Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерной техники

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ 1

РЕШЕНИЕ АЛГЕБРАИЧЕСКОЙ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

ВАРИАНТ 12

­

Студент: Пышкин Никита Сергеевич, P3213

Преподаватель:

Санкт Петербург 2025

Содержание

[**Цель работы** 3](#_Toc190877005)

[**Описание метода** 3](#_Toc190877006)

[**Листинг программы** 3](#_Toc190877007)

[**Примеры и результаты работы программы** 5](#_Toc190877008)

[**Заключение** 6](#_Toc190877009)

# **Цель работы**

Изучить численные методы решения СЛАУ и реализовать один из них средствами языка программирования

# **Описание метода**

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению систем уравнений.

Идея метода: при вычислении компонента 𝑥𝑖(𝑘+1) на (k+1)-й итерации используются 𝑥1(𝑘+1), 𝑥2(𝑘+1), …, 𝑥𝑖−1(𝑘+1), уже вычисленные на (k+1)-й итерации. Значения остальных компонент 𝑥𝑖+1(𝑘+1), 𝑥𝑖+2(𝑘+1), …, 𝑥𝑛(𝑘+1) берутся из предыдущей итерации.

# **Листинг программы**

class AbstractIterMethod(ABC):

    @classmethod

    @abstractmethod

    def solve(cls, accuracy: float, matrix: List[List[float]], max\_iterations: int) -> Any:

        pass

    @classmethod

    def make\_diagonally\_dominant(cls, matrix: List[List[float]]) -> Optional[Tuple[List[int], List[List[float]]]]:

        def swap\_columns(order: List[int], matrix: List[List[float]], i: int, j: int) -> List[List[float]]:

            order[i], order[j] = order[j], order[i]

            for row in matrix:

                row[i], row[j] = row[j], row[i]

        order = [i for i in range(len(matrix))]

        for row\_index, row in enumerate(matrix):

            max\_value, max\_index = float("-inf"), None

            for i, value in enumerate(row):

                if value > max\_value:

                    max\_value = value

                    max\_index = i

            if max\_index is None:

                return

            swap\_columns(order, matrix, row\_index, max\_index)

        if cls.is\_diagonally\_dominant(matrix):

            return order, matrix

    @classmethod

    def transform\_answer\_order(cls, vector: List[float], order: List[int]) -> List[float]:

        ordered\_x = [0] \* len(vector)

        for i, elem in enumerate(vector):

            ordered\_x[order.index(i)] = elem

        return ordered\_x

    @classmethod

    def is\_diagonally\_dominant(cls, matrix: List[List[float]]) -> bool:

        n = len(matrix)

        if n == 1:

            return True

        strict\_equality = False

        for i in range(n):

            row\_sum = sum(abs(matrix[i][j]) for j in range(n) if i != j)

            if abs(matrix[i][i]) < row\_sum:

                return False

            if not strict\_equality and abs(matrix[i][i]) > row\_sum:

                strict\_equality = True

        return strict\_equality

class GaussSeidelMethod(AbstractIterMethod):

    @classmethod

    def solve(cls, accuracy: float, matrix: List[List[float]], max\_iterations: int = 100000) -> Any:

        A, b = [], []

        for row in matrix:

            A.append(row[:-1])

            b.append(row[-1])

        if (res := cls.make\_diagonally\_dominant(A)) is None:

            return "Невозможно привести матрицу в диагональный вид"

        order, A = res

        x = [0] \* len(A)

        for iteration\_number in range(1, max\_iterations + 1):

            x\_old = x.copy()

            for i in range(len(A)):

                sum\_new = sum(A[i][j] \* x[j] for j in range(i))

                sum\_old = sum(A[i][j] \* x\_old[j] for j in range(i + 1, len(A)))

                x[i] = (b[i] - sum\_new - sum\_old) / A[i][i]

            e\_vector = list(abs(x[i] - x\_old[i]) for i in range(len(A)))

            if max(e\_vector) < accuracy:

                norma = max([sum(abs(x) for x in row) for row in A])

                return (

                    cls.transform\_answer\_order(x, order),

                    norma,

                    iteration\_number,

                    e\_vector

                )

        return "Метод не сошелся за заданное количество итераций"

