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

Кафедра инфокоммуникаций

Анализ данных

Отчет по лабораторной работе №3.3

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

Выполнил студент группы

ИВТ-б-о-21-1

Богадуров В.И. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь 2023

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

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

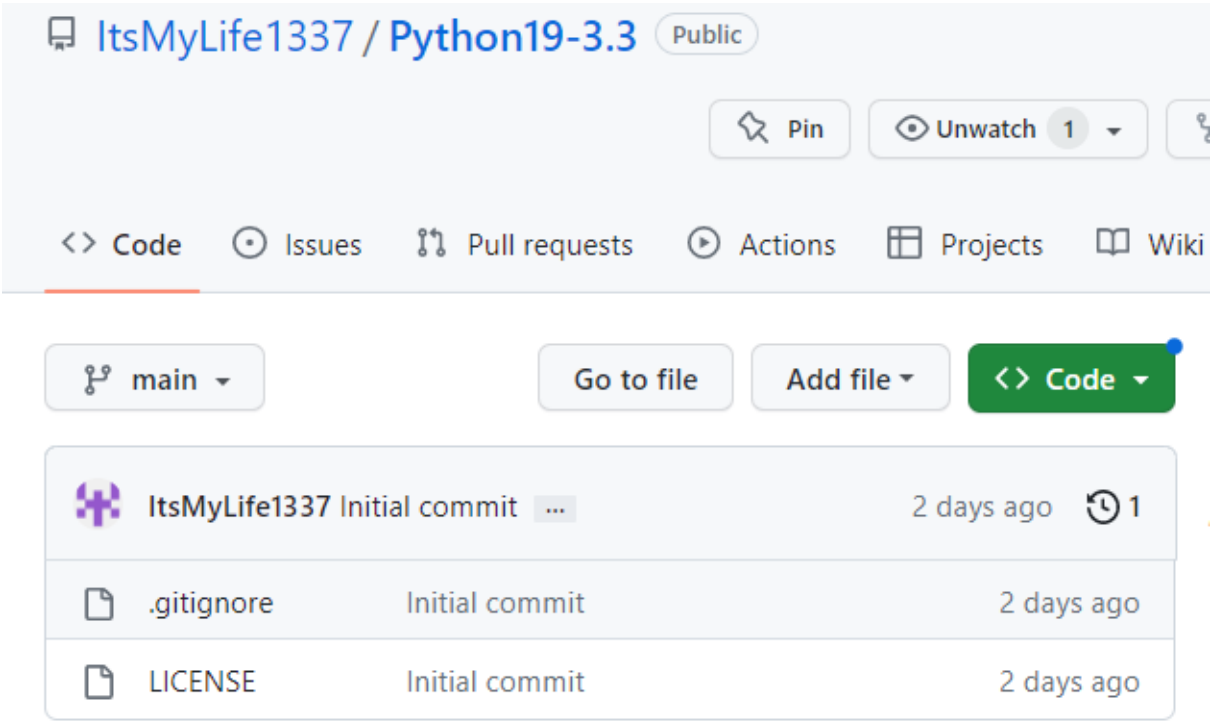

Рисунок 1.1 – Созданный репозиторий

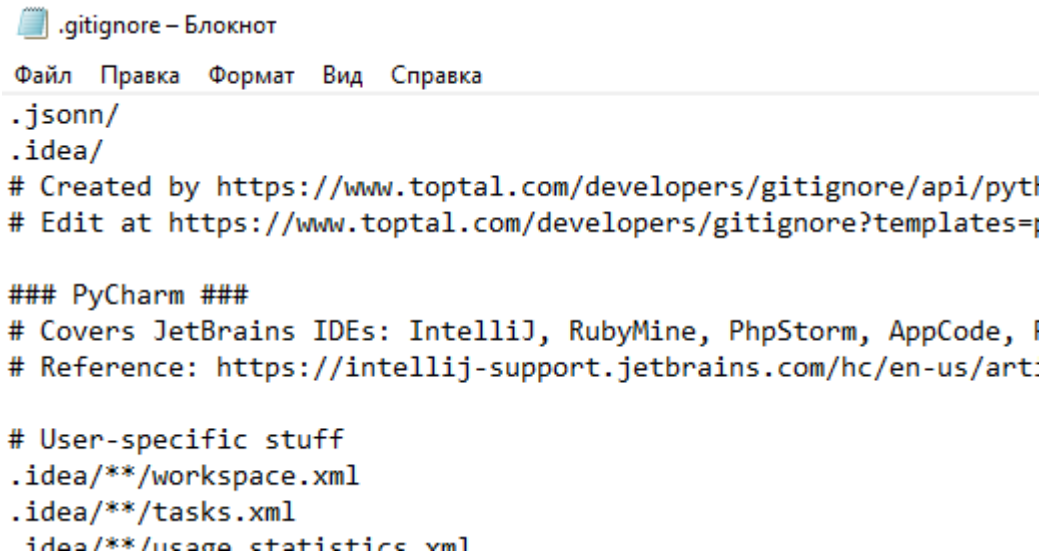

Рисунок 1.2 – Изменения в .gitignore

```
C:\Users\Admin\Desktop\git\Python19-3.3>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/Python19-3.3/.git/hooks]

C:\Users\Admin\Desktop\git\Python19-3.3>
```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Проработка примеров лабораторной работы:

