Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Методы численного анализа»

**ОТЧЕТ**

к лабораторной работе №1

на тему:

**«Решение систем линейных алгебраических уравнений (СЛАУ)**

**методом Гаусса и с помощью его модификаций»**

БГУИР 6-05-0612-01

|  |
| --- |
| Выполнил студент группы 453503  АВРАМЕНКО Роман Александрович |
|  |
| (дата, подпись студента) |
| Проверил доцент каф. Информатики  АНИСИМОВ Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

Минск 2025

**Содержание**

[**1 ЦЕЛЬ ВЫПОЛНЕНИЯ ЗАДАНИЯ** 3](#_Toc208394545)

[**2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ** 4](#_Toc208394546)

[**3 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ И ХОД ВЫПОЛНЕНИЯ РАБОТЫ** 8](#_Toc208394547)

[**4 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ** 1](#_Toc208394548)5

[**ВЫВОД**](#_Toc208394550) 17

# **1 ЦЕЛЬ ВЫПОЛНЕНИЯ ЗАДАНИЯ**

1 Изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;

2 Составить алгоритм решения СЛАУ указанными методами,

применимый для организации вычислений на ЭВМ;

3 Составить программу решения СЛАУ по разработанному алгоритму;

4 Выполнить тестовые примеры и проверить правильность работы

программы.

# **2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**



Здесь ***A*** и ***b*** заданы, требуется найти ***x***.

Методы решения СЛАУ делятся на прямые и итерационные.

*Прямые методы* дают в принципе точное решение(если не учитывать ошибок округления) за конечное число арифметических операций. Они просты и наиболее универсальны. Для хорошо обусловленных систем небольшого порядка n < *200* применяются практически только прямые методы.

Наибольшее распространение среди прямых методов получили метод Гаусса и его модификации.

**Метод Гаусса.** Одним из самых распространенных методов решения систем линейных уравнений является метод Гаусса. Этот метод (который также называют *методом последовательного исключения неизвестных*) широко известен в различных вариантах.

Вычисления с помощью метода Гаусса заключаются в последовательном исключении неизвестных из системы для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисления значений неизвестных производят на этапе обратного хода.

Рассмотрим сначала простейший вариант метода Гаусса, называемый *схемой единственного деления*.

