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

Решение систем линейных алгебраических уравнений (СЛАУ)

методом Гаусса и с помощью его модификаций

Выполнил: студент группы 153501

Миролюбов Илья Игоревич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

**Оглавление**

[**Цели выполнения задания**3](#_Toc65424824)

[**Краткие теоритические сведения**4](#_Toc65424825)

[**Задание**9](#_Toc65424826)

[**Программная реализацияОшибка! Закладка не определена.**](#_Toc65424827)

[**Полученные результаты**14](#_Toc65424828)

[**Оценка**19](#_Toc65424829)

[**Выводы**20](#_Toc65424830)

Вариант 8

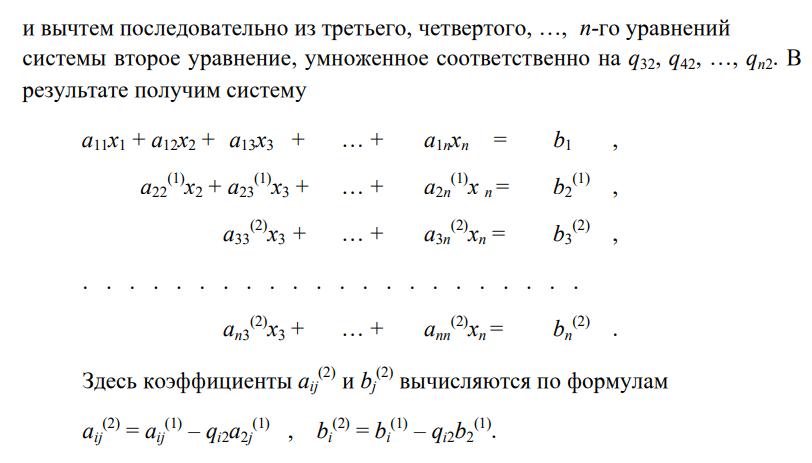
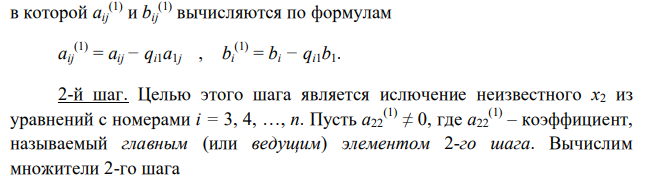
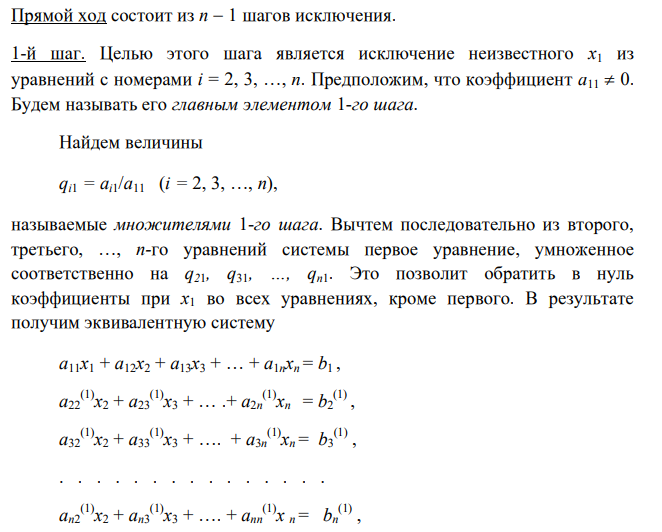
# **Цели выполнения задания**

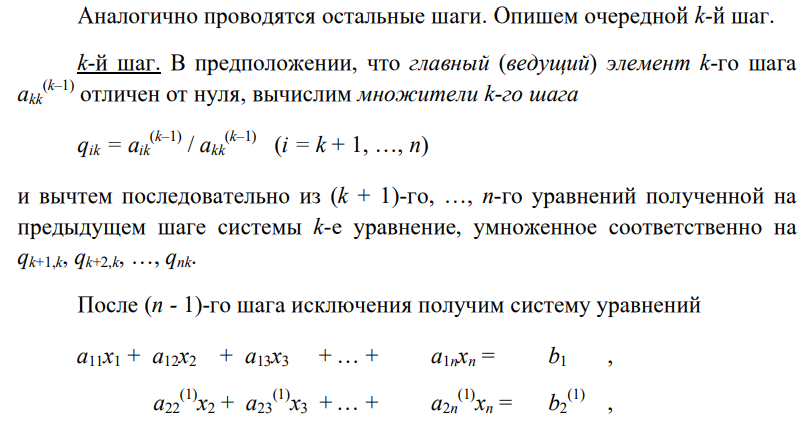
* изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;
* составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
* составить программу решения СЛАУ по разработанному алгоритму;
* выполнить тестовые примеры и проверить правильность работы программы.

