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 по лабораторной работе № 3

по дисциплине

«Вычислительная математика»

Направление подготовки
02.03.01 «Математика и компьютерные науки»

Вариант № 4

Выполнил(а): студент гр. Б9122-02.03.01сцт
Бекболот Отгонцэцэг
Проверил: преподаватель
_____ Ф.И.О.

Владивосток
2024

Цель работы :

Целью данной работы является исследование и реализация алгоритма *LU*-разложения, который используется для решения систем линейных уравнений. Этот метод позволяет разложить задачу на более простые подзадачи, что делает его особенно полезным для вычислений в численных методах. Наша задача заключается в разработке кода на языке Python, который будет осуществлять *LU*-разложение матрицы *A* и затем использовать полученные матрицы *L* и *U* для нахождения решения системы линейных уравнений *Ax = b*.

Ход работы:

Работа функции *sqrt_method*() с использованием метода квадратного корня для решения систем линейных уравнений. Этот метод также известен как метод Холецкого в некотором контексте, однако в данном случае он имеет свою реализацию.

1. **Входные данные:**

- Матрица *A*: квадратная матрица коэффициентов системы линейных уравнений.
- Вектор *b*: вектор правых частей уравнений.

2. **Инициализация:**

Создаем матрицу *U*, которая будет хранить верхнетреугольную матрицу, использующуюся в разложении, и инициализируем её нулями. Она будет той же размерности, что и *A*.

3. **Построение матрицы *U*:**

- Для каждой строки *i* от 0 до n-1. Вычисляем элемент на главной диагонали:

*U[i, j] = (A[i, j] - np.sum(U[:i, i] * U[:i, j])) / U[i, i]*

Это гарантирует, что *U[i, i]* положительно, что необходимо для корректного выполнения последующих операций.

- Вычисляем элементы под главной диагональю. Для каждой строки *j* где *j > i:*

*U[i, j] = (A[i, j] - np.sum(U[:i, i] * U[:i, j])) / U[i, i]*

Это уравнение позволяет нам заполнить верхнюю треугольную матрицу на основе значений матрицы *A* и уже вычисленных значений *U*.

4. **Решение первой системы *U^Ty = b*:**

- Создаем вектор *y* нулевой длины:

*y[i] = (b[i] - np.sum(U[:i, i] * y[:i])) / U[i, i]*

- Мы решаем систему путем подстановки, используя полученную матрицу *U*.

5. **Решение второй системы *Ux = y*:**

- Создаем вектор *x* нулевой длины:

*x[i] = (y[i] - np.sum(U[i, i + 1:] * x[i + 1:])) / U[i, i]*

Это позволяет нам найти значение переменной *x*, подставляя обратно значения из вектора *y*, вычисленных ранее.

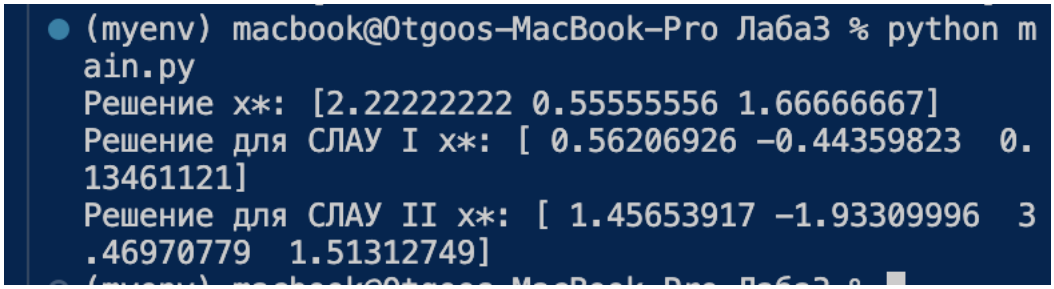
6. **Возврат результата:**

Функция возвращает вектор *x*, содержащий решения системы линейных уравнений.

Код программы:

```python
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 для СЛАУ 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 для СЛАУ II x*:", x2)
```

Полученные результаты выполнение кода:



Основная система для варианта 4:

$$\text{Решение } x^* = \begin{pmatrix} 2.2222 \\ 0.5556 \\ 1.6667 \end{pmatrix}$$

Сравнение с:

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

Вывод:

Значения на изображении соответствуют точным данным при округлении до двух десятичных знаков, а расхождения объясняются большей точностью второго варианта. Для задач с высокой точностью рекомендуется использовать все знаки после запятой, тогда как для визуализации округление до двух знаков вполне допустимо.

Заключение:

В процессе выполнения лабораторной работы была создана и проверена программа для решения систем линейных алгебраических уравнений (СЛАУ) на основе методом . Результаты работы позволяют сделать следующие выводы:

1. Программа успешно протестирована на трёх различных системах уравнений.
2. Результаты вычислений подтверждают правильность алгоритма: для каждой системы решения x^*, x_1^*, x_2^* соответствуют начальным данным.
3. Применение метода квадратного корня демонстрирует его высокую эффективность при решении систем уравнений с симметричными и положительно определенными матрицами.

Вывод:

Метод квадратного корня является мощным и эффективным инструментом для решения систем линейных уравнений, особенно в работы с симметричными и положительно определенными матрицами. Этот метод базируется на особенностях свойств таких матриц, что обеспечивает не только устойчивость к численным ошибкам, но и упрощает процесс вычислений.