РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

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

Отчет по лабораторной работе № 3.3

по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«27» апреля 2023г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

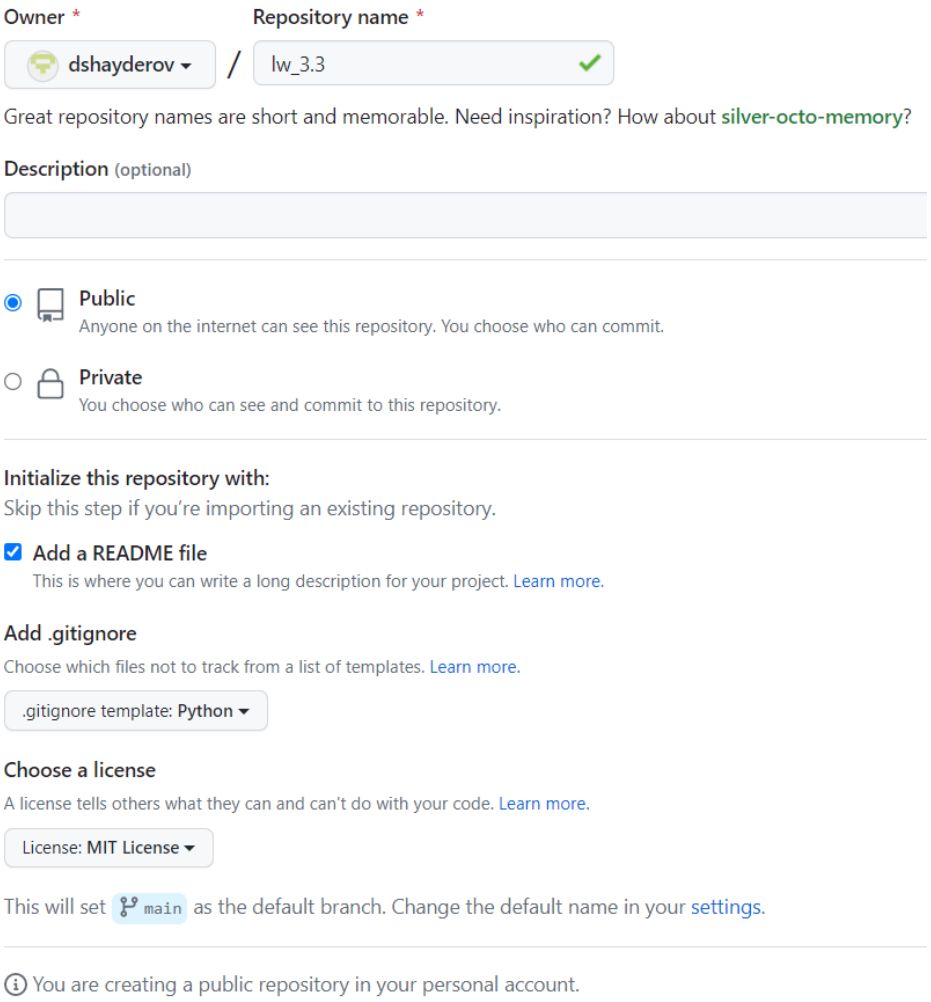


Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных>git clone https://github.com/dshayderov/lw_3.3.git
Cloning into 'lw_3.3'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.3>git checkout -b develop
Switched to a new branch 'develop'

C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.3>
```

Рисунок 3 - Ветвление по модели git-flow

4. Проработать примеры лабораторной работы.

Пример 1.

Пример 1

Транспонирование матрицы

```
In [1]: import numpy as np
```

Свойство 1. Дважды транспонированная матрица равна исходной матрице:

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

```
[[1 2 3]
 [4 5 6]]
```

```
In [3]: R = (A.T).T
        print(R)
```

```
[[1 2 3]
 [4 5 6]]
```

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц:

```
In [4]: B = np.matrix('7 8 9; 0 7 5')
        L = (A + B).T
        R = A.T + B.T
        print(L)
```

```
[[ 8  4]
 [10 12]
 [12 11]]
```

Рисунок 4 - Результат выполнения примера 1

Пример 2.

Пример 2

Умножение матрицы на число

```
In [1]: import numpy as np
```

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

```
In [2]: A = np.matrix('1 2; 3 4')
        L = 1 * A
        R = A
        print(L)
```