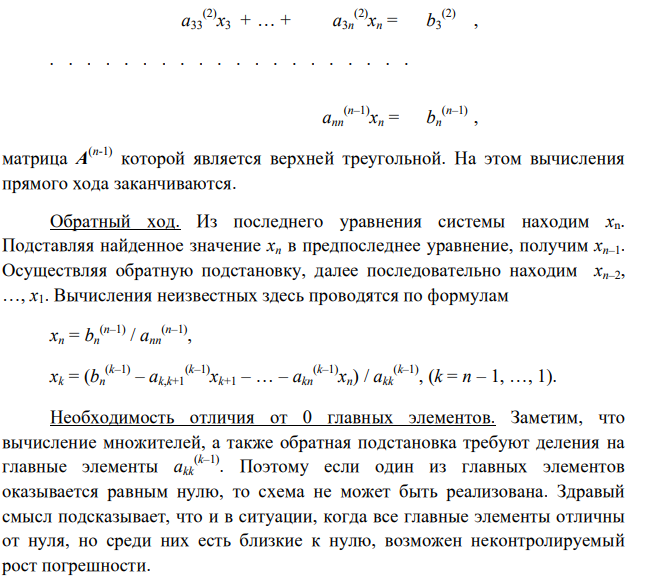
# **Краткие теоретические сведения**

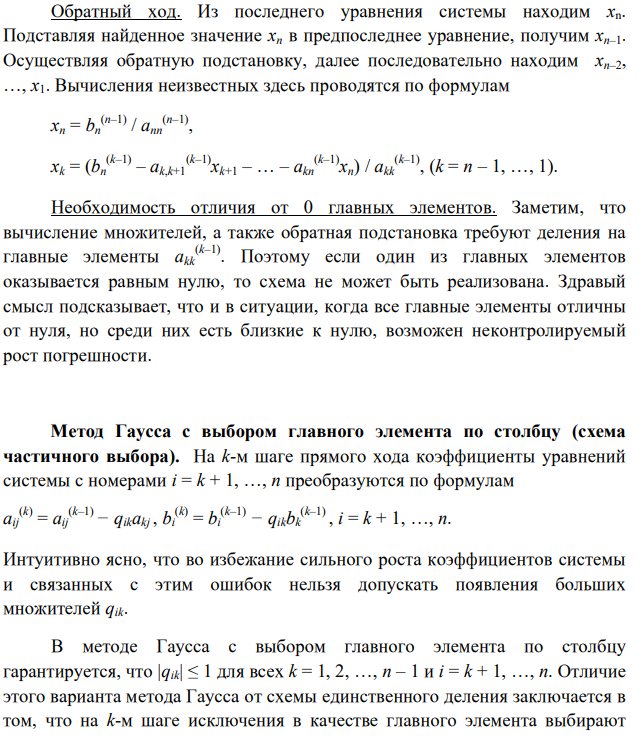
Метод Гаусса прекрасно подходит для решения систем линейных алгебраических уравнений (СЛАУ). Он обладает рядом преимуществ по сравнению с другими методами.

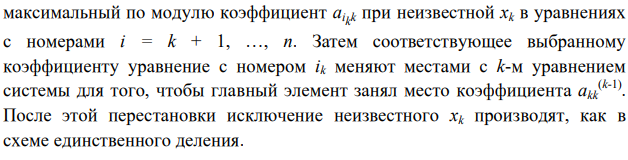
Данный метод включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный - методом Гаусса-Жордана, который отличается от первого только последовательностью исключения переменных.

