - параметры первого уравнения

2)a\_s,b\_s,c\_s,d\_s - параметры второго уравнения

3)a\_t,b\_t,c\_t,d\_t - параметры третьего уравнения

Принцип работы:

1)Формируются начальные решения системы уравнений - нули

2)Из каждого уравнения выражается корень (из первого-первого, из второго-второй, из третьего-третьего)

3)Используя корни, найденные на предыдущем шаге, ищутся новые корни до тех пор, пока точность корней не достигнет заданной

\*/

vector<double> iteration(double a\_f, double b\_f, double c\_f, double d\_f, double a\_s, double b\_s, double c\_s, double d\_s, double a\_t, double b\_t, double c\_t, double d\_t) {

vector<double> x\_prev(3);

vector<double>x(3,DBL\_MAX);

int n = 0;

cout << "\n\u001B[32m" << setw(5) << "n|" << setw(16)

<< "X\_1|" << setw(16) << "X\_2|" << setw(17) << "X\_3|\n"

<< "---- --------------- --------------- ---------------\n"

<<setw(4)<<n<<"|"<<setw(15)<<0<<"|" << setw(15) << 0 << "|"

<< setw(15) << 0 << "|\n"

<< "---- --------------- --------------- ---------------\n";

while (true){

x[0] = (d\_f - b\_f \* x\_prev[1] - c\_f \* x\_prev[2]) / a\_f;

x[1] = (d\_s - a\_s \* x\_prev[0] - c\_s \* x\_prev[2]) / b\_s;

x[2] = (d\_t - a\_t \* x\_prev[0] - b\_t \* x\_prev[1]) / c\_t;

n++;

cout << setw(4) << n << "|" << setw(15) << x[0] << "|"

<< setw(15) << x[1] << "|" << setw(15) << x[2] << "|\n"

<< "---- --------------- --------------- ---------------\n";

double max\_lim = 0;

for (int i = 0; i < 3; i++){

if (abs(x[i] - x\_prev[i]) > max\_lim) {

max\_lim = abs(x[i] - x\_prev[i]);

}

}

if (max\_lim < E) {break;}

else {

x\_prev[0] = x[0];

x\_prev[1] = x[1];

x\_prev[2] = x[2];

}

}

return x;

}

/\*Метод Гауса-Зейделя

Уравнения задаются в виде a\*x\_1+b\*x\_2+c\*x\_3=d

Параметры функции:

1)a\_f,b\_f,c\_f,d\_f - параметры первого уравнения

2)a\_s,b\_s,c\_s,d\_s - параметры второго уравнения

3)a\_t,b\_t,c\_t,d\_t - параметры третьего уравнения

Принцип работы:

1)Формируются начальные решения системы уравнений - нули

2)Из каждого уравнения выражается корень (из первого-первого, из второго-второй, из третьего-третьего)

3)Используя корни, найденные на предыдущем и текущем шаге, ищутся новые корни до тех пор, пока точность корней не достигнет заданной

\*/

vector<double> Gauss\_Seidel(double a\_f, double b\_f, double c\_f, double d\_f, double a\_s, double b\_s, double c\_s, double d\_s, double a\_t, double b\_t, double c\_t, double d\_t) {

vector<double> x\_prev(3);

vector<double>x(3, DBL\_MAX);

int n = 0;

cout << "\n\u001B[32m" << setw(5) << "n|" << setw(16)

<< "X\_1|" << setw(16) << "X\_2|" << setw(17) << "X\_3|\n"

<< "---- --------------- --------------- ---------------\n"

<< setw(4) << n << "|" << setw(15) << 0 << "|" << setw(15)

<< 0 << "|"<< setw(15) << 0 << "|\n"

<< "---- --------------- --------------- ---------------\n";

while (true) {

x[0] = (d\_f - b\_f \* x\_prev[1] - c\_f \* x\_prev[2]) / a\_f;

x[1] = (d\_s - a\_s \* x[0] - c\_s \* x\_prev[2]) / b\_s;

x[2] = (d\_t - a\_t \* x[0] - b\_t \* x[1]) / c\_t;

n++;

cout << setw(4) << n << "|" << setw(15) << x[0] << "|"

<< setw(15) << x[1] << "|" << setw(15) << x[2] << "|\n"

<< "---- --------------- --------------- ---------------\n";

double max\_lim = 0;

for (int i = 0; i < 3; i++) {

if (abs(x[i] - x\_prev[i]) > max\_lim) {

max\_lim = abs(x[i] - x\_prev[i]);

}

}

if (max\_lim < E) { break; }

else {

x\_prev[0] = x[0];

x\_prev[1] = x[1];

x\_prev[2] = x[2];

}

}

return x;

}

int main() {

double a\_f = 0, b\_f =0, c\_f = 0,d\_f=0;

double a\_s = 0, b\_s =0, c\_s = 0, d\_s = 0;

double a\_t = 0, b\_t = 0, c\_t = 0, d\_t = 0;

char metod=0;

setlocale(LC\_ALL, "Russian");

//Ввод исходных данных

cout <<"\u001B[33mДанная программа предназначения для решения системы"

<<" трёх линейных уравнений.\n"

<<"Уравнение представляется в виде a\*x\_1+b\*x\_2+c\*x\_3=d.\n"

<<"Введите коэффициенты первого уравнения\n"

<<"\u001B[36m\ta = ";

cin >> a\_f;

cout << "\tb = ";

cin >> b\_f;

cout << "\tc = ";

cin >> c\_f;

cout << "\td = ";

cin >> d\_f;

cout<< "\u001B[33m\nВведите коэффициенты второго уравнения\n"

