Университет ИТМО

Факультет Программной Инженерии и Компьютерных Техники

Лабораторная работа №1

Перевод чисел между различными системами счисления

Вариант № 15

Выполнила:
Студент группы P3213
Юсупова Алиса Ильясовна
Преподаватель:
Машина Екатерина Алексеевна
Преподаватель практики

Санкт-Петербург

2025

Цель:

Изучение численных методов решения СЛАУ(системы линейных алгебраических уравнений), разработка программы, которая будет использовать метод простых итераций для решения таких уравнений.

Описание метода:

Метод простых итераций решает СЛАУ путём последовательных приближений к решению. Заключается в преобразовании системы уравнений в итерационную форму $x = Cx + D$, где $x -$ вектор неизвестных, $C -$ матрица коэффициентов преобразованной системы размерности $n * n, D -$ вектор правых частей преобразованной системы.

Точность – задаётся параметром ε, которая определяет минимальную разницу между последними итерациями, при которой процесс можно завершить.

Диагональное преобладание – необходимо, чтобы матрица имела диагональный вид, если его нет, выполняется перестановка строк/столбцов.

Норма матрицы: $||A||\infty = \max \sum_j |a_{ij}|$

Листинг программы:

```
import sys
from PyQt6.QtWidgets import QApplication
from ui import SLAUSolverUI
from solver import SimpleIterationSolver

if __name__ == "__main__":
    app = QApplication(sys.argv)

    # Создаем объект метода решения
    solver_method = SimpleIterationSolver()

    # Передаем метод решения в интерфейс
    solver = SLAUSolverUI(solver_method)
    solver.show()

    sys.exit(app.exec())
```

```
import numpy as np

DEFAULT_EPSILON = 1e-6
MAX_ITERATIONS = 1000
MAX_MATRIX_SIZE = 20

# Абстракция для решения СЛАУ
class SLAUSolverMethod:
    def solve(self, A, b, eps=DEFAULT_EPSILON, max_iter=MAX_ITERATIONS):
        pass

# Реализация метода простых итераций
class SimpleIterationSolver(SLAUSolverMethod):
    def solve(self, A, b, eps=DEFAULT_EPSILON, max_iter=MAX_ITERATIONS):
        A_new, b_new, warning = rearrange_for_diagonal_dominance(A, b)
        result_message = warning or ""

        n = len(A_new)
        x = np.zeros(n)
        B = np.zeros((n, n))
        c = np.zeros(n)
```

```
        # Приведение к форме x = Bx + C
        for i in range(n):
            c[i] = b_new[i] / A_new[i, i]
            for j in range(n):
                if i != j:
                    B[i, j] = -A_new[i, j] / A_new[i, i]

        norm_B = np.max(np.sum(np.abs(B), axis=1))  # Норма по строкам
        if norm_B >= 1:
            return None, f"{result_message}Решений нет. Метод не сходится,
так как норма матрицы B ≥ 1.\n", norm_B

        iter_count = 0
        while iter_count < max_iter:
            x_new = np.dot(B, x) + c
            if np.linalg.norm(x_new - x, ord=np.inf) < eps:
                residual = np.dot(A_new, x_new) - b_new
                return x_new, iter_count, residual, result_message, norm_B
            x = x_new
            iter_count += 1

        return None, f"{result_message}Решения не найдены. Превышено
максимальное количество итераций.", norm_B


def rearrange_for_diagonal_dominance(A, b):
    n = len(A)
    A = A.copy()
    b = b.copy()

    for i in range(n):
        max_col = np.argmax(np.abs(A[i, :]))  # Находим индекс максимального
по модулю элемента в строке
        if max_col != i:
            A[:, [i, max_col]] = A[:, [max_col, i]]  # Переставляем столбцы
    # Проверка на диагональное преобладание
    for i in range(n):
        row_sum = np.sum(np.abs(A[i, :])) - np.abs(A[i, i])
        if np.abs(A[i, i]) < row_sum:
            return A, b, "Не удалось достичь диагонального преобладания.
Пробуем решить без него.\n"
    return A, b, None
```