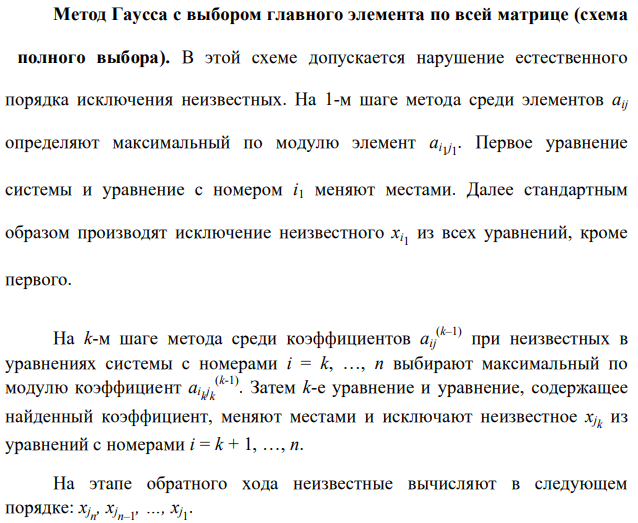




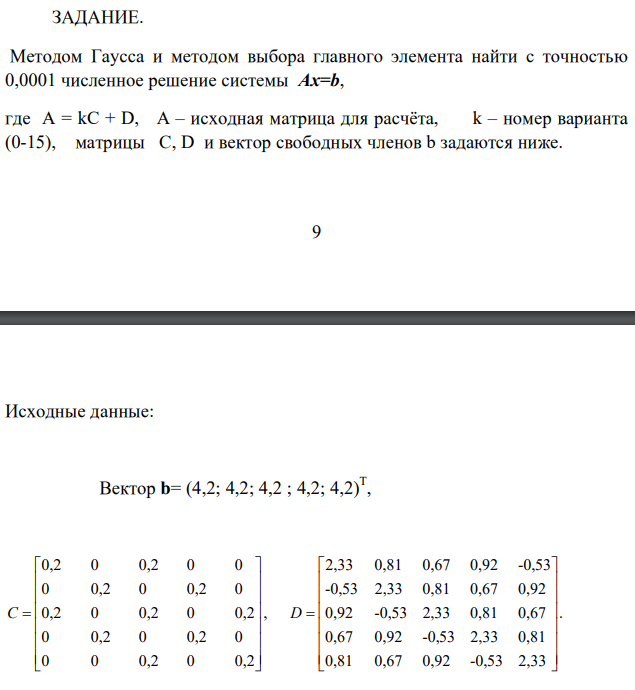




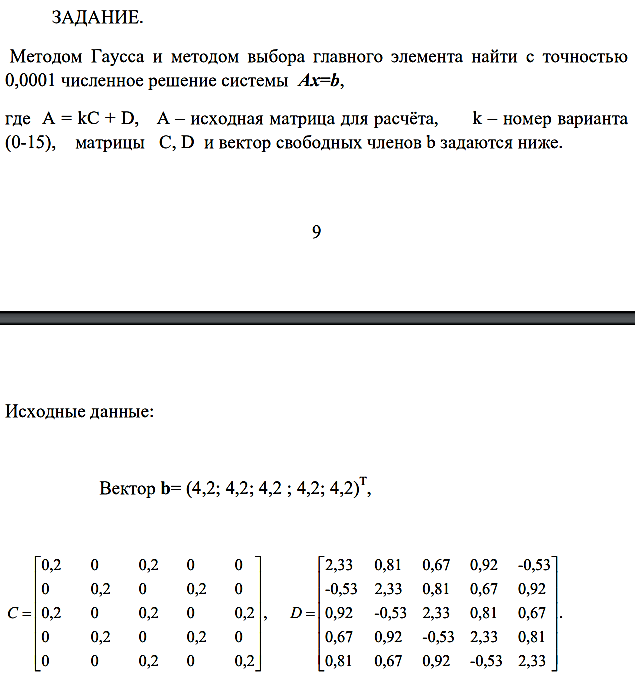




# **Задание**



Вариант 8



Полученные результаты будем сверять с решением, полученным используя подмодуль numpy linalg:

**print(numpy.linalg.solve(A, b))**

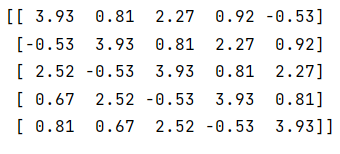
Для исходных данных получим следующий ответ:



