Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

**Лабораторная работа №1**

Выполнил:

Шишминцев Д. В.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc115866768)

[1. ЗАДАНИЕ 2 4](#_Toc115866769)

[1.1 Интерфейс программы 4](#_Toc115866770)

[1.2 Транспонирование матрицы 4](#_Toc115866771)

[1.3 Умножение матриц 5](#_Toc115866772)

[1.4 Ранг матрицы 6](#_Toc115866773)

[2. ЗАДАНИЕ 3 8](#_Toc115866774)

[2.1 Реализация второго задания с использованием библиотеки NumPy 8](#_Toc115866775)

[2.2 Достоинства и недостатки использования NumPy 8](#_Toc115866776)

[2 ЗАДАНИЕ 4 9](#_Toc115866777)

[3.1 Возведение матрицы в степень -1 9](#_Toc115866778)

[3.2 Сравнение быстродействия 10](#_Toc115866779)

[ВЫВОД 11](#_Toc115866780)

[СПИСОК ЛИТЕРАТУРЫ 12](#_Toc115866781)

[ПРИЛОЖЕНИЕ 13](#_Toc115866782)

# ВВЕДЕНИЕ

Цель данной работы – знакомство с языком программирования Python. В процессе выполнения поставленных задач на операции с матрицами будут задействованы базовые конструкции языка программирования, а также одна из популярных библиотек.

# ЗАДАНИЕ 2

## 1.1 Интерфейс программы

В качестве интерфейса используется интерфейс командной строки. Пользователь выбирает операцию над матрицами вводом числа. Задается размерность матрицы, после чего ввод производиться построчно. Для хранения и работы с матрицами используются вложенные списки. Интерфейс программы демонстрируется ниже. (Рисунок 1)

Text

Description automatically generated

Рисунок 1

Данный вид интерфейса удобен для этой программы и прост в реализации.

## Транспонирование матрицы

Первая задача – реализовать функцию для транспонирования матриц.

Транспонирование — в линейной алгебре это операция над матрицами в результате которой матрица поворачивается относительно своей главной диагонали. При этом столбцы исходной матрицы становятся строками результирующей. Для реализации использовалось два вложенных цикла. Элементы многомерного списка перебирались и добавлялись в новый список. Функция возвращает многомерный список. Реализация представлена ниже. (Фрагмент кода 1)

def transpose(m):

new\_matrix = []

for i in range(0, len(m[0])):

new\_matrix.append([])

for q in range(0,len(m)):

new\_matrix[i].append(m[q][i])

return new\_matrix

Фрагмент кода 1

## Умножение матриц

Вторая задача – реализовать функцию умножения двух матриц. Умножение матриц – одна из основных операций над матрицами. Матрицы могут быть перемножены если число строк матрицы A равно числу столбцов матрицы B. Принцип умножения демонстрируется на рисунке ниже. (Рисунок 2)

A screenshot of a video game

Description automatically generated with medium confidence

Рисунок 2

Для реализации алгоритма используется 3 вложенных цикла, а так же дополнительная функция getCollumn возвращающая столбец матрицы в виде списка. Функция multiplication возвращает результирующую матрицу в виде многомерного списка. Реализация представлена ниже. (Фрагмент кода 2)

def getCollumn(m, index):

collumn = []

for i in range(0, len(m)):

collumn.append(m[i][index])

return collumn

def multiplication(a, b):

if len(a[0]) != len(b):

return False

new\_matrix = []

for i in range(0,len(a)):

new\_matrix.append([])

for q in range(0, len(b[0])):

row = a[i]

collumn = getCollumn(b, q)

element = 0

for p in range(0, len(row)):

element+=row[p]\*collumn[p]

new\_matrix[i].append(element)

return new\_matrix

Фрагмент кода 2

## Ранг матрицы

Третьей задачей было написать функцию нахождения ранга матрицы. Рангом матрицы A с m строками и n столбцами называется максимальное число линейно независимых строк (столбцов). Для реализации алгоритма используется метод элементарный преобразований (Метод Гаусса). Написанная функция принимает в качестве аргумента матрицу в виде многомерного списка, а возвращает число – ранг матрицы.

Функция имеет следующий порядок действий:

1. Из многомерного списка (далее - матрицы) удаляются строки содержащие в себе только нули.
2. Перебирается каждая строка матрицы
   1. Находиться первый ненулевой элемент в строке
   2. Строка делится на найденный элемент
   3. Для каждой последующей строки находиться коэффициент k отношения элемента строки с индексом p к элементу строки с индексом r .
   4. Из последующих строк вычитается строка с индексом r умноженная на коэффициент k.
3. В полученной матрице считается количество ненулевых строк, что и является рангом матрицы.

