Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №1**

**По вычислительной математике**

**Вариант: 3**

Выполнил:

студент 2 курса

Батманов Даниил Евгеньевич

Группа: Р3207

Принял:

Рыбаков Степан Дмитриевич

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

**Оглавление**

[Задание 3](#_Toc159280598)

[Описание метода 3](#_Toc159280599)

[Блок-схема метода Гаусса 4](#_Toc159280600)

[Исходный код программы 5](#_Toc159280601)

[Результат работы программы 8](#_Toc159280602)

[Вывод 9](#_Toc159280603)

# 

# 

# 

# Задание

Написать программный продукт, реализующий решение СЛАУ методом Гаусса.

Для прямых методов должно быть реализовано:

* Вычисление определителя
* Вывод треугольной матрицы (включая преобразованный столбец В)
* Вывод вектора неизвестных: 𝑥1, 𝑥2, ... , 𝑥𝑛
* Вывод вектора невязок: 𝑟 , 𝑟, ... , 𝑟

Должна быть реализована возможность ввода коэффициентов матрицы,

как с клавиатуры, так и из файла (по выбору конечного пользователя).

# Описание метода

Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы.

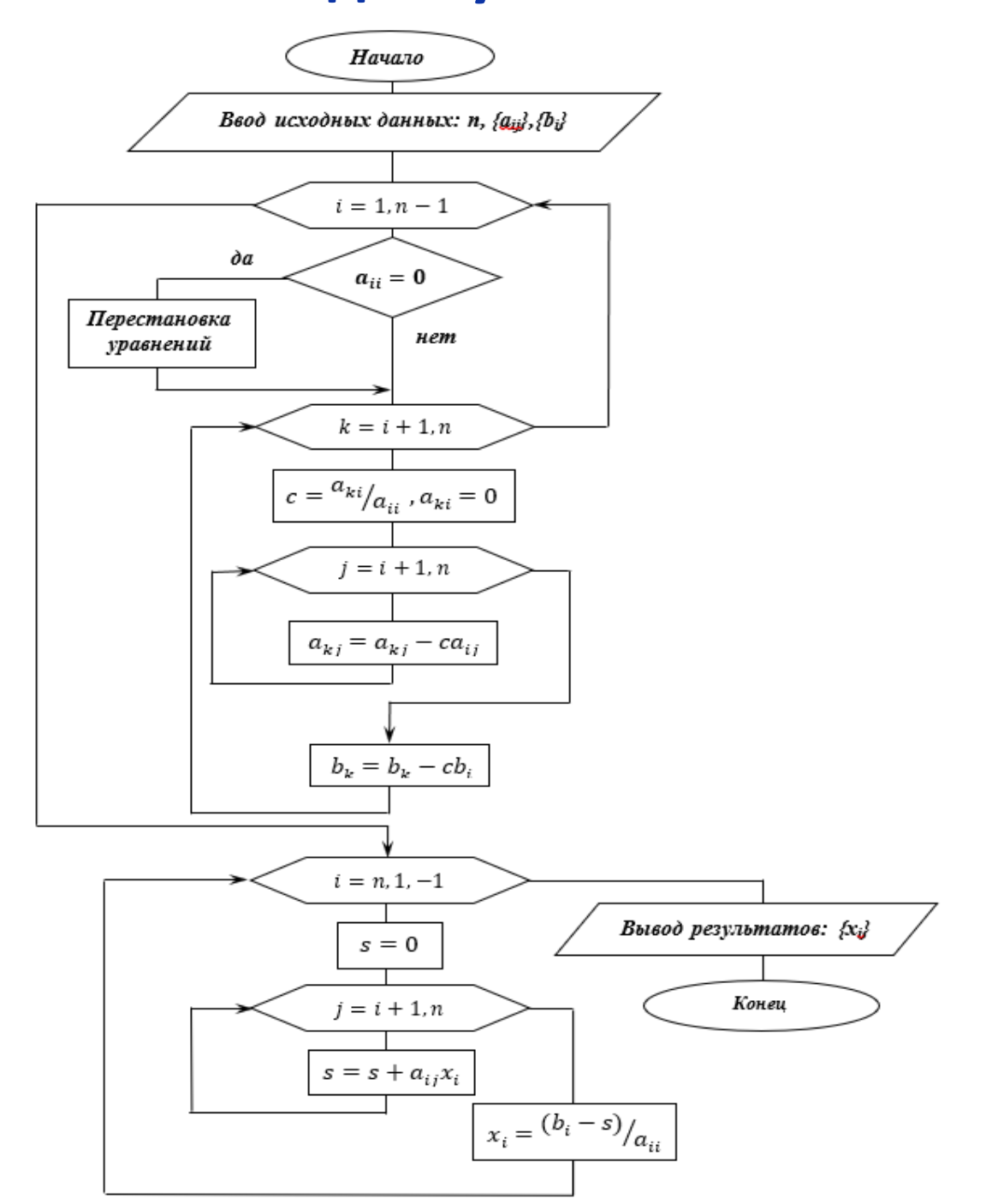
Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается 𝑥1 из всех последующих уравнений системы. Затем с помощью второго уравнения исключается 𝑥2 из третьего и всех последующих уравнений и т.д.

Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным 𝑥𝑛 , т. е. матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим неизвестное 𝑥𝑛 . Далее, из предыдущего уравнения вычисляем 𝑥𝑛−1 и т. д. Последним найдем 𝑥1 из первого уравнения.

Метод имеет много различных вычислительных схем. Основное требование - det A ≠ 0 .

# Блок-схема метода Гаусса



# Исходный код программы

# variant: №3  
  
def get\_data\_from\_keyboard():  
 final\_data = []  
 n = int(input("Введите количество строк/столбцов матрицы (n): "))  
 while n < 1 or n > 20:  
 print("Значение n должно быть в диапазоне [1; 20]\n")  
 n = int(input("Введите количество строк/столбцов матрицы (n): "))  
 else:  
 final\_data.append(n)  
 print("Спасибо, введите значения коэффициентов матрицы через пробел (a\_ij):")  
 raw\_line = input()  
 values = []  
 sp\_line = raw\_line.split(' ')  
 while len(sp\_line) != (n \* n):  
 print("Количество значений должно быть кратно n \* n, "  
 "введите значения коэффициентов матрицы через пробел (a\_ij):")  
 raw\_line = input()  
 values = []  
 sp\_line = raw\_line.split(' ')  
 else:  
 for i in sp\_line:  
 values.append(float(i))  
 table = []  
 line = []  
 for i in range(n \* n):  
 if (i + 1) % n == 0:  
 line.append(values[i])  
 table.append(line)  
 line = []  
 else:  
 line.append(values[i])  
 final\_data.append(table)  
 print("Спасибо, введите решения через пробел (b\_i):")  
 res\_line = input()  
 result = []  
 res\_line = res\_line.split(' ')  
 while len(res\_line) != n:  
 print("Количество решений должно быть кратным n, введите решения через пробел (b\_i):")  
 res\_line = input()  
 print("------------------")  
 res\_line = res\_line.split(' ')  
 else:  
 for i in res\_line:  
 result.append(float(i))  
 final\_data.append(result)  
 return final\_data  
  
  
def get\_data\_from\_file(file\_path):  
 final\_data = []  
 try:  
 with open(file\_path, 'r') as file:  
 n = int(file.readline().strip())  
 final\_data.append(n)  
 raw\_line = file.readline()  
 values = []  
 sp\_line = raw\_line.split(' ')  
 for i in sp\_line:  
 values.append(float(i))  
 table = []  
 line = []  
 for i in range(n \* n):  
 if (i + 1) % n == 0:  
 line.append(values[i])  
 table.append(line)  
 line = []  
 else:  
 line.append(values[i])  
 final\_data.append(table)  
 res\_line = file.readline()  
 result = []  
 res\_line = res\_line.split(' ')  
 for i in res\_line:  
 result.append(float(i))  
 final\_data.append(result)  
 print("Спасибо, n, a\_ij и b\_i считанны корректно")  
 except FileNotFoundError:  
 print("Файл не найден")  
 except ValueError:  
 print("Неверный формат данных в файле")  
 return final\_data  
  