```
[[1 2]
 [3 4]]
```

```
In [3]: print(R)
```

```
[[1 2]
 [3 4]]
```

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

```
In [4]: A = np.matrix('1 2; 3 4')
        Z = np.matrix('0 0; 0 0')
        L = 0 * A
        R = Z
        print(L)
```

```
[[0 0]
```

Рисунок 5 - Результат выполнения примера 2

Пример 3.

Пример 3

Сложение матриц

In [1]:
```
import numpy as np
```

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

In [2]:
```
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
L = A + B
R = B + A
print(L)
```

```
[[ 6  8]
 [10 12]]
```

In [3]:
```
print(R)
```

```
[[ 6  8]
 [10 12]]
```

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

In [4]:
```
C = np.matrix('1 7; 9 3')
L = A + (B + C)
R = (A + B) + C
print(L)
```

```
[[ 7 15]
```

Рисунок 6 - Результат выполнения примера 3

Пример 4.

Пример 4

Умножение матриц

In [1]:
```
import numpy as np
```

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

In [2]:
```
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
C = np.matrix('2 4; 7 8')
L = A.dot(B.dot(C))
R = (A.dot(B)).dot(C)
print(L)
```

```
[[192 252]
 [436 572]]
```

In [3]:
```
print(R)
```

```
[[192 252]
 [436 572]]
```

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

In [4]:
```
L = A.dot(B + C)
R = A.dot(B) + A.dot(C)
print(L)
```

```
[[35 42]
```

Рисунок 7 - Результат выполнения примера 4

Пример 5.

Пример 5

Определитель матрицы

In [1]:
```
import numpy as np
```

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании:

In [2]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
```

In [3]:
```
print(A.T)
```

```
[[-4 10  8]
 [-1  4  3]
 [ 2 -1  1]]
```

In [4]:
```
det_A = round(np.linalg.det(A), 3)
det_A_t = round(np.linalg.det(A.T), 3)
print(det_A)
```

```
-14.0
```

In [5]:
```
print(det_A_t)
```

```
-14.0
```

Рисунок 8 - Результат выполнения примера 5

Пример 6.

Пример 6

Обратная матрица

In [1]:
```
import numpy as np
```

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица:

In [2]:
```
A = np.matrix('1. -3.; 2. 5.')
A_inv = np.linalg.inv(A)
A_inv_inv = np.linalg.inv(A_inv)
print(A)
```

```
[[ 1. -3.]
 [ 2.  5.]]
```

In [3]:
```
print(A_inv_inv)
```

```
[[ 1. -3.]
 [ 2.  5.]]
```

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

In [4]:
```
L = np.linalg.inv(A.T)
R = (np.linalg.inv(A)).T
print(L)
```

```
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
```