**Программная реализация**

**1. Метод Гаусса**

Матрица А, полученная в результате вычисления A=8C+D



**Код программы:**

import numpy

*#Исходные данные*

C = numpy.array([[0.2, 0, 0.2, 0, 0],

                [0, 0.2, 0, 0.2, 0],

                [0.2, 0, 0.2, 0, 0.2],

                [0, 0.2, 0, 0.2, 0],

                [0, 0, 0.2, 0, 0.2]])

D = numpy.array([[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]])

b = numpy.array([4.2, 4.2, 4.2, 4.2, 4.2])

b = b.transpose()

A = 8 \* C + D

X = numpy.zeros(len(A))

*#Прямой обход метод Гаусса***def** StraightRun(matrix, b):  
 **for** numRow, row **in** enumerate(matrix):  
*# где numRow равен номеру строки, а row это сама строка матрицы* divider = row[numRow]  
 row /= divider  
 b[numRow] /= divider  
 bfactor = b[numRow]

*# вычитаем из всех нижележащих строчек* **for** lowerRow **in** range(numRow + 1, len(matrix)):  
 factor = matrix[lowerRow][numRow]  
*# элемент строки в колонке nrow* matrix[lowerRow] -= factor \* row  
*# вычитаем, чтобы получить ноль в колонке nrow* b[lowerRow] -= factor \* bfactor  
 **return** matrix, b

**def** revRun(a, b):  
 n = len(a)  
 **for** k **in** reversed(range(0, n)):  
 X[k] = (b[k] - sum(a[k][i] \* X[i] **for** i **in** range(k + 1, n)))/a[k][k]

*#Перестановка колонок***def** swapClmn(a, i, j):  
 **for** k **in** range(len(a)):  
 a[k][i], a[k][j] = a[k][j], a[k][i]

*#Проверка на 0 на главной диагонали*

**def** zerOnDiag(matrix, b):  
 nstr = len(matrix)  
 **for** i **in** range(0, nstr):  
 **if** matrix[i][i] == 0:  
 check = **True  
 for** j **in** range(0, nstr):  
 **if** matrix[i][j] != 0 **and** matrix[j][i] != 0:  
 swapClmn(matrix, i, j)  
 check = **False  
 break  
 if** check:  
 **if** b[i] == 0:  
 print(**'The system has infinitive amount of solutions'**)  
 **else**:  
 print(**'The system has no solutions'**)  
 exit()

Результат работы вы можете увидеть в тестовом примере 1. Заметим так же, что недостатком данного метода является необходимость отличия от 0 главных элементов. Решением данном проблемы занимается функция zerOnDiag, которая проверяет каждый главный элемент и в случае, если он равен 0 заменяет колонку с этим элементом на любую другую, подходящую для решения. Результат работы вы можете увидеть в тестовом примере 2. Заметим, что если матрица окажется несовместной (все элементы строки будут равны 0, а элемент вектор-столбца b отличен от 0) решений не будет (в тестовом примере 3), а если все элементы строки, включая элемент вектор столбца равен 0, матрица имеет бесконечное количество решений. (в тестовом примере 4)

**2. Метод Гаусса с выбором главного элемента по столбцу**

*# Максимальный элемент по столбцу***def** straightRunClmn(matrix, b):  
 **for** nrow **in** range(len(matrix)): *# nrow равен номеру строки* pivot = nrow + numpy.argmax(abs(matrix[nrow:, nrow]))  
 **if** pivot != nrow:  
 matrix[[nrow, pivot]] = matrix[[pivot, nrow]] *# swap* change = b[pivot]  
 b[pivot] = b[nrow]  
 b[nrow] = change  
 row = matrix[nrow]  
 divider = row[nrow] *# диагональный элемент* **if** abs(divider) < 1e-10:  
 *# почти нуль на диагонали.  
 # Продолжать не имеет смысла, результат счёта неустойчив* print(**f"Matrix is incompatible. Max element in column {**nrow**}: {**divider**:.3g}"**)  
 exit()  
 row /= divider *# делим на диагональный элемент.* b[nrow] /= divider  
 bfactor = b[nrow]  
 *# теперь надо вычесть приведённую строку из всех нижележащих строчек* **for** lowerRow **in** range(nrow + 1, len(matrix)):  
 factor = matrix[lowerRow][nrow]  
 *# элемент строки в колонке nrow* matrix[lowerRow] -= factor \* row  
 *# вычитаем, чтобы получить ноль в колонке nrow* b[lowerRow] -= factor \* bfactor

