Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Лабораторная работа № 1

«Решение системы линейных алгебраических уравнений СЛАУ»

Вычислительная математика

Студент

Дубинин Артём Сергеевич

группа P3215

Преподаватель

Малышева Татьяна Алексеевна

Санкт - Петербург 2025 год

Цель работы:

- Реализация численного метода решения системы линейных алгебраических уравнений (СЛАУ) в соответствии с заданным вариантом (Метод Якоби).

Описание метода, расчетные формулы:

Метод Якоби:

- Метод Якоби — это итерационный метод решения СЛАУ, который используется для нахождения приближенного решения системы линейных уравнений.
- Основная идея метода заключается в последовательном уточнении решения на каждой итерации.
- Формула для вычисления нового приближения:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} * \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), i = 1,2, \dots, n$$

- Условие остановки итераций: (где ε — заданная точность)

$$\left\| x^{(k+1)} - x^{(k)} \right\| < \varepsilon$$

Диагональное преобладание:

- Для сходимости метода Якоби необходимо, чтобы матрица системы имела диагональное преобладание:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \;\; i = 1,2, \dots, n$$

- Если диагональное преобладание отсутствует, программа пытается переставить строки матрицы для его достижения.

Спектральная норма:

- Спектральная норма вычисляется как квадратный корень из максимального собственного значения матрицы $A^T A$, где A^T — транспонированная матрица A.

Листинг программы:

Основные функции:

- **read_matrix_from_file** — чтение матрицы из файла.

```
    def read_matrix_from_file(filename):
        try:
            with open(filename, 'r') as file:
                # Чтение размерности матрицы
                n = int(file.readline().strip())  # Убираем лишние
пробелы и символы новой строки
                matrix = []
                # Чтение матрицы A
                for _ in range(n):
                    row = list(map(float,
file.readline().strip().split()))
                    matrix.append(row)
                # Чтение вектора b
                b = list(map(float, file.readline().strip().split()))
                # Чтение точности
                tolerance = float(file.readline().strip())
                return np.array(matrix), np.array(b), n, tolerance
        except FileNotFoundError:
            print(f"Ошибка: Файл '{filename}' не найден.")
            return None, None, None, None
        except ValueError:
            print(f"Ошибка: Файл '{filename}' имеет неправильный
формат.")
            return None, None, None, None
```

- **read_matrix_from_keyboard** — ввод матрицы с клавиатуры.

```
    def read_matrix_from_keyboard(n):
        matrix = []
        print("Введите коэффициенты матрицы построчно:")
        for _ in range(n):
            row = list(map(float, input().split()))
            matrix.append(row)
        b = np.array(list(map(float, input("Введите вектор правых
частей (b): ").split())))
        tolerance = float(input("Введите точность: "))
        return np.array(matrix), b, tolerance
```

- **check_diagonal_dominance** — проверка диагонального преобладания.

```
    def check_diagonal_dominance(matrix, n):
        """Проверяет, является ли матрица диагонально доминирующей."""
        for i in range(n):
            diagonal_element = abs(matrix[i][i])
            sum_of_other_elements = sum(abs(matrix[i][j]) for j in
range(n) if j != i)
            if diagonal_element < sum_of_other_elements:
                return False
        return True

```

- **rearrange_for_diagonal_dominance** — перестановка строк для достижения диагонального преобладания.

```
def rearrange_for_diagonal_dominance(matrix, b, n):
    """Перебирает все возможные перестановки строк для достижения диагонального преобладания."""
    # Генерируем все возможные перестановки индексов строк
    for perm in permutations(range(n)):
        # Создаем копии матрицы и вектора b
        new_matrix = matrix.copy()
        new_b = b.copy()

        # Применяем перестановку
        for i in range(n):
            new_matrix[i] = matrix[perm[i]]
            new_b[i] = b[perm[i]]

        # Проверяем, достигнуто ли диагональное преобладание
        if check_diagonal_dominance(new_matrix, n):
            return new_matrix, new_b

    return matrix, b
```

- **jacobi_method** — реализация метода Якоби.

```
def jacobi_method(A, b, tolerance):
    n = len(b)
    x = np.zeros(n)
    x_new = np.zeros(n)
    iterations = 0
    errors = []  # Вектор погрешностей
    max_iterations = 1000  # Максимальное количество итераций

    for _ in range(max_iterations):
        for i in range(n):
            sum_ = sum(A[i][j] * x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - sum_) / A[i][i]

        # Вычисление погрешности (норма разности между текущим и предыдущим приближением)
        error = np.linalg.norm(x_new - x)
        errors.append(error)

        if error < tolerance:
            x = x_new.copy()  # Обновляем x на последней итерации
            break

        x = x_new.copy()
        iterations += 1

    return x, iterations, errors
```

Примеры и результаты работы программы:

Пример №1 (Решение есть):

Метод простой итерации. Пример

Методом простых итераций с точностью $\varepsilon = 0{,}01$ решить систему линейных алгебраических уравнений:

$$\begin{cases} 2x_1 + 2x_2 + 10x_3 = 14 \\ 10x_1 + x_2 + x_3 = 12 \\ 2x_1 + 10x_2 + x_3 = 13 \end{cases}$$

```
Введите '1' для ввода данных с клавиатуры или '2' для ввода из файла: 1
Введите размерность матрицы (n <= 20): 3
Введите коэффициенты матрицы построчно:
2 2 10
10 1 1
2 10 1
Введите вектор правых частей (b): 14 12 13
Введите точность: 0.01
Диагональное преобладание отсутствует. Пытаемся переставить строки...
Диагональное преобладание достигнуто после перестановки строк.

Норма матрицы A (спектральная норма): 13.030989593863206
Используемая точность: 0.01

Вектор неизвестных (x):
[0.999568 0.99946  0.999316]

Количество итераций: 5

Вектор погрешностей (на каждой итерации):
[2.256103, 0.683593, 0.188308, 0.054392, 0.01539, 0.004393]

Вектор невязок:
[-0.005544 -0.006948 -0.008784]

Решение, полученное с помощью библиотеки numpy:
[1. 1. 1.]
```

Пример №2 (Решения нет):

Входные данные:

- Матрица A:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

- Вектор b:

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

- Точность: 0.001.

```
Введите '1' для ввода данных с клавиатуры или '2' для ввода из файла: 1
Введите размерность матрицы (n <= 20): 3
Введите коэффициенты матрицы построчно:
1 2 3
4 5 6
7 8 9
Введите вектор правых частей (b): 1 2 3
Введите точность: 0.001
Диагональное преобладание отсутствует. Пытаемся переставить строки...
Невозможно достичь диагонального преобладания. Метод может не сойтись.
```

Выводы:

- Метод Якоби успешно решает СЛАУ при условии диагонального преобладания матрицы.
- Программа корректно проверяет и переставляет строки для достижения диагонального преобладания.
- Результаты работы программы совпадают с решением, полученным с помощью библиотеки numpy.
- Метод Якоби эффективен для систем с диагональным преобладанием, но может не сходиться в противном случае.