<< "\u001B[36m\ta = ";

cin >> a\_s;

cout << "\tb = ";

cin >> b\_s;

cout << "\tc = ";

cin >> c\_s;

cout << "\td = ";

cin >> d\_s;

cout << "\u001B[33m\nВведите коэффициенты третьего уравнения\n"

<< "\u001B[36m\ta = ";

cin >> a\_t;

cout << "\tb = ";

cin >> b\_t;

cout << "\tc = ";

cin >> c\_t;

cout << "\td = ";

cin >> d\_t;

while (metod!='q') {

cout << "\u001B[31m\n\n---------------------------------------\n";

//Выбор метода решения системы

cout << "\u001B[33m\nВыберите метод решения уравнения:\n"

<< "\u001B[36m\t{1} - Метод Гаусса\n"

<< "\t{2} - Метод простой итерации\n"

<< "\t{3} - Метод Гауса-Зейдена\n"

<< "\t{q} - для выхода из программы\n\t";

cin >> metod;

//Решение выбранным методом

vector<double>x(3);

time\_t start = clock();

switch (metod) {

case '1':

x = Gauss(a\_f, b\_f, c\_f, d\_f, a\_s, b\_s, c\_s, d\_s, a\_t, b\_t,

c\_t, d\_t);

break;

case '2':

x = iteration(a\_f, b\_f, c\_f, d\_f, a\_s, b\_s, c\_s, d\_s, a\_t,

b\_t, c\_t, d\_t);

break;

case '3':

x = Gauss\_Seidel(a\_f, b\_f, c\_f, d\_f, a\_s, b\_s, c\_s, d\_s, a\_t,

b\_t, c\_t, d\_t);

break;

}

if (metod != 'q') {

//Вывод результата решения

cout << "\u001B[36m\nВремя, затраченное на нахождение корней = "

<< (double)(clock() - start) / CLOCKS\_PER\_SEC

<< " секунд\n";

for (int i = 0; i < 3; i++) {

cout << i + 1 << " корень = "<< x[i] << "\n";

}

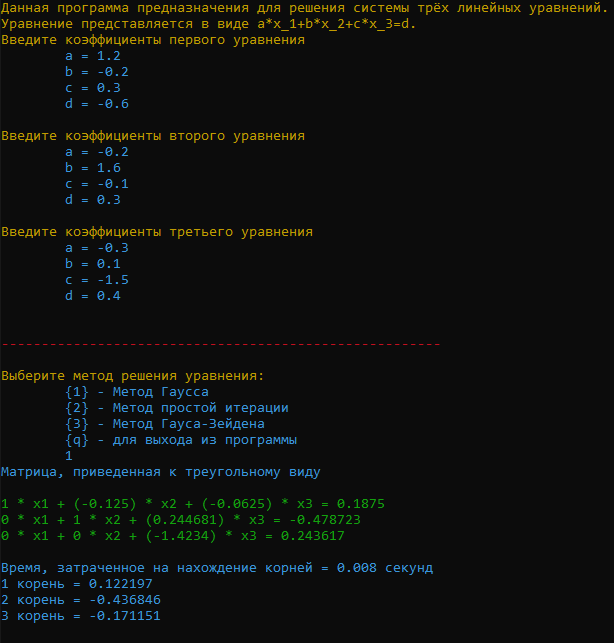
}

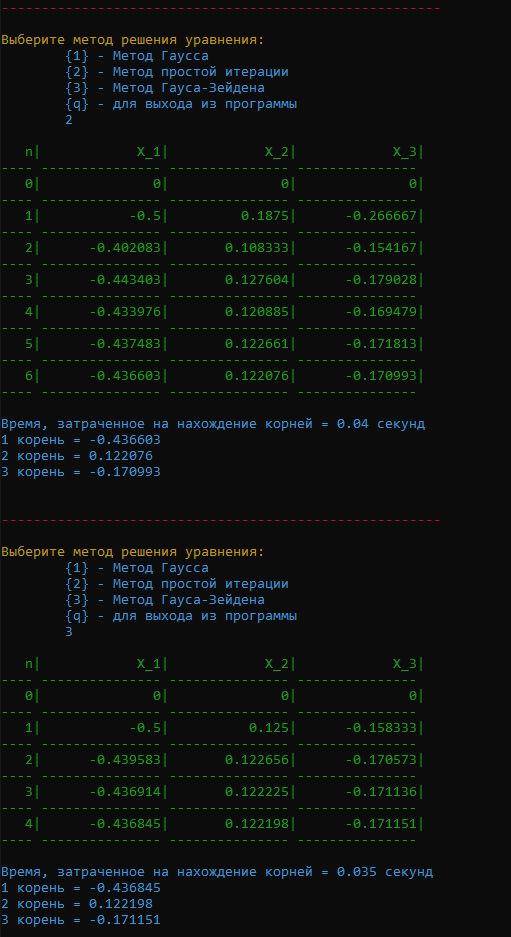
}

return 0;

}

# Результаты работы программы





# Выводы

Закрепил знания и умения по нахождению корней системы линейных уравнений. Реализовал три метода решения системы. Самым быстро действенным методом оказался метод Гаусс, время его работы равно 0.008 секунд. Самым медленным оказался метод простой итерации, время его работы 0.04 секунды. Метод Гаусса-Зейделя оказался более быстро действенным, ему потребовалось на 2 итерации меньше, что сэкономило 0.005 секунд.