МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРОНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине
«Вычислительная математика»
Вариант № 13 (метод простых итераций)

Выполнил работу:

Студент группы P3218

Рамеев Тимур Ильгизович

Преподаватель:

Бострикова Дарья

Константиновна

Санкт-Петербург 2024

Содержание

Цель работы

Цель работы заключается в ознакомлении с методами решения СЛАУ и их способами их реализации на языках программирования

Задание

Написать программу для решения СЛАУ с использованием метода простых итераций.

Требования к программе

Общее

- В программе численный метод должен быть реализован в виде отдельной подпрограммы/метода/класса, в который исходные/выходные данные передаются в качестве параметров.
- Размерность матрицы n<=20 (задается из файла или с клавиатуры – по выбору конечного пользователя).
- Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Для итерационных методов 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 вектора неизвестных: $x1, x2, \dots, xn$
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей: $|xi\,(k) - xi\,(k-1)|$

Описание метода выполнения

Программа для решения СЛАУ при помощи метода простых итераций была написана на языке Python 3.12.
В ходе решения задачи была использована библиотека random для генерации случайной матрицы.

Код класса для численного метода

Исходный код программы

GitHub Link

```
      class Solver:

    def simple_iterations(self, entity):
        print("Проверяю достаточные условия...")
        if self.changing_the_order(entity) != simple_iterations_codes.OK:
            print(f"{bcolors.FAIL}Достаточные условия для уравнения не выполняются!
Я не могу его решить :({bcolors.ENDC}")
            return
        else:
            print(f"{bcolors.OKGREEN}Достаточные условия для уравнения выполняются!
Решаю...{bcolors.ENDC}")
            self.expressing_coefficients(entity)
            self.doing_iterations(entity)
            print(f"{bcolors.OKGREEN}Система успешно решена!{bcolors.ENDC}")
    def changing_the_order(self, entity):
        control_flag = False
        main_map = dict()
        d_map = dict()
        entity_main_matrix = entity.get_main_matrix()
        entity_d_matrix = entity.get_d_matrix()
        for i in entity_main_matrix:
            for j in range(len(i)):
                if i[j] < 0:
                    i[j] = abs(i[j])
        for i in range(0, len(entity_main_matrix)):
            max_item = entity_main_matrix[i].index(max(entity_main_matrix[i])) + 1
            main_map[max_item] = entity_main_matrix[i]
            d_map[max_item] = entity_d_matrix[i]
        if (len(main_map) < len(entity_main_matrix)):
            return simple_iterations_codes.INVALID_EQUATION
        for i in main_map:
            max_item = max(main_map[i])
            if sum(main_map[i]) - max_item > max_item:
                return simple_iterations_codes.INVALID_EQUATION
            elif max_item > sum(main_map[i]) - max_item:
                control_flag = True
        if control_flag == False:
            return simple_iterations_codes.INVALID_EQUATION
```

```
        entity.set_main_matrix([i[1] for i in sorted(main_map.items())])
        entity.set_d_matrix([i[1] for i in sorted(d_map.items())])
        return simple_iterations_codes.OK

    def expressing_coefficients(self, entity):
        new_main_arr = list()
        new_d_arr = list()
        old_main_arr = entity.get_main_matrix()
        old_d_arr = entity.get_d_matrix()
        for i in range (1, len(old_main_arr) + 1):
            current_arr = []
            for j in range(1, len(old_main_arr[i - 1]) + 1):
                if (i == j):
                    current_arr.append(0)
                else:
                    current_arr.append(old_main_arr[i - 1][j - 1] / -old_main_arr[i
- 1][i - 1])
            new_main_arr.append(current_arr)
            new_d_arr.append(old_d_arr[i - 1] / old_main_arr[i - 1][i - 1])
        entity.set_d_matrix(new_d_arr)
        entity.set_main_matrix(new_main_arr)

    def doing_iterations(self, entity):
        current_vector = entity.get_d_matrix()
        counter = 1
        while True:
            counter += 1
            new_vector = []
            for i in range(len(entity.get_main_matrix())):
                current_coord = 0
                for j in range(len(entity.get_main_matrix()[i])):
                    current_coord += entity.get_main_matrix()[i][j] *
current_vector[j]
                current_coord += entity.get_d_matrix()[i]
                new_vector.append(current_coord)
            # Определим максимальное отклонение
            variance = -100000
            for i in range(len(current_vector)):
                if variance < abs(current_vector[i] - new_vector[i]):
                    variance = abs(current_vector[i] - new_vector[i])
            if variance <= entity.get_accuracy():
                error_vector = []
                for i in range(len(current_vector)):
                    error_vector.append(abs(current_vector[i] - new_vector[i]))

                print(f'Количество итераций:
{bcolors.OKCYAN}{counter}{bcolors.ENDC}')
                print(f'Вектор погрешностей: {bcolors.OKCYAN}({str(error_vector)[1:-
1]}){bcolors.ENDC}')
                print(f'Решение: {bcolors.OKCYAN}({str(new_vector)[1:-
1]}){bcolors.ENDC}')
                break
            current_vector = new_vector
```