*#Метод Гаусса поиск максимального по столбцу*

**def** GaussMax(A, b):  
 zerOnDiag(A, b)  
 straightRunMax(A, b)  
 revRun(A, b)  
 **for** i **in** range(len(X)):  
 print(**"%.4f"** % X[i], end=**" "**)  
 print(**'\n'**)

**3. Метод Гаусса с выбором главного элемента по всей матрице**

*#Поиск макс элемента по всей матрице***def** straightRunMax(matrix, b):  
 **for** nrow **in** range(len(matrix)): *# nrow равен номеру строки* mrow, mcol = MaxElement(matrix, nrow)  
 **if** mrow != nrow:  
 matrix[[nrow, mrow]] = matrix[[mrow, nrow]] *# swap* change = b[mrow]  
 b[mrow] = b[nrow]  
 b[nrow] = change  
 swapClmn(matrix, nrow, mcol)  
 row = matrix[nrow]  
 divider = row[nrow] *# диагональный элемент* **if** abs(divider) < 1e-10:  
 *# почти нуль на диагонали. Продолжать не имеет смысла, результат счёта неустойчив* print(**f"Matrix is incompatible. Max element in matrix: {**divider**:.3g}"**)  
 exit()  
 row /= divider *# делим на диагональный элемент.* b[nrow] /= divider  
 bfactor = b[nrow]  
 *# теперь надо вычесть приведённую строку из всех нижележащих строчек* **for** lower\_row **in** range(nrow + 1, len(matrix)):  
 factor = matrix[lower\_row][nrow]  
*# элемент строки в колонке nrow* matrix[lower\_row] -= factor \* row  
*# вычитаем, чтобы получить ноль в колонке nrow* b[lower\_row] -= factor \* bfactor

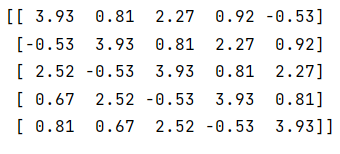
**def** MaxElement(A, k):  
 maximum = A[k-1][k-1]  
 maxIndex = [k-1, k-1]  
 **for** i **in** range(k-1, len(A)):  
 **for** j **in** range(k-1, len(A)):  
 **if** maximum < A[i][j]:  
 maximum = A[i][j]  
 maxIndex = [i, j]  
 **return** maxIndex

**def** GaussMax(A, b):  
 zerOnDiag(A, b)  
 straightRunMax(A, b)  
 revRun(A, b)  
 **for** i **in** range(len(X)):  
 print(**"%.4f"** % X[i], end=**" "**)  
 print(**'\n'**)

**Полученные результаты**

**Тестовый пример 1.**

Матрица А, полученная в результате вычисления A=8C+D

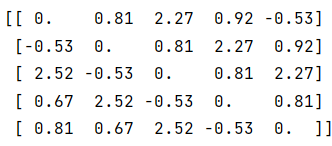


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод Гаусса | | Метод Гаусса с выбором главного элемента по столбцу | Метод Гаусса с выбором главного элемента по всей матрице | Вычисление при помощи встроенной функции |
| 0.8036  0.7979  0.2069  0.3087  0.6760 | | 0.8036  0.7979  0.2069  0.3087  0.6760 | 0.8036  0.7979  0.2069  0.3087  0.6760 | 0.8036  0.7979  0.2069  0.3086  0.6759 |
|  |

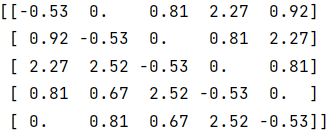
**Тестовый пример 2.**

Проверка наличия 0 на главной диагонали.

