Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе

на тему

Численное решение систем линейных уравнений методом простых итераций и методом Зейделя

Выполнил: студент группы 253505

Снежко Максим Андреевич

Проверил: Анисимов Владимир Яковлевич 

Минск 2023

**Вариант 27**

**Цель выполнения задания:**

* Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя);
* Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
* Составить программу решения СЛАУ по разработанному алгоритму;
* Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоёмкость решения методом простых итераций и методом Зейделя.

**Краткие теоретические сведения**

Итерационные методы основаны на построении сходящейся к точному решению х рекуррентной последовательности.

Для решения СЛАУ методом простых итераций преобразуем систему от первоначальной формы Ax = b или

a11 x1 + a12 x2 + … + a1n xn = b1 ,

a21 x1 + a22 x2 + … + a2n xn = b2 ,

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

(2.1) an1 x1 + an2 x2 + … + ann xn = bn

к виду

    x = Bx + c                                     (2.2)

Здесь B - квадратная матрица с элементами bij (i, j = 1, 2. ..., n), с – вектор столбец с элементами сi (i = 1,2,..., n).

В развернутой форме записи система (2.2) имеет следующий вид:

x1 = b11x1 + b12x2 + b13x3 + … + b1nxn + c1

x2 = b21x1 + b22x2 + b23x3 + … + b2nxn + c2

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

xn = bn1x1 + bn2x2 + bn3x3 + … + bnnxn + cn

Вообще говоря, операция приведения системы к виду, удобному для итераций, не является простой и требует специальных знаний, а также существенного использования специфики системы.

Можно, например, преобразовать систему (2.1) следующим образом

x1 = (b1 - a11 x1 – a12 x2 - … - a1n xn) / a11 + x1,

x2 = (b2 - a21 x1 – a22 x2 - … - a2n xn) / a22 + x2,

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

xn = (bn - an1 x1 – an2 x2 - … - ann xn) / ann + xn

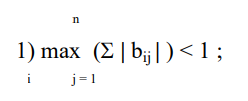
если диагональные элементы матрицы А отличны от нуля.

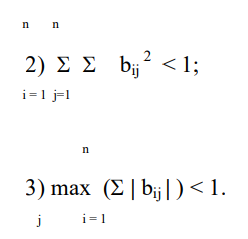
Можно преобразовать систему (2.1) в эквивалентную ей систему

x = (E-A)x+b.

    Задав произвольным образом столбец начальных приближений x0 = (x10, x20, ... . xn0 )Т . подставим их в правые части системы (2.2) и вычислим новые приближения x1 = (x11, x21, ... . xn1), которые опять подставим в систему (2.2) и т.д. Таким образом, организуется итерационный процесс      xk = Bxk-1 + c, k = 1,2 ..., . Известно, что система (2.1) имеет единственное решение x\* и последовательность {xk} сходится к этому решению со скоростью геометрической прогрессии, если  || В || < 1 в любой матричной норме.

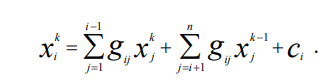
Т.e. для того, чтобы последовательность простых итераций сходилась к единственному решению достаточно, чтобы выполнялось одно из следующих условий:

****

****

**Метод Зейделя**

Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего хik : 2 в формуле xk = Bxk-1 + c , k =1,2, ...  используются вместо х1k-1 ,…,  уже вычисленные ранее хik,..., т.e.

 (2.3)

Такое усовершенствование позволяет ускорить сходимость итераций почти в два раза. Кроме того, данный метод может быть реализован на ЭВМ без привлечения дополнительного массива, т.к. полученное новое   сразу засыпается на место старого.

Схема алгоритма аналогична схеме метода простых итераций.

Самый простой способ приведения системы к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы выразим неизвестное х1 :

 x1 = a11 –1 (b1 – a12x2 – a13x3 – … – a1nxn),

из второго уравнения – неизвестное х2:

x2: x2 = a21 –1 (b2 – a22x2 – a23x3 – … – a2nxn),

и т. д.

