Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 2

Дисциплина: Вычислительная математика

Выполнил студент гр. 3530901/10003 _____Рубцов Е.А.
(подпись)
Принял старший преподаватель _____ Цыган В.Н.
(подпись)
“_” _____ 2023 г.

Санкт-Петербург
2023

Содержание

Задание:

Вариант №29

Составить процедуру вычисления по заданной матрице A(N, N) матрицы $R = A^{-1}\text{A} - E$ и ее нормы $||R|| = \max_k \sum_j^n |R_{jk}|$.

Построить три матрицы A при $x_k = \left(1 + \frac{cos(k)}{sin^2(k)}\right), k = 1, \dots, 4$; и $x_5 = 1 + \frac{cos(1)}{sin^2(1+\epsilon)}$ для трех значений ε = 0.001, 0.00001, 0.000001 и N = 5.

$$\text{A} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ X_1 & X_2 & \dots & X_N \\ \dots & \dots & \dots & \dots \\ X_1^{N-1} & X_2^{N-1} & \dots & X_N^{N-1} \end{pmatrix}$$

Исследовать зависимость погрешности вычисления $||R||$ от ε. Предусмотреть вычисление и вывод числа обусловленности *cond* матрицы A.

Инструменты:

Для выполнения поставленного задания был выбран язык python версии 3.10. Были использованы следующие библиотеки:

1. NumPy – эта библиотека предостовляет функции для более быстрого выполнения расчетов
2. SciPy – эта библиотека содержит необходимые функции для выполнения вычислений

Ход выполнения работы:

1. Получение обратной матрицы

Алгоритм получения обратной матрицы с использованием библиотеки scipy и аналогов функций DECOMP и SOLVE:

1. Розложить исходную матрицу A, размерности N, на матрицы L и U – нижнетреугольную и верхнетреугольную матрицу соответственно
2. Из уравнения $LZ_i = E_i$, где E_i i-й столбец единичной матрицы размерности N×N, получаем значение Z_i – столбцевая матрица размерности N×1
3. Из уравнения $UX_i = Z_i$ получаем значение X – i-го столбца искомой обратной матрицы A^{-1}
4. Повторяем шаги 2 и 3 N раз для получения всех столбцов обратной матрицы

Реализация данного алгоритма в коде на языке python:

```
def get_inverse(A: np.matrix):
    A_inv = np.zeros(A.shape)
    E = np.identity(A.shape[0])
    P, L, U = lu(A)

    for i in range(0, A.shape[0]):
        Ei = E[:, i]
```

```
        Zi = solve(L, Ei)
        X = solve(U, Zi)
        A_inv[:, i] = X.flatten()

    return A_inv
```

2. *Получение матрицы R и её нормы:*

```
    def get_R_matrix(A: np.matrix):
        A_inv = get_inverse(A)
        E = np.identity(A_inv.shape[0])
        return np.subtract(np.matmul(A_inv, A), E)

    def get_norm(A: np.matrix):
        return np.max([np.sum(np.abs(k)) for k in A], axis=None)
```

3. *Построение матриц и получение числа обусловленности:*

В рамках данной лабораторной работы, число обасловленности матрицы A вычисляется как $cond = ||A|| * |A^{-1}|$:

```
def get_cond(A: np.matrix):
    return get_norm(A) * get_norm(get_inverse(A))
```

Функция для построения матрицы по заданному числу ε:

```
def construct(eps):
    x_k = lambda k: 1 + np.cos(k) / np.power(np.sin(k), 2)
    x_eps = lambda eps: 1 + np.cos(1) / np.power(np.sin(1 + eps), 2)

    A = np.matrix([[np.power(x_k(k), N) for k in range(1, 5)] for N in range(0, 4)])

    new_row = [np.power(x_k(k), 4) for k in range(1, 5)]
    new_column = [[np.power(x_eps(eps), N)] for N in range(0, 5)]

    A = np.vstack((A, new_row))
    A = np.hstack((A, new_column))

    return A
```

Полученные результаты:

1. Для ε = 0.001:

```
                                Matrix:
 [[ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [ 1.76305972e+00  4.96691027e-01 -4.87112539e+01 -1.41237192e-01  1.76208152e+00]
 [ 3.10837958e+00  2.46701976e-01  2.37278626e+03  1.99479443e-02  3.10493127e+00]
 [ 5.48025885e+00  1.22534658e-01 -1.15581394e+05 -2.81739164e-03  5.47114201e+00]
 [ 9.66202364e+00  6.08618649e-02  5.63011462e+06  3.97920483e-04  9.64059821e+00]]

                                     R:
 [[-5.75942041e+03  1.05624481e+02 -4.87652186e+09  8.07533621e+00 -5.74344255e+03]
 [-1.36976884e+01 -4.32436956e-01 -1.84478899e+07 -3.05814239e-02 -1.36552255e+01]
 [ 4.15553441e-06  1.48569472e-08  2.98110603e+00  1.57253123e-07  4.14395631e-06]
 [ 4.89996429e+00  7.24041734e-02  9.81908051e+06 -1.10479864e+00  4.88331599e+00]
 [ 5.76898119e+03 -1.05767757e+02  4.88515062e+09 -8.08119349e+00  5.75297654e+03]]

Norm: 5630133.984277477
Conditionality: 8687641145.88789
```

