МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное

автономное образовательное учреждение высшего образования

«**Национальный исследовательский университет ИТМО**»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

«Вычислительная математика»

Вариант 4

Выполнила:

Студент группы Р3218

Горло Евгений

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:

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).

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.

```
# Метод простых итераций
def simple_iterations(c, d, eps):
    current_d = [0]*len(d)
    last_d = d.copy()
    count_iter = 0
    while count_iter < 25:
        count_iter += 1
        for i in range(len(c)):
            x_i = 0
            for j in range(len(c[i])):
                x_i += c[i][j]*last_d[j]
            x_i += d[i]
            current_d[i] = x_i
        print(f'Итерация #{count_iter}\nПриближение: ')
        print(current_d)
        if absolute_deviations_check(last_d, current_d, eps):
            break
        last_d = current_d.copy()
    return current_d, count_iter
```

Исходный код программы.

Репозиторий на GitHub: https://github.com/Djerden/computation-math/tree/main/lab1

Расчетные формулы.

Метод простых итераций состоит из следующих этапов:

1. Проверка матрицы на соответствие условию диагонального преобладания
2. В случае, если условие преобладания не соблюдается, попытаться переставить строки/столбцы таким образом, чтобы условие выполнялось
3. Выражение неизвестных x из каждого уравнения СЛАУ

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

4. Обозначим:

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

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

5. Тогда получим

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

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.

6. Представим систему в сокращенном виде:

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

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

7. За начальное приближение выберем вектор свободных членов x0 = D

Рабочая формула метода простой итерации:

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

где k – номер итерации.

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

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

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

$$\|C\| < 1$$

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

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

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

Примеры вывода программы.

--Чтение из файла--

Решение Слау методом простых итераций

Справка по командам:

Считать матрицу с консоли - "1"

Считать матрицу из файла - "2"

Сгенерировать случайную матрицу - "3"

Выйти из программы - "4"

Введите команду: 2

--Чтение из файла--

Первым значением должна подаваться точность (epsilon)

Максимальная размерность матрицы <= 20

Матрица в файле должна иметь следующий вид:

a11 a12 a13 ... a1n | b1

a21 a22 a23 ... a2n | b2

... ... | ...

an1 an2 an3 ... ann | bn

Введите абсолютный путь до файла: C:\Users\Number_One\Documents\VsCode\computational-math\lab1\test1.txt

Получена матрица и вектор:

Матрица:

[[2.0, 2.0, 10.0]

[10.0, 1.0, 1.0]

[2.0, 10.0, 1.0]]

Вектор:

[14.0, 12.0, 13.0]

Проверка матрицы на диагональное преобладание

Диагональный элемент = 2.0, сумма других элементов = 12.0

Матрица не соответствует условию диагонального преобладания. Попытаемся привести...

Матрица приведена к диагональному преобладанию

Матрица:

[[10.0, 1.0, 1.0]

[2.0, 10.0, 1.0]

[2.0, 2.0, 10.0]]

Вектор:

[12.0, 13.0, 14.0]

Получаем коэффициенты

Матрица:

[[0, -0.1, -0.1]

[-0.2, 0, -0.1]

[-0.2, -0.2, 0]]

Вектор:

[1.2, 1.3, 1.4]

||C|| < 1, удовлетворяет условию нормы преобразования

Итерация #1

Приближение:

[0.9299999999999999, 0.92, 0.8999999999999999]

Критерий по абсоютным отклонениям

max = 0.5 > 0.01

Итерация #2

Приближение:

[1.018, 1.024, 1.0299999999999998]

Критерий по абсоютным отклонениям

max = 0.1299999999999999 > 0.01

Итерация #3

Приближение:

[0.9945999999999999, 0.9934000000000001, 0.9915999999999999]

Критерий по абсоютным отклонениям

max = 0.03839999999999988 > 0.01

Итерация #4

Приближение:

[1.0015, 1.0019200000000001, 1.0024]

Критерий по абсоютным отклонениям

max = 0.010800000000000032 > 0.01

Итерация #5

Приближение:

[0.999568, 0.99946, 0.9993159999999999]

Критерий по абсоютным отклонениям

max = 0.0030840000000000867 < 0.01

Заканчиваем вычисления...

