

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, Р3213

Преподаватель:

Санкт Петербург 2025

Содержание

Цель работы	3
Описание метода	3
Листинг программы	3
Примеры и результаты работы программы.....	5
Заключение.....	6

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 $x_i^{(k+1)}$ на $(k+1)$ -й итерации используются $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{i-1}^{(k+1)}$, уже вычисленные на $(k+1)$ -й итерации. Значения остальных компонент $x_{i+1}^{(k+1)}, x_{i+2}^{(k+1)}, \dots, x_n^{(k+1)}$ 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]

Входные данные	Результат												
<table><tbody><tr><td>1</td><td>8</td><td>1</td><td>4</td></tr><tr><td>9</td><td>1</td><td>2</td><td>3</td></tr><tr><td>3</td><td>4</td><td>7</td><td>2</td></tr></tbody></table> <div><div>0.000001</div><div><div></div></div><div>3</div><div>Загрузить из файла</div></div> <div>Очистить ввод</div> <div>Запустить расчет</div>	1	8	1	4	9	1	2	3	3	4	7	2	<div>Ответ: [0.3066037541389659, 0.4764150981963455, -0.1179245221717542]</div> <div>Норма: 14.0</div> <div>Количество итераций: 8</div> <div>Вектор погрешностей: [6.6294753353624e-08, 9.6986210562644e-08, 7.94482349769643e-08]</div>
1	8	1	4										
9	1	2	3										
3	4	7	2										

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.