2. Для ε = 0.00001:

```
                                Matrix:
 [[ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [ 1.76305972e+00  4.96691027e-01 -4.87112539e+01 -1.41237192e-01  1.76304992e+00]
 [ 3.10837958e+00  2.46701976e-01  2.37278626e+03  1.99479443e-02  3.10834503e+00]
 [ 5.48025885e+00  1.22534658e-01 -1.15581394e+05 -2.81739164e-03  5.48016747e+00]
 [ 9.66202364e+00  6.08618649e-02  5.63011462e+06  3.97920483e-04  9.66180884e+00]]

                                     R:
 [[-5.75150888e+05  1.05447742e+04 -4.87034981e+11  8.05679363e+02 -5.75134889e+05]
 [-1.36872596e+01 -4.32628158e-01 -1.84390588e+07 -3.05960327e-02 -1.36868340e+01]
 [ 4.15769602e-06  1.48173165e-08  2.98293648e+00  1.57250095e-07  4.15757987e-06]
 [ 4.89685783e+00  7.24611269e-02  9.81644997e+06 -1.10479429e+00  4.89669095e+00]
 [ 5.75160441e+05 -1.05449173e+04  4.87043603e+11 -8.05685210e+02  5.75144442e+05]]

Norm: 5630134.005488109
Conditionality: 866147309711.7546
```

3. Для ε = 0.000001:

```
                                Matrix:
 [[ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [ 1.76305972e+00  4.96691027e-01 -4.87112539e+01 -1.41237192e-01  1.76305874e+00]
 [ 3.10837958e+00  2.46701976e-01  2.37278626e+03  1.99479443e-02  3.10837613e+00]
 [ 5.48025885e+00  1.22534658e-01 -1.15581394e+05 -2.81739164e-03  5.48024971e+00]
 [ 9.66202364e+00  6.08618649e-02  5.63011462e+06  3.97920483e-04  9.66200216e+00]]

                                     R:
 [[-5.75143700e+06  1.05446136e+05 -4.87029373e+12  8.05662512e+03 -5.75142100e+06]
 [-1.36871646e+01 -4.32629898e-01 -1.84389785e+07 -3.05961656e-02 -1.36871221e+01]
 [ 4.15771571e-06  1.48169556e-08  2.98295315e+00  1.57250067e-07  4.15770409e-06]
 [ 4.89682955e+00  7.24616454e-02  9.81642603e+06 -1.10479425e+00  4.89681286e+00]
 [ 5.75144655e+06 -1.05446279e+05  4.87030236e+12 -8.05663096e+03  5.75143055e+06]]

Norm: 5630134.005681429
Conditionality: 8661235368829.015
```

Как видно из полученных значений, при уменьшении числа ε, число обусловленности матрицы уменьшается, в то время как норма матрицы R увеличивается, однако учитывая размер этих величин, данные изменения можно считать незначительными.

Вывод:

В ходе выполнения данной лабораторной работы, мной были получены навыки работы с функциями DECOMP и SOLVE, а так же навыки алгоритмического получения обратной матрицы, значений нормы и обусловленности матрицы.

Приложение:

```
import numpy as np
from scipy.linalg import lu, solve


# n - размер матрицы
# Гарантировано генерирует инвертированную матрицу
def generate_random_matrix(n):
    m = np.random.rand(n, n)
    mx = np.sum(np.abs(m), axis=1)
    np.fill_diagonal(m, mx)

    return m


def get_inverse(A: np.matrix):
    A_inv = np.zeros(A.shape)
    E = np.identity(A.shape[0])
    P, L, U = lu(A)

    for i in range(0, A.shape[0]):
        Ei = E[:, i]

        Zi = solve(L, Ei)
        X = solve(U, Zi)
        A_inv[:, i] = X.flatten()

    return A_inv


def get_R_matrix(A: np.matrix):
    A_inv = get_inverse(A)
    E = np.identity(A_inv.shape[0])
    return np.subtract(np.matmul(A_inv, A), E)


def get_norm(A: np.matrix):
    return np.max([np.sum(np.abs(k)) for k in A], axis=None)


def get_cond(A: np.matrix):
    return get_norm(A) * get_norm(get_inverse(A))


def construct(eps):
    x_k = lambda k: 1 + np.cos(k) / np.power(np.sin(k), 2)
    x_eps = lambda eps: 1 + np.cos(1) / np.power(np.sin(1 + eps), 2)

    A = np.matrix([[np.power(x_k(k), N) for k in range(1, 5)] for N in range(0, 4)])

    new_row = [np.power(x_k(k), 4) for k in range(1, 5)]
    new_column = [[np.power(x_eps(eps), N)] for N in range(0, 5)]

    A = np.vstack((A, new_row))
    A = np.hstack((A, new_column))
```

```
    return A


def main():
    eps_arr = [0.001, 0.00001, 0.000001]
    A_arr = [construct(eps) for eps in eps_arr]

    np.set_printoptions(linewidth=150)

    for i in range(0, 3):
        print("Epsilon:", eps_arr[i])
        print("\t\t\t\t   Matrix:\n", A_arr[i], "\n")
        print("\t\t\t\t\tR:\n", get_R_matrix(A_arr[i]), "\n")
        print("Norm:", get_norm(A_arr[i]))
        print("Conditionality:", get_cond(A_arr[i]))
        print("\n", "--" * 40, "\n")


if __name__ == "__main__":
    main()
```

Листинг №1: файл matrix.py