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 253504

Фроленко Кирилл Юрьевич

Проверил: Анисимов Владимир Яковлевич

Минск 2023

Оглавление

[**Цели выполнения задания** 3](#_30j0zll)

[**Краткие теоретические сведения** 4](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.1fob9te)

[**Задание** 9](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.3znysh7)

[**Программная реализация** 1](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.2et92p0)0

[**Полученные результаты** 1](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.tyjcwt)9

[**Оценка** 20](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.3dy6vkm)

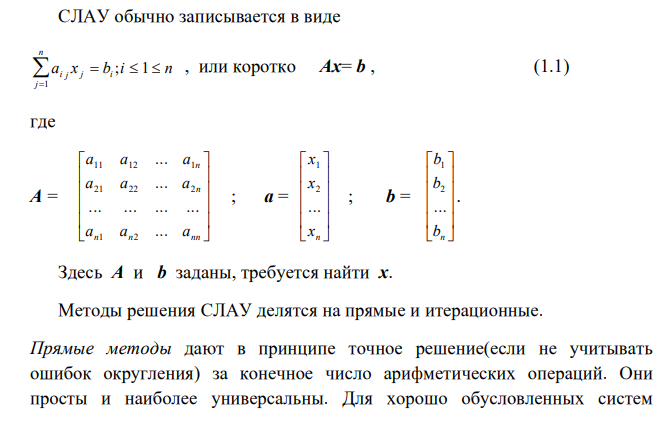
**Тесты** 20

[**Выводы** 21](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.1t3h5sf)

Вариант 28

# **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.

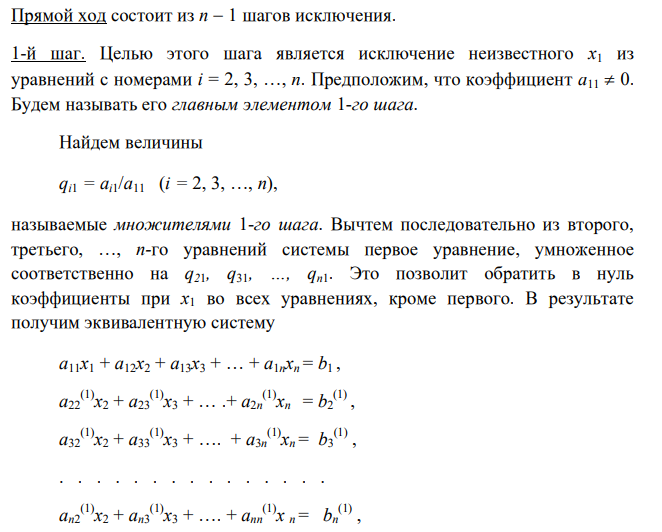
**Краткие теоретические сведения**

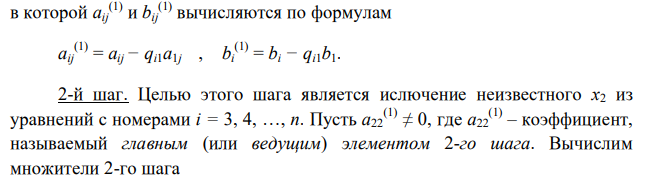
Метод Гаусса прекрасно подходит для решения систем линейных алгебраических уравнений (СЛАУ). Он обладает рядом преимуществ по сравнению с другими методами:

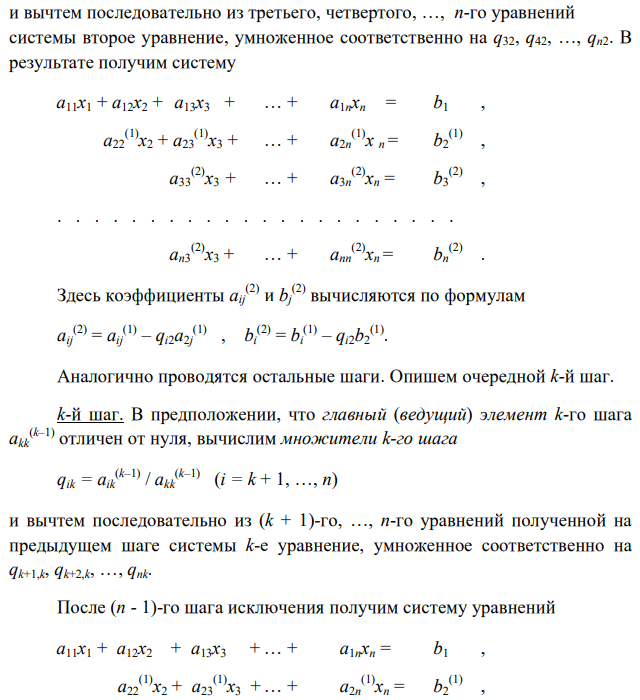
* во-первых, нет необходимости предварительно исследовать систему уравнений на совместность;
* во-вторых, методом Гаусса можно решать не только СЛАУ, в которых число уравнений совпадает с количеством неизвестных переменных и основная матрица системы невырожденная, но и системы уравнений, в которых число уравнений не совпадает с количеством неизвестных переменных или определитель основной матрицы равен нулю;
* в-третьих, метод Гаусса приводит к результату при сравнительно небольшом количестве вычислительных операций.

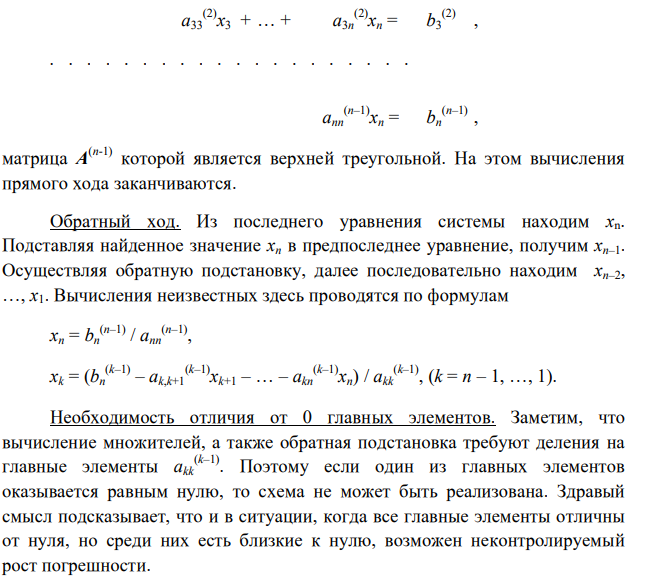
Метод Гаусса включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный - методом Гаусса-Жордана, который отличается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

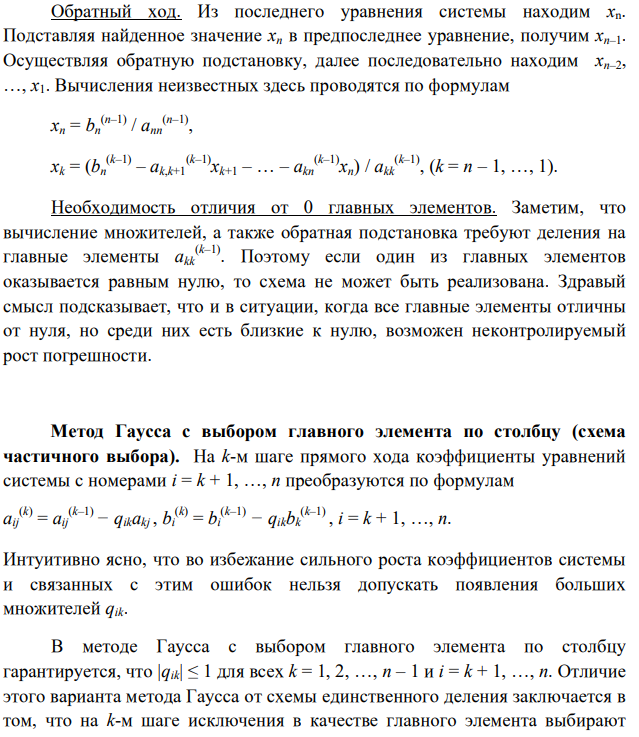
Метод Гаусса идеально подходит для решения систем содержащих больше трех линейных уравнений, для решения систем уравнений, которые не являются квадратными (чего не скажешь про метод Крамера и матричный метод). То есть метод Гаусса - наиболее универсальный метод для нахождения решения любой системы линейных уравнений, он работает в случае, когда система имеет бесконечно много решений или*.*