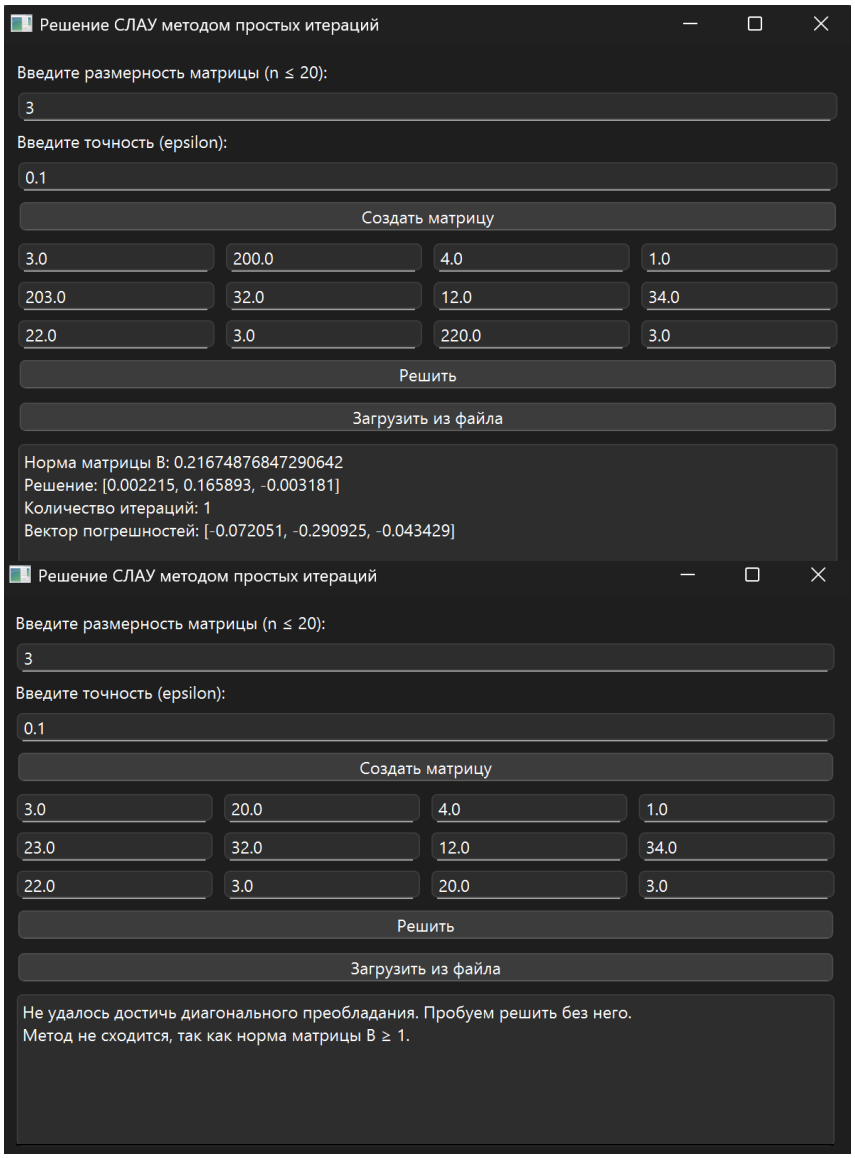
```
from PyQt6.QtWidgets import (
    QApplication, QWidget, QPushButton, QVBoxLayout, QTextEdit, QFileDialog,
QLabel, QLineEdit, QGridLayout,
    QMessageBox
)
import numpy as np


class SLAUSolverUI(QWidget):
    def __init__(self, solver_method):
        super().__init__()
        self.solver_method = solver_method
        self.initUI()

    def initUI(self):
        self.layout = QVBoxLayout()

        self.label = QLabel("Введите размерность матрицы (n ≤ 20):")
        self.layout.addWidget(self.label)
```

