МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра

инфокоммуникаций

Институт цифрового

развития

ОТЧЁТ

по лабораторной работе №3.3

Дисциплина: «Исследование методов работы с матрицами и векторами с помощью библиотеки NumPy»

Выполнила:
студентка 2 курса
группы Пиж-б-о-21-1
Джолдошова Мээрим
Бекболотовна

Ставрополь 2023

Цель работы: исследовать методы работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.

1. Проработать примеры лабораторной работы.

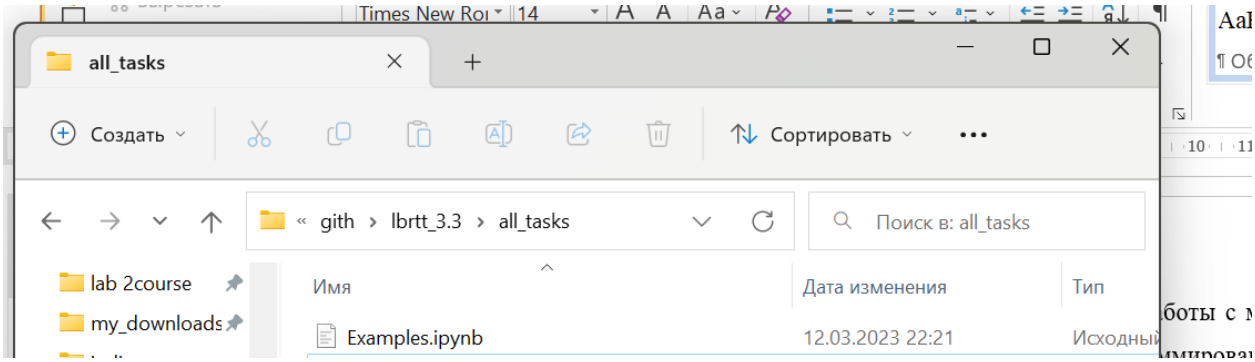

Рисунок 1 – Было выполнено 1 задание

2. Создать ноутбук, в котором будут приведены собственные примеры на языке Python для каждого из представленных свойств матричных вычислений.

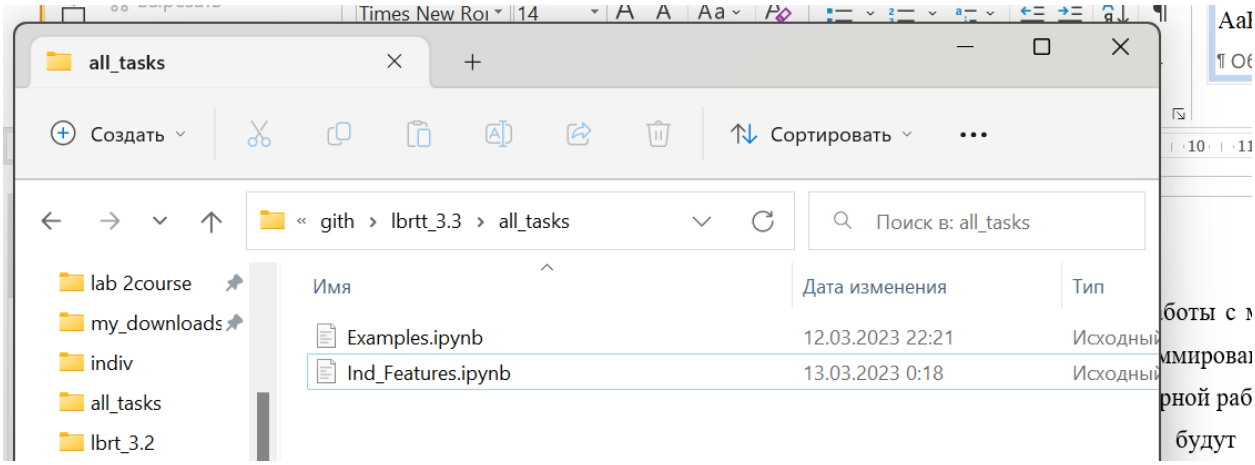

Рисунок 2 – Было выполнено 2 задание

3. Создать ноутбук, в котором будут приведены собственные примеры решения систем линейных уравнений матричным методом и методом Крамера.