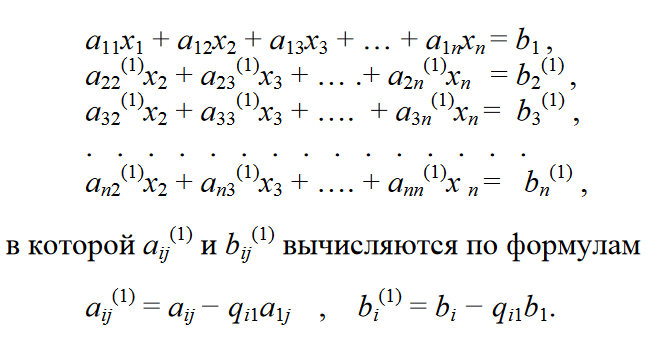
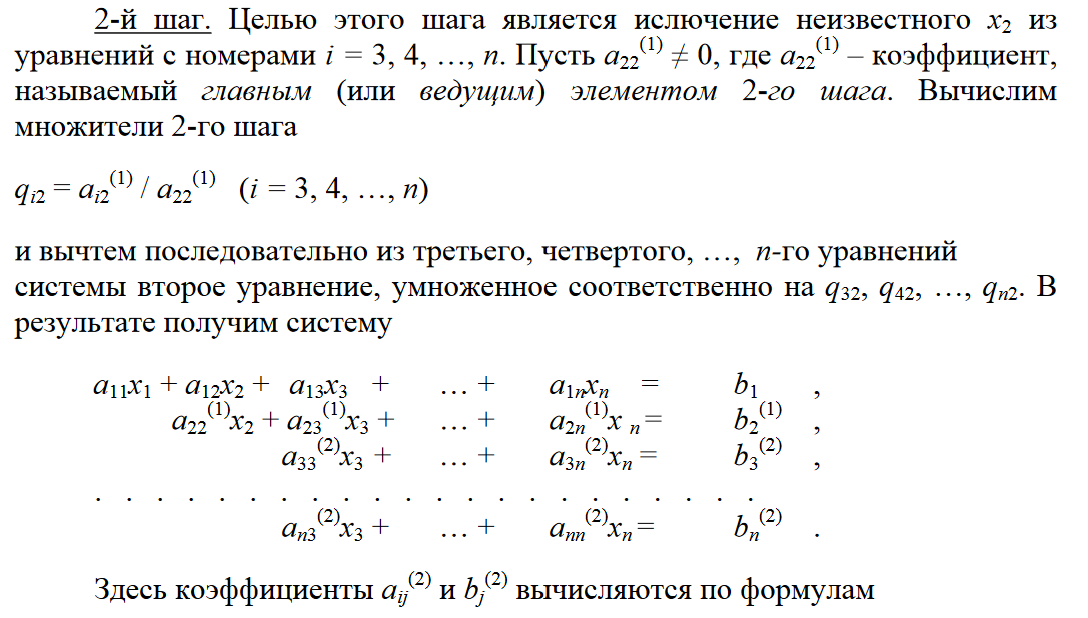
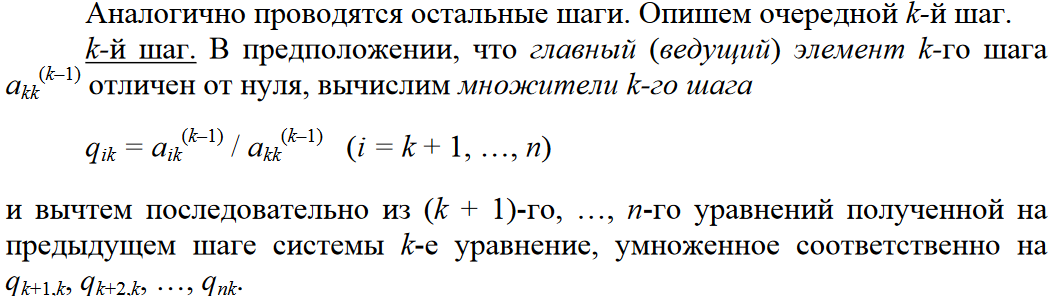
Прямой ход состоит из *n*  1 шагов исключения.

