Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

Лабораторная работа №1

Вариант №1

Выполнил(и:)

Гусев Я.А.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

Введение

В этой работе я напишу несколько функций, позволяющих производить некоторые манипуляции над матрицами, а также проанализирую рациональность использования библиотеки numpy в Python для реализации этих же функций.

Задание 1

Цель – создать программу на языке Python, которая будет содержать следующие функции: умножение матриц, транспонирование матрицы.

Начнём с ввода произвольной матрицы. Считываем её высоту (n) и ширину(m), далее создаём 2д-список (далее – матрица) из нулей с заданными размерами и считываем в неё элементы построчно, используя цикл for (Рисунок 1).

```
n, m = map(int, input('Введите размер матрицы в формате высота:ширина ').split(':'))
print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
mat = [[0] * m for i in range(n)]
for i in range(n):
    mat[i] = list(map(int, input().split()))
```

Рисунок 1 – Ввод

Транспонирование

Создадим функцию transp (Рисунок 2), которая будет принимать на вход нашу матрицу, а возвращать транспонированную матрицу.

```
def transp(mat, n, m):
    mat_transp = [[0] * n for i in range(m)]
    for i in range(n):
        for j in range(m):
            mat_transp[j][i] = mat[i][j]
    return mat_transp
```

Рисунок 2 – Функция transp

В качестве аргументов наша функция принимает матрицу и её размеры. Внутри самой функции создаём новую матрицу, у которой высота равна ширине нашей введённой матрицы, а ширина соответственно равна высоте нашей матрицы. Далее с помощью вложенного цикла проходимся по нашему списку mat и присваиваем в список mat_transp элементы с противоположными индексами (столбец становится строчкой, строчка – столбцом). Таким образом у нас получается транспонированная матрица, которую мы благополучно возвращаем из функции.

Интерфейс

Так как для умножения матриц нам потребуется считать у пользователя вторую матрицу, создадим простенький интерфейс для нашей программы (Рисунок 3). Для начала

попросим выбрать действие: транспонирование или умножение. Если пользователь вводит «Т», инициируется выполнение вышеописанной функции транспонирования с присваиванием её результата в переменную mat_transp и дальнейшим выводом результата в удобном для чтения виде (построчно).

В случае если пользователь вводит «У», мы считываем вторую матрицу аналогично первой, присваиваем результат функции умножения (она будет описана далее) в переменную u и выводим его построчно

```
action = input('Выберите действие: Транспонировать (T), Умножить (У): ')
if action == 'Транспонировать' or action == 'T':
    mat_transp = transp(mat, n, m)
    print('Ваша транспонированная матрица:')
    for i in range(m):
        print(*mat_transp[i])
elif action == 'Умножить' or action == 'У':
    n2, m2 = map(int, input('Введите размер второй матрицы в формате высота:ширина ').split(':'))
    print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
    mat2 = [[0] * m2 for i in range(n2)]
    for i in range(n2):
        mat2[i] = list(map(int, input().split()))
    u = mult(mat, mat2)
    print('Ваше произведение:')
    for i in range(n):
        print(*u[i])
```

Рисунок 3 – Интерфейс

Умножение

Создаём функцию умножения матриц (Рисунок 4), принимающую 2 матрицы в качестве аргументов.

```
def mult(mat, mat2):
    mult_mat = [[0] * m2 for i in range(n)]
    mat2 = transp(mat2, n2, m2)
    for i in range(n):
        for j in range(m2):
            for elem in range(n2):
                mult_mat[i][j] += mat[i][elem] * mat2[j][elem]
    return mult_mat
```

Рисунок 4 – Функция умножения

Внутри функции создаём матрицу, состоящую из нулей, шириной второй матрицы, высотой первой матрицы. Транспонируем матрицу 2 с помощью нашей первой функции. Далее проходимся по строкам первой матрицы, по столбцам второй и по элементам в них (элементов одинаковое количество, т.к. высота первой матрицы должна быть равна ширине второй, иначе их нельзя было бы умножить). К элементу mult_mat, имеющему координаты (строка первой матрицы; столбец второй матрицы) прибавляем соответственные произведения элементов этих строки и столбца (Рисунок 5).

