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: 3
Инструменты: 3
Ход выполнения работы: 3
1. Получение обратной матрицы 3
2. Получение матрицы R и её нормы: 4
3. Построение матриц и получение числа обусловленности: 4
Полученные результаты: 5
Вывод: 6
Приложение: 7

Задание:

Вариант №29

Составить процедуру вычисления по заданной матрице A(N, N) матрицы $R = A^{-1}A - E$ 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.

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

Исследовать зависимость погрешности вычисления $||R||$ от ε. Предусмтореть 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 исхоную 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

Так как, в функции lu() для получения LU разложения выполняется перестановка столбцов матрицы A, то при выполнении вышеописанного алгоритма мы получим обратную матрицу, умноженную на некоторую матрицу перестановок P: A^{-1}*P. Для того, чтобы учесть это, необходимо умножить получившеюся матрицу A^{-1} на P^{-1}, для получения обратной матрицы перестановок, достаточно выполнить транспонирование этой матрицы.

Реализация данного алгоритма в коде на языке python:

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

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

Для того, чтобы получить LU разложение матрицы A, метод lu из библиотеки scipy так же выполняет перестановку столбцов матрицы A, а значит, что при получании обратной матрицы по вышеописанному алгоритму,

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.          1.          1.          1.          1.        ]
 [ 2.17534265  0.70614146  0.50251446  0.60472522  2.17255397]
 [ 4.73211564  0.49863577  0.25252078  0.36569259  4.71999074]
 [10.29397298  0.35210739  0.12689534  0.22114353 10.2544346 ]
 [22.39291846  0.24863763  0.06376674  0.13373107 22.27831257]]

                           inverse Matrix:
 [[  43.30786302 -239.06246634  466.84223372 -370.27812025   92.89617376]
 [  32.27933364 -147.31066333  221.25623435 -122.6098263    22.47603354]
 [  34.70866895 -138.47981133  186.64787951  -97.32209425   17.19847414]
 [ -65.69999507  284.22637724 -404.92966291  217.68687229  -39.1766065 ]
 [ -43.59587054  240.62656376 -469.81668468  372.5231685   -93.39407494]]

                           A^(-1) * A:
 [[ 1.00000000e+00 -2.53562161e-14  1.50476687e-14 -5.26986306e-14  2.21874180e-13]
 [-6.38194183e-13  1.00000000e+00 -2.49658102e-14 -7.41693159e-14 -4.32104926e-13]
 [ 2.71097844e-13  1.51504803e-14  1.00000000e+00  2.70629514e-15  3.37376095e-13]
 [ 7.08593071e-14  1.08173754e-14 -2.28101567e-14  1.00000000e+00 -1.75809524e-13]
 [-1.93923820e-13  1.89854384e-14 -3.64664725e-14  4.04505248e-14  1.00000000e+00]]

                                        R:
 [[-2.03170814e-13  4.21647043e-14  7.09849076e-14  1.64552101e-14  3.60027667e-13]
 [-2.63002459e-13  6.21724894e-15  1.92165658e-14 -1.46923808e-14 -5.65060412e-14]
 [ 1.68830008e-13  1.60483390e-14  2.70894418e-14 -2.68481336e-15  3.49609420e-13]
 [-3.38463626e-13 -2.64957985e-14 -5.43314899e-14  5.99520433e-15 -5.85946781e-13]
 [-1.93923820e-13 -3.78579805e-14 -6.48881819e-14 -1.63928941e-14 -1.06370468e-12]]

Norm: 45.11736647297931
Conditionality: 55041.218284029506
```

2. Для ε = 0.00001:

```
                                   Matrix:
 [[ 1.          1.          1.          1.          1.        ]
 [ 2.17534265  0.70614146  0.50251446  0.60472522  2.17531471]
 [ 4.73211564  0.49863577  0.25252078  0.36569259  4.73199411]
 [10.29397298  0.35210739  0.12689534  0.22114353 10.29357641]
 [22.39291846  0.24863763  0.06376674  0.13373107 22.39176824]]

                           inverse Matrix:
 [[ 4.32882121e+03 -2.38928704e+04  4.66502456e+04 -3.69896412e+04  9.27361787e+03]
 [ 3.22596185e+01 -1.47201846e+02  2.21043772e+02 -1.22441362e+02  2.24337985e+01]
 [ 3.46954195e+01 -1.38406681e+02  1.86505095e+02 -9.72088786e+01  1.71700902e+01]
 [-6.56678498e+01  2.84048952e+02 -4.04583245e+02  2.17412194e+02 -3.91077427e+01]
 [-4.32910840e+03  2.38944300e+04 -4.66532112e+04  3.69918792e+04 -9.27411402e+03]]

