Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Вариант № 1

Выполнил: студент группы P3208,
Васильев Н. А.

Преподаватель: Машина Е.А.

Санкт-Петербург 2025

Текст задания

Решение системы линейных алгебраических уравнений СЛАУ методом простых итераций.

В программе должно быть реализовано:

- Точность задается с клавиатуры/файла,
- 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.
- Вывод нормы матрицы (любой, на Ваш выбор),
- Вывод вектора неизвестных: $x_1, x_2, \dots, x_n$,
- Вывод количества итераций, за которое было найдено решение,
- Вывод вектора погрешностей: $\left|x_i^{(k)} - x_i^{(k-1)}\right|$.

Цель работы

Решение системы линейных алгебраических уравнений СЛАУ методом простых итераций.

Описание метода, расчётные формулы

Рабочая формула метода простой итерации:

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

где k – номер итерации.

Диагональное доминирование:

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

Норма матрицы:

$$\|A\|_\infty = \max\left(\sum_{j=1}^{n} |a_{ij}|\right), i = 1, 2, \dots, n$$

Погрешность:

$$\Delta = \left\|x^* - x^{(k)}\right\| \leq \frac{\|C\|}{1 - \|C\|} \left\|x^{(k)} - x^{(k-1)}\right\|$$

Листинг программы

```python
import csv
import numpy as np
import random


def generate_random_matrix(min_val=-25, max_val=25):
    size = random.randint(2, 20)
    matrix = [[random.uniform(min_val, max_val) for _ in range(size)] for _
in range(size)]
    vector = [random.uniform(min_val, max_val) for _ in range(size)]

    for i in range(size):
        matrix[i].append(vector[i])

    return matrix


def read_matrix_from_file(filename):
    matrix = []
    try:
        with open(filename, newline='') as file:
            if filename.endswith('.csv'):
                reader = csv.reader(file)
                rows = [row for row in reader]
            else:
                rows = [line.split() for line in file]

            for row in rows:
                if not all(num.replace('.', '', 1).replace('-', '',
1).isdigit() for num in row):
                    raise ValueError("Ошибка: все элементы должны быть
вещественными числами")
                matrix.append([float(num) for num in row])
        return matrix
    except FileNotFoundError:
        print("Такой файл не найден. Попробуйте снова.")
        return read_matrix_from_file(input())


def validate_matrix(matrix):
    n = len(matrix)
    for row in matrix:
        if len(row) != n + 1:
            raise ValueError("Ошибка: каждая строка должна содержать n
элементов матрицы и один элемент вектора")


def diagonal_dominance(matrix):
    matrix = np.array(matrix)
    n = len(matrix)
    for i in range(n):
        row_sum = sum(abs(matrix[i, j]) for j in range(n) if j != i)
        if abs(matrix[i, i]) <= row_sum:
            return False
    return True
```

```python
def rearrange_matrix(matrix, vector):
    n = len(matrix)
    matrix = np.array(matrix, dtype=float)
    vector = np.array(vector, dtype=float)
    for i in range(n):
        max_index = max(range(i, n),
                        key=lambda k: abs(matrix[k, i]) - sum(abs(matrix[k,
j]) for j in range(n) if j != i))
        if max_index != i:
            matrix[[i, max_index]] = matrix[[max_index, i]]
            vector[i], vector[max_index] = vector[max_index], vector[i]
    return matrix, vector


def norm_matrix(matrix):
    return max(sum(abs(element) for element in row) for row in matrix)


def iteration(matrix, x, vector):
    n = len(matrix)
    x_new = np.zeros(n)
    for i in range(n):
        sum_ax = sum(matrix[i][j] * x[j] for j in range(n) if j != i)
        x_new[i] = (vector[i] - sum_ax) / matrix[i][i]
    return x_new


def iterative_method(matrix, vector, precision, max_iterations=1000):
    n = len(matrix)
    x = np.zeros(n)
    iterations = 0
    errors = []

    while iterations <= max_iterations:
        x_new = iteration(matrix, x, vector)
        errors.append(np.abs(x_new - x))

        iterations += 1

        if np.max(np.abs(x_new - x)) < precision:
            return x_new, iterations, errors

        x = x_new

    print("Метод не сошёлся за", max_iterations, "итераций")
    return x, max_iterations, errors


def calculate_matrix(matrix, precision):
    validate_matrix(matrix)
    matrix = np.array(matrix, dtype=float)
    vector = matrix[:, -1]
    matrix = matrix[:, :-1]
```

```
    if not diagonal_dominance(matrix):
        matrix, vector = rearrange_matrix(matrix, vector)
        if not diagonal_dominance(matrix):
            print("Ошибка: матрица не является диагонально доминирующей и не
может быть преобразована")
            return

    norm = norm_matrix(matrix)
    x, iterations, errors = iterative_method(matrix, vector, precision)

    print("Матрица после преобразований: ")
    for i in range(len(matrix)):
        print([float('%.2f' % elem) for elem in matrix[i]])
    print("Вектор свободных членов после преобразований: ", vector)
    print("Норма матрицы: ", norm)
