**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего  
 образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

«Методы искусственного интеллекта»

Отчет по лабораторной работе №2

Вариант – 20

Выполнил:

студент группы ИСТбд-42

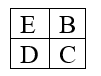
Ходарев Д.А.

Проверил:

Преподаватель Сазонов С.Н.

Ульяновск,

2022

**Задание:** С клавиатуры вводится два числа K и N. Квадратная матрица А(N,N), состоящая из 4-х равных по размерам подматриц, B,C,D,E заполняется случайным образом целыми числами в интервале [-10,10]. Для отладки использовать не случайное заполнение, а целенаправленное. Вид матрицы А:

Для простоты все индексы в подматрицах относительные.

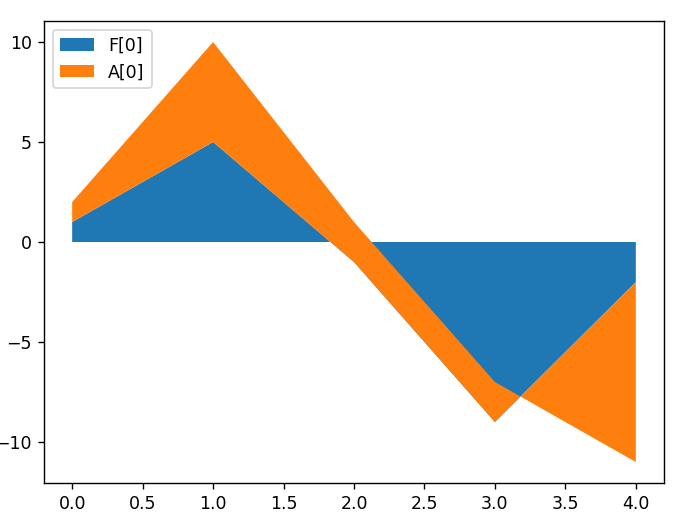
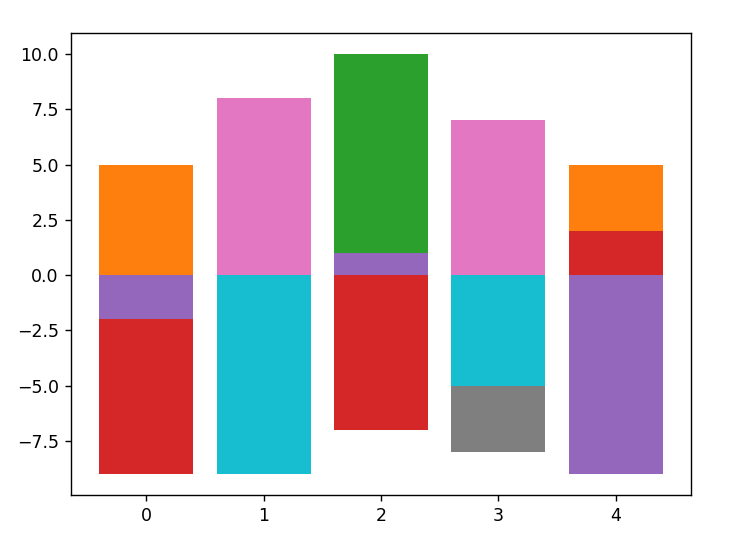
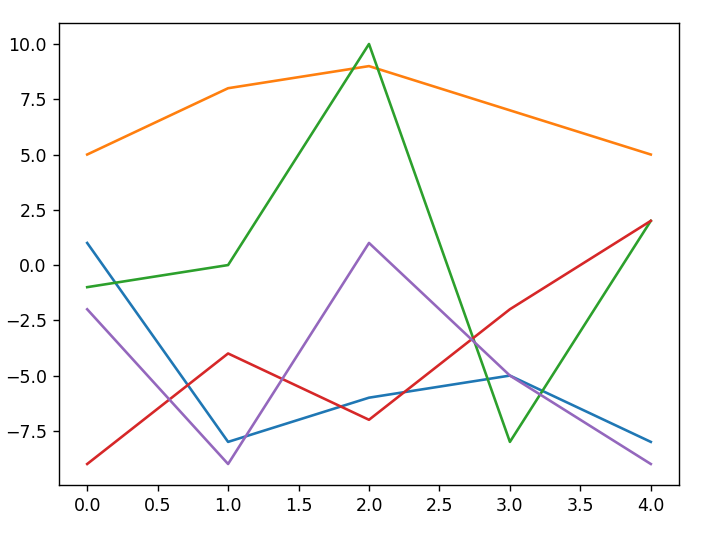
По сформированной матрице F (или ее частям) необходимо вывести не менее 3 разных графиков.

Допускается использование библиотек numpy и mathplotlib.

**Вариант: 20**

Формируется матрица F следующим образом: скопировать в нее А и если в Е количество чисел, больших К в четных столбцах , чем произведение чисел в нечетных строках , то поменять местами С и Е симметрично, иначе С и В поменять местами несимметрично. При этом матрица А не меняется. После чего если определитель матрицы А больше суммы диагональных элементов матрицы F, то вычисляется выражение: A\*AT – K \* F-1, иначе вычисляется выражение (A-1 +G-FТ)\*K, где G-нижняя треугольная матрица, полученная из А. Выводятся по мере формирования А, F и все матричные операции последовательно.

**Графики:**



Графики построены на основе следующих данных:

Введите N: 5

Введите K: 5

=====MatrixA=====

[[ 1 5 2 2 -9]

[-8 8 -8 -2 -5]

[-6 9 -1 -9 -2]

[-5 7 0 -4 -9]

[-8 5 10 -7 1]]

================

=====MatrixF=====

Description: Copying matrix A.