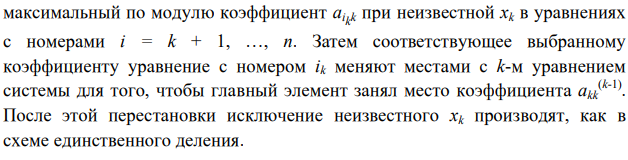


 qi2 = ai2(1) / a22(1) (i = 3, 4, …, n)







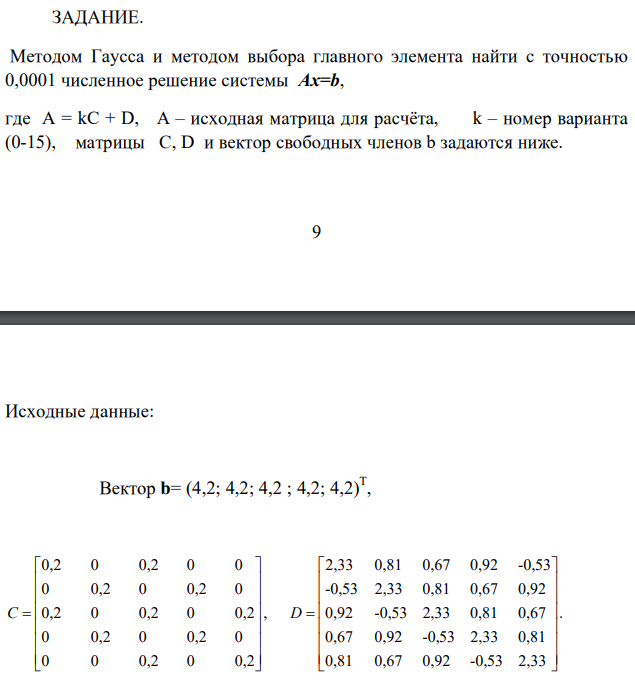


**Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора).** В этой схеме допускается нарушение естественного порядка исключения неизвестных. На 1-м шаге метода среди элементов аij определяют максимальный по модулю элемент аij. Первое уравнение системы и уравнение с номером іi меняют местами. Далее стандартным образом производят исключение неизвестного хi из всех уравнений, кроме первого.

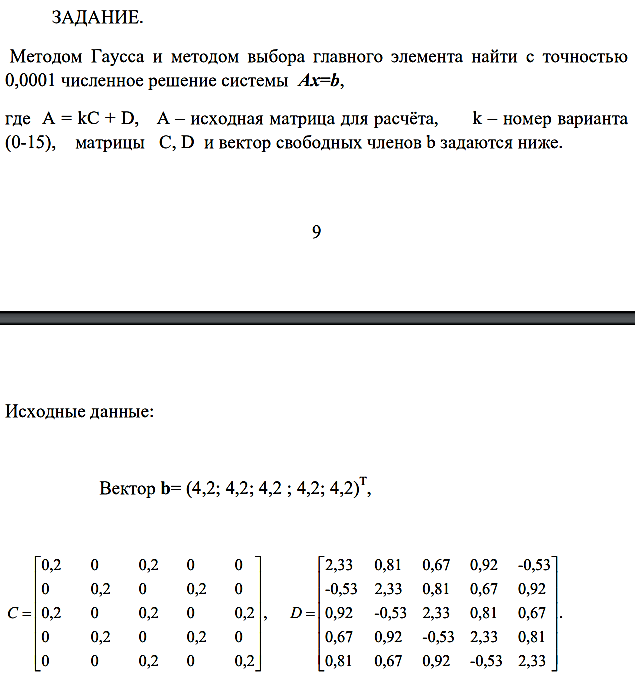
На *k*-м шаге метода среди коэффициентов aij(k-1) при неизвестных в уравнениях системы с номерами i = k, ..., n и выбирают максимальный по модулю коэффициент aij(k-1) . Затем *k-*е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное xjk из уравнений с номерами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

На этапе обратного хода неизвестные вычисляют в следующем порядке: хjn **,** хjn-1 **, …,** хj1 **.**

# **Задание**



Так же надо учесть что в четном варианте данные – приближенные значения.

