МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Гасанов Г. М. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2023

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

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.
6. Проработать примеры лабораторной работы.

Примеры лабораторной работы

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

Вектор строка

In [3]:
```
v_hor_np = np.array([1, 2])
print(v_hor_np )
```

```
[1 2]
```

In [7]:
```
v_hor_zeros_v1 = np.zeros((5,))
print(v_hor_zeros_v1 )

v_hor_zeros_v2 = np.zeros((1, 5))
print(v_hor_zeros_v2 )
```

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

In [8]:
```
v_hor_one_v1 = np.ones((5,))
print(v_hor_one_v1)

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

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

Рисунок 1 – Проработка примеров

Вектор столбец

In [9]:
```
v_vert_np = np.array([[1], [2]])
print(v_vert_np)
```

```
[[1]
 [2]]
```

In [10]:
```
v_vert_zeros = np.zeros((5, 1))
print(v_vert_zeros)
```

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

In [11]:
```
v_vert_ones = np.ones((5, 1))
print(v_vert_ones)
```

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

Квадратная матрица

In [12]:
```
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]]
```

Рисунок 2 – Проработка примеров

In [13]:
```
m_sqr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
m_sqr_arr = np.array(m_sqr)
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [14]:
```
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]]
```

In [15]:
```
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

In [16]:
```
m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
m_diag_np = np.matrix(m_diag)
print(m_diag_np)
```

```
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

Рисунок 3 – Проработка примеров

In [17]:
```
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
diag = np.diag(m_sqr_mx)
print(diag)
```

```
[1 5 9]
```

In [18]:
```
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

In [19]:
```
m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
m_e_np = np.matrix(m_e)
print(m_e_np)
```

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

In [20]:
```
m_eye = np.eye(3)
print(m_eye)
```

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

In [21]:
```
m_idnt = np.identity(3)
print(m_idnt)
```

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

Рисунок 4 – Проработка примеров

In [22]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

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

Задание матрицы в общем виде

In [23]:
```
m_mx = np.matrix('1 2 3; 4 5 6')
>>> print(m_mx)
```

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

In [24]:
```
m_var = np.zeros((2, 5))
print(m_var)
```

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

Транспонирование матрицы

In [25]:
```
A = np.matrix('1 2 3; 4 5 6')
print(A)
```

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

In [26]:
```
A_t = A.transpose()
print(A_t)
```

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

Рисунок 5 – Проработка примеров

In [27]:
```
A = np.matrix('1 2 3; 4 5 6')
print(A)
```

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

In [28]:
```
R = (A.T).T
print(R)
```

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

Действия над матрицами

In [33]:
```
A = np.matrix('1 2 3; 4 5 6')
C = 3 * A
print(C)
```

```
[[ 3  6  9]
 [12 15 18]]
```

In [34]:
```
A = np.matrix('1 6 3; 8 2 7')
B = np.matrix('8 1 5; 6 9 12')
C = A + B
print(C)
```

```
[[ 9  7  8]
 [14 11 19]]
```

In [35]:
```
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]]
```

Рисунок 6 – Проработка примеров

Определитель матрицы

```
In [37]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         print(A)
         np.linalg.det(A)
```

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

```
Out[37]: -14.000000000000009
```

Обратная матрица

```
In [38]: A = np.matrix('1 -3; 2 5')
         A_inv = np.linalg.inv(A)
         print(A_inv)
```

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

Ранг матрицы

```
In [39]: m_eye = np.eye(4)
         print(m_eye)

         rank = np.linalg.matrix_rank(m_eye)
         print(rank)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
4
```

Рисунок 7 – Проработка примеров

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

Собственные примеры для каждого из представленных свойств матричных вычислений.

In [3]:
```
import numpy as np
```

Транспонирование матрицы

Свойство 1. Дважды транспонированная матрица равна исходной матрице:

In [10]:
```
A = np.matrix('11 22 33; 44 55 66; 7 3 4')
print('Исходная матрица:\n', A)
print('Первое транспонирование матрицы:\n', A.T)
print('Второе транспонирование матрицы:\n', (A.T).T)
```

```
Исходная матрица:
 [[11 22 33]
 [44 55 66]
 [ 7  3  4]]
Первое транспонирование матрицы:
 [[11 44  7]
 [22 55  3]
 [33 66  4]]
Второе транспонирование матрицы:
 [[11 22 33]
 [44 55 66]
 [ 7  3  4]]
```

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц:

In [11]:
```
A = np.matrix('5 7 15; 2 33 11; 3 4 1')
print('Матрица A:\n', A)
B = np.matrix('11 22 33; 44 55 66; 7 3 4')
print('Матрица B:\n', B)

L = (A + B).T
print('Транспонирование суммы матриц:\n', L)

R = A.T + B.T
print('Сумма транспонированных матриц:\n', R)
```

```
Матрица A:
 [[ 5  7 15]
 [ 2 33 11]
 [ 3  4  1]]
Матрица B:
 [[11 22 33]
 [44 55 66]
 [ 7  3  4]]
Транспонирование суммы матриц:
 [[16 46 10]
 [29 88  7]
 [48 77  5]]
Сумма транспонированных матриц:
 [[16 46 10]
 [29 88  7]
 [48 77  5]]
```

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

In [13]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)

L = (A.dot(B)).T
print('Транспонирование произведения матриц:\n', L)

R = (B.T).dot(A.T)
print('Произведение транспонированных матриц'
      'расставленных в обратном порядке:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Транспонирование произведения матриц:
 [[ 363 1474]
 [ 495 1859]]
Произведение транспонированных матрицрасставленных в обратном порядке:
 [[ 363 1474]
 [ 495 1859]]
```

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

In [14]:
```
A = np.matrix('5 7 7; 2 33 7')
k = 2
print('Исходная матрица:\n', A)

L = (k * A).T
print('Транспонирование произведения матрицы на число:\n', L)

R = k * (A.T)
print('Произведение числа на транспонированную матрицу:\n', R)
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]]
Транспонирование произведения матрицы на число:
 [[10  4]
 [14 66]
 [14 14]]
Произведение числа на транспонированную матрицу:
 [[10  4]
 [14 66]
 [14 14]]
```

Свойство 5. Определители исходной и транспонированной матрицы совпадают:

In [17]:
```
A = np.matrix('5 7 7; 2 33 7; 1 2 3')
print('Исходная матрица:\n', A)

A_det = np.linalg.det(A)
print('Определитель исходной матрицы:', format(A_det, '.9g'))

A_T_det = np.linalg.det(A.T)
print('Определитель транспонированной матрицы:', format(A_T_det, '.9g'))
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]
 [ 1  2  3]]
Определитель исходной матрицы: 229
Определитель транспонированной матрицы: 229
```

Действия над матрицами

Умножение матрицы на число

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

In [18]:
```
A = np.matrix('5 7 7; 2 33 7; 1 2 3')
print('Исходная матрица:\n', A)

L = 1 * A
print('Произведение единицы и матрицы:\n', L)
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]
 [ 1  2  3]]
Произведение единицы и матрицы:
 [[ 5  7  7]
 [ 2 33  7]
 [ 1  2  3]]
```

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

In [24]:
```
A = np.matrix('5 7; 2 33')
Z = np.matrix('0 0; 0 0')
print(f"Исходная матрица:\n {A}\n", f"Нулевая матрица:\n {Z}")

L = 0 * A
print('Произведение нуля и матрицы A:\n', L)
```

```