[[ 1 5 2 2 -9]

[-8 8 -8 -2 -5]

[-6 9 -1 -9 -2]

[-5 7 0 -4 -9]

[-8 5 10 -7 1]]

================

=====MatrixF=====

Description: Non-symmetrical swap C and B.

[[ 1 5 -1 -9 -2]

[-8 8 0 -4 -9]

[-6 9 10 -7 1]

[-5 7 -8 -2 -5]

[-8 5 2 2 -9]]

================

DetA = 32189.99999999996.

Sum of diagonal elements = 8.

=====MatrixA=====

Description: Transposition.

[[ 1 -8 -6 -5 -8]

[ 5 8 9 7 5]

[ 2 -8 -1 0 10]

[ 2 -2 -9 -4 -7]

[-9 -5 -2 -9 1]]

================

=====MatrixF=====

Description: Exponentiation to -1.

[[ 0.4070411 -0.70592505 0.01070683 0.10244079 0.55974957]

[ 0.30650576 -0.71926322 0.10525361 0.22738408 0.53652119]

[ 0.07322385 -0.12476182 0.03402595 -0.05580256 0.14327194]

[ 0.12884493 -0.48471101 0.04128482 0.1322929 0.38716995]

[-0.14662916 0.09245985 0.06569277 0.05226386 -0.19272298]]

================

Result = [[ -1.03520552 -36.47037474 -12.05353416 -10.51220397 69.20125215]

[-41.53252881 67.59631612 -72.52626803 -15.13692042 -27.68260593]

[-12.36611923 -71.37619091 0.82987025 0.27901279 -20.71635968]

[-10.64422466 -11.57644497 -0.2064241 15.33853552 61.06415026]

[ 72.73314581 -25.46229925 -20.32846384 62.7386807 1.96361492]].

**Листинг программы:**

# Created by Danila Khodarev.

# Option: 20.

from math import floor, ceil

import matplotlib.pyplot as plt

import numpy

def printMatrix(matrix, matrixName = None, description = None):

    if (matrixName != None):

        print(f"====={matrixName}=====")

    else:

        print("=====NoNameMatrix=====")

    if (description != None):

        print(f'Description: {description}')

    print(matrix)

    print("================")

def createMatrix(n, m, minRandValue, maxRandValue):

    return numpy.random.randint(minRandValue, maxRandValue, size=(n, m))

def copyMatrix(matrix):

    return numpy.copy(matrix)

try:

    n = int(input('Введите N: '))

    k = int(input('Введите K: '))

    matrixA = createMatrix(n, n, -10, 11)

    matrixF = copyMatrix(matrixA)

    printMatrix(matrixA, "MatrixA")

    printMatrix(matrixF, "MatrixF", "Copying matrix A.")

    numNumbers = 0

    proNumbers = 1

    for i in range(0, ceil(n / 2)):

        for j in range(0, ceil(n / 2)):

            if (j % 2 == 0 and matrixF[i][j] > k):

                numNumbers += 1

            if (i % 2 == 1):

                proNumbers \*= matrixF[i][j]

    if (numNumbers > proNumbers):

        for i in range(0, ceil(n / 2)): # Symmetrical swap C and E.

            for j in range(0, ceil(n / 2)):

                matrixF[i][j], matrixF[n - 1 - j][n - 1 - i] = matrixF[n - 1 - j][n - 1 - i], matrixF[i][j]

        printMatrix(matrixF, "MatrixF", "Symmetrical swap C and E.")

    else:

        for i in range(0, ceil(n / 2)): # Non-symmetrical swap C and B.

            for j in range(floor(n / 2), n):

                matrixF[i][j], matrixF[floor(n / 2) + i][j] = matrixF[floor(n / 2) + i][j], matrixF[i][j]

        printMatrix(matrixF, "MatrixF", "Non-symmetrical swap C and B.")

    detA = numpy.linalg.det(matrixA)

    print(f"DetA = {detA}.")

    sumDiagonalElements = numpy.trace(matrixF)

    print(f"Sum of diagonal elements = {sumDiagonalElements}.")

    result = 0

    if (detA > sumDiagonalElements):

        result = matrixA \* matrixA.transpose() - k \* numpy.linalg.matrix\_power(matrixF, -1)

        printMatrix(matrixA.transpose(), "MatrixA", "Transposition.")

        printMatrix(numpy.linalg.matrix\_power(matrixF, -1), "MatrixF", "Exponentiation to -1.")

    else:

        result = (numpy.linalg.matrix\_power(matrixA, -1) + numpy.tril(matrixA) - numpy.linalg.matrix\_power(matrixF, -1)) \* k

        printMatrix(numpy.linalg.matrix\_power(matrixA, -1), "MatrixA", "Exponentiation to -1.")

        printMatrix(numpy.tril(matrixA), "MatrixA", "Lower triangular matrix.")

        printMatrix(numpy.linalg.matrix\_power(matrixF, -1), "MatrixF", "Exponentiation to -1.")

    print(f"Result = {result}.")

    plt.plot(matrixF)         # Graph 1.

    plt.show()

    for i in range(0, n):     # Graph 2.

        for j in range(0, n):

            plt.bar(i, matrixF[i][j])

    plt.show()

    x = numpy.arange(0, n, 1) # Graph 3.

    f0 = matrixF[0]

    a0 = matrixA[0]

    labels = ["F[0]", "A[0]"]

    fig, ax = plt.subplots()

    ax.stackplot(x, f0, a0, labels=labels)

    ax.legend(loc='upper left')

    plt.show()

except:

    print('[EXCEPT] An unknown exception has been encountered!')