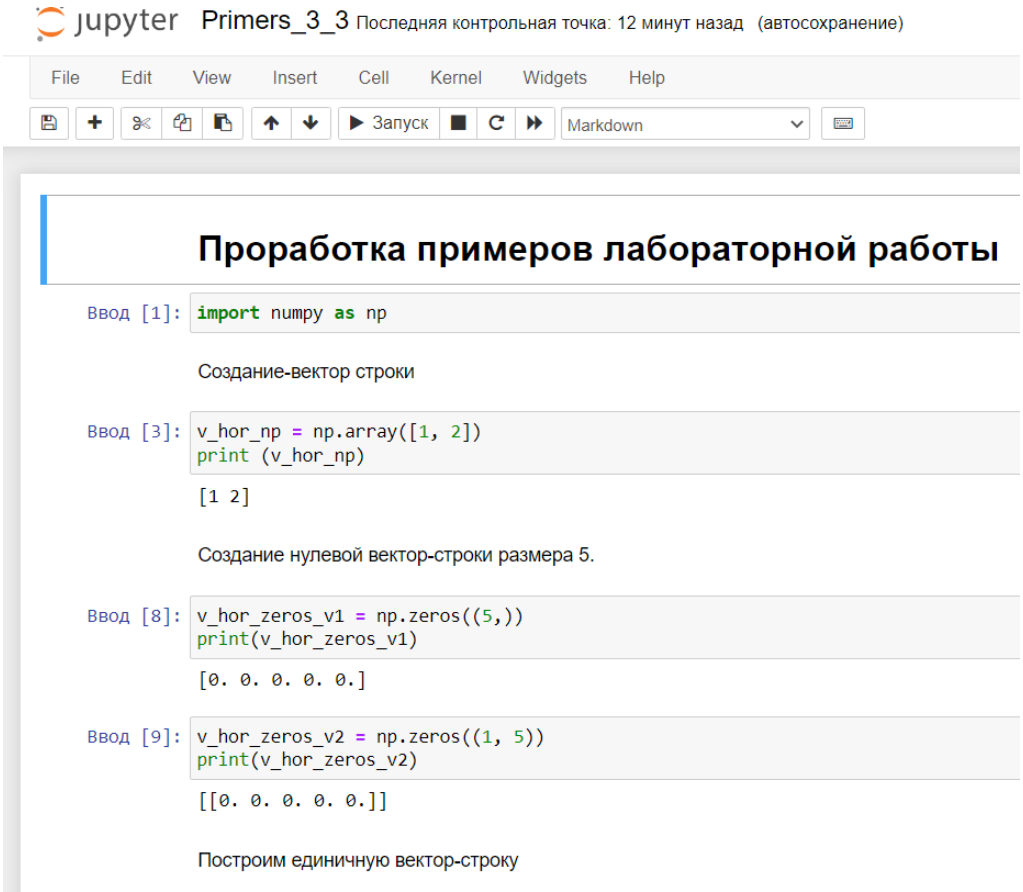

Рисунок 2 – Проработка примеров

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

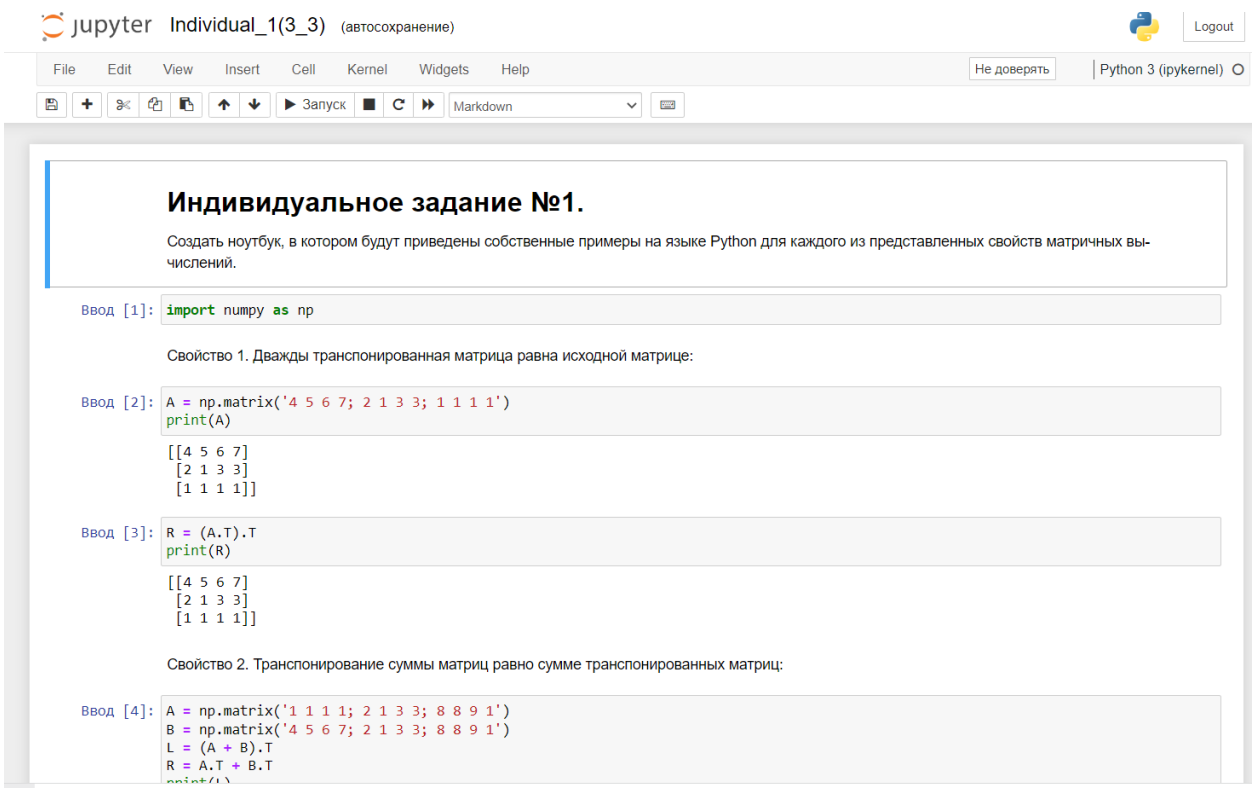

Рисунок 3 – Выполненное индивидуальное задание №1

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

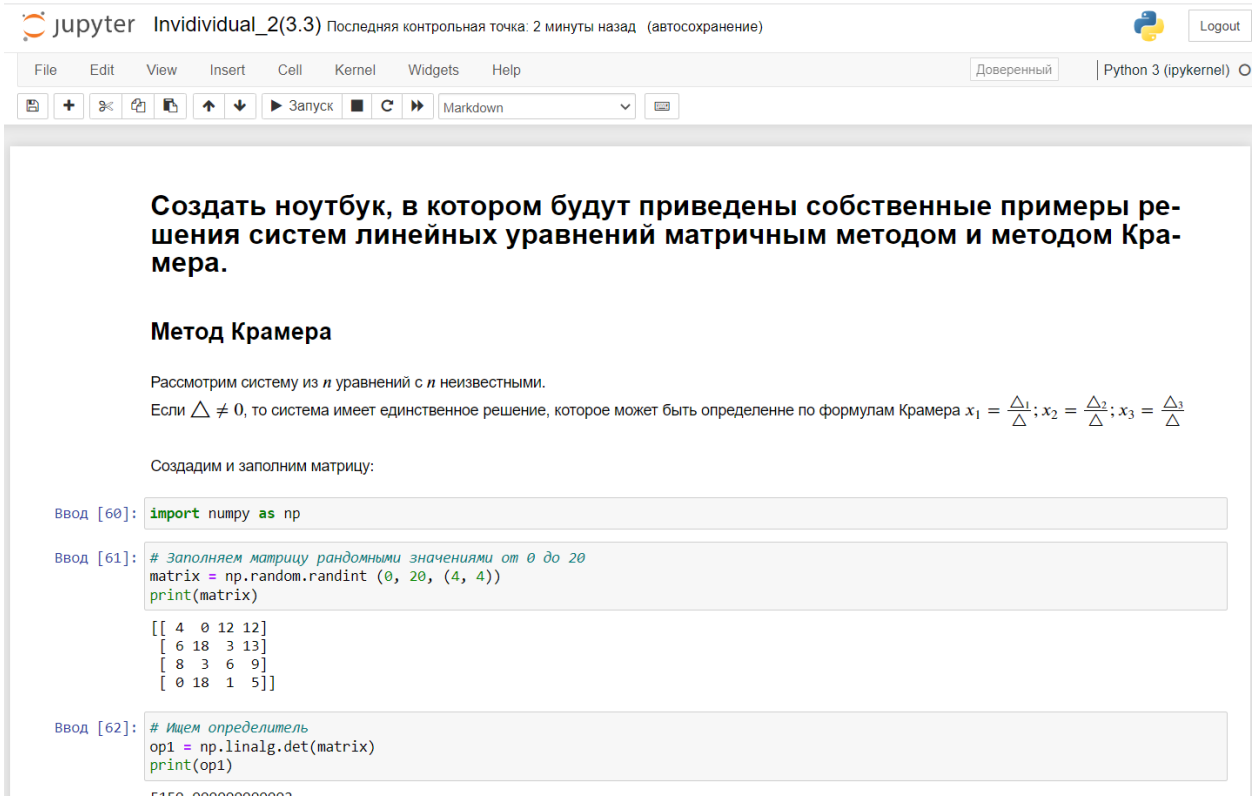

Рисунок 4 – Выполненное индивидуальное задание №2

Вывод: в результате выполнения лабораторнйо работы были получены практические знания и теоретические сведения о методах работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.

Ответы на контрольные вопросы:

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

Вектор-столбец:

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

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

print(v_vert_np)

[[1]

[2]]
```

Нулевой вектор-столбец.

```
v_vert_zeros = np.zeros((5, 1))

print(v_vert_zeros)

[[0.]

[0.]

[0.]

[0.]

[0.]]
```

Единичный вектор-столбец.

```
v_vert_ones = np.ones((5, 1))

print(v_vert_ones)
```

[[1.]

[1.]

[1.]

[1.]

[1.]]

Квадратная матрица:

Довольно часто, на практике, приходится работать с квадратными матрицами. Квадратной называется матрица, у которой количество столбцов и строк совпадает.

В Numpy можно создать квадратную матрицу с помощью метода array():

```
m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_arr)
```

[[1 2 3]

[4 5 6]

[7 8 9]]

Но в Numpy есть еще одни способ создания матриц – это построение объекта типа matrix с помощью одноименного метода. Задать матрицу можно в виде списка.