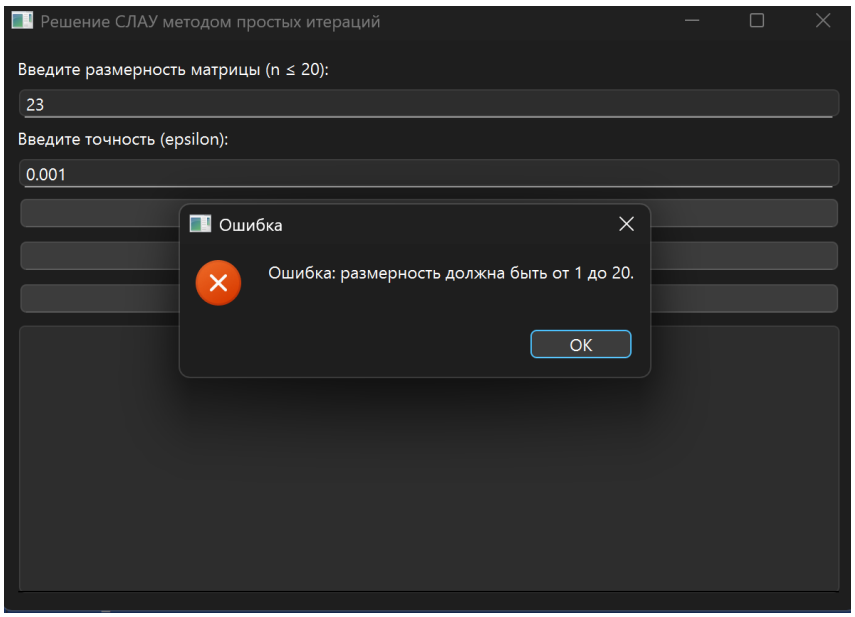
```
        self.size_input = QLineEdit()
        self.layout.addWidget(self.size_input)

        self.label_eps = QLabel("Введите точность (epsilon):")
        self.layout.addWidget(self.label_eps)

        self.eps_input = QLineEdit()
        self.layout.addWidget(self.eps_input)

        self.generate_matrix_button = QPushButton("Создать матрицу")
        self.generate_matrix_button.clicked.connect(self.create_matrix_input)
        self.layout.addWidget(self.generate_matrix_button)

        self.matrix_layout = QGridLayout()
        self.layout.addLayout(self.matrix_layout)

        self.solveButton = QPushButton("Решить")
        self.solveButton.clicked.connect(self.solve_slae)
        self.layout.addWidget(self.solveButton)

        self.loadButton = QPushButton("Загрузить из файла")
        self.loadButton.clicked.connect(self.load_from_file)
        self.layout.addWidget(self.loadButton)

        # Текстовое поле для результатов
        self.resultText = QTextEdit()
        self.resultText.setReadOnly(True)
        self.layout.addWidget(self.resultText)

        self.setLayout(self.layout)
        self.setWindowTitle("Решение СЛАУ методом простых итераций")
        self.resize(600, 400)

        self.matrix_inputs = []

    def create_matrix_input(self):
        # Очистка текущих полей
        for i in reversed(range(self.matrix_layout.count())):
            self.matrix_layout.itemAt(i).widget().setParent(None)

        try:
            n = int(self.size_input.text())
            if not (1 <= n <= 20):
                self.show_error_message("Ошибка: размерность должна быть от 1
до 20.")
                return
            self.matrix_inputs = [[QLineEdit() for _ in range(n + 1)] for _
in range(n)]
            for i in range(n):
                for j in range(n + 1):
                    self.matrix_layout.addWidget(self.matrix_inputs[i][j], i,
j)
        except ValueError:
            self.show_error_message("Введите корректное число.")

    def solve_slae(self):
        try:
            n = len(self.matrix_inputs)
            A = np.array([[float(self.matrix_inputs[i][j].text()) for j in
range(n)] for i in range(n)])
            B = np.array([float(self.matrix_inputs[i][-1].text()) for i in
range(n)])
            eps = float(self.eps_input.text()) if self.eps_input.text() else
1e-6
```

