МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

Отчет
Вариант №4

по дисциплине
«Вычислительная математика»

Направление подготовки
02.03.01 «Математика и компьютерные науки»

Выполнила студентка
группы Б9122-02.03.01сцт
Винницкая Д. ______________________
(ФИО) *(подпись)*

« 21 » ноябрь 20 24 г.

г. Владивосток
2024

1 Цель работы

Целью данной лабораторной работы является изучение и реализация метода квадратного корня для решения системы линейных алгебраических уравнений (СЛАУ) вида $Ax = b$, где матрица A является симметричной и положительно определенной.

Основная задача заключается в вычислении вектора неизвестных x с использованием матричного разложения, что позволяет решить систему методом поэтапного вычисления промежуточных переменных.

2 Постановка задачи

1. Решить систему линейных алгебраических уравнений вида $Ax = b$, используя метод квадратного корня.
2. Входными данными являются матрица A и вектор b. Матрица A должна быть симметричной и положительно определенной.
3. Реализовать алгоритм
4. Проверить корректность работы алгоритма на тестовых примерах.

3 Теоретические сведения

Метод квадратного корня (или метод Холецкого) является численным методом решения систем линейных алгебраических уравнений, применяемым в случае, когда матрица A симметрична и положительно определена. Основная идея метода заключается в представлении матрицы A в виде произведения $A = U^T U$, где U – верхнетреугольная матрица. Такое разложение позволяет упростить решение системы $Ax = b$ до двух последовательных этапов:

1. Решение системы $U^T y = b$ для вектора y.
2. Решение системы $Ux = y$ для искомого вектора x.

Для нахождения элементов матрицы U используются следующие формулы:

- Первый элемент диагонали вычисляется как $u_{11} = \sqrt{a_{11}}$.
- Для остальных элементов первой строки: $u_{1j} = \frac{a_{1j}}{u_{11}}, \quad j > 1$.
- Для диагональных элементов: $u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2}, \quad i > 1$.
- Для недиагональных элементов: $u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}}, \quad i < j$.

После нахождения всех элементов матрицы U можно решить исходную систему, выполняя два этапа: сначала решается система с нижнетреугольной матрицей U^T, а затем с верхнетреугольной матрицей U. Это позволяет эффективно находить решение системы $Ax = b$.

Код программы

```
import numpy as np

def sqrt_method(A, b):
    n = A.shape[0]
    U = np.zeros_like(A, dtype=float)

    for i in range(n):
        U[i, i] = np.sqrt(A[i, i] - np.sum(U[:i, i] ** 2))
        for j in range(i + 1, n):
            U[i, j] = (A[i, j] - np.sum(U[:i, i] * U[:i, j])) / U[i, i]

    y = np.zeros(n)
    for i in range(n):
        y[i] = (b[i] - np.sum(U[:i, i] * y[:i])) / U[i, i]

    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = (y[i] - np.sum(U[i, i + 1:] * x[i + 1:])) / U[i, i]

    return x

A = np.array([
    [1, 2, 4],
    [2, 13, 23],
    [4, 23, 77]
], dtype=float)

b = np.array([10, 50, 150], dtype=float)
x = sqrt_method(A, b)
print("Решение x*:", x)

A1 = np.array([
    [5.8, 0.3, -0.2],
    [0.3, 4.0, -0.7],
    [-0.2, -0.7, 6.7]
], dtype=float)

b1 = np.array([3.1, -1.7, 1.1], dtype=float)
x1 = sqrt_method(A1, b1)
print("Решение дляСЛАУ I x*:", x1)

A2 = np.array([
    [4.12, 0.42, 1.34, 0.88],
    [0.42, 3.95, 1.87, 0.43],
    [1.34, 1.87, 3.20, 0.31],
    [0.88, 0.43, 0.31, 5.17]
], dtype=float)

b2 = np.array([11.17, 0.115, 9.909, 9.349], dtype=float)
x2 = sqrt_method(A2, b2)
print("Решение дляСЛАУ II x*:", x2)
```

Листинг 1: Реализация метода квадратного корня

4 Описание

Функция $\mathrm{sqrt\_method(A, b)}$ решает систему линейных уравнений вида $Ax = b$ методом квадратного корня. Давайте разберемся, как она работает.

Сначала функция создает пустую матрицу U такого же размера, как и входная матрица A. Матрица U будет верхнетреугольной и используется для хранения элементов разложения $A = U^T U$.

Затем в цикле функция заполняет элементы матрицы U. Для диагональных элементов вычисляется квадратный корень, а для остальных элементов используется формула, которая учитывает предыдущие значения. Здесь важно, что матрица A должна быть симметричной и положительно определенной, иначе метод не сработает корректно.