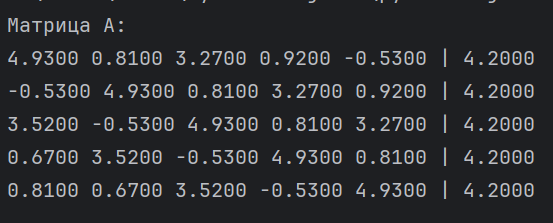


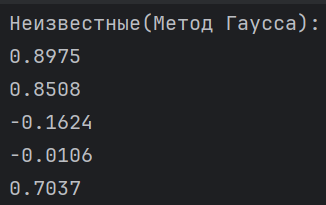
# **Программная реализация**

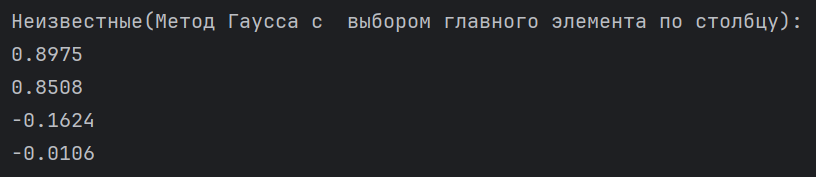
import numpy as np  
  
matrixC = np.array([[0.2, 0, 0.2, 0, 0],  
 [0, 0.2, 0, 0.2, 0],  
 [0.2, 0, 0.2, 0, 0.2],  
 [0, 0.2, 0, 0.2, 0],  
 [0.2, 0, 0.2, 0.0, 0.2]])  
  
matrixD = np.array([[2.33, 0.81, 0.67, 0.92, -0.53],  
 [-0.53, 2.33, 0.81, 0.67, 0.92],  
 [0.92, -0.53, 2.33, 0.81, 0.67],  
 [0.67, 0.92, -0.53, 2.33, 0.81],  
 [0.81, 0.67, 0.92, -0.53, 2.33]])  
  
b = np.array([4.1, 4.2, 4.2, 4.2, 4.2])  
b = b.transpose()  
  
matrixA = 13 \* matrixC + matrixD  
  
  
# Проверка на совместность-----------------------------------------------------------------  
def checkCompatibility(matrixA):  
 matrixBmultidimensial = np.array([[4.1], [4.2], [4.2], [4.2], [4.2]])  
 matrixAexpanded = matrixA.copy()  
 matrixAexpanded = np.hstack((matrixAexpanded, matrixBmultidimensial))  
  
 rankA = np.linalg.matrix\_rank(matrixA)  
 rankAexpanded = np.linalg.matrix\_rank(matrixAexpanded)  
 return rankA == rankAexpanded  
  
  
print("Матрица А:")  
for i in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in matrixA[i]) + " | " + "{:0.4f}".format(b[0]))  
  
  
# Метод Гаусса-------------------------------------------------  
def GaussMethod(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Прямой ход-----------------------------------------------  
 for i in range(1, 5):  
 qArr = np.array([0, 0, 0, 0], float)  
 for p in range(i - 1, 5):  
 if Amatrix[p][p] == 0.:  
 print("--------------------------------")  
 print("Метод не удалось выполнить")  
 print("--------------------------------")  
 return  
 for j in range(i, 5):  
 qArr[j - 1] = Amatrix[j][i - 1] / Amatrix[i - 1][i - 1]  
 for j in range(i, 5):  
 l = i - 1  
 Bmatrix[j] -= qArr[j - 1] \* Bmatrix[i - 1]  
 for m in range(i - 1, 5):  
 Amatrix[j][m] -= qArr[j - 1] \* Amatrix[l][m]  
  
 print("")  
 print("Матрица после преобразований")  
 for i in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[i]) + " " + "| " + "{:0.4f}".format(Bmatrix[i]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
  
 # Обратный ход-----------------------------------------  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные------------------------------------------  
 print("")  
 print("Неизвестные(Метод Гаусса):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
  
GaussMethod(matrixA, b)  
  