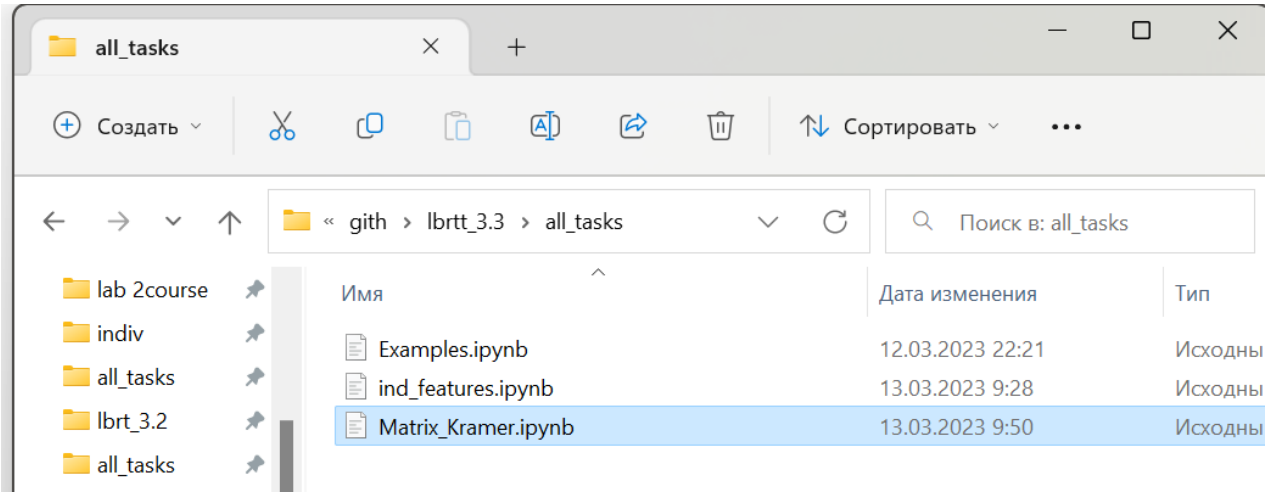

Рисунок 3 – Было выполнено 3 задание

Вопросы для защиты

1. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python.

Матрицей в математике называют объект, записываемый в виде прямоугольной таблицы, элементами которой являются числа (могут быть как действительные, так и комплексные). Пример матрицы приведен ниже.

Матрица в Python – это двумерный массив, поэтому задание матриц того или иного вида предполагает создание соответствующего массива. Для работы с массивами в Python используется тип данных список (англ. list). Но с точки зрения представления матриц и проведения вычислений с ними списки – не очень удобный инструмент, для этих целей хорошо подходит библиотека Numpy,

Вектором называется матрица, у которой есть только один столбец или одна строка. Более подробно свойства векторов, их геометрическая интерпретация и операции над ними будут рассмотрены в “Главе 2 Векторная алгебра”.

Вектор-строка

Вектор-строка имеет следующую математическую запись.

v=(1 2)

```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np )
[1 2]
```

Если необходимо создать нулевой или единичный вектор, то есть вектор, у которого все элементы нули либо единицы, то можно использовать специальные функции из библиотеки Numpy.

```
>>> v_hor_zeros_v2 = np.zeros((1, 5))
>>> print(v_hor_zeros_v2 )
[[0. 0. 0. 0. 0.]]
```

```
>>> v_hor_one_v2 = np.ones((1, 5))
>>> print(v_hor_one_v2)
[[1. 1. 1. 1. 1.]]
```

Вектор-столбец

Вектор-столбец имеет следующую математическую запись.

$$v = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

```
>>> v_vert_np = np.array([[1], [2]])
>>> print(v_vert_np)
[[1]
[2]]
```

2. Как выполняется транспонирование матриц?

С помощью transpose()

```
A = np.matrix('1 2 3; 4 5 6')
A_t = A.transpose()
```

Единичный вектор-столбец можно создать с помощью функции ones().

```
>>> v_vert_ones = np.ones((5, 1))
>>> print(v_vert_ones)
[[1.]
[1.]
[1.]
[1.]
[1.]]
```

Квадратной называется матрица, у которой количество столбцов и строк совпадает.

Особым видом квадратной матрицы является диагональная – это такая матрица, у которой все элементы, кроме тех, что расположены на главной диагонали, равны нулю.