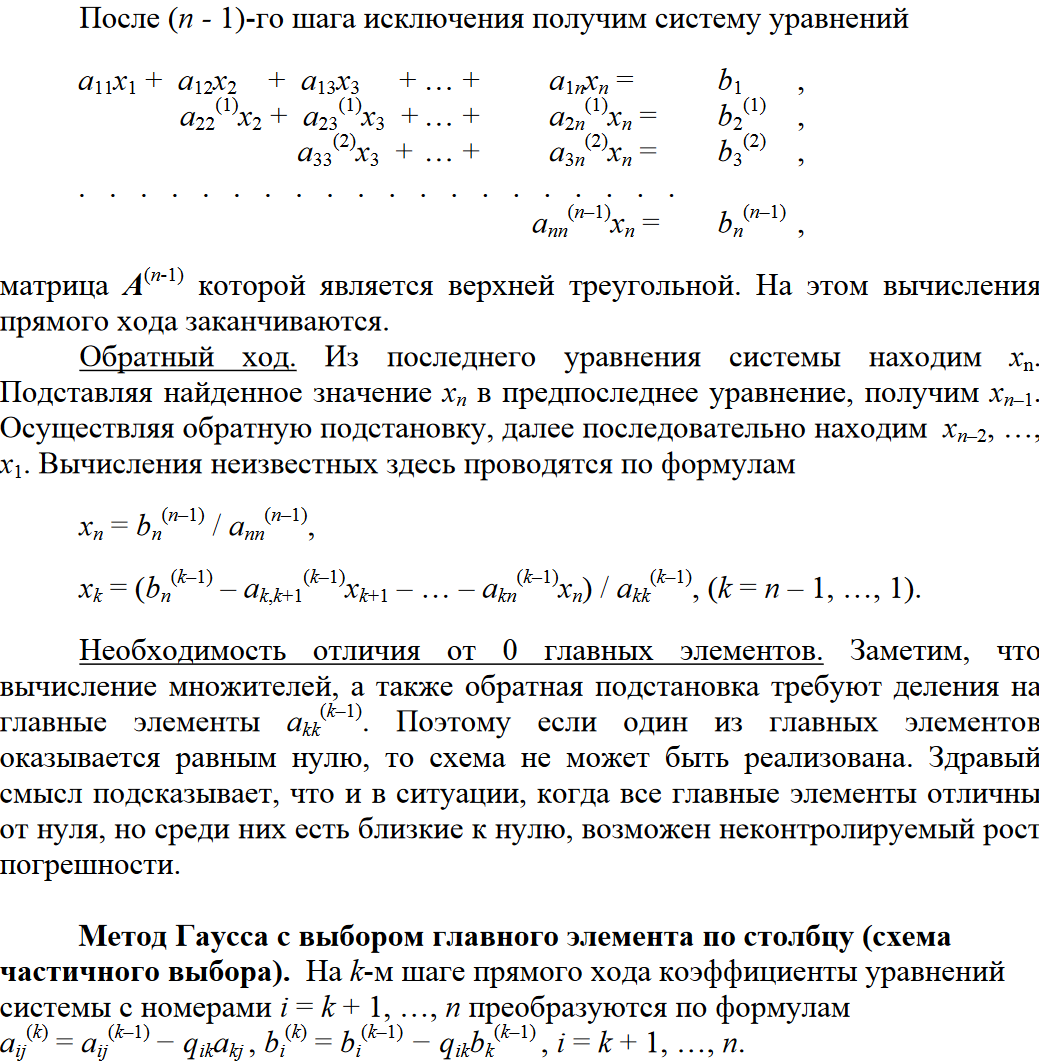
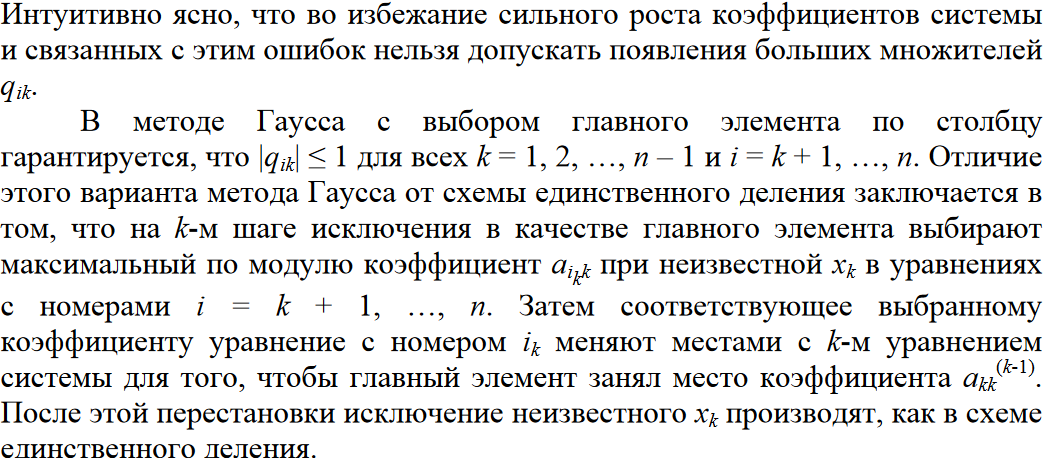
1-й шаг. Целью этого шага является исключение неизвестного *x*1 из уравнений с номерами *i* = 2, 3, …, *n*. Предположим, что коэффициент *a*11  0. Будем называть его *главным элементом* 1-*го шага*.

Найдем величины

*qi*1 *= ai*1/*a*11 (*i =* 2, 3, …, *n*),

называемые *множителями* 1-*го шага*. Вычтем последовательно из второго, третьего, …, *n-*го уравнений системы первое уравнение, умноженное соответственно на *q2*1*, q*31*, …, qn*1. Это позволит обратить в нуль коэффициенты при *x*1 во всех уравнениях, кроме первого. В результате получим эквивалентную систему





**Метод Гаусса с выбором главного элемента по всей матрице (схема**

**полного выбора).** В этой схеме допускается нарушение естественного порядка исключения неизвестных. На 1-м шаге метода среди элементов *aij* определяют максимальный по модулю элемент *ai*1*j*1. Первое уравнение системы и уравнение с номером *i*1 меняют местами. Далее стандартным образом производят исключение неизвестного *xi*1 из всех уравнений, кроме первого.