# Метод Гаусса с выбором главного элемента по столбцу----------  
  
def GausMethod2(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Поиск главного по столбцу элемента  
 for i in range(5):  
 maxIndex = i  
 max = Amatrix[i][i]  
 for j in range(i + 1, 5):  
 if (abs(max) < abs(Amatrix[j][i])):  
 maxIndex = j  
 max = Amatrix[j][i]  
  
 # Перестановка строк местами  
 if (i != maxIndex):  
 temp = Bmatrix[i]  
 Bmatrix[i] = Bmatrix[maxIndex]  
 Bmatrix[maxIndex] = temp  
 for s in range(i, 5):  
 tempp = Amatrix[i][s]  
 Amatrix[i][s] = Amatrix[maxIndex][s]  
 Amatrix[maxIndex][s] = tempp  
  
 # Треуг. вид  
 for v in range(i + 1, 5):  
 tmp = Amatrix[v][i] / Amatrix[i][i]  
 Bmatrix[v] -= tmp \* Bmatrix[i]  
 Amatrix[v][i] = 0  
 for l in range(i + 1, 5):  
 Amatrix[v][l] -= tmp \* Amatrix[i][l]  
  
 print("")  
 print("Матрица после преобразований")  
 for c in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[c]) + " " + "| " + "{:0.4f}".format(Bmatrix[c]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
 # Поиск решения  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные  
 print("")  
 print("Неизвестные(Метод Гаусса с выбором главного элемента по столбцу):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
  
GausMethod2(matrixA, b)  
  
# Метод Гаусса с выбором главного элемента по всей матрице-----------  
def GausMethod3(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Поиск главн. элемента по всей матрице  
 for i in range(5):  
  
 maxIndex = i  
 max = Amatrix[i][i]  
 for j in range(i, 5):  
 for k in range(i, 5):  
 if (abs(max) < abs(Amatrix[j][k])):  
 maxIndex = j  
 max = Amatrix[j][k]  
  
 # Перестановка строк местами  
 if (i != maxIndex):  
 temp = Bmatrix[i]  
 Bmatrix[i] = Bmatrix[maxIndex]  
 Bmatrix[maxIndex] = temp  
 for j in range(i, 5):  
 tmp = Amatrix[i][j]  
 Amatrix[i][j] = Amatrix[maxIndex][j]  
 Amatrix[maxIndex][j] = tmp  
 # Привидение к треуг. виду  
 for j in range(i + 1, 5):  
 tmp = Amatrix[j][i] / Amatrix[i][i]  
 Bmatrix[j] -= tmp \* Bmatrix[i]  
 Amatrix[j][i] = 0  
 for l in range(i + 1, 5):  
 Amatrix[j][l] -= tmp \* Amatrix[i][l]  
  
 print("")  
 print("Матрица после преобразований")  
 for c in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[c]) + " " + "| " + "{:0.4f}".format(Bmatrix[c]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
 # Поиск решения  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные  
 print("")  
 print("Неизвестные(Метод Гаусса с выбором главного элемента по всей матрице):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
GausMethod3(matrixA, b)  
  
matrixC = np.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]])  
  
matrixD = np.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 = np.array([4.2, 4.2, 4.2, 4.2, 4.2])  
b = b.transpose()  
  
matrixA = 13 \* matrixC + matrixD  
  
  
# Проверка на совместность-----------------------------------------------------------------  
def checkCompatibility(matrixA):  
 matrixBmultidimensial = np.array([[4.2], [4.2], [4.2], [4.2], [4.2]])  
 matrixAexpanded = matrixA.copy()  
 matrixAexpanded = np.hstack((matrixAexpanded, matrixBmultidimensial))  
  
 rankA = np.linalg.matrix\_rank(matrixA)  
 rankAexpanded = np.linalg.matrix\_rank(matrixAexpanded)  
 return rankA == rankAexpanded  
  
  
print("Матрица А:")  
for i in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in matrixA[i]) + " | " + "{:0.4f}".format(b[0]))  
  