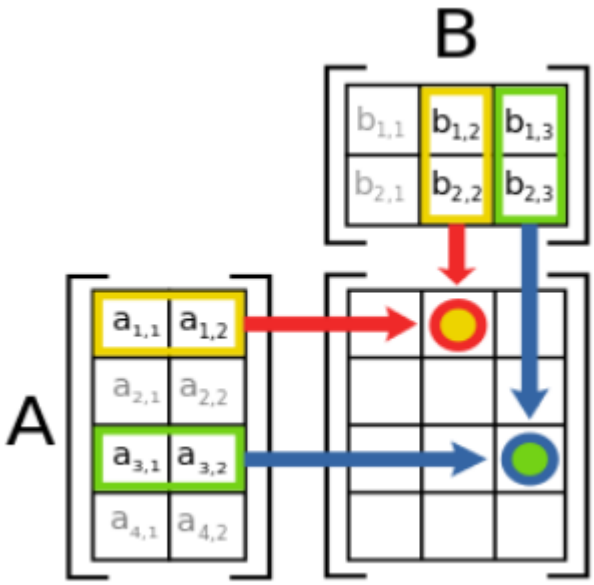

Рисунок 5 – Принцип умножения матриц

Таким образом, мы получили программу, способную транспонировать и умножать матрицы.

Задание 2

Цель – реализовать программу, созданную нами в задании 1, при помощи библиотеки numpy.

Импортируем библиотеку numpy (Рисунок 6).

```
import numpy as np
```

Рисунок 6 – Импорт библиотеки numpy в Python

Создаём 2д-список с нулями и принимаем в него значения от пользователя аналогично первому заданию. Используем функцию array() библиотеки numpy, для того чтобы превратить наш список в матрицу, с которой мы будем производить дальнейшие манипуляции (Рисунок 7).

```
import numpy as np
n, m = map(int, input('Введите размер матрицы в формате высота:ширина ').split(':'))
print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
mat = [[0] * m for i in range(n)]
for i in range(n):
    mat[i] = list(map(int, input().split()))
mat = np.array(mat)
```

Рисунок 7 – Ввод матрицы

Транспонирование

Позаимствуем интерфейс из первого задания и реализуем транспонирование матрицы при помощи функции transpose() (Рисунок 8).

```
action = input('Выберите действие: Транспонировать (T), Умножить (Y): ')
if action == 'Транспонировать' or action == 'T':
    transposed_mat = mat.transpose()
    print('Ваша транспонированная матрица:')
    print(transposed_mat)
```

Рисунок 8 – Транспонирование матрицы с помощью numpy

Выводим результат.

Умножение

Считываем вторую матрицу. Реализуем умножение при помощи функции matmul() (Рисунок 9).

```
13    elif action == 'Умножить' or action == 'У':
14        n2, m2 = map(int, input('Введите размер второй матрицы в формате высота:ширина ').split(':'))
15        print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
16        mat2 = [[0] * m2 for i in range(n2)]
17        for i in range(n2):
18            mat2[i] = list(map(int, input().split()))
19        mat2 = np.array(mat2)
20        mult_mat = np.matmul(mat, mat2)
21        print('Ваше произведение:')
22        print(mult_mat)
```

Рисунок 9 – Умножение матриц с помощью numpy

Выводим результат.

Преимущества и недостатки numpy

Очевидным преимуществом библиотеки numpy по сравнению с созданием функций вручную является упрощение кода в несколько десятков раз с сохранением скорости

Недостаток – numpy сильно зависит от библиотек C++.

Задание 3

Цель – создание функции для возведения матрицы A размерности 3x3 в степень -1 и сравнение быстродействия нашей функции с её аналогом из numpy.

Считываем матрицу 3 на 3 (Рисунок 10).

```
print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
mat = [[0] * 3 for i in range(3)]
for i in range(3):
    mat[i] = list(map(int, input().split()))
```

Рисунок 10 – Считывание матрицы

Создаём функцию reverse_mat. Позаимствуем функцию транспонирования из задания 1 (Рисунок 11).

```
def reverse_mat(mat):
    def transp(mat):
        mat_transp = [[0] * 3 for i in range(3)]
        for i in range(3):
            for j in range(3):
                mat_transp[j][i] = mat[i][j]
        return mat_transp
```

Рисунок 11 – Начало функции

Находим детерминант матрицы методом треугольников (Рисунок 12).

```
det_mat = (mat[0][0] * (mat[1][1]*mat[2][2] - mat[1][2]*mat[2][1]))\
    - (mat[0][1]*(mat[1][0]*mat[2][2] - mat[1][2]*mat[2][0]))\
    + (mat[0][2]*(mat[1][0]*mat[2][1] - mat[2][0]*mat[1][1]))              # Находим детерминанту
if det_mat == 0:
    print('Определитель равен нулю. Обратую матрицу вычислить невозможно.')
else:
```

Рисунок 12 – Нахождение детерминанта

Если детерминант не равен нулю, продолжаем выполнение функции.