```
m_sqr_mx = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_mx)
```

[[1 2 3]

[4 5 6]

[7 8 9]]

Также доступен стиль Matlab, когда между элементами ставятся пробелы, а

строки разделяются точкой с запятой, при этом такое описание должно быть передано в виде строки.

m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')

print(m_sqr_mx)

[[1 2 3]

[4 5 6]

[7 8 9]]

Диагональная матрица:

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

Можно построить вручную или:

Создадим матрицу:

m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')

Извлекаем её диагональ:

diag = np.diag(m_sqr_mx)

print(diag)

[1 5 9]

Построим диагональную матрицу на базе полученной диагонали.

m_diag_np = np.diag(np.diag(m_sqr_mx))

print(m_diag_np)

[[1 0 0]

[0 5 0]

[0 0 9]]

Единичная матрица:

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

В ручную или:

Такой способ не очень удобен, к счастью для нас, для построения такого типа матриц в библиотеке Numpy есть специальная функция – eye().

```
m_eye = np.eye(3)

print(m_eye)

[[ 1. 0. 0.]

[ 0. 1. 0.]

[ 0. 0. 1.]]
```

В качестве аргумента функции передается размерность матрицы, в нашем примере – это матрица 3 3. Тот же результат можно получить с помощью функции identity().

```
m_idnt = np.identity(3)

print(m_idnt)

[[ 1. 0. 0.]

[ 0. 1. 0.]

[ 0. 0. 1.]]
```

Нулевая матрица:

У нулевой матрицы все элементы равны нулю. Пример того, как создать такую матрицу с использованием списков, мы приводить не будем, он делается по аналогии с предыдущим разделом. Что касается Numpy, то в составе этой библиотеки есть функция zeros(), которая создает нужную нам матрицу.

```
m_zeros = np.zeros((3, 3))

print(m_zeros)

[[ 0. 0. 0.]

[ 0. 0. 0.]

[ 0. 0. 0.]]
```

В качестве параметра функции zeros() передается размерность требуемой матрицы в виде кортежа из двух элементов, первый из которых – число строк, второй – столбцов. Если функции zeros() передать в качестве аргумента число, то будет построен нулевой вектор-строка, это мы делали в параграфе, посвященном векторам.

Прямоугольная матрица.

```
m_mx = np.matrix('1 2 3 5; 4 5 6 4')

print(m_mx)

[[1 2 3 5]

[4 5 6 4]]
```

2. Как выполняется транспонирование матриц? С помощью функции .transpone() и сокращённый вариант .T

Транспонирование матрицы – это процесс замены строк матрицы на ее столбцы, а столбцов соответственно на строки. Полученная в результате матрица называется транспонированной. Символ операции транспонирования – буква Т.

Для исходной матрицы:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

Транспонированная будет выглядеть так:

$$A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
[[1 2 3]
[4 5 6]]
```

Транспонируем матрицу с помощью метода **transpose():**

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
[2 5]
[3 6]]
```

Существует сокращенный вариант получения транспонированной матрицы, он очень удобен в практическом применении:

```
>>> print(A.T)
```

[[1 4]

[2 5]

[3 6]]

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:

$$(AB)^T = B^T A^T.$$

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?

С помощью функции .transpone() и сокращённый вариант .T.

5. Какие существуют основные действия над матрицами?

Сложения, вычитания, умножение, умножение на число матрицы

Транспонирование матрицы, вынесение минуса из матрицы(внесение), нахождение обратной матрицы.

6. Как осуществляется умножение матрицы на число?

Необходимо каждый элемент матрицы умножить на это число

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?

Складывать можно только матрицы одинаковой размерности — то есть матрицы, у которых совпадает количество столбцов и строк.

Матрицы складываются поэлементно. Например: 1,1 с элементом 1,1; 1,2 с 1,2 и т. д..

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. A + (-A) = Z (нулевая матрица)

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

+ -

11. Как осуществляется операция умножения матриц?

Умножение матриц это уже более сложная операция, по сравнению с рассмотренными выше. Умножать можно только матрицы, отвечающие следующему требованию: количество столбцов первой матрицы должно быть равно числу строк второй матрицы.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \; B = \begin{pmatrix} 7 & 8 \\ 9 & 1 \\ 2 & 3 \end{pmatrix},$$

$$C = A \times B,$$

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \times \begin{pmatrix} 7 & 8 \\ 9 & 1 \\ 2 & 3 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 \cdot 7 + 2 \cdot 9 + 3 \cdot 2 & 1 \cdot 8 + 2 \cdot 1 + 3 \cdot 3 \\ 4 \cdot 7 + 5 \cdot 9 + 6 \cdot 2 & 4 \cdot 8 + 5 \cdot 1 + 6 \cdot 3 \end{pmatrix} = \begin{pmatrix} 31 & 19 \\ 85 & 55 \end{pmatrix}.$$

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?

функция dot(), Например:

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

B = np.matrix('7 8; 9 1; 2 3')

C = A.dot(B)

print(C)

[[31 19]

[85 55]]
```

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.

Перед тем, как привести методику расчета определителя в общем виде, введем понятие минора элемента определителя. *Минор элемента определителя* – это определитель, полученный из данного, путем вычеркивания всех элементов строки и столбца, на пересечении которых стоит данный элемент. Для матрицы 3×3 следующего вида:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

Минор *M23* будет выглядеть так:

$$M_{23} = \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}.$$

Введем еще одно понятие – *алгебраическое дополнение элемента определителя* – это минор этого элемента, взятый со знаком *плюс* или *минус*:

$$A_{ij} = (-1)^{i+j} M_{ij}.$$

В общем виде вычислить определитель матрицы можно через разложение определителя по элементам строки или столбца. Суть в том, что определитель равен сумме произведений элементов любой строки или столбца на их алгебраические дополнения. Для матрицы 3×3 это правило будет выполняться следующим образом:

$$|A| = \begin{vmatrix} -4 & -1 & 2 \\ 10 & 4 & -1 \\ 8 & 3 & 1 \end{vmatrix} = (-4) \cdot A_{11} + (-1) \cdot A_{12} + 2 \cdot A_{13} =$$

$$= (-4) \cdot (-1)^{1+1} M_{11} + (-1) \cdot (-1)^{1+2} M_{12} + 2 \cdot (-1)^{1+3} M_{13} =$$

$$= (-4) \cdot \begin{vmatrix} 4 & -1 \\ 3 & 1 \end{vmatrix} - (-1) \cdot \begin{vmatrix} 10 & -1 \\ 8 & 1 \end{vmatrix} + 2 \cdot \begin{vmatrix} 10 & 4 \\ 8 & 3 \end{vmatrix} =$$

$$= (-4) \cdot (4 + 3) + (10 + 8) + 2 \cdot (30 - 32) =$$

$$= -28 + 18 - 4 = -14.$$

Свойства определителя матрицы.

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.

Свойство 9. Если матрица содержит пропорциональные строки, то ее определитель равен нулю.

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

Для вычисления определителя матрицы воспользуемся функцией det() из пакета linalg.

np.linalg.det(A)

-14.000000000000009

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

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

$$A \times A^{-1} = A^{-1} \times A = E,$$

где – E это единичная матрица.

Для того, чтобы у квадратной матрицы A была обратная матрица необходимо и достаточно чтобы определитель $|A|$ был не равен нулю. Введем понятие **союзной матрицы**. Союзная матрица A строится на базе исходной A путем замены всех элементов матрицы A на их алгебраические дополнения.

Исходная матрица:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Союзная ей матрица A:

$$A^* = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

Транспонируя матрицу A, мы получим так называемую присоединенную матрицу A^T:

$$A^{*T} = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}.$$

Теперь, зная как вычислять определитель и присоединенную матрицу, мы можем определить матрицу A^{-1}, обратную матрице A:

$$A^{-1} = \frac{1}{det(A)} \times A^{*T}.$$

17. Каковы свойства обратной матрицы?