  
# Метод Гаусса-------------------------------------------------  
def GaussMethod(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Прямой ход-----------------------------------------------  
 for i in range(1, 5):  
 qArr = np.array([0, 0, 0, 0], float)  
 for p in range(i - 1, 5):  
 if Amatrix[p][p] == 0.:  
 print("--------------------------------")  
 print("Метод не удалось выполнить")  
 print("--------------------------------")  
 return  
 for j in range(i, 5):  
 qArr[j - 1] = Amatrix[j][i - 1] / Amatrix[i - 1][i - 1]  
 for j in range(i, 5):  
 l = i - 1  
 Bmatrix[j] -= qArr[j - 1] \* Bmatrix[i - 1]  
 for m in range(i - 1, 5):  
 Amatrix[j][m] -= qArr[j - 1] \* Amatrix[l][m]  
  
 print("")  
 print("Матрица после преобразований")  
 for i in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[i]) + " " + "| " + "{:0.4f}".format(Bmatrix[i]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
  
 # Обратный ход-----------------------------------------  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные------------------------------------------  
 print("")  
 print("Неизвестные(Метод Гаусса):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
  
GaussMethod(matrixA, b)  
  
# Метод Гаусса с выбором главного элемента по столбцу----------  
  
def GausMethod2(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Поиск главного по столбцу элемента  
 for i in range(5):  
 maxIndex = i  
 max = Amatrix[i][i]  
 for j in range(i + 1, 5):  
 if (abs(max) < abs(Amatrix[j][i])):  
 maxIndex = j  
 max = Amatrix[j][i]  
  
 # Перестановка строк местами  
 if (i != maxIndex):  
 temp = Bmatrix[i]  
 Bmatrix[i] = Bmatrix[maxIndex]  
 Bmatrix[maxIndex] = temp  
 for s in range(i, 5):  
 tempp = Amatrix[i][s]  
 Amatrix[i][s] = Amatrix[maxIndex][s]  
 Amatrix[maxIndex][s] = tempp  
  
 # Треуг. вид  
 for v in range(i + 1, 5):  
 tmp = Amatrix[v][i] / Amatrix[i][i]  
 Bmatrix[v] -= tmp \* Bmatrix[i]  
 Amatrix[v][i] = 0  
 for l in range(i + 1, 5):  
 Amatrix[v][l] -= tmp \* Amatrix[i][l]  
  
 print("")  
 print("Матрица после преобразований")  
 for c in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[c]) + " " + "| " + "{:0.4f}".format(Bmatrix[c]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
 # Поиск решения  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные  
 print("")  
 print("Неизвестные(Метод Гаусса с выбором главного элемента по столбцу):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
  
GausMethod2(matrixA, b)  
  
# Метод Гаусса с выбором главного элемента по всей матрице-----------  
def GausMethod3(matrixA, matrixB):  
 Amatrix = matrixA.copy()  
 Bmatrix = matrixB.copy()  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 check = checkCompatibility(Amatrix)  
 if (check):  
 pass  
 else:  
 print("Нет решений")  
 return  
  
 # Поиск главн. элемента по всей матрице  
 for i in range(5):  
  
 maxIndex = i  
 max = Amatrix[i][i]  
 for j in range(i, 5):  
 for k in range(i, 5):  
 if (abs(max) < abs(Amatrix[j][k])):  
 maxIndex = j  
 max = Amatrix[j][k]  
  
 # Перестановка строк местами  
 if (i != maxIndex):  
 temp = Bmatrix[i]  
 Bmatrix[i] = Bmatrix[maxIndex]  
 Bmatrix[maxIndex] = temp  
 for j in range(i, 5):  
 tmp = Amatrix[i][j]  
 Amatrix[i][j] = Amatrix[maxIndex][j]  
 Amatrix[maxIndex][j] = tmp  
 # Привидение к треуг. виду  
 for j in range(i + 1, 5):  
 tmp = Amatrix[j][i] / Amatrix[i][i]  
 Bmatrix[j] -= tmp \* Bmatrix[i]  
 Amatrix[j][i] = 0  
 for l in range(i + 1, 5):  
 Amatrix[j][l] -= tmp \* Amatrix[i][l]  
  
 print("")  
 print("Матрица после преобразований")  
 for c in range(5):  
 print(' '.join("{:0.4f}".format(row) for row in Amatrix[c]) + " " + "| " + "{:0.4f}".format(Bmatrix[c]))  
  
 xArr = np.array([0, 0, 0, 0, 0], float)  
 # Поиск решения  
 for i in range(4, -1, -1):  
 for j in range(i, 5):  
 if (j == 4):  
 break  
 else:  
 Bmatrix[i] -= Amatrix[i][j + 1] \* xArr[j + 1]  
  
 xArr[i] = Bmatrix[i] / Amatrix[i][i]  
  