После этого функция решает вспомогательную систему уравнений $U^T y = b$. Для этого используется прямой ход — постепенно вычисляются значения вектора y.

Затем решается система $Ux = y$, используя обратный ход, чтобы найти значения вектора x, который и является решением исходной системы. Функция возвращает вектор x, который является решением системы.

Вот что делает каждая часть функции:

- $\mathrm{U = np.zero\_like(A, dtype{=}float)}$ — создаем пустую матрицу U того же размера, что и A.
- $\mathrm{for\ i\ in\ range(n)}$ — заполняем матрицу U по диагонали и выше.
- $\mathrm{y = np.zeros(n)}$ — создаем пустой вектор y для хранения промежуточных значений.
- $\mathrm{for\ i\ in\ range(n)}$ — решаем систему для вектора y.
- $\mathrm{x = np.zeros(n)}$ — создаем пустой вектор x для хранения решения.
- $\mathrm{for\ i\ in\ range(n\ \text{-}\ 1,\ \text{-}1,\ \text{-}1)}$ — решаем систему для вектора x с помощью обратного хода.

Основная система для варианта VI

$$\textbf{Решение } \mathbf{x}^* = \begin{pmatrix} 2.22222222 \\ 0.55555556 \\ 1.66666667 \end{pmatrix}$$

Сравнение с:

$$x^* = \begin{pmatrix} 2.22 \\ 0.55 \\ 1.67 \end{pmatrix}$$

Вывод:

Значения из изображения и точные данные совпадают с округлением до двух десятичных знаков.

Различия появляются из-за более высокой точности представления во втором случае. Для задач, где требуется высокая точность, рекомендуется использовать все доступные знаки после запятой.

Для визуального представления результатов округление до двух знаков считается приемлемым.

$$\textbf{Решение для СЛАУ I } \mathbf{x}^* = \begin{pmatrix} 0.56206926 \\ -0.44359823 \\ 0.13461121 \end{pmatrix}$$

$$\textbf{Решение для СЛАУ II } \mathbf{x}^* = \begin{pmatrix} 1.45653917 \\ -1.93309996 \\ 3.46970779 \\ 1.51312749 \end{pmatrix}$$

5 Заключение

В данной лабораторной работе была реализована и протестирована программа для решения систем линейных алгебраических уравнений (СЛАУ) методом квадратного корня. По итогам выполнения работы можно сделать следующие выводы:

1. Реализована функция $\mathrm{sqrt\_method}$, которая позволяет решать СЛАУ, преобразуя матрицу A в верхнетреугольную матрицу U с использованием метода квадратного корня.

2. Программа успешно протестирована на трёх различных системах уравнений:

   - Для матрицы $A = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{pmatrix}$ и вектора $b = \begin{pmatrix} 10 \\ 50 \\ 150 \end{pmatrix}$ получено решение $x^* = \begin{pmatrix} 2.222 \\ 0.556 \\ 1.667 \end{pmatrix}$.

   - Для матрицы $A_1 = \begin{pmatrix} 5.8 & 0.3 & -0.2 \\ 0.3 & 4.0 & -0.7 \\ -0.2 & -0.7 & 6.7 \end{pmatrix}$ и вектора $b_1 = \begin{pmatrix} 3.1 \\ -1.7 \\ 1.1 \end{pmatrix}$ получено решение $x_1^* = \begin{pmatrix} 0.562 \\ -0.444 \\ 0.135 \end{pmatrix}$.

   - Для матрицы $A_2 = \begin{pmatrix} 4.12 & 0.42 & 1.34 & 0.88 \\ 0.42 & 3.95 & 1.87 & 0.43 \\ 1.34 & 1.87 & 3.20 & 0.31 \\ 0.88 & 0.43 & 0.31 & 5.17 \end{pmatrix}$ и вектора $b_2 = \begin{pmatrix} 11.17 \\ 0.115 \\ 9.909 \\ 9.349 \end{pmatrix}$ получено решение $x_2^* = \begin{pmatrix} 1.457 \\ -1.933 \\ 3.470 \\ 1.513 \end{pmatrix}$.

3. Результаты расчетов демонстрируют корректность работы алгоритма: для каждой из систем решений x^*, x_1^* и x_2^* были получены значения, которые удовлетворяют начальным данным.

4. Реализация метода квадратного корня подтверждает его эффективность для решения систем с симметричными и положительно определёнными матрицами.

5. Программа написана с использованием библиотеки numpy, что обеспечивает высокую производительность и лёгкость работы с массивами.

Вывод: Метод квадратного корня является надёжным инструментом для решения СЛАУ. Алгоритм хорошо справляется с симметричными матрицами и может быть использован для численных расчётов в прикладных задачах.