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица.

$$(A^{-1})^{-1} = A.$$

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

$$(A^T)^{-1} = (A^{-1})^T.$$

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

$$(A_1 \times A_2)^{-1} = A_2^{-1} \times A_1^{-1}.$$

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

Для получения обратной матрицы будем используем функцию *inv()*.

```
A = np.matrix('1 -3; 2 5')

A_inv = np.linalg.inv(A)

print(A_inv)

[[ 0.45454545 0.27272727]

[-0.18181818 0.09090909]]
```

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

Создадим и заполним матрицу:

In [60]:
```
import numpy as np
```

In [61]:
```
# Заполняем матрицу рандомными значениями от 0 до 20
matrix = np.random.randint (0, 20, (4, 4))
print(matrix)
```

```
[[ 4  0 12 12]
 [ 6 18  3 13]
 [ 8  3  6  9]
 [ 0 18  1  5]]
```

In [62]:
```
# Ищем определитель
op1 = np.linalg.det(matrix)
print(op1)
```

```
5159.999999999992
```

In [63]:
```
# Заполняем рандомными значенями столбец свободных членов
svobod = np.random.randint (0, 20, (4, 1))
print(svobod)
x = np.ones((4, 1))
```

```
[[16]
 [ 7]
 [ 7]
 [14]]
```

Найдем определитель и дополнительные определители, а также решения по формулам Крамера

In [64]:
```
# Решаем матрицу методом крамера
if op1 != 0:
    for i in range(4):
        # Копируем значения в дополнительную матрицу
        matrix_dop = matrix.copy()
        # Подставляем столбец свободных членов в дополнительную матрицу
        matrix_dop[:, i] = svobod[:, 0]
        # Считаем определители и выводим решения СЛАУ
        x[i,0] = np.linalg.det(matrix_dop) / op1
    print(x)
else:
    print("Матрица вырожденная, нельзя продолжить.")
```

```
[[ 0.1627907 ]
 [ 1.09689922]
 [ 3.03488372]
 [-1.75581395]]
```

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

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

Создадим и заполним матрицу:

Ввод [7]:
```
# Заполняем матрицу рандомными значениями
matrix = np.random.randint (0, 20, (4, 4))
print(matrix)
```

```
[[ 5 15  5 15]
 [19  8  2  7]
 [ 1 11  4 19]
 [16 17  4  1]]
```

Найдем определитель $|A|$, обратную матрицу $A^{-1} = E/A$ и присоединенную A^*.

Ввод [8]:
```
# Для решения матричным методом необходимо найти обратную матрицу:
matrix_inv = np.linalg.inv(matrix)
print(matrix_inv)
```

```
[[ 0.0776699   0.0776699  -0.08737864 -0.04854369]
 [-0.76504854 -0.16504854  0.64401294  0.39482201]
 [ 2.98834951  0.38834951 -2.4368932  -1.24271845]
 [-0.19029126  0.00970874  0.197411    0.03559871]]
```

Ввод [9]:
```
# находим определитель матрицы
op1 = np.linalg.det(matrix)
print(op1)
```

```
-1544.9999999999995
```

Ввод [10]:
```
# Заполняем столбец свободных членов (B)
svobod = np.random.randint (0, 20, (4, 1))
print(svobod)
```

```
[[10]
 [ 1]
 [ 5]
 [ 8]]
```

Найдём решения СЛАУ:

Ввод [11]:
```
# Умножаем нашу обратную матрицу на столбец свободных членов и получаем решение СЛАУ
if op1 != 0:
    x = matrix_inv.dot(svobod)
    print(x)
else:
    print("Невозможно решить, т.к определитель матрицы равен 0")
```

```
[[ 0.02912621]
 [-1.4368932 ]
 [ 8.14563107]
 [-0.62135922]]
```

Найденные решения