На *k*-м шаге метода среди коэффициентов *a* (*k*–1) при неизвестных в уравнениях системы с номерами *i* = *k*, …, *n* выбирают максимальный по модулю коэффициент *ai j* (*k*-1). Затем *k*-е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное *xjk* из уравнений с номерами *i* = *k* + 1, …, *n*.

*ij*

*k k*

На этапе обратного хода неизвестные вычисляют в следующем порядке: *xjn, xjn–*1*, …, xj*1.

# **3 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ И ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

**ЗАДАНИЕ** 1: Вариант 1. Методом Гаусса и методом выбора главного элементанайти с точностью 0,0001 численное решение системы Ax=b, где A = kC + D, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы С, D и вектор свободных членов b задаются программно.

При выполнении программы была использована система компьютерной алгебры Maple с пакетом LinearAlgebra, а также язык программирования C++.

Для начала в системе Maple зададим начальные значения для матриц C, D и b и посчитаем матрицу A для варианта 1.

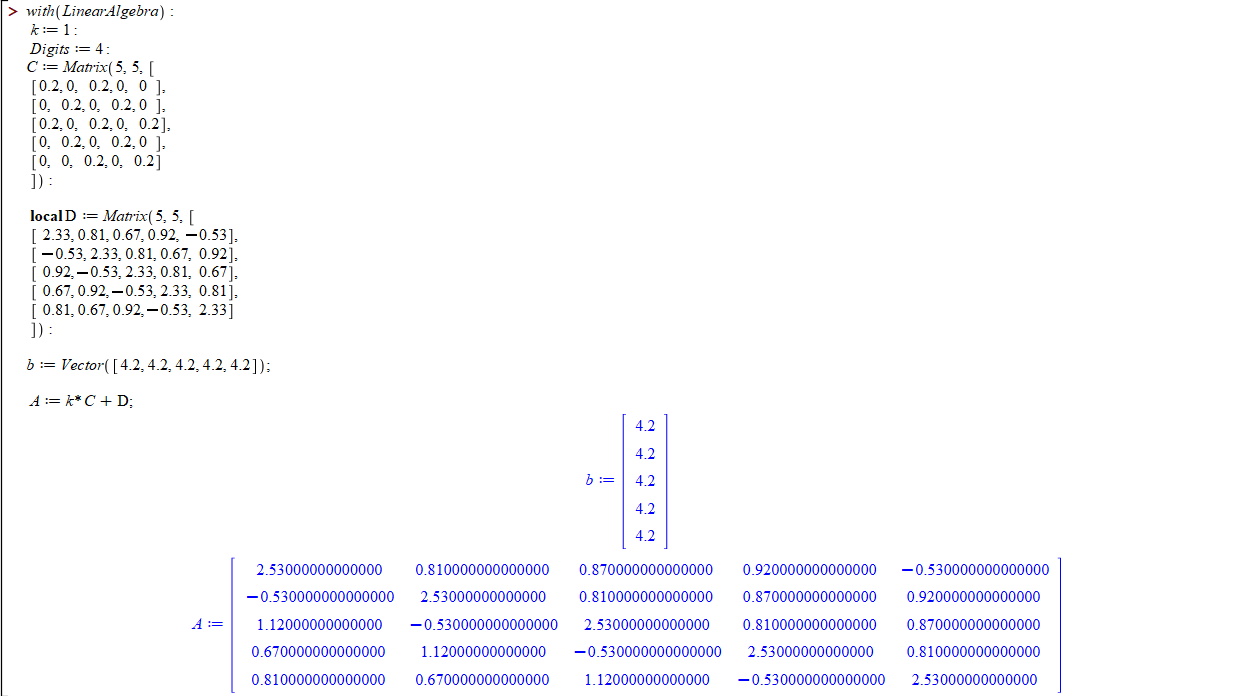


Рисунок 1 – Инициализация входных данных

Затем, объявим вспомогательные процедуры для поиска ненулевого коэффициента в матрице и для выполнения этапа обратного хода в методе Гаусса.