# **Примеры и результаты работы программы**

Матрица:

1 8 1 4

9 1 2 3

3 4 7 2  
Точность: 0.0000001

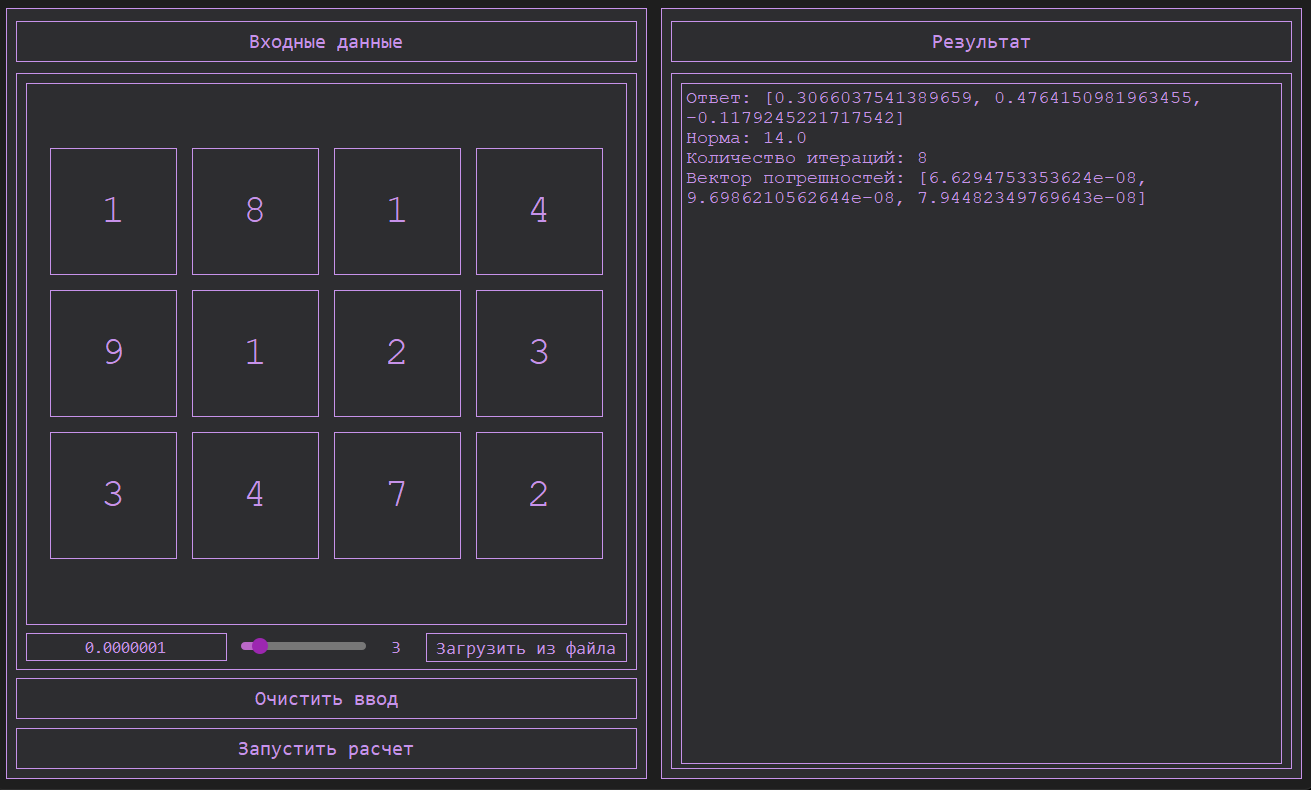
Результат:

Ответ: [0.3066037541389659, 0.4764150981963455, -0.1179245221717542]

Норма: 14.0

Количество итераций: 8

Вектор погрешностей: [6.6294753353624e-08, 9.6986210562644e-08, 7.94482349769643e-08]



# **Заключение**

В ходе выполнения данной лабораторной работы я изучил численные методы решения системы линейных уравнений и реализовал один из итерационных методов с помощью языка программирования Python.