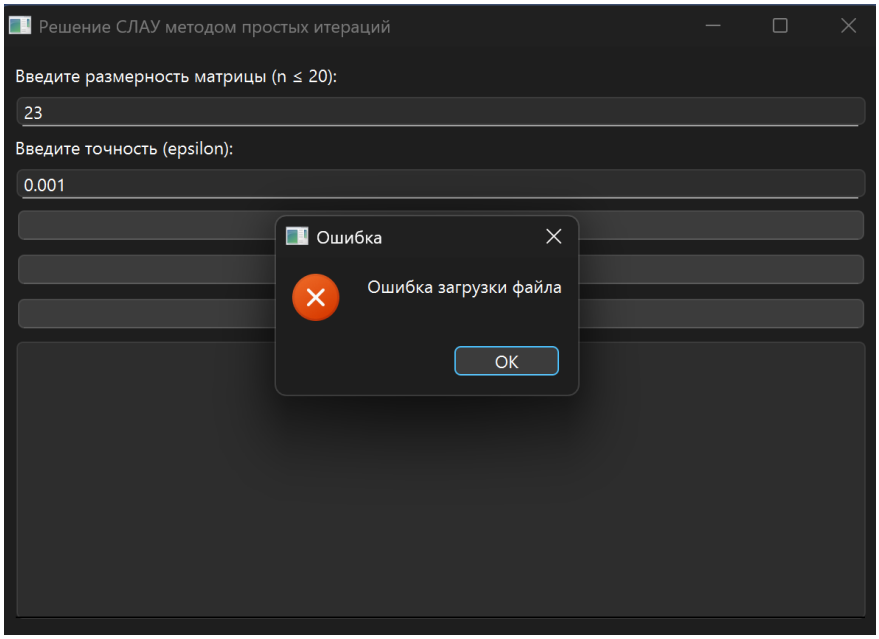
```
            result = self.solver_method.solve(A, B, eps)
            if result[0] is None:
                self.resultText.setText(result[1])  # Вывод ошибки в
текстовое поле
            else:
                x, iter_count, residual, warning, norm_B = result

                # Форматирование вывода
                x_str = ', '.join([f"{int(val) if val.is_integer() else
val:.6f}" for val in x])
                residual_str = ', '.join([f"{int(val) if val.is_integer()
else val:.6f}" for val in residual])

                # Вывод решения
                self.resultText.setText(
                    f"Норма матрицы B: {norm_B}\n"
                    f"Решение: [{x_str}]\n"
                    f"Количество итераций: {iter_count}\n"
                    f"Вектор погрешностей: [{residual_str}]\n"
                )

        except Exception as e:
            self.show_error_message(f"Ошибка ввода данных: {e}")

    def load_from_file(self):
        filename, _ = QFileDialog.getOpenFileName(self, "Выберите файл", "",
"Text Files (*.txt);;All Files (*)")
        if filename:
            try:
                with open(filename, 'r') as file:
                    lines = file.readlines()

                n = int(lines[0].strip())  # размерность
                eps = float(lines[1].strip())  # точность

                A = []
                b = []

                for line in lines[2:n + 2]:
                    values = list(map(float, line.split()))
                    A.append(values[:-1])
                    b.append(values[-1])

                A = np.array(A)
                b = np.array(b)

                # Автоматически подставляем в поля ввода
                self.size_input.setText(str(n))
                self.eps_input.setText(str(eps))
                self.create_matrix_input()

                # Заполняем GUI полями из файла
                for i in range(n):
                    for j in range(n):
                        self.matrix_inputs[i][j].setText(str(A[i, j]))
                    self.matrix_inputs[i][-1].setText(str(b[i]))

                self.resultText.setText("Файл загружен успешно.")
            except Exception as e:
                self.show_error_message(f"Ошибка загрузки файла")

    def show_error_message(self, message):
        """Всплывающее окно с сообщением об ошибке."""
```