Алгоритм

Создаём пустую матрицу 3 на 3. Транспонируем нашу изначальную матрицу (Рисунок 13).

Рисунок 13 – Главный алгоритм

Далее нам требуется найти минор для каждого элемента матрицы. Для этого мы проходимся по матрице с помощью вложенного цикла, удаляем из нашей транспонированной матрицы строчку с индексом i, при помощи метода pop(). Проходимся по оставшимся строчкам и из каждой удаляем элемент с индексом j (таким образом убираем j-ный столбец). У нас получился минор opr. Определяем его детерминант и алгебраическое дополнение матрицы. Делим на детерминант нашей входной матрицы и присваиваем это значение элементу reverse_mat[i][j]. Проделываем данный алгоритм для каждого из 9 элементов матрицы и получаем обратную матрицу opr.

Реализация в numpy

Та же функция с помощью numpy (Рисунок 14).

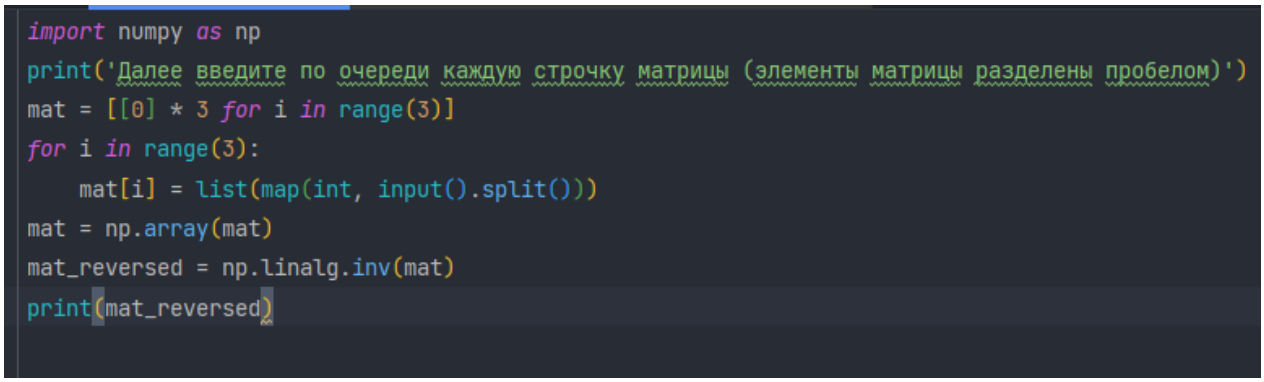

Рисунок 14 – Нахождение обратной матрицы с помощью numpy

Сравнение быстродействия

Импортируем библиотеки time и timeit для измерения быстродействия кода. Сначала замерим скорость функции в numpy (Рисунок 15).

```python
import numpy as np
import time
import timeit

# print('Далее введите по очереди каждую строчку матрицы (элементы матрицы разделены пробелом)')
# mat = [[0] * 3 for i in range(3)]
# for i in range(3):
#     mat[i] = list(map(int, input().split()))
# mat = np.array(mat)
# mat_reversed = np.linalg.inv(mat)
# print(mat_reversed)
print(timeit.timeit(stmt='np.linalg.inv(np.array([[1,2,3],[4,5,6],[7,8,8]]))', setup='import numpy as np', number=1))
```

Рисунок 15 – Измерение быстродействия функции с numpy

В консоль вывелось 0.0001915. Это и есть наша скорость. Далее замерим скорость самописной функции (Рисунок 16).

```python
print(timeit.timeit(stmt='reverse_mat([[1,2,3],[4,5,6],[7,8,8]])', setup='from __main__ import reverse_mat', number=1))
```

Рисунок 16 – Измерение быстродействия самописной функци

В консоль вывелось 0,00432999. Мы видим, что использование numpy повышает быстродействие функции в 40 раз.

Вывод

Я написал функции, позволяющие транспонировать, умножать и находить обратные матрицы. Также я понял, что библиотека numpy – это лучшее, что придумало человечество, так как она позволяет сохранить кучу времени, а также не только не замедляет код, но и ускоряет его в десятки раз.

Список литературы.

1. Wikipedia - https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F (Дата последнего обращения 28.09.2022)