findNotZeroCoefficient := proc(A::Matrix, i::integer)

local j, k;

for j from i to RowDimension(A) do

for k from i to ColumnDimension(A) do

if A[j, k] <> 0 then

return [j, k];

end if;

end do;

end do;

return [-1, -1];

end proc:

backSubstitution := proc(A::Matrix, b::Vector, varSubst::Vector)

local i, j, temp, res;

res := Vector(ColumnDimension(A));

for i from RowDimension(A) by -1 to 1 do

temp := 0;

for j from i + 1 to ColumnDimension(A) do

temp := temp + A[i, j]\*res[varSubst[j]];

end do;

res[varSubst[i]] := (b[i] - temp)/A[i, i];

end do;

return res;

end proc:

Реализуем алгоритм нахождения решений СЛАУ обычным методом Гаусса. При этом будем учитывать, что элемент на диагонали может быть нулевым, в такой ситуации необходимо найти в матрице любой другой ненулевой элемент и поменять местами соответствующие строки и столбы. При этом системы линейных уравнений будут равносильными. Если получится так, что ненулевых элементов в матрице не осталось, то необходимо проверить столбец правых частей, если в ней остался хотя бы один ненулевой элемент, то система несовместна, иначе система имеет бесконечно много решений.

solveGauss := proc(A::Matrix, b::Vector)

local varSubst, i, j, idx, firstIdx, secondIdx, q, k, temp;

varSubst := Vector(RowDimension(A), i -> i);

for i to RowDimension(A) - 1 do

if A[i, i] = 0 then

idx := findNotZeroCoefficient(A, i);

firstIdx := idx[1];

secondIdx := idx[2];

if firstIdx < 0 then

for j from i to Dimension(b) do

if b[j] <> 0 then

error "No Solutions";

end if;

end do;

error "Infinite Number of Solutions";

end if;

if firstIdx <> i then

for k to ColumnDimension(A) do

temp := A[i, k];

A[i, k] := A[firstIdx, k];

A[firstIdx, k] := temp;

end do;

temp := b[i];

b[i] := b[j];

b[j] := temp;

end if;

if secondIdx <> i then

for k to RowDimension(A) do

temp := A[k, i];

A[k, i] := A[k, secondIdx];

A[k, secondIdx] := temp;

end do;

temp := varSubst[i];

varSubst[i] := varSubst[j];

varSubst[j] := temp;

end if;

end if;

for j from i + 1 to RowDimension(A) do

q := A[j, i]/A[i, i];

for k from i to ColumnDimension(A) do

A[j, k] := A[j, k] - q\*A[i, k];

end do;

b[j] := b[j] - q\*b[i];

end do;

end do;

return backSubstitution(A, b, varSubst);

end proc:

Реализуем решение СЛАУ с выбором главного элемента по столбцу.

solveChooseInColumn := proc(A::Matrix, b::Vector)

local varSubst, i, j, idx, firstIdx, secondIdx, q, k, maxElement, maxIndex,temp;

varSubst := Vector(RowDimension(A), i -> i);

for i to RowDimension(A) - 1 do

maxElement := abs(A[i,i]);

maxIndex := i;

for j from i + 1 to RowDimension(A) do

if (abs(A[j,i]) > maxElement) then

maxElement := abs(A[j,i]);

maxIndex := j;

end if;

end do;

if maxIndex <> i then

for k to ColumnDimension(A) do

temp := A[i, k];

A[i, k] := A[maxIndex, k];

A[maxIndex, k] := temp;

end do;

temp := b[i];

b[i] := b[maxIndex];

b[maxIndex] := temp;

end if;

if A[i, i] = 0 then

idx := findNotZeroCoefficient(A, i);

firstIdx := idx[1]; secondIdx := idx[2];

if firstIdx < 0 then

for j from i to Dimension(b) do

if b[j] <> 0 then

error "No Solutions";

end if;

end do;

error "Infinite Number of Solutions";

end if;

if firstIdx <> i then

for k to ColumnDimension(A) do

temp := A[i, k];

A[i, k] := A[firstIdx, k];