В результате получим систему:

x1 = b12x2 + b13x3 + … + b1,n–1xn–1 + b1nxn+c1 ,

x2 = b21x1 + b23x3 + … + b2,n–1xn–1 + b2nxn+c2 ,

x3 = b31x1 + b32x2 + … + b3,n–1xn–1 + b3nxn+c3 ,

. . . . . . . . . . . . . . . . . . . . . . .

xn = bn1x1 + bn2x2 + bn3x3 + … + bn,n–1xn–1 + cn ,

в которой на главной диагонали матрицы B находятся нулевые элементы.

Остальные элементы выражаются по формулам



    Конечно, для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы А были ненулевыми.

Введем нижнюю В1 (получается из В заменой нулями элементов стоявших на главной диагонали и выше ее) и верхнюю В2 (получается из В заменой нулями элементов стоявших на главной диагонали и ниже ее) треугольные матрицы.

    Заметим, что B = B1 + B2 и поэтому решение х исходной системы удовлетворяет равенству

x = B1x + B2x+c                    (2.5)

Выберем начальное приближение x(0) = [x1(0), x2(0), … , xn(0)]T. Подставляя его в правую часть равенства при верхней треугольной матрице B2 и вычисляя полученное выражение, находим первое приближение

x(1) = B1x(0) + B2x(1)                    (2.6)

Подставляя приближение x(1), получим

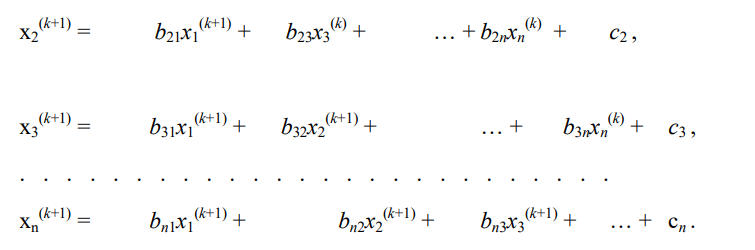
        X(2) = B1x(1) + B2x(2)                    (2.7)

Продолжая этот процесс далее, получим последовательность x(0) ,x(1), … , x(n) , … приближений к вычисляемых по формуле

X(k+1) = B1(k+1) + B2(k) + c                (2.8)

или в развернутой форме записи:





Объединив приведение системы к виду, удобному для итераций и метод Зейделя в одну формулу, получим



Тогда достаточным условием сходимости метода Зейделя будет условие доминирования диагональных элементов в строках или столбцах матрицы

A, т.e.

aii>ai1+ ……..+ain для всех i=1,…n,

или ajj>a1j+…….+anj для всех j=1,….,n.

Методы простой итерации и Зейделя сходятся примерно так же, как геометрическая прогрессия со знаменателем || В || .

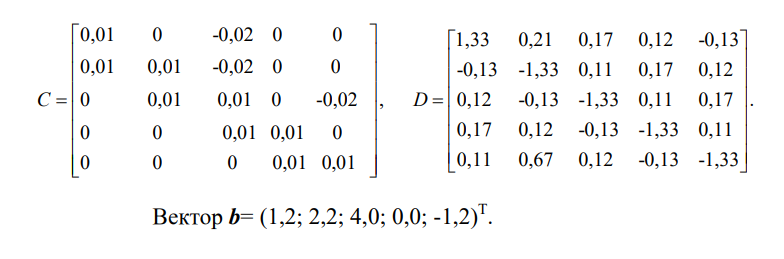
**ЗАДАНИЕ**

Методом простых итераций и методом Зейделя найти с

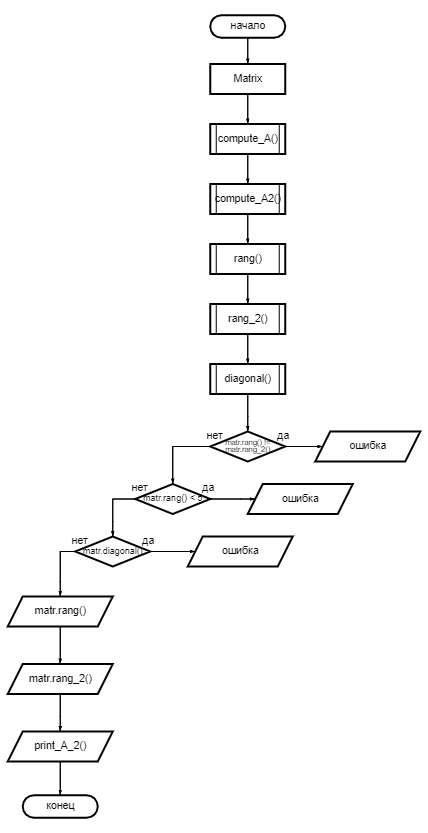
точностью 0,0001 численное решение системы Аx=b,

где A = kC + D,  A - исходная матрица для расчёта,  k - номер варианта

(0-15), матрицы C, D и вектор свободных членов ь задаются ниже.



