Оглавление

[Задача 3](#_Toc53163652)

[Теоретическая часть 3](#_Toc53163653)

[LU-разложение 3](#_Toc53163654)

[Метод вращений 3](#_Toc53163655)

[Практическая часть 4](#_Toc53163656)

[LU-разложение 4](#_Toc53163657)

[Метод вращений 4](#_Toc53163658)

**Лабораторная работа №3.**

**«Прямые методы решения СЛАУ»**

***Кущ Артем, БПМ-18-1***

# Задача

Решить СЛАУ вида:

с использованием двух методов: LU-разложение и метод вращений.

# Теоретическая часть

## LU-разложение

Необходимо применить LU-разложение матрицы A и преобразовать уравнение к системе равенств:

Далее система элементарно решается, путем домножения каждого уравнения на обратную матрицу.

## Метод вращений

Метод вращений, как и метод Гаусса, состоит из прямого хода и обратного. Цель прямого хода - последовательное исключение неизвестных и приведение матрицы к треугольному виду.

Условие, обеспечивающие исключение из второго уравнения:

Условие нормировки, коэффициенты рассчитываются по следующим формулам:

Коэффициенты возможно интерпретировать как синус и косинус некоторого угла. Таким образом, каждый шаг прямого хода будет поворачивать в пространстве, определяемым индексами.

В итоге, должна получиться система вида:



После успешного приведения матрицы к треугольному виду применяется обратный ход, аналогичный методу Гаусса. Для этого необходимо применить следующую формулу:

# Практическая часть

def find\_x(A, b):

# LU

def create\_LU(A):

L, U = LU(A, permute\_l=True)

assert np.allclose(L @ U, A), ValueError("Not correct LU")

return L, U

L, U = create\_LU(A)

# обратные матрицы

L\_inv = np.linalg.inv(L)

U\_inv = np.linalg.inv(U)

# новый вектор y

y = L\_inv @ b

# искомый вектор x

x = U\_inv @ y

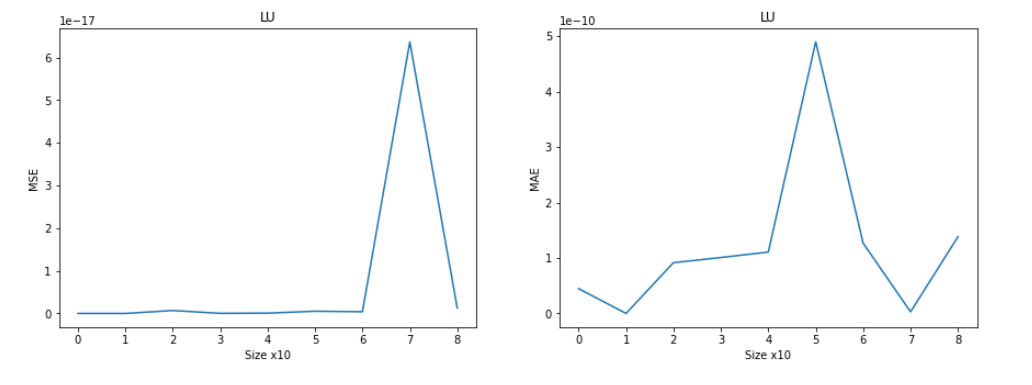
mse = ((A @ x - b)\*\*2).mean(axis=-1)

mae = np.abs((A @ x - b).mean(axis=-1))

return x, mse, mae

## LU-разложение

Тестировался алгоритм на матрицах размера от 2х2 до 99х99 и в диапазоне значений от 0 до 100000. Ошибка на всех итерациях достаточно мала (< -10 порядка), чаще всего ошибка меньше машинной точности.

Ошибка рассчитывалась на исходном векторе b и векторе, найденном по формуле Ax = b.

## Метод вращений

def solve\_with\_rotation(A, b):

m = np.concatenate((A, np.expand\_dims(b, 1)), axis=-1)

n = m.shape[0]

# forward trace

for i in range(n-1):

for j in range(i + 1, n):

c = m[i, i] / (m[i, i]\*\*2 + m[j, i]\*\*2) \*\* .5

s = m[j, i] / (m[i, i]\*\*2 + m[j, i]\*\*2) \*\* .5

tmp1 = m[i, :] \* c + m[j, :] \* s

tmp2 = m[i, :] \* -s + m[j, :] \* c

m[i, :] = tmp1

m[j, :] = tmp2

# find x, backward trace

x = np.array([0.0 for i in range(n)])

for k in range(n - 1, -1, -1):

x[k] = (m[k, -1] - m[k, k:n] @ x[k:n]) / m[k, k]

mse = ((A @ x - b)\*\*2).mean(axis=-1)

mae = np.abs((A @ x - b).mean(axis=-1))

return x, mse, mae

Были проведены аналогичные тесты. Как и ожидалось, ошибка также очень мала.