A[firstIdx, k] := temp;

end do;

temp := b[i];

b[i] := b[firstIdx];

b[firstIdx] := temp;

end if;

if secondIdx <> i then

for k to RowDimension(A) do

temp := A[k, i];

A[k, i] := A[k, secondIdx];

A[k, secondIdx] := temp;

end do;

temp := varSubst[i];

varSubst[i] := varSubst[secondIdx];

varSubst[secondIdx] := temp;

end if;

end if;

for j from i + 1 to RowDimension(A) do

q := A[j, i]/A[i, i];

for k from i to ColumnDimension(A) do

A[j, k] := A[j, k] - q\*A[i, k];

end do;

b[j] := b[j] - q\*b[i];

end do;

end do;

return backSubstitution(A, b, varSubst);

end proc:

И наконец реализуем метод Гаусса с поиском главного элемента по всей матрице. Особенность данного метода в том, что для него нет необходимости дополнительно искать ненулевой элемент, если на главной диагонали находиться ноль. Если так случилось, что после выбора главного элемента он оказался нулевым, то система либо не имеет решения, либо имеет бесконечно много решений.

solveChooseInMatrix := proc(A::Matrix, b::Vector)

local varSubst, i, j, q, k, maxElement, maxIndexFirst, maxIndexSecond, temp;

varSubst := Vector(RowDimension(A), i -> i);

for i to RowDimension(A) - 1 do

maxElement := abs(A[i,i]);

maxIndexFirst := i;

maxIndexSecond := i;

for j from i to RowDimension(A) do

for k from i to ColumnDimension(A) do

if (abs(A[j,k]) > maxElement) then

maxElement := abs(A[j,k]);

maxIndexFirst := j;

maxIndexSecond := k;

end if;

end do;

end do;

if maxIndexFirst <> i then

for k to ColumnDimension(A) do

temp := A[i, k];

A[i, k] := A[maxIndexFirst, k];

A[maxIndexFirst, k] := temp;

end do;

temp := b[i];

b[i] := b[maxIndexFirst];

b[maxIndexFirst] := temp;

end if;

if maxIndexSecond <> i then

for k to RowDimension(A) do

temp := A[k, i];

A[k, i] := A[k, maxIndexSecond];

A[k, maxIndexSecond] := temp;

end do;

temp := varSubst[i];

varSubst[i] := varSubst[maxIndexSecond];

varSubst[maxIndexSecond] := temp;

end if;

if A[i, i] = 0 then

for j from i to Dimension(b) do

if b[j] <> 0 then

error "No Solutions";

end if;

end do;

error "Infinite Number of Solutions";

end if;

for j from i + 1 to RowDimension(A) do

q := A[j, i]/A[i, i];

for k from i to ColumnDimension(A) do

A[j, k] := A[j, k] - q\*A[i, k];

end do;

b[j] := b[j] - q\*b[i];

end do;

end do;

return backSubstitution(A, b, varSubst);

end proc:

Теперь осуществим поиск решения СЛАУ с помощью разработанных алгоритмов и запишем их в переменные x1, x2, x3 обычным методом Гаусса, с выбором по столбцу и с выбором по всей матрице, соответственно.

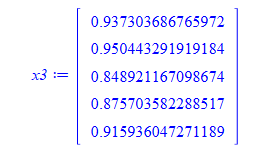
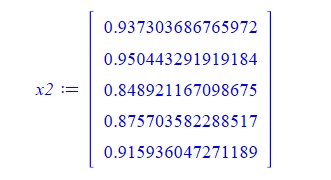
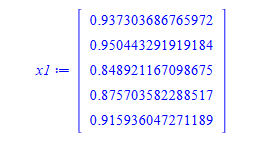


Рисунок 2 – Решения СЛАУ, найденные тремя методами

# **4 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТ**

В системе Maple присутствует функция LinearSolve, которая решает системы линейных уравнений по заданной матрице коэффициентов и вектору правых частей. Сравним полученные решения в трех методах с встроенной функцией Maple и посчитаем модуль разности решений.