Ответ: [0.999568, 0.99946, 0.9993159999999999]

Количество итераций: 5

--Генерирование случайной матрицы--

Справка по командам:

Считать матрицу с консоли - "1"

Считать матрицу из файла - "2"

Сгенерировать случайную матрицу - "3"

Выйти из программы - "4"

Введите команду: 3

--Генерирование случайной матрицы--

Введите точность (epsilon): 0.01

Введите размерность матрицы (n <= 20): 3

Получена матрица и вектор:

Матрица:

[[13.856409688509741, 3.0, -4.0]

[1.0, 12.27496299598409, 10.0]

[-4.0, -3.0, 13.0702318866667]]

Вектор:

[2.0, 1.0, 2.0]

Проверка матрицы на диагональное преобладание

Матрица соблюдает условие диагонального преобладания

Получаем коэффициенты

Матрица:

[[0, -0.21650630050926636, 0.2886750673456885]

[-0.08146664070003003, 0, -0.8146664070003002]

[0.3060389467214051, 0.2295292100410538, 0]]

Вектор:

[0.14433753367284424, 0.08146664070003003, 0.15301947336070254]

||C|| < 1, удовлетворяет условию нормы преобразования

Итерация #1

Приближение:

[0.17087239945756572, -0.05495187785906576, 0.21589135382288138]

Критерий по абсоютным отклонениям

max = 0.13641851855909579 > 0.01

Итерация #2

Приближение:

[0.21855741255831979, -0.10833319319344879, 0.1927000213991913]

Критерий по абсоютным отклонениям

max = 0.053381315334383034 > 0.01

Итерация #3

Приближение:

[0.22342004420844064, -0.09332473156334699, 0.19504092144328916]

Критерий по абсоютным отклонениям

max = 0.015008461630101805 > 0.01

Итерация #4

Приближение:

[0.22084637718245148, -0.09562792645691602, 0.19997395645367505]

Критерий по абсоютным отклонениям

max = 0.004933035010385889 < 0.01

Заканчиваем вычисления...

Ответ: [0.22084637718245148, -0.09562792645691602, 0.19997395645367505]

Количество итераций: 4

Вывод:

Метод Гаусса

Достоинства:

1. Гарантирует нахождение точного решения системы линейных уравнений после конечного числа шагов.
2. Эффективен для любых систем линейных уравнений.
3. Подходит для систем с любым порядком сложности.

Недостатки:

1. Требует значительных вычислительных ресурсов, особенно для больших систем.
2. Сложен в реализации, особенно при наличии необходимости выбора главного элемента для устойчивости.
3. Восприимчив к ошибкам округления, что может влиять на точность при работе с числами плавающей точки.

Метод простых итераций

Достоинства:

1. Простота реализации.
2. Устойчивость к ошибкам округления, особенно при подходящем выборе параметра релаксации.
3. Подходит для широкого спектра линейных и нелинейных уравнений.

Недостатки:

1. Сходимость не гарантирована для всех систем уравнений.
2. Может требовать много итераций для достижения приемлемой точности.
3. Необходимость в тщательном подборе итерационного параметра для обеспечения сходимости.

Метод Гаусса-Зейделя

Достоинства:

1. Обычно быстрее сходится по сравнению с методом простых итераций благодаря использованию уже обновлённых значений переменных.
2. Прост в реализации и модификации для различных типов задач.
3. Эффективен для диагонально доминантных матриц.

Недостатки:

1. Сходимость не гарантирована для всех систем.
2. Может быть неэффективен для некоторых типов матриц.

3. Также требует подбора релаксационного параметра в некоторых случаях.

Сравнительная характеристика

- **Простота реализации**: Метод простых итераций и метод Гаусса-Зейделя считаются более простыми в реализации по сравнению с методом Гаусса, который требует выполнения элементарных преобразований и обратного хода.
- **Скорость сходимости**: Метод Гаусса обеспечивает нахождение решения за конечное число шагов, что делает его наиболее быстрым. Метод Гаусса-Зейделя обычно сходится быстрее, чем метод простых итераций.
- **Универсальность и точность**: Метод Гаусса гарантирует точное решение для всех систем линейных уравнений, в то время как методы итераций могут не сходиться в зависимости от характеристик системы.
- **Требования к ресурсам**: Метод Гаусса требует больше вычислительных ресурсов, особенно для больших систем, в то время как итерационные методы могут быть более легковесными в плане вычислений, но требуют большего количества итераций.

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.