Рисунок 9 - Результат выполнения примера 6

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

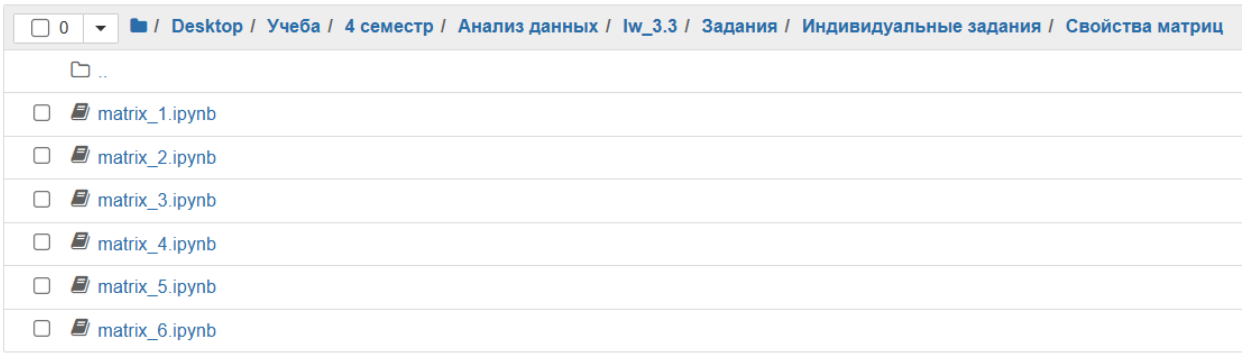


Рисунок 10 – Ноутбуки с примерами свойств матриц

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

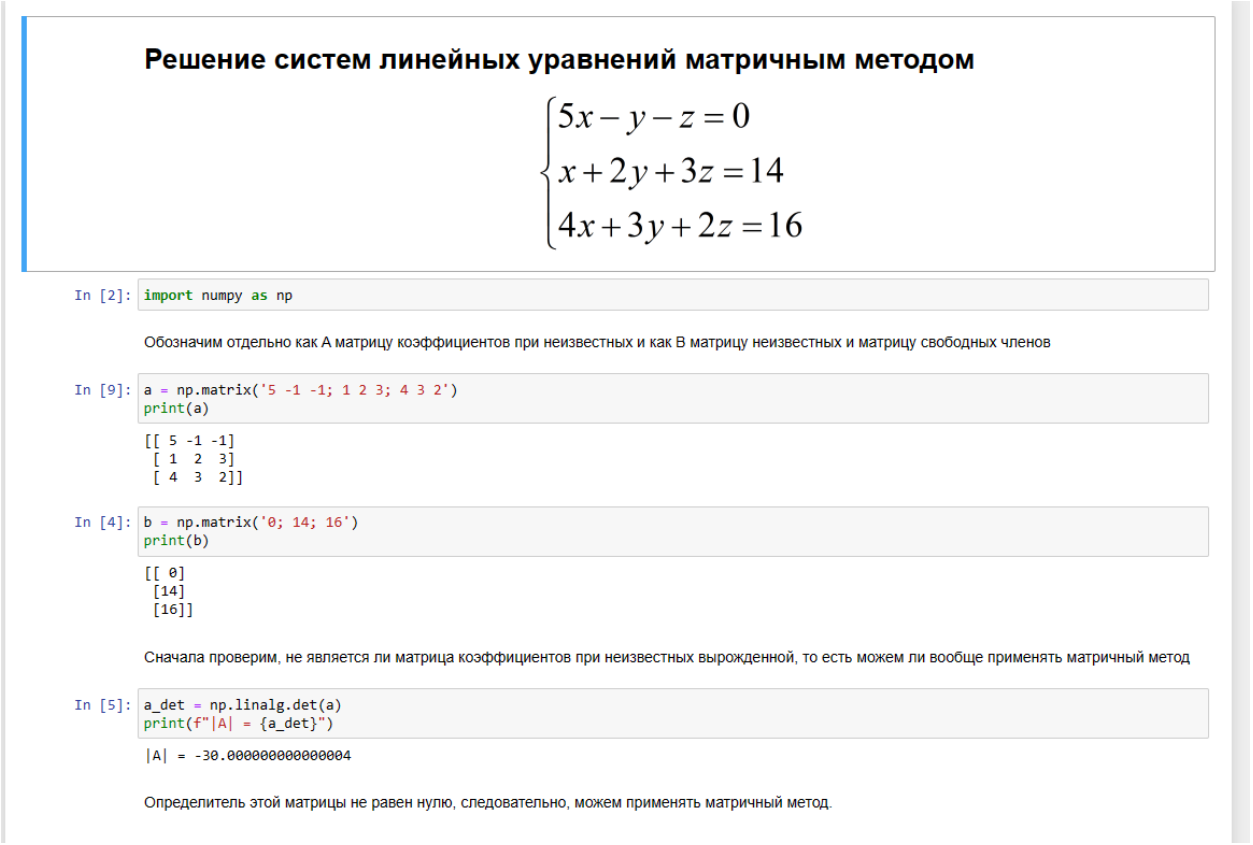
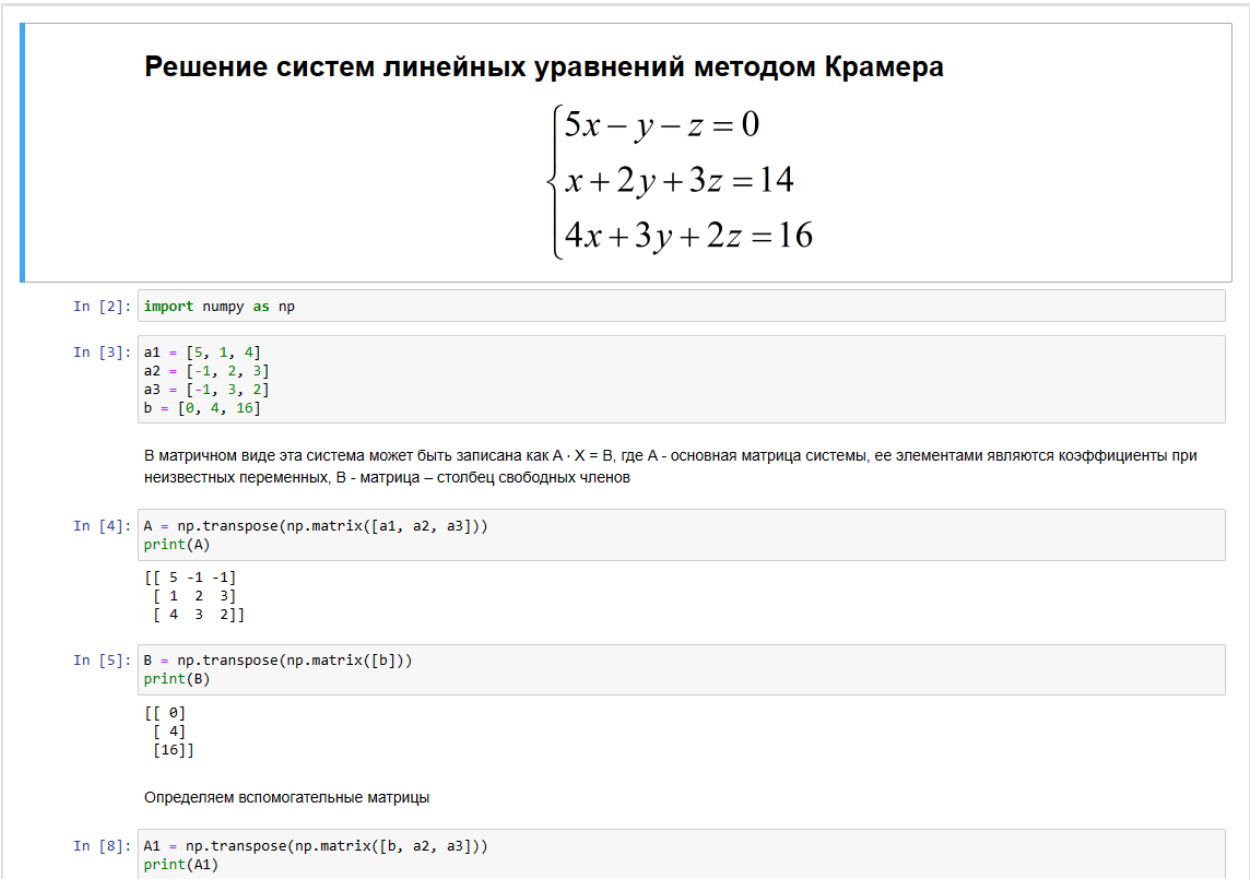