**Алгоритм выполнения задания**

****

**Программная реализация**

#include <iostream>

#include <iomanip>

#include <vector>

class Matrix

{

private:

int V;

double A[5][5];

double A\_2[5][6];

double C[5][5]

{

{1.0 / 100, 0, -2.0 / 100, 0, 0},

{1.0 / 100, 1.0 / 100, -2.0 / 100, 0, 0},

{0, 1.0 / 100, 1.0 / 100, 0, -2.0 / 100},

{0, 0, 1.0 / 100, 1.0 / 100, 0},

{0, 0, 0, 1.0 / 100, 1.0 / 100}

};

double D[5][5]

{

{133.0 / 100, 21.0 / 100, 17.0 / 100, 12.0 / 100, -13.0 / 100},

{-13.0 / 100, -133.0 / 100, 11.0 / 100, 17.0 / 100, 12.0 / 100},

{12.0 / 100, -13.0 / 100, -133.0 / 100, 11.0 / 100, 17.0 / 100},

{17.0 / 100, 12.0 / 100, -13.0 / 100, -133.0 / 100, 11.0 / 100},

{11.0 / 100, 67.0 / 100, 12.0 / 100, -13.0 / 100, -133.0 / 100}

};

double b[5]{ 12.0 / 10, 22.0 / 10, 40.0 / 10, 0, -12.0 / 10 };

double roots[5]{ 0, 0, 0, 0, 0 };

public:

Matrix(const int V)

{

this->V = V;

std::cout << std::fixed << std::setprecision(4);

compute\_A();

compute\_A\_2();

}

void compute\_A()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

A[i][j] = V \* C[i][j] + D[i][j];

}

}

}

void compute\_A\_2()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

A\_2[i][j] = A[i][j];

}

}

A\_2[0][5] = 12.0 / 10;

A\_2[1][5] = 22.0 / 10;

A\_2[2][5] = 40.0 / 10;

A\_2[3][5] = 0;

A\_2[4][5] = -12.0 / 10;

}

void print\_A()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

std::cout << std::setw(10) << A[i][j];

}

std::cout << std::setw(5) << '|' << std::setw(5) << b[i] << '\n';

}

}

void print\_A\_2()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

std::cout << std::setw(10) << A\_2[i][j];

}

std::cout << std::setw(5) << '|' << std::setw(5) << A\_2[i][5] << std::endl;

}

}

// Метод простых итераций

std::vector<double> simpleIterations(int& iterations)

{

// Размер матрицы

int size = 5;

// Неизвестные на предыдущей итерации

std::vector <double> prev\_X(size, 0.0);

// Пока не будет достигнута точность

bool stop = false;

while (!stop)

{

++iterations;

// Неизвестные на текущей итерации

std::vector <double> current\_X(size);

for (int i = 0; i < size; i++)

{

// x\_i = b\_i

current\_X[i] = A\_2[i][size];

// Вычитаем сумму по всем отличным от i-ой неизвестным

for (int j = 0; j < size; j++)

{

// С прошлой итерации

if (j != i)

{

current\_X[i] -= A\_2[i][j] \* prev\_X[j];

}

}

// x\_i /= b\_i

current\_X[i] /= A\_2[i][i];

}

// Максимальная погрешность

long double max\_error = 0.0;

for (int i = 0; i < size; i++)

{

double new\_max\_error = abs(current\_X[i] - prev\_X[i]);

max\_error = new\_max\_error > max\_error ? new\_max\_error : max\_error;

}

// Дотигнута ли точность

if (max\_error < 0.0001)

{

stop = true;

}

// Переход к следующей итерации

prev\_X = current\_X;

}

return prev\_X;