Единичной матрицей называют такую квадратную матрицу, у которой элементы главной диагонали равны единицы, а все остальные нулю.

У нулевой матрицы все элементы равны нулю.

3. Приведите свойства операции транспонирования матриц.

Свойство 1. Дважды транспонированная матрица равна исходной матрице.

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц.

Свойство 3. Транспонирование произведения матриц равно произведению транспонированных матриц, расставленных в обратном порядке.

Свойство 4. Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу.

Свойство 5. Определители исходной и транспонированной матрицы совпадают.

4. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц?

В библиотеки NumPy для транспонирования двумерных матриц используется метод transpose().

5. Какие существуют основные действия над матрицами?

- Сложение матриц
- Умножение матрицы на число
- Умножение матриц

6. Как осуществляется умножение матрицы на число?

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
[[ 3  6 9]
[12 15 18]]
```

7. Какие свойства операции умножения матрицы на число?

Свойство 1. Произведение единицы и любой заданной матрицы равно заданной матрице.

Свойство 2. Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы.

Свойство 3. Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел.

Свойство 4. Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число.

Свойство 5. Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число.

8. Как осуществляется операции сложения и вычитания матриц?

Suppose we have two matrices A and B.

A = [[1,2], [3,4]]

B = [[4,5], [6,7]]

then we get

A+B = [[5,7], [9,11]]

A-B = [[-3,-3], [-3,-3]]

9. Каковы свойства операций сложения и вычитания матриц?

Свойство 1. Коммутативность сложения. От перестановки матриц их сумма не изменяется.

Свойство 2. Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться.

Свойство 3. Для любой матрицы существует противоположная ей, такая, что их сумма является нулевой матрицей

10. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц?

Для сложения и вычитания используется + и -.

11. Как осуществляется операция умножения матриц?

```
mat1 = [[10, 13, 44],
      [11, 2, 3],
      [5, 3, 1]]

mat2 = [[7, 16, -6],
          [9, 20, -4],
          [-1, 3 , 27]]

mat3  = [[0,0,0],
       [0,0,0],
       [0,0,0]]
matrix_length = len(mat1)

#To Add mat1 and mat2 matrices
for i in range(len(mat1)):
    for k in range(len(mat2)):
        mat3[i][k] = mat1[i][k]  * mat2[i][k]
```

12. Каковы свойства операции умножения матриц?

Свойство 1. Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет выполняться эта операция.

Свойство 2. Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц.

Свойство 3. Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется правило независимости произведения от перестановки множителей.

Свойство 4. Произведение заданной матрицы на единичную равно исходной матрице.

Свойство 5. Произведение заданной матрицы на нулевую матрицу равно нулевой матрице.

13. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц?

В библиотеки NumPy для выполнения операции умножения матриц используется функция dot().

14. Что такое определитель матрицы? Каковы свойства определителя матрицы?

Определитель матрицы размера (n-го порядка) является одной из ее численных характеристик. Определитель матрицы A обозначается как |A| или det(A), его также называют детерминантом.

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании.

Свойство 2. Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю.

Свойство 3. При перестановке строк матрицы знак ее определителя меняется на противоположный.

Свойство 4. Если у матрицы есть две одинаковые строки, то ее определитель равен нулю.

Свойство 5. Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число.

Свойство 6. Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц.

Свойство 7. Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится.

Свойство 8. Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю

15. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?

В библиотеки NumPy для нахождения значения определителя матрицы используется функция linalg.det().

16. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству: $A * A^{-1} = E$

Обратную матрицу A − 1 A^{-1}A −1 к матрице A AA можно найти по формуле:

$A^{-1} = 1/\det A \cdot A^{*}$,

где detA — определитель матрицы A,

A^{*} – транспонированная матрица алгебраических дополнений к матрице A.

17. Каковы свойства обратной матрицы?

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица.

Свойство 2. Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы.

Свойство 3. Обратная матрица произведения матриц равна произведению обратных матриц.

18. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы?

В библиотеки NumPy для нахождения обратной матрицы используется функция linalg.inv().

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Решение методом Крамера

Метод Крамера предназначен для того, чтобы решать системы линейных алгебраических уравнений (СЛАУ), в которых число неизвестных переменных равняется числу уравнений, а определитель основной матрицы не равен нулю.

Создадим и заполним матрицу

In [10]:
```
np.random.seed(0)
slae = np.random.randint(0, 25, size = (4, 5))
slae
```

Out[10]:
```
array([[12, 15, 21,  0,  3],
       [ 3,  7,  9, 19, 21],
       [18,  4, 23,  6, 24],
       [24, 12,  1,  6,  7]])
```

Осуществим перевод к матричному виду

In [17]:
```
slae_a = slae[:, 0:4]
slae_b = np.ones((4, 1))
x = np.ones((4, 1))