Реализация представлена ниже (Фрагмент кода 3)

def rank(matrix):

matrix\_copy = matrix.copy()

for row in matrix\_copy:

if row.count(0) == len(row):

matrix.remove(row)

for r in range(len(matrix)):

div = 1

for i in range(len(matrix[r])):

if matrix[r][i] != 0:

div = matrix[r][i]

break

for q in range(len(matrix[r])):

matrix[r][q] = matrix[r][q] / div

for p in range(r+1, len(matrix)):

k = matrix[p][i] / matrix[r][i]

for n in range(len(matrix[p])):

matrix[p][n] = matrix[p][n] - k\*matrix[r][n]

matrix\_rank = len(matrix)

for row in matrix:

if row.count(0) == len(row):

matrix\_rank = matrix\_rank - 1

return matrix\_rank

Фрагмент кода 3

# ЗАДАНИЕ 3

## 2.1 Реализация второго задания с использованием библиотеки NumPy

Используя популярную Python библиотеку NumPy были реализованы все те же функции (Транспонирование, умножение, определение ранга матрицы). Код представлен ниже. (Фрагмент кода 4)

np.array(ioMatrix.inputMatrix()).T

np.dot(

np.array(ioMatrix.inputMatrix()),

np.array(ioMatrix.inputMatrix()))

np.linalg.matrix\_rank(ioMatrix.inputMatrix())

Фрагмент кода 4

## 2.2 Достоинства и недостатки использования NumPy

Библиотека NumPy имеет широкий функционал. Доступно множество математических алгоритмов, работа с многомерными массивами. Реализация тех или иных задач с использованием библиотеки NumPy легка и не занимает много времени. Алгоритмы работают быстро, библиотека написана с использованием Си и Фортрана.

Недостатком может быть то, что разработчик зачастую не понимает что скрывается за готовыми алгоритмами NumPy, что может негативно повлиять на процесс изучения программирования.

# ЗАДАНИЕ 4

## 3.1 Возведение матрицы в степень -1

Для реализации функции получения обратной матрицы была написана вспомогательная функция det принимающая многомерный список на вход и возвращающая число – определитель матрицы. Определитель высчитывается по формуле для матрицы размерности 3х3.

Используя вложенные циклы перебираются все миноры матрицы и вычисляются алгебраические дополнения. Полученный многомерный массив домножается на (1/d), где d – определитель полученный с помощью функции det. Реализация алгоритма представлена ниже. (Фрагмент кода 5)

def det(m):

d = m[0][0]\*m[1][1]\*m[2][2]+\

m[2][0]\*m[0][1]\*m[1][2]+\

m[1][0]\*m[2][1]\*m[0][2]-\

m[2][0]\*m[1][1]\*m[0][2]-\

m[0][0]\*m[2][1]\*m[1][2]-\

m[1][0]\*m[0][1]\*m[2][2]

return d

def reverse(matrix):

d = det(matrix)

if d == 0:

print('Обратная матрица невозможна, определитель равен нулю!')

return

new\_matrix = [[0,0,0],[0,0,0],[0,0,0]]

k = 1

for n in range(len(matrix)):

for m in range(len(matrix[n])):

a = []

for q in range(len(matrix)):

for p in range(len(matrix[q])):

if m != p and n != q:

a.append(matrix[q][p])

new\_matrix[m][n] = (a[0] \* a[3] - a[1] \* a[2]) \* k

k = k\*(-1)

for n in range(len(new\_matrix)):

for m in range(len(new\_matrix[n])):

new\_matrix[n][m] = round(new\_matrix[n][m] \* (1/d), 15)

return new\_matrix

Фрагмент кода 5

## 3.2 Сравнение быстродействия

Сравним быстродействия написанной функции и функции, представленной в библиотеке NumPy. Для сравнения был использован модуль timeit. Было произведено 10 замеров быстродействия. Результаты продемонстрированы на изображении ниже (Рисунок 3)Text

Description automatically generated

Рисунок 3

Мы видим, что в среднем функция из библиотеки NumPy справляется с поставленной задачей в 2,23 раза быстрее, чем написанная мною функция.

# ВЫВОД

Все поставленные задачи были выполнены. В процессе выполнения задач я познакомился с языком программирования Python и библиотекой NumPy, а также закрепил некоторые знания по линейной алгебре. Сравнив быстродействие функций, можно сделать вывод – использование функций из библиотеки NumPy более эффективно, а так же это экономит время разработчика.

# СПИСОК ЛИТЕРАТУРЫ

1. Transpose – Wikipedia // URL: <https://en.wikipedia.org/wiki/Transpose>
2. Matrix Multiplication – Wikipedia // URL: <https://en.wikipedia.org/wiki/Matrix_multiplication>
3. Rank (Linear algebra) – Wikipedia // URL: <https://en.wikipedia.org/wiki/Rank_(linear_algebra)>
4. Invertible matrix – Wikipedia // URL: <https://en.wikipedia.org/wiki/Invertible_matrix>
5. NumPy – Wikipedia // URL: <https://en.wikipedia.org/wiki/NumPy>

# ПРИЛОЖЕНИЕ

Репозиторий с кодом лабораторной работы на GitHub: <https://github.com/Evisom/itmo-ads-lab1>