Рисунок 11 – Решение системы уравнений матричным методом



Решение систем линейных уравнений методом Крамера

$$\begin{cases} 5x - y - z = 0 \\ x + 2y + 3z = 14 \\ 4x + 3y + 2z = 16 \end{cases}$$

```
In [2]: import numpy as np

In [3]: a1 = [5, 1, 4]
        a2 = [-1, 2, 3]
        a3 = [-1, 3, 2]
        b = [0, 4, 16]
```

В матричном виде эта система может быть записана как A · X = B, где A - основная матрица системы, ее элементами являются коэффициенты при неизвестных переменных, B - матрица – столбец свободных членов

```
In [4]: A = np.transpose(np.matrix([a1, a2, a3]))
        print(A)

[[ 5 -1 -1]
 [ 1  2  3]
 [ 4  3  2]]

In [5]: B = np.transpose(np.matrix([b]))
        print(B)

[[ 0]
 [ 4]
 [16]]
```

Определяем вспомогательные матрицы

```
In [8]: A1 = np.transpose(np.matrix([b, a2, a3]))
        print(A1)
```

Рисунок 12 - Решение системы уравнений методом Крамера

Контрольные вопросы:

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

Вектор-строка имеет следующую математическую запись.

v = (1 2)

v_hor_np = np.array([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_v1 = np.zeros((5,))

v_hor_one_v1 = np.ones((5,))

Вектор-столбец имеет следующую математическую запись.

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

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

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

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

в Numpy есть еще одни способ создания матриц – это построение объекта типа matrix с помощью одноименного метода.

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

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

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

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

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

m_eye = np.eye(3)

m_idnt = np.identity(3)

У **нулевой** матрицы все элементы равны нулю.

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

2. Как выполняется транспонирование матриц?

Транспонирование матрицы – это процесс замены строк матрицы на ее столбцы, а столбцов соответственно на строки.

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?

transpose()

5. Какие существуют основные действия над матрицами?