slae_b[0, 0] = slae[0, 4]
slae_b[1, 0] = slae[1, 4]
slae_b[2, 0] = slae[2, 4]
slae_b[3, 0] = slae[3, 4]

print(f"slae_a\n{slae_a}")
print(f"\nslae_b\n{slae_b}")
```

```
slae_a
[[12 15 21  0]
 [ 3  7  9 19]
 [18  4 23  6]
 [24 12  1  6]]

slae_b
[[ 3.]
 [21.]
 [24.]
 [ 7.]]
```

Осуществим нахождение определителя и дополнительных определителей и самого решения по формулам Крамера

$$x_1 = \frac{\Delta_1}{\Delta}; x_2 = \frac{\Delta_2}{\Delta}; x_3 = \frac{\Delta_3}{\Delta}$$

In [20]:
```
#Нахождения определителя
A_deter = np.linalg.det(slae_a)

if A_deter != 0:
    #Нахождение дополнительных определителей
    for i in range(4):
        A_extra = slae_a.copy()
        A_extra[:, i] = slae_b[:, 0]
        x[i,0] = round(np.linalg.det(A_extra), 2) / round(np.linalg.det(slae_a), 2)
    print(f"Решения:\n{x}")
    print(x)
else:
    print("Матрица вырожденная")
```

```
Решения:
[[ 0.46213836]
 [-0.94625293]
 [ 0.55467303]
 [ 1.11817359]]
[[ 0.46213836]
 [-0.94625293]
 [ 0.55467303]
 [ 1.11817359]]
```

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy

Задача:

Создать ноутбук, в котором будут приведены собственные примеры решения систем линейных уравнений матричным методом и методом Крамера.

Решение матричный методом

решение системы линейных алгебраических уравнений по матричному методу определяется равенством

$$X = A^{-1} * B$$

1. Создание и заполнение матрицы

In [4]:
```
import numpy as np

np.random.seed(0)
slae = np.random.randint(0, 25, size = (4, 5))
slae
```

Out[4]:
```
array([[12, 15, 21,  0,  3],
       [ 3,  7,  9, 19, 21],
       [18,  4, 23,  6, 24],
       [24, 12,  1,  6,  7]])
```

$$\begin{cases} 12x_1 + 15x_2 + 21x_3 = 3 \\ 3x_1 + 7x_2 + 9x_3 + 19x_4 = 21 \\ 18x_1 + 4x_2 + 23x_3 + 6x_4 = 24 \\ 24x_1 + 12x_2 + 1x_3 + 6x_4 = 7 \end{cases}$$

1. Осуществим перевод к матричному виду:

In [5]:
```
slae_a = slae[:, 0:4]
slae_b = slae[:, 4]
x = np.ones((4, 1))

print(f"slae_a\n{slae_a}")
print(f"\nslae_b\n{slae_b}")
```

```
slae_a
[[12 15 21  0]
 [ 3  7  9 19]
 [18  4 23  6]
 [24 12  1  6]]

slae_b
[ 3 21 24  7]
```

Получаем

$$\bar{A} = \left(\begin{array}{cccc|c} 12 & 15 & 21 & 0 & 3 \\ 3 & 7 & 9 & 19 & 21 \\ 18 & 4 & 23 & 6 & 24 \\ 24 & 12 & 1 & 6 & 7 \end{array}\right)$$

Нужно найти определитель |A|

находим его по формуле

$$A^{-1} = E/A$$

In [9]:
```
#Нахождение определителя
A_deter = np.linalg.det(slae_a)

if A_deter != 0:
    #Нахождение обратной и присоединенной матриц
    A_inv = np.linalg.inv(slae_a)
    x = A_inv.dot(slae_b)
    print(f"Решение:\n{x}")
else:
    print("Матрица вырождена")
```

```
Решение:
[ 0.46213836 -0.94625293  0.55467303  1.11817359]
```