

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ     
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**ДАЛЬНЕВОСТОЧНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**Институт математики и компьютерных технологий**

**Департамент математического и компьютерного моделирования**

**ОТЧЕТ**

**по лабораторной работе** **№ 1**

«Плохо обусловленная СЛАУ»

Выполнил: студент гр. Б9120-02.03.01сцт

Пограничный К. О.

Проверил: преподаватель

Кузнецов К. С.

Владивосток 2023

**Постановка задачи**

Даны матрицы А, вектор b:

A = b =

Решение системы Ax=b единственно: x = (0, 0, ..., 1) \*.

Нужно решить систему (A+ɛN) x = b

A = N = b =

Где:

ɛ — это малый параметр, который можно брать в широком диапазоне от 10–3 до 10–6. Систему следует решать при увеличивающейся размерности матрицы и вектора b.

**Метод решения**

Задача была выполнена на языке программирования Python с использованием средств библиотеки NumPy. Использовал отдельные функции для создания матриц A и N, а также вектора b. Была написана функция для нахождения точных значений вектора x и числа обусловленности матриц для каждого эпсилон.

**Алгоритм решения**

1. Создание матриц А и N. Использовал встроенные функции библиотеки NumPy: add, triu, ones, identity
2. Вектора создал с помощью функции ones
3. С помощью написанной функции “solve”, был выполнен подсчёт обусловленности матрицы А (использовалась встроенная функция linalg.cond() библиотеки NumPy) и написано решение. Обусловленность квадратной матрицы:

k(A) = ||A||·||A^-1||

**Использованные функции**

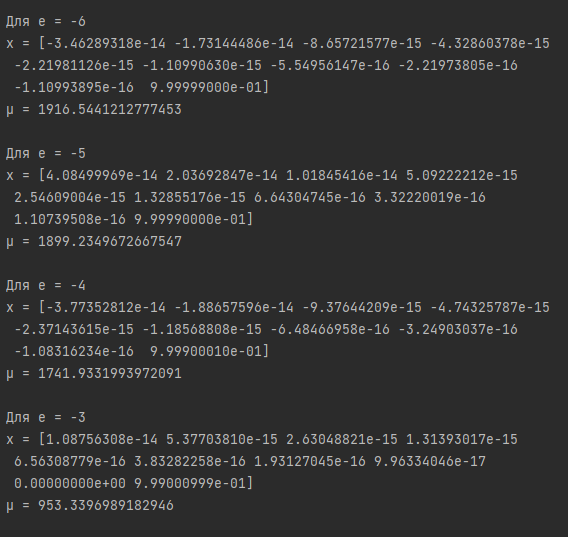
1. Функция numpy.ones() - возвращает новый массив заданной формы и типа данных, где значение элемента равно 1.
2. Функция numpy.triu() - используется для создания верхней треугольной матрицы.
3. Функция numpy.identity() - используется для создания квадратной матрицы из нулей, с единицами на главной диагонали.
4. Функция numpy.linalg.cond() - находит число обусловленности матрицы.
5. Функция numpy.linalg.tensorsolve() - используется для разрешения тензорного уравнения Ax=b относительно x.

**Описание тестов**

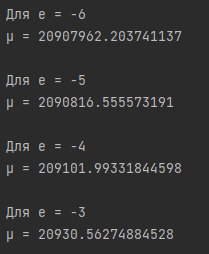
1. Размерность матрицы = 10, = , …,
2. Размерность матрицы = 100, = , …,

**Результаты тестов**

Отработка для первого теста:



Отработка для второго теста (из-за достаточно большой размерности, во избежание нагромождения опциональной информации, опущу необходимость демонстрации значений вектора x для каждого из значений эпсилон):



**Вывод**

В ходе лабораторной работы выяснилось, что размерность матрицы влияет на обусловленность самой матрицы. Влияние параметра на обусловленность матрицы прямо пропорционально её размерности. При малой размерности влияние почти незаметно, но с увеличением размерности, влияние также повышается.

**Приложение**

Листинг программы:

import numpy as np  
  
  
def initial\_N(dim):  
 N = np.add(-np.triu(np.ones((dim, dim), dtype=int), 1), np.tri(dim, dim, 0, dtype=int))  
 return N  
  
  
def initial\_A(dim):  
 A = np.add(-np.triu(np.ones((dim, dim), dtype=int), 1), np.identity(dim, dtype=int))  
 return A  
  
  
def initial\_b(dim):  
 b = -np.ones(dim, dtype=int)  
 b[dim - 1] = 1  
 return b  
  
  
def solve(e, N, A, b):  
 x = np.linalg.tensorsolve(np.add(A, (pow(10, e) \* N)), b)  
 m = np.linalg.cond(A + pow(10, e) \* N)  
 print("\nДля e =", e)  
 print("x =", x)  
 print("μ =", m)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 dim = int(input("Введите размерность матрицы: "))  
  
 N = initial\_N(dim)  
 print("N:\n", N)  
 A = initial\_A(dim)  
 print("\nA:\n", A)  
 b = initial\_b(dim)  
 print("\nb:\n", b)  
  
 for e in range(-6, -2): solve(e, N, A, b)