Умножение матрицы на число.

Сложение матриц.

Умножение матриц.

6. Как осуществляется умножение матрицы на число?

При умножении матрицы на число, все элементы матрицы умножаются 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.

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?

Сложение и вычитание матриц производится при помощи операторов «+» и «-».

11. Как осуществляется операция умножения матриц?

Умножать можно только матрицы, отвечающие следующему требованию: количество столбцов первой матрицы должно быть равно числу строк второй матрицы.

Каждый элемент cij новой матрицы является суммой произведений элементов i-ой строки первой матрицы и j-го столбца второй матрицы.

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?

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.

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 этой матрицы используется функция det() из пакета linalg.

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

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству: $A \times A^{-1} = A^{-1} \times A = E$, где – Е это единичная матрица.

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

Транспонируя матрицу А, мы получим так называемую присоединенную матрицу A^T.

Теперь, зная как вычислять определитель и присоединенную матрицу, мы можем определить матрицу A^{-1}, обратную матрице А: $A^{-1} = \frac{1}{\det(A)} \times A^T$.

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?

Для получения обратной матрицы используется функция *inv()*.

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

В матричном виде эта система может быть записана как A · X = B, где A - основная матрица системы, ее элементами являются коэффициенты при неизвестных переменных, B - матрица – столбец свободных членов

In [4]:
```
A = np.transpose(np.matrix([a1, a2, a3]))
print(A)
```

```
[[ 5 -1 -1]
 [ 1  2  3]
 [ 4  3  2]]
```

In [5]:
```
B = np.transpose(np.matrix([b]))
print(B)
```

```
[[ 0]
 [ 4]
 [16]]
```

Определяем вспомогательные матрицы

In [8]:
```
A1 = np.transpose(np.matrix([b, a2, a3]))
print(A1)
```

```
[[ 0 -1 -1]
 [ 4  2  3]
 [16  3  2]]
```

In [9]:
```
A2 = np.transpose(np.matrix([a1, b, a3]))
print(A2)
```

```
[[ 5  0 -1]
 [ 1  4  3]
 [ 4 16  2]]
```

In [10]:
```
A3 = np.transpose(np.matrix([a1, a2, b]))
print(A3)
```

```
[[ 5 -1  0]
 [ 1  2  4]
 [ 4  3 16]]
```

Находим определители главной и вспомогательной матриц

In [11]:
```
A_det = round(np.linalg.det(A))
A1_det = round(np.linalg.det(A1))
A2_det = round(np.linalg.det(A2))
A3_det = round(np.linalg.det(A3))
print(f"|A| = {A_det}")
print(f"|A1| = {A1_det}")
print(f"|A2| = {A2_det}")
print(f"|A3| = {A3_det}")
```

```
|A| = -30
|A1| = -20
|A2| = -200
|A3| = 100
```

Находим неизвестные

In [27]:
```
x = A1_det / A_det
y = A2_det / A_det
z = A3_det / A_det
print(f"x = {x:.3f}")
print(f"y = {y:.3f}")
print(f"z = {z:.3f}")
```

```
x = 0.667
y = 6.667
z = -3.333
```

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы

линейных уравнений матричным методом средствами библиотеки NumPy.

Сначала проверим, не является ли матрица коэффициентов при неизвестных вырожденной, то есть можем ли вообще применять матричный метод

In [5]:
```
a_det = np.linalg.det(a)
print(f"|A| = {a_det}")
```

```
|A| = -30.000000000000004
```

Определитель этой матрицы не равен нулю, следовательно, можем применять матричный метод.

Решение системы линейных алгебраических уравнений матричным методом определяется по формуле X = A^(-1) * B

Определим обратную матрицу A

In [6]:
```
a_inv = np.linalg.inv(a)
print(a_inv)
```

```
[[ 0.16666667  0.03333333  0.03333333]
 [-0.33333333 -0.46666667  0.53333333]
 [ 0.16666667  0.63333333 -0.36666667]]
```

Находим матрицу неизвестных

In [7]:
```
X = a_inv.dot(b)
print(X)
```

```
[[1.]
 [2.]
 [3.]]
```

Итак, получили решение

In [8]:
```
x = int(round(X[0, 0]))
y = int(round(X[1, 0]))
z = int(round(X[2, 0]))
print(f"x = {x}")
print(f"y = {y}")
print(f"z = {z}")
```

```
x = 1
y = 2
z = 3
```

Вывод: были исследованы методы работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.