Исходная матрица:



Матрица после функции zerOnDiag:

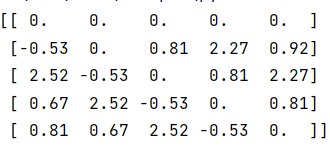


Как видно из примера, нулей на главной диагонали нет и получен ответ:

|  |  |
| --- | --- |
| Метод Гаусса | Вычисление при помощи встроенной функции |
| 1.4912  0.3445  1.3828  1.3648  0.8393 | 1.4911  0.3444  1.3828  1.3648  0.8392 |

**Тестовый пример 3.**

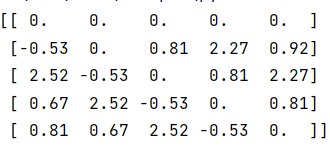
Исходная матрица и вектор столбец:



Получен ответ: The system has no solutions

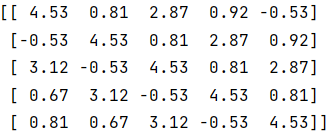
**Тестовый пример 4.**

Исходная матрица и вектор столбец:



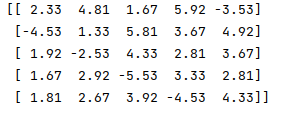
Получен ответ: The system has infinitive amount of solutions

**Тестовый пример 5.**

****

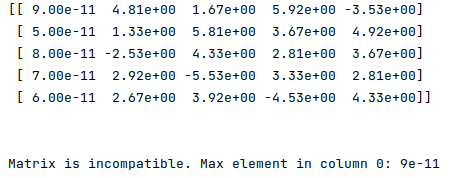
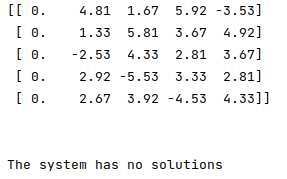
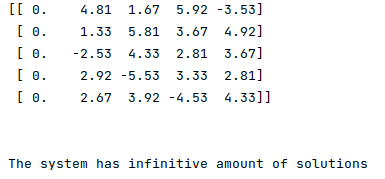
|  |  |
| --- | --- |
| Метод Гаусса | Вычисление при помощи встроенной функции |
| 9.4372  9.1215  -0.0698  1.3721  7.5870 | 9.4372  9.1214  -0.0697  1.3721  7.5870 |
|  |  |

**Тестовый пример 6.**

****

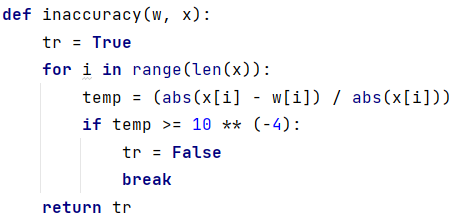
|  |  |
| --- | --- |
| Метод Гаусса | Вычисление при помощи встроенной функции |
| 0.0006  0.0006  0.0003  0.0005  0.0008 | 0.0006  0.0006  0.0003  0.0005  0.0008 |
|  |

**Тестовый пример 7.**

**Тестовый пример 8.**

# **Оценка**

Следующая функция производит проверку на допустимость погрешности, однако для полной уверенности посчитаем относительную погрешность вручную.



0,2069-0.20694999=0,00004999

0,00004493

# **Выводы**

Таким образом, в ходе выполнения лабораторной работы был применён метод Гаусса, метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора) и метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора) для решения системы линейных уравнений, рассмотрены решения СЛАУ методом Гаусса на конкретном примере,составлены алгоритмы и созданы реализации соответствующих программ на языке Python для решения поставленной задачи, также проведена оценка и проверена правильность работы программы.

Итак, метод Гаусса применим к любой системе линейных уравнений,

он идеально подходит для решения систем, содержащих больше трех

линейных уравнений. Метод Гаусса решения СЛАУ с числовыми

коэффициентами в силу простоты и однотипности выполняемых операций

пригоден для счета на электронно-вычислительных машинах.

Достоинства метода:

1. менее трудоёмкий по сравнению с другими методами;

2. позволяет однозначно установить, совместна система или нет, и если

совместна, найти её решение;

3. позволяет найти максимальное число линейно независимых

уравнений – ранг матрицы системы.

Существенным недостатком этого метода является невозможность

сформулировать условия совместности и определенности системы в

зависимости от значений коэффициентов и свободных членов. С другой

стороны, даже в случае определенной системы этот метод не позволяет найти общие формулы, выражающие решение системы через ее коэффициенты и свободные члены, которые необходимо иметь при теоретических исследованиях.