 # Неизвестные  
 print("")  
 print("Неизвестные(Метод Гаусса с выбором главного элемента по всей матрице):")  
 for i in range(5):  
 print("{:0.4f}".format(xArr[i]))  
  
GausMethod3(matrixA, b)

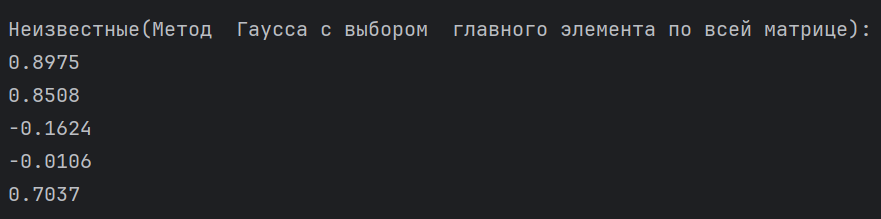
# **Полученные результаты**

Матрица А, полученная в результате вычисления A=6C+D:

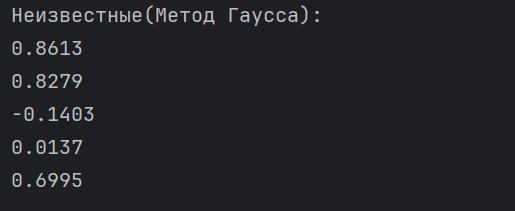


T





Если учесть то, что наши данные приближенные числа к примеру вектор b: [4.1, 4.2, 4.2, 4.2, 4.2]



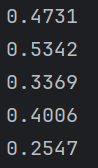
Следовательно максимальная точность для приближенных входных данных будет равна 1 знак после целой части. И для приближенных входных данных точность заданная в условии недостижима.

# **Оценка**

Учитывая то, что наши входные данные являются приближенными числами а именно +-0.1, следовательно невозможно точно посчитать точность. К примеру при изменении всего одного значения на 0.1 мы получим следующую разницы в ответах

# 

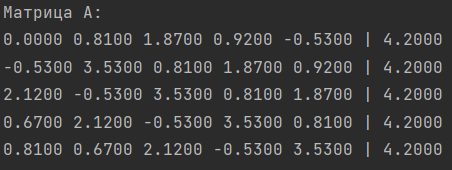
И



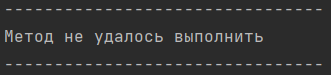
Этим мы доказываем, что точность теряется сразу же после перехода из целой части в дробную.

# **Тесты**

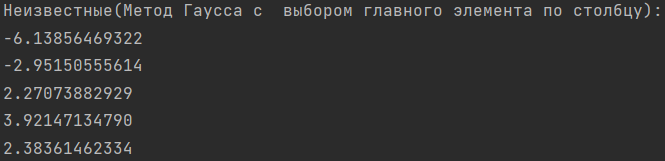
Матрица А(в тестах программа вычисляет значения относительно точно):



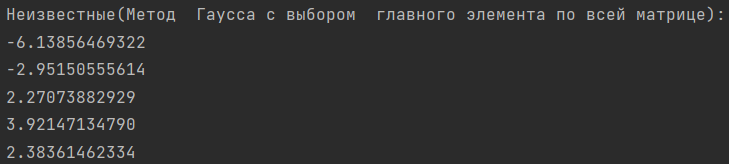
Результат, полученный решением методом Гаусса:



Результат, полученный решением методом Гаусса выбором главного элемента по столбцу:



Результат, полученный решением методом Гаусса выбором главного элемента по всей матрице:



# **Выводы**

Таким образом, в ходе выполнения данной 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, составлены алгоритмы и созданы реализации соответствующих программ на языке Python для решения поставленной задачи, также проведена оценка.

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.