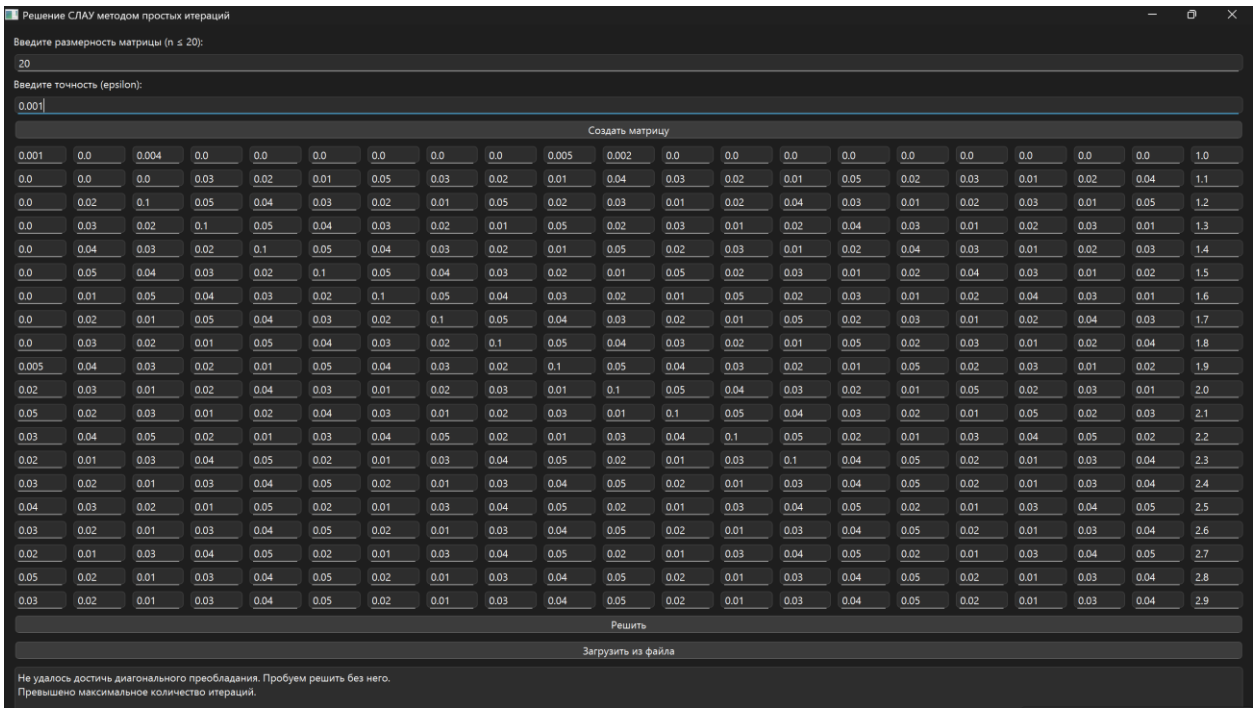
```
        error_dialog = QMessageBox(self)
        error_dialog.setIcon(QMessageBox.Icon.Critical)
        error_dialog.setWindowTitle("Ошибка")
        error_dialog.setText(message)
        error_dialog.exec()
```

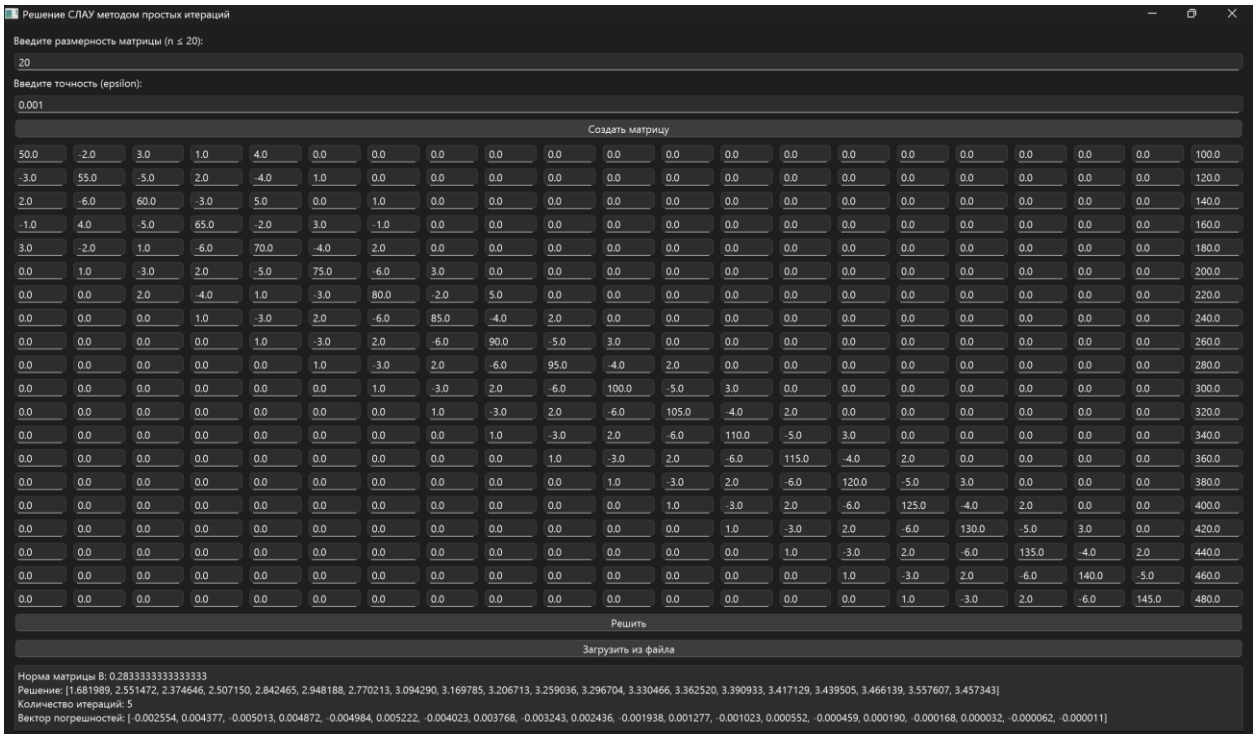
Пример работы программы:

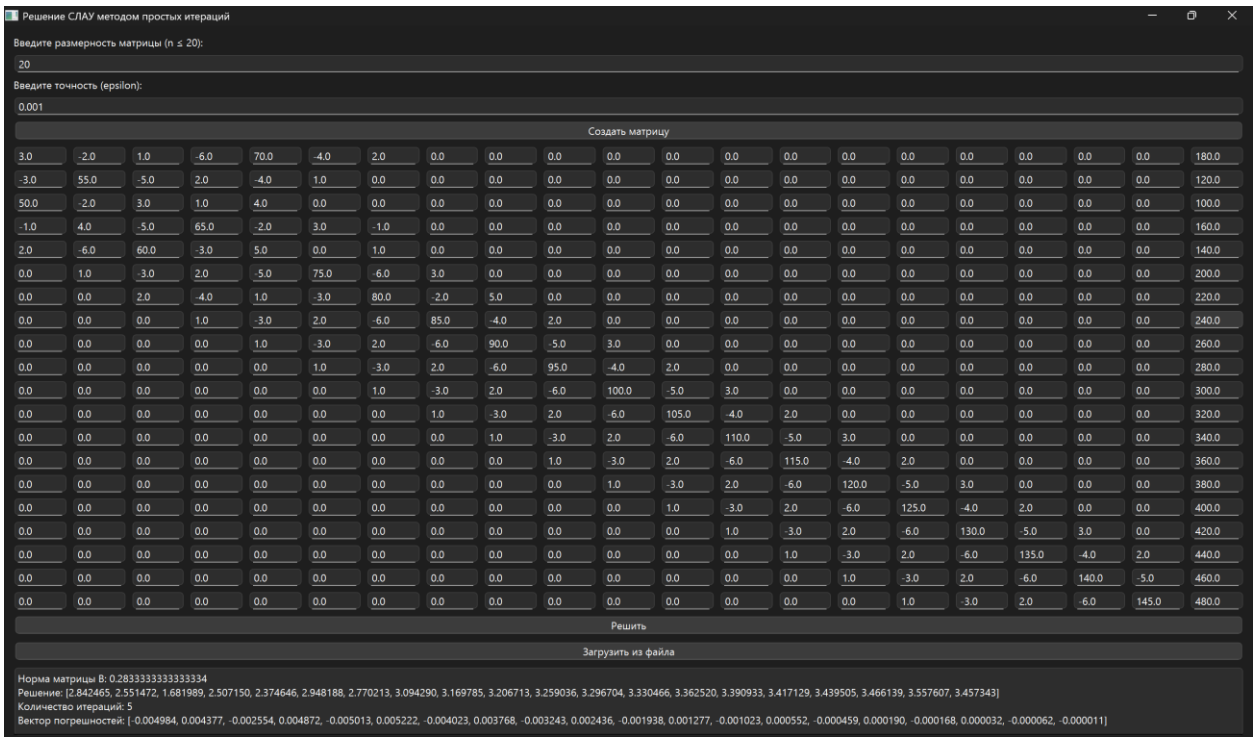



Решение СЛАУ методом простых итераций
Введите размерность матрицы (n ≤ 20):
23
Введите точность (epsilon):
0.001
Ошибка
Ошибка: размерность должна быть от 1 до 20.
OK