  
def triangular\_matrix(matrix, answers):  
 n = len(matrix)  
 for i in range(n):  
 if matrix[i][i] == 0:  
 print("ERROR: Диагональный элемент равен нулю(((")  
 return None, None  
 for k in range(i+1, n):  
 coef = round(-matrix[k][i] / matrix[i][i], 20)  
 for j in range(i, n):  
 matrix[k][j] = round(matrix[k][j] + coef \* matrix[i][j], 20)  
 answers[k] = round(answers[k] + coef \* answers[i], 20)  
  
 return matrix, answers  
  
  
def determinant(matrix):  
 det = 1  
 for i in range(len(matrix)):  
 det \*= matrix[i][i]  
 return det  
  
def calculate\_results(matrix, answers):  
 n = len(matrix)  
 results = [0] \* n  
 for i in range(n - 1, -1, -1):  
 results[i] = answers[i]  
 for j in range(i + 1, n):  
 results[i] = round(results[i] - matrix[i][j] \* results[j], 20)  
 results[i] = round(results[i] / matrix[i][i], 20)  
 return results  
  
  
def show\_miss(matrix, answers, result):  
 n = len(matrix)  
 miss\_mas = []  
 for i in range(n):  
 miss = 0  
 for j in range(n):  
 miss += matrix[i][j] \* result[j]  
 miss -= answers[i]  
 miss\_mas.append(miss)  
 return miss\_mas  
  
  
  
way\_of\_input = int(input("Решатор СЛАУ методом Гаусса 3000\n"  
 "-------------------\n"  
 "Выберите способ ввода данных:\n"  
 "0 – ввод данных с клавиатуры\n"  
 "1 – ввод из файла\n"  
 "------------------\n"))  
  
if not way\_of\_input:  
 data = get\_data\_from\_keyboard()  
else:  
 data = get\_data\_from\_file(input("Укажите путь к файлу: "))  
  
n = data[0]  
table = data[1]  
table\_copy = table  
answers = data[2]  
answers\_copy = answers  
  
triangular\_table, new\_answers = triangular\_matrix(table, answers)  
det = determinant(triangular\_table)  
print("------------------\nОпределитель матрицы равен:", det)  
if det == 0:  
 print("------------------\nERROR: Метод Гаусса неприменим к матрицам с определителем равным нулю\n"  
 "------------------\n")  
else:  
 print("------------------\nТреугольный вид матрицы:")  
 for i in range(n):  
 print(triangular\_table[i], "|", answers[i])  
 print("------------------")  
 results = calculate\_results(triangular\_table, new\_answers)  
 print("Вычисленные коэффициенты:")  
 for i in range(n):  
 print("x" + str(i + 1), "=", results[i])  
 print("------------------")  
 miss = show\_miss(table\_copy, answers\_copy, results)  
 print("Невязки:")  
 for i in range(n):  
 print("r" + str(i + 1), "=", miss[i])  
 print("------------------")

# Результат работы программы

Ввод:

3  
10 -7 0 -3 2 6 5 -1 5  
7 4 6

Вывод:

Решатор СЛАУ методом Гаусса 3000  
-------------------  
Выберите способ ввода данных:  
0 – ввод данных с клавиатуры  
1 – ввод из файла  
------------------  
1  
Укажите путь к файлу: lex.txt  
Спасибо, n, a\_ij и b\_i считанны корректно  
------------------  
Определитель матрицы равен: -155.0  
------------------  
Треугольный вид матрицы:  
[10.0, -7.0, 0.0] | 7.0  
[0.0, -0.1, 6.0] | 6.1  
[0.0, 0.0, 155.0] | 155.0  
------------------  
Вычисленные коэффициенты:  
x1 = 0.0  
x2 = -1.0  
x3 = 1.0  
------------------  
Невязки:  
r1 = 0.0  
r2 = 0.0  
r3 = 0.0  
------------------

# Вывод

В ходе выполнения данной лабораторной работы я вспомнил метод решения СЛАУ методом Крамера и реализовал его на языке Python. Узнал, что такое невязки, это очень пригодится мне в дальнейшей жизни, но я пока не очень понял где.