}

// Метод Зейделя

std::vector<double> seidelMethod(int& iterations)

{

// Размер матрицы

int size = 5;

// Неизвестные на предыдущей итерации

std::vector <double> prev\_X(size, 0.0);

// Пока не будет достигнута точность

bool stop = false;

while (!stop)

{

++iterations;

// Неизвестные на текущей итерации

std::vector <double> current\_X(size);

for (int i = 0; i < size; i++)

{

// x\_i = b\_i

current\_X[i] = A\_2[i][size];

// Вычитаем сумму по всем отличным от i-ой неизвестным

for (int j = 0; j < size; j++)

{

// С этой итерации

if (j < i)

{

current\_X[i] -= A\_2[i][j] \* current\_X[j];

}

// С прошой итерации

if (j > i)

{

current\_X[i] -= A\_2[i][j] \* prev\_X[j];

}

}

// x\_i /= b\_i

current\_X[i] /= A\_2[i][i];

}

// Максимальная погрешность

long double max\_error = 0.0;

for (int i = 0; i < size; i++)

{

double new\_max\_error = abs(current\_X[i] - prev\_X[i]);

max\_error = new\_max\_error > max\_error ? new\_max\_error : max\_error;

}

// Дотигнута ли точность

if (max\_error < 0.0001)

{

stop = true;

}

// Переход к следующей итерации

prev\_X = current\_X;

}

return prev\_X;

}

// Функции для проверки исходной матрицы :

// Нахождение ранга матрицы

int rang()

{

int rang = 5;

double sum = 0;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

sum += A[i][j];

}

if (sum == 0)

{

rang = rang - 1;

}

sum = 0;

}

return rang;

}

int rang\_2()

{

int rang = 6;

double sum = 0;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

sum += A\_2[i][j];

}

if (sum == 0)

{

rang = rang - 1;

}

sum = 0;

}

return rang;

}

// Проверка на преобладание главной диагонали

bool diagonal()

{

int i, j, k = 1;

double sum;

for (i = 0; i < 5; i++)

{

sum = 0;

for (j = 0; j < 5; j++)

{

sum += A[i][j];

}

sum -= A[i][i];

if (sum < A[i][i])

{

return 0;

}

}

return k == 1;

}

};

int main()

{

setlocale(0, "");

Matrix matr(27);

if (matr.rang() != matr.rang\_2())

{

std::cout << "Ранг исходной матрицы не равен рангу расширенной матрицы.\nСЛАУ не имеет решений.";

return 0;

}

if (matr.rang() < 5)

{

std::cout << "Ранги матрицы равны или меньше числа неизвестных системы.\nСЛАУ имеет бесконечное множество решений.";

return 0;

}

if (matr.diagonal())

{

std::cout << "Исходная матрица не обладает свойством диагонального преобладания.\nРешение методом итераций/Зейделя невозможен.";

return 0;

}

std::cout << matr.rang() << ", " << matr.rang\_2();

std::cout << "\t\t\tМетод итераций:" << "\n\n";

std::cout << "\tИсходная матрица:" << "\n";

matr.print\_A\_2();

int iterations = 0;

std::vector<double> vect = matr.seidelMethod(iterations);

std::cout << iterations << "\n";

int t = 0;

for (auto i : vect)

{

std::cout << ++t << ". " << i << "\n";

}

std::cout << "\n\n\t\t\tМетод Зейделя" << std::endl << std::endl;

std::cout << "\tИсходная матрица:" << std::endl;

matr.compute\_A();

matr.compute\_A\_2();

matr.print\_A\_2();

iterations = 0;

vect = matr.simpleIterations(iterations);

std::cout << iterations << "\n";

for (auto i : vect)

