Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

По вычислительной математике

Вариант №7. Метод простых итераций

Выполнила:

Мельник Фёдор Александрович

Группа № P3206

Проверил:

Зинченко Антон Андреевич

Санкт-Петербург 2025

Цель

Реализовать программу, решающую систему линейных алгебраических уравнений СЛАУ методом простых итераций с учетом следующих требований:

1. В программе численный метод должен быть реализован в виде отдельной подпрограммы/метода/класса, в который исходные/выходные данные передаются в качестве параметров.
2. Размерность матрицы n<=20 (задается из файла или с клавиатуры - по выбору конечного пользователя).
3. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).
4. Точность задается с клавиатуры/файла,
5. Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
6. Вывод нормы матрицы (любой, на Ваш выбор),
7. Вывод вектора неизвестных: $x_1, x_2, \dots, x_n$,
8. Вывод количества итераций, за которое было найдено решение,
9. Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$.

Описание метода

Метод простых итераций – метод для решения систем линейных алгебраических уравнений СЛАУ, идеей которого является:

Система вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Приводится к виду:

$$x = Bx + c$$

После этого строится итерационный процесс:

$$x^{(k+1)} = Bx^{(k)} + c$$

Метод сходится, если для каждой строки выполняется диагональное преобладание матрицы:

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|$$

Условие остановки:

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

где ε - точность

Листинг кода

```
import sys


def safe_float_input(value):
    try:
        return float(value)
    except ValueError:
        raise ValueError("Ошибка: введено не число.")


def input_from_keyboard():
    try:
        n = int(input("Введите размерность матрицы (n <= 20): "))
        if n <= 0 or n > 20:
            raise ValueError("Размерность должна быть 1 <= n <= 20")

        print("Введите матрицу A построчно:")
        A = []
        for i in range(n):
            row = input().split()
            if len(row) != n:
                raise ValueError("Неверное количество элементов в строке.")
            A.append([safe_float_input(x) for x in row])

        print("Введите вектор b:")
        b_input = input().split()
        if len(b_input) != n:
            raise ValueError("Неверное количество элементов в векторе b.")
        b = [safe_float_input(x) for x in b_input]

        eps = safe_float_input(input("Введите точность: "))
        if eps <= 0:
            raise ValueError("Точность должна быть положительной.")

        return A, b, eps
    except Exception as e:
        print("Ошибка ввода:", e)
        sys.exit(1)


def input_from_file(filename):
    try:
        with open(filename, 'r') as f:
            lines = f.readlines()

        n = int(lines[0])
        if n <= 0 or n > 20:
            raise ValueError("Размерность должна быть 1 <= n <= 20")

        A = []
        for i in range(1, n + 1):
            row = lines[i].split()
            if len(row) != n:
                raise ValueError("Ошибка в матрице.")
            A.append([float(x) for x in row])

        b = [float(x) for x in lines[n + 1].split()]
        if len(b) != n:
            raise ValueError("Ошибка в векторе b.")

        eps = float(lines[n + 2])
```

```
        if eps <= 0:
            raise ValueError("Точность должна быть положительной.")

        return A, b, eps

    except FileNotFoundError:
        print("Файл не найден.")
        sys.exit(1)
    except Exception as e:
        print("Ошибка чтения файла:", e)
        sys.exit(1)


def is_diagonally_dominant(A):
    n = len(A)
    for i in range(n):
        if abs(A[i][i]) <= sum(abs(A[i][j]) for j in range(n) if j != i):
            return False
    return True


def make_diagonally_dominant(A, b):
    n = len(A)
    used = [False] * n
    current_perm = []

    def backtrack():
        if len(current_perm) == n:
            new_A = [A[i] for i in current_perm]
            new_b = [b[i] for i in current_perm]
            if is_diagonally_dominant(new_A):
                return new_A, new_b
            return None

        for i in range(n):
            if not used[i]:
                used[i] = True
                current_perm.append(i)

                result = backtrack()
                if result:
                    return result

                current_perm.pop()
                used[i] = False

        return None

    result = backtrack()
    if result:
        return result
    return None, None


def matrix_norm(A):
    return max(sum(abs(x) for x in row) for row in A)


def simple_iteration(A, b, eps, max_iter=10000):
    n = len(A)
    x = [0.0] * n
    x_new = x.copy()
    iterations = 0
```

```
    while iterations < max_iter:
        iterations += 1
        for i in range(n):
            s = sum(A[i][j] * x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i][i]

        errors = [abs(x_new[i] - x[i]) for i in range(n)]

        if max(errors) < eps:
            return x_new, iterations, errors

        x = x_new.copy()

    raise ValueError("Метод не сошелся за максимальное число итераций.")


def main():
    print("Выберите способ ввода:")
    print("1 - С клавиатуры")
    print("2 - Из файла")
    choice = input("Ваш выбор: ")

    if choice == '1':
        A, b, eps = input_from_keyboard()
    elif choice == '2':
        filename = input("Введите имя файла: ")
        A, b, eps = input_from_file(filename)
    else:
        print("Некорректный выбор.")
        sys.exit(1)

    if not is_diagonally_dominant(A):
        print("Диагонального преобладания нет. Выполняется попытка
перестановки строк...")
        A, b = make_diagonally_dominant(A, b)
        if A is None:
            print("Невозможно достичь диагонального преобладания.")
            sys.exit(1)
        else:
            print("Диагональное преобладание достигнуто.")

    print("\nНорма матрицы:", matrix_norm(A))

    try:
        solution, iterations, errors = simple_iteration(A, b, eps)

        print("\nРешение:")
        for i, x in enumerate(solution):
            print(f"x{i + 1} = {x}")

        print("\nКоличество итераций:", iterations)

        print("\nВектор погрешностей:")
        for i, e in enumerate(errors):
            print(f"|x{i + 1}(k) - x{i + 1}(k-1)| = {e:.9f}")

    except Exception as e:
        print("Ошибка вычислений:", e)


if __name__ == "__main__":
    main()
```