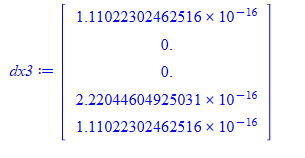
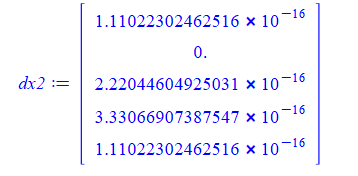
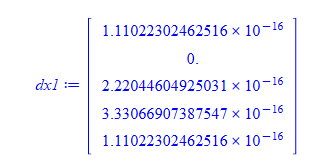


Рисунок 3 – Сравнение решений, найденных с помощью реализованных алгоритмов, и функцией LinearSolve

Как можно заметить алгоритм с выбором главного элемента по всей матрице находит решения, которые ближе к встроенной функции, чем два других алгоритма, что соответствует сути метода с выбором главного элемента.

Теперь для каждого из найденных решений посчитаем векторы невязок и их нормы.

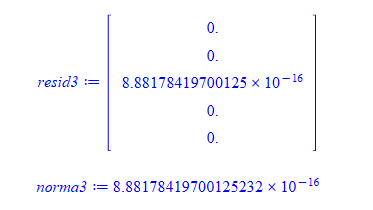
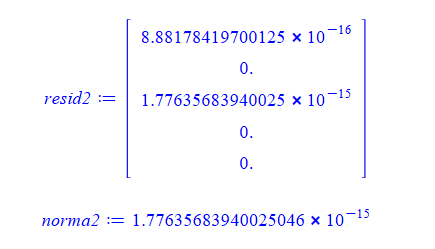
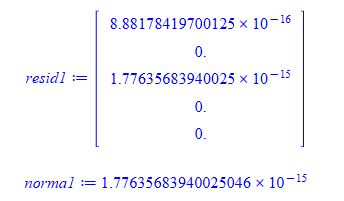


Рисунок 4 – Векторы невязок и их нормы

Снова легко заметить, что третий метод дает более точный результат, чем два остальных.

Теперь сравним работу аналогичного алгоритма, написанного на языке C++. При этом в каждой операции будем учисывать только первые четыре знака после запятой.

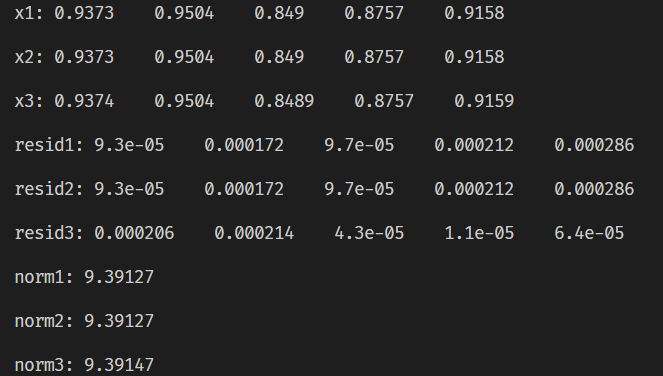


Рисунок 5 – Результат работы алгоритмов на языке C++

Исходя из полученного результата, можно сказать, что алгоритм, реализованный в Maple находит более точные значения, что связанно с тем, что в случае реализации на C++ учитывались только первые четыре знака после запятой. Также свою роль сыграло неточность представления чисел с плавающей точкой в double.

**ВЫВОД**

Таким образом, были спроектированы и реализованы три алгоритма решения систем алгебраических линейных уравнений, а именно методом Гаусса, методом с выбором главного элемента по столбцу, методом выбора главного элемента по всей матрице. Алгоритмы были реализованы в системе Maple и на языке программирования C++.

Были проанализированы результаты работы алгоритмов по поиску решения. Каждое найденное решение сравнивалось с решением системы, найденным с помощью функции LinearSolve пакета LinearAlgebra для Maple. Также для найденных с помощью трех методов решений были посчитаны и проанализированы векторы невязок и их нормы. Сравнивался результат работы программы написанной на языке C++ и в системе Maple.

В результате проведенной работы, выяснилось, что все три метода способны достаточно точно находить решения заданной системы линейных уравнений. При этом наиболее точные решения были найдены с использованием метода поиска главного элемента по всей матрице.

Метод Гаусса легко применяется к любой системе линейных уравнений и демонстрирует свою эффективность, благодаря своей простоте и универсальности.