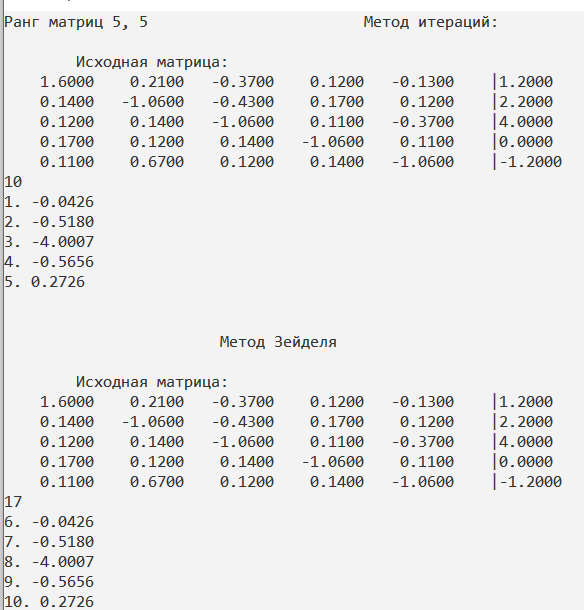
{

std::cout << ++t << ". " << i << "\n";

}

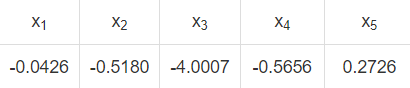
}

**Полученные результаты программы**

****

**Результаты, полученные в онлайн-калькуляторе:**

С сайта math.semestr.ru :



**Оценка**

Таким образом, мы видим, что результаты, полученные в онлайн калькуляторе, практически идентичны нашим. Различия лишь в погрешности вычисления нашей программы и онлайн-калькулятора

Рассчитаем относительную погрешность:

Δ(x1) = |x – x1| = |-0.0426 – (–0.0425653426)| = 0.0000346574

Δ(x2) = |x – x2| = |–0.5180 – (–0.5180234215)| = 0.0000234215

Δ(x3) = |x – x3| = |–4.0007 – (–4.0006356637)| = 0.0000643365

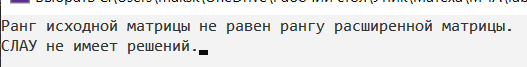
Δ(x4) = |x – x4| = |–0.5656 – (–0.5656431278)| = 0.0000431278

Δ(x5) = |x – x5| = |0.2726 – 0.2726403142| = 0.0000403142

**Тестовые примеры  
  
Пример 1**

Возьмем в качестве матрицы А нулевую матрицу. В этом случае СЛАУ не имеет решений, так как ранг исходной матрицы не равен рангу расширенной матрицы.

Результат выполнения программы:



**Пример 2**

A = { {2.33,0.81,0.67,0.92,-0.53},

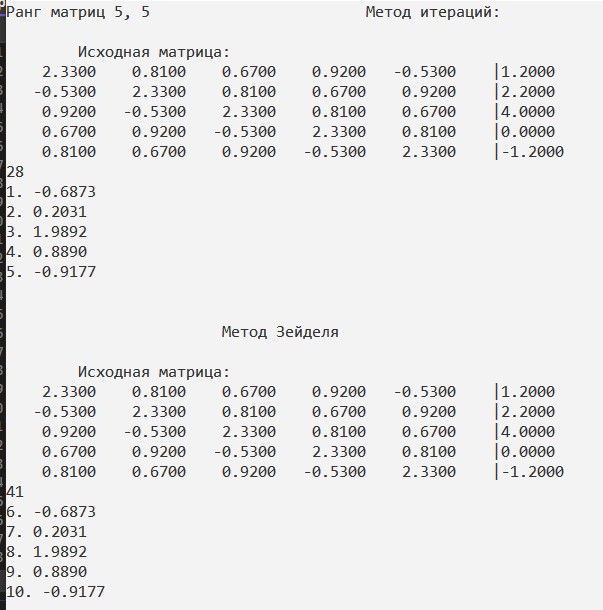
{-0.53,2.33,0.81,0.67,0.92},

{0.92,-0.53,2.33,0.81,0.67},

{0.67,0.92,-0.53,2.33,0.81},

{0.81,0.67,0.92,-0.53,2.33} };

Результат выполнения программы:



**Пример 3**

Пусть матрица А содержит в себе следующие данные:

A { {1,2,3,4,5},

{0,0,0,0,0},

{0,0,0,0,0},

{0,0,0,0,0},

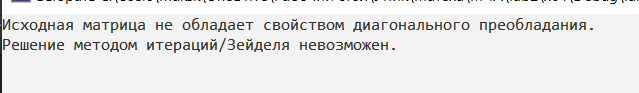
{0,0,0,0,0} };

Можно заметить, что в первой строке данной матрицы не выполняется следующее правило:



А это значит, что матрица не обладает свойством диагонального преобладания, следовательно решать ее методом итераций или Зейделя нельзя.