    print("Вектор неизвестных: ", x)
    print("Количество итераций: ", iterations)
    print("Погрешность: ", errors[-1] if errors else None)


def is_valid_number(value):
    try:
        float(value)
        return True
    except ValueError:
        return False


def main():
    print("Решение системы линейных алгебраических уравнений СЛАУ")
    while True:
        try:
            print("Введите точность:")
            precision = float(input())
            if precision <= 0:
                raise ValueError("Ошибка: точность должна быть положительным
числом.")
            break
        except ValueError:
            print("Ошибка: введите корректное положительное число для
точности.")
    print(
        "Введите матрицу в формате 'a11 a12 a13 ... a1n b1; a21 a22 a23 ...
a2n b2; ... a3n1 an2 an3 ... ann bn' или введите имя файла с матрицей (.txt
или .csv)")
    print("Матрица должна быть квадратной. n <= 20")
    print("Введите имя файла или первую строку матрицы: (Для генерации
случайной матрицы введите RANDOM)")
    while True:
        try:
            input_data = input().strip()
            if input_data.endswith('.csv') or input_data.endswith('.txt'):
                matrix = read_matrix_from_file(input_data)
            elif input_data == 'RANDOM':
```

```
                matrix = generate_random_matrix()
                print("Сгенерированная матрица:")
                for i in range(len(matrix)):
                    print([float('%.2f' % elem) for elem in matrix[i]])
            else:
                matrix = [[float(num) for num in input_data.split()]]
                if len(matrix[0]) > 21:
                    raise ValueError("Ошибка: размерность матрицы должна быть
не больше 20.")
                current_row = 1
                while current_row < len(matrix[0]) - 1:
                    input_data = input().strip()
                    row = input_data.split()
                    if len(row) != len(matrix[0]) or not
all(is_valid_number(num) for num in row):
                        print(f"Ошибка: строка должна содержать
{len(matrix[0])} вещественных чисел.")
                        continue
                    matrix.append([float(num) for num in row])
                    current_row += 1
            break
        except ValueError as e:
            print(f"Ошибка: {e}. Попробуйте снова.")
    try:
        calculate_matrix(matrix, precision)
    except ValueError as e:
        print(e)


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

Примеры и результаты работы программы

Пример №1. Ввод матрицы 3*3 с точностью 0,01:

12 1 3 1

0 0 5 3

1 6 2 2

Вывод:

Матрица после преобразований:

[12.0, 1.0, 3.0]

[1.0, 6.0, 2.0]

[0.0, 0.0, 5.0]

Вектор свободных членов после преобразований: [1. 2. 3.]

Норма матрицы: 16.0

Вектор неизвестных: [-0.07908951 0.1461034   0.6   ]

Количество итераций: 4

Погрешность: [0.00246914 0.00297068 0.   ]

Пример №2. Ввод дробных и отрицательных чисел с точностью 1:

1.3 -2.1 4 3

-8.3 -1.1 3 3

2.1 5.5 0.7 2.2

Вывод:

Матрица после преобразований:

[-8.3, -1.1, 3.0]

[2.1, 5.5, 0.7]

[1.3, -2.1, 4.0]

Вектор свободных членов после преобразований: [3. 2.2 3.]

Норма матрицы: 12.4

Вектор неизвестных: [-0.36144578 0.4   0.75   ]

Количество итераций: 1

Погрешность: [0.36144578 0.4   0.75   ]

Пример №3. Ввод матрицы, которую нельзя диагонализировать:

1 1 1 2

2 4 2 2

1 1 1 1

Вывод:

Ошибка: матрица не является диагонально доминирующей и не может быть преобразована

Пример №4. Ввод матрицы, из файла:

matrix.txt

Содержание файла:

1 22 1 3

12 3 4 2

1 3 10 3

Вывод:

Матрица после преобразований:

[12.0, 3.0, 4.0]

[1.0, 22.0, 1.0]

[1.0, 3.0, 10.0]

Вектор свободных членов после преобразований: [2. 3. 3.]

Норма матрицы: 24.0

Вектор неизвестных: [0.16666667 0.13636364 0.3   ]

Количество итераций: 1

Погрешность: [0.16666667 0.13636364 0.3   ]

Пример №5. Случайная матрица:

Сгенерированная матрица:

[15.47, -18.42, -13.58, 17.51, -4.01, -17.05, 1.73, 15.2, 20.22, 6.1]

[-4.49, -23.7, 7.38, -24.66, 16.33, -23.35, -20.7, 14.2, -15.03, -19.88]

[8.66, -12.81, -16.26, -5.27, 0.6, 24.57, 15.58, 1.75, 7.06, 3.55]

[21.72, 7.84, 12.2, 1.51, 3.44, 3.44, 1.0, 20.59, -7.8, 18.36]

[5.03, 10.64, -24.3, 5.97, -20.43, -5.95, -24.67, -18.57, -19.05, 3.39]

[5.87, -14.11, -10.43, -19.42, -3.26, 17.02, 13.46, -3.72, -7.5, 16.25]

[16.94, 18.95, 24.99, -5.38, -22.26, 9.0, 0.52, -1.17, -21.26, 6.63]

[14.56, -11.44, -24.71, -23.42, 23.69, 23.63, 13.51, 8.18, -2.69, -15.23]

[-21.04, 18.88, -23.15, 8.39, -18.25, -18.34, -6.65, -15.25, 0.37, 2.78]

Ошибка: матрица не является диагонально доминирующей и не может быть преобразована

Вывод

В ходе выполнения данной работы была реализована программа для решения системы линейных алгебраических уравнений (СЛАУ) с использованием итерационных методов.

Реализована возможность ввода коэффициентов матрицы как с клавиатуры, так и из файла (форматы .txt, .csv). Пользователь может задать точность вычислений вручную.