Решение СЛАУ методом простых итераций
Введите размерность матрицы (n ≤ 20):
23
Введите точность (epsilon):
0.001
Ошибка
Ошибка загрузки файла
OK


Решение СЛАУ методом простых итераций
Введите размерность матрицы (n ≤ 20):
20
Введите точность (epsilon):
0.001
Создать матрицу
0.001 0.0 0.004 0.0 0.0 0.0 0.0 0.0 0.0 0.005 0.002 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.03 0.02 0.01 0.05 0.03 0.02 0.01 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 1.1
0.0 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 0.03 0.01 0.05 1.2
0.0 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 0.03 0.01 1.3
0.0 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 0.03 1.4
0.0 0.05 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 1.5
0.0 0.01 0.05 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 1.6
0.0 0.02 0.01 0.05 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 0.03 1.7
0.0 0.03 0.02 0.01 0.05 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 0.04 1.8
0.005 0.04 0.03 0.02 0.01 0.05 0.04 0.03 0.02 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 0.02 1.9
0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 0.03 0.01 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 0.01 2.0
0.05 0.02 0.03 0.01 0.02 0.04 0.03 0.01 0.02 0.03 0.01 0.1 0.05 0.04 0.03 0.02 0.01 0.05 0.02 0.03 2.1
0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.1 0.05 0.02 0.01 0.03 0.04 0.05 0.02 2.2
0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.1 0.04 0.05 0.02 0.01 0.03 0.04 2.3
0.03 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 2.4
0.04 0.03 0.02 0.01 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 2.5
0.03 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 2.6
0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 2.7
0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 2.8
0.03 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 0.05 0.02 0.01 0.03 0.04 2.9
Решить
Загрузить из файла
Не удалось достичь диагонального преобладания. Пробуем решить без него.
Превышено максимальное количество итераций.


Решение СЛАУ методом простых итераций
Введите размерность матрицы (n ≤ 20):
20
Введите точность (epsilon):
0.001
Создать матрицу
50.0 -2.0 3.0 1.0 4.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 100.0
-3.0 55.0 -5.0 2.0 -4.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 120.0
2.0 -6.0 60.0 -3.0 5.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 140.0
-1.0 4.0 -5.0 65.0 -2.0 3.0 -1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 160.0
3.0 -2.0 1.0 -6.0 70.0 -4.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 180.0
0.0 1.0 -3.0 2.0 -5.0 75.0 -6.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 200.0
0.0 0.0 2.0 -4.0 1.0 -3.0 80.0 -2.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 220.0
0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 85.0 -4.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 240.0
0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 90.0 -5.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 260.0
0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 95.0 -4.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 280.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 100.0 -5.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 300.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 105.0 -4.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 320.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 110.0 -5.0 3.0 0.0 0.0 0.0 0.0 0.0 340.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 115.0 -4.0 2.0 0.0 0.0 0.0 0.0 360.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 120.0 -5.0 3.0 0.0 0.0 0.0 380.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 125.0 -4.0 2.0 0.0 0.0 400.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 130.0 -5.0 3.0 0.0 420.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 135.0 -4.0 2.0 440.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 140.0 -5.0 460.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -3.0 2.0 -6.0 145.0 480.0
Решить
Загрузить из файла
Норма матрицы B: 0.2833333333333333
Решение: [1.681989, 2.551472, 2.374646, 2.507150, 2.842465, 2.948188, 2.770213, 3.094290, 3.169785, 3.206713, 3.259036, 3.296704, 3.330466, 3.362520, 3.390933, 3.417129, 3.439505, 3.466139, 3.557607, 3.457343]
Количество итераций: 5
Вектор погрешностей: [0.002554, 0.004377, -0.005013, 0.004872, -0.004984, 0.005222, -0.004023, 0.003768, -0.003243, 0.002436, -0.001938, 0.001277, -0.001023, 0.000552, -0.000459, 0.000190, -0.000168, 0.000032, -0.000062, -0.000011]



Вывод:

Метод простых итераций является простым и эффективным методом для решения СЛАУ при условии, что матрица обладает диагональным преобладанием или может быть преобразована в такую форму. Важным аспектом является то, что метод требует анализа матрицы на наличие диагонального преобладания и вычисления нормы матрицы для оценки сходимости. При этом метод может быть неустойчив, если матрица не удовлетворяет условиям сходимости.