                           A^(-1) * A:
 [[ 1.00000000e+00 -6.83704076e-12  1.09740482e-12 -5.30681913e-12  1.80961405e-11]
 [ 1.33736132e-13  1.00000000e+00  3.80188350e-14  5.09224044e-15  4.11376099e-13]
 [ 8.93457209e-15 -3.26076167e-14  1.00000000e+00 -4.02640882e-14  1.69779454e-13]
 [ 9.33852501e-14  1.90103536e-14 -9.80163236e-15  1.00000000e+00 -3.54091674e-13]
 [ 2.17377096e-11  2.04555038e-12 -3.09616708e-12  3.39698280e-12  1.00000000e+00]]

                                        R:
 [[ 6.02855543e-11  8.91186062e-13  5.76086936e-12 -5.16090225e-13  8.79303601e-11]
 [ 5.42933272e-13  2.26485497e-14  4.04748268e-14  6.07347583e-15  7.06874142e-13]
 [ 8.93457209e-15 -1.12913346e-14  1.06581410e-14 -2.60532335e-14  2.83466291e-13]
 [-4.52210936e-13 -1.29447091e-14 -3.91295217e-14  2.17603713e-14 -8.99671514e-13]
 [-4.80986022e-11 -2.04469763e-12 -5.94064222e-12  1.33473800e-12 -9.74491599e-11]]
```

3. Для ε = 0.000001:

```
                                   Matrix:
 [[ 1.          1.          1.          1.          1.        ]
 [ 2.17534265  0.70614146  0.50251446  0.60472522  2.17533986]
 [ 4.73211564  0.49863577  0.25252078  0.36569259  4.73210349]
 [10.29397298  0.35210739  0.12689534  0.22114353 10.29393332]
 [22.39291846  0.24863763  0.06376674  0.13373107 22.39280344]]

                           inverse Matrix:
 [[ 4.32880338e+04 -2.38927490e+05  4.66499370e+05 -3.69892944e+05  9.27347249e+04]
 [ 3.22594393e+01 -1.47200857e+02  2.21041840e+02 -1.22439831e+02  2.24334145e+01]
 [ 3.46952990e+01 -1.38406016e+02  1.86503797e+02 -9.72078492e+01  1.71698321e+01]
 [-6.56675576e+01  2.84047339e+02 -4.04580096e+02  2.17409697e+02 -3.91071167e+01]
 [-4.32883210e+04  2.38929050e+05 -4.66502336e+05  3.69895182e+05 -9.27352211e+04]]

                           A^(-1) * A:
 [[ 1.00000000e+00 -2.81914292e-11  3.37972466e-11 -2.95334083e-11  3.03760828e-10]
 [-1.35206143e-13  1.00000000e+00  1.94064166e-14 -1.40165184e-14  1.32603953e-13]
 [-7.45787263e-14 -3.29499347e-14  1.00000000e+00 -3.72097891e-14  2.91896965e-14]
 [ 3.84835054e-13  4.95181594e-14  3.91476591e-15  1.00000000e+00 -2.53251625e-14]
 [ 1.06597755e-10  3.01480507e-11 -3.38593226e-11  3.13043933e-11  1.00000000e+00]]

                                        R:
 [[ 7.30979277e-10  7.36918025e-11  9.83529369e-11  3.94611353e-11  1.14205587e-09]
 [-2.15193050e-14  1.06581410e-14  2.11827734e-14 -1.09078939e-14  3.03134210e-13]
 [ 3.57456416e-13  4.35839905e-15  1.79856130e-14 -1.32599908e-14  4.61225247e-13]
 [-4.10972810e-13 -1.79834005e-14 -5.64813666e-14  1.42108547e-14 -7.07446189e-13]
 [-3.59063532e-10 -2.80596102e-11 -6.29631530e-11  2.20056288e-12 -1.00963848e-09]]

Norm: 45.23185734282182
Conditionality: 54791615.36800836
```

Как видно из полученных значений, при уменьшении числа ε, число обусловленности матрицы значительно увеличивается, в то время как норма матрицы R изменяется незничетельно.

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, norm


# n - размер матрицы
# Гарантировано генерирует инвертируемую 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)
    P, L, U = lu(A)
    E = np.identity(A.shape[0])

    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 np.matmul(A_inv, P.transpose())


def get_R_matrix(A: np.matrix):
    A_inv = np.linalg.inv(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   inverse Matrix:\n", get_inverse(A_arr[i]), "\n")

        print("\t\t\t   A^(-1) * A:\n", np.matmul(get_inverse(A_arr[i]), 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