Примеры работы

Простая система

```
3
10 1 1
2 10 1
2 2 10
12 13 14
0.0001
```

```
Норма матрицы: 14.0

Решение:
x1 = 1.000009936
x2 = 1.000012528
x3 = 1.0000157680000001

Количество итераций: 9

Вектор погрешностей:
|x1(k) - x1(k-1)| = 0.000044856
|x2(k) - x2(k-1)| = 0.000056448
|x3(k) - x3(k-1)| = 0.000071208
```

Невозможно получить диагональное преобладание

```
3
1 2 3
4 5 6
7 8 9
1 1 1
0.0001
```

```
Диагонального преобладания нет. Выполняется попытка перестановки строк...
Невозможно достичь диагонального преобладания.
```

Диагональное преобладание через перестановку

```
3
1 10 1
2 1 10
10 2 1
12 13 14
0.0001
```

```
Диагонального преобладания нет. Выполняется попытка перестановки строк...
Диагональное преобладание достигнуто.

Норма матрицы: 13.0

Решение:
x1 = 1.103671475
x2 = 0.991628175
x3 = 0.9801116499999999

Количество итераций: 9

Вектор погрешностей:
|x1(k) - x1(k-1)| = 0.000031475
|x2(k) - x2(k-1)| = 0.000024675
|x3(k) - x3(k-1)| = 0.000033400
```

Вывод

В ходе лабораторной работы был реализован метод простых итераций для решения СЛАУ с проверкой и достижением диагонального преобладания. Программа поддерживает ввод данных с клавиатуры и из файла, обрабатывает ошибки ввода, вычисляет норму матрицы, вектор неизвестных, вектор погрешностей и количество итераций. Эксперименты показали, что метод сходится при выполнении условия диагонального преобладания, а полученные результаты соответствуют заданной точности.