Пример вывода программы

Выберите режим работы:

a) Консоль

b) Файл

c) Рандомная генерация

b

Введите путь до файла:

main/example_files/test.txt

Проверяю достаточные условия...

Достаточные условия для уравнения выполняются! Решаю...

Количество итераций: 6

Вектор погрешностей: (0.0019320000000000448, 0.0024600000000001288, 0.0030840000000000867)

Решение: (0.999568, 0.99946, 0.9993159999999999)

Система успешно решена!

Расчетные формулы.

Метод простых итераций состоит из следующих этапов:

Проверка матрицы на соответствие условию диагонального преобладания

В случае, если условие преобладания не соблюдается, попытаться переставить строки/столбцы таким образом, чтобы условие выполнялось

Выражение неизвестных x из каждого уравнения СЛАУ

$$\begin{cases} x_1 = -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \cdots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\ x_2 = -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \cdots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\ \cdots\ \cdots \\ x_n = -\frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \cdots - \frac{a_{n-1n-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}} \end{cases} \quad (6)$$

1. Обозначим:

$$c_{ij} = \begin{cases} 0, & \text{при } i = j \\ -\frac{a_{ij}}{a_{ii}}, & \text{при } i \neq j \end{cases}$$

$$d_i = \frac{b_i}{a_{ii}} \quad i = 1, 2, \dots, n$$

2. Тогда получим

$$\begin{cases} x_1 = c_{11}x_1 + c_{12}x_2 + \dots + c_{1n}x_n + d_1 \\ x_2 = c_{21}x_1 + c_{22}x_2 + \dots + c_{2n}x_n + d_2 \\ \cdots\ \cdots \\ x_n = c_{n1}x_1 + c_{n2}x_2 + \dots + c_{nn}x_n + d_n \end{cases}$$

Или в векторно-матричном виде: $\boldsymbol{x = Cx + D}$, где x – вектор неизвестных, C – матрица коэффициентов преобразованной системы размерности $n*n$, D – вектор правых частей преобразованной системы.

3. Представим систему в сокращенном виде:

$$\boldsymbol{x_i = \sum_{j=1}^{n} c_{ij}x_j + d_i}\ , \qquad i = 1, 2, \dots, n$$

$$c_{ij} = \begin{cases} 0, & \text{при } i = j \\ -\frac{a_{ij}}{a_{ii}}, & \text{при } i \neq j \end{cases} \qquad d_i = \frac{b_i}{a_{ii}} \quad i = 1, 2, \dots, n$$

4. За начальное приближение выберем вектор свободных членов x0 = D:

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.

5. Достаточным условием сходимости итерационного процесса является выполнение условие преобладания диагональных элементов, при этом хотя бы одно из неравенств должно быть строгим:

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

6. Достаточным условием сходимости итерационного метода к решению системы при любом начальном векторе является требование к норме матрицы ||C||:

$$\|C\| < 1$$

$$\|C\| = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |c_{ij}| < 1$$

$$\|C\| = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |c_{ij}| < 1$$

$$\|C\| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^2} < 1$$

Вывод

В ходе выполнения лабораторной работы были изучены численные прямые и итерационные методы решения СЛАУ, а также при помощи метода простых итераций была реализована программа на языке программирования Python. Были проанализированы условия возможности применения тех или иных численных методов решения СЛАУ, их достоинства и недостатки. Была проделана работа по нахождению оптимального способа проведения точных вычислений и визуального представления числовых данных.