Результат выполнения программы:



Данный результат соответствует ожидаемому исходу.

**Пример 4**

Пусть матрица А содержит в себе следующие данные:

A { {1,2,3,4,5},

{0,0,0,0,0},

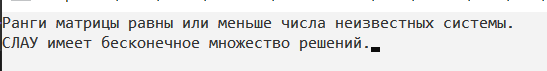
{0,0,5,0,0},

{0,0,0,4,0},

{0,3,0,0,0} };

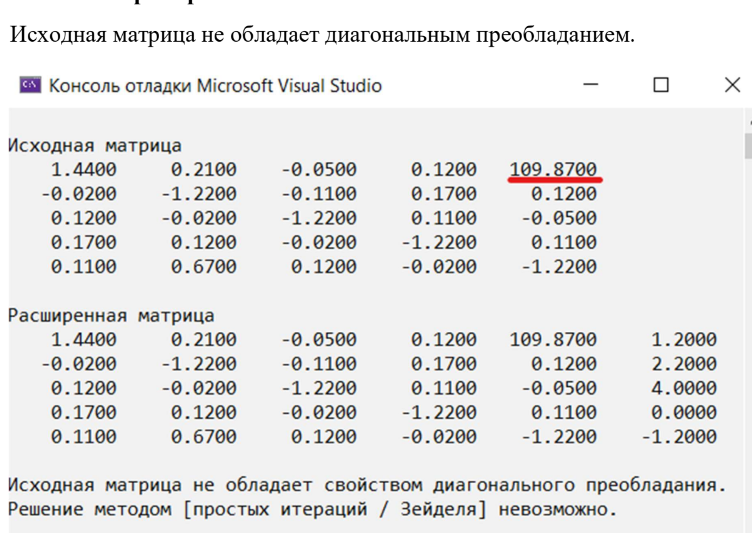
Ранг такой матрицы равен 4, что меньше чем количество неизвестных (5), поэтому по следствию из теоремы Кронекера-Капелли система имеет бесконечное множество решений.

Результат выполнения программы:

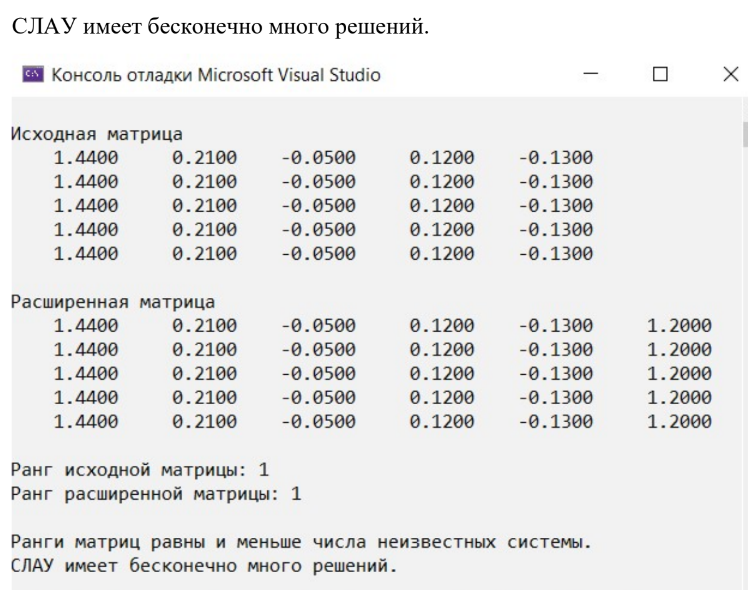


Данный результат соответствует ожидаемому исходу.

**Пример 5**



**Пример 6**



**Вывод:** в ходе выполнения данной лабораторной работы были изучены и применены итерационные методы решения СЛАУ, в частности метод простых итераций и метод Зейделя для решения системы линейных уравнений; были созданы соответствующие алгоритмы, применимые для организации вычислений на ЭВМ, и реализации программ на языке С++ для решения поставленной задачи; для проверки правильности работы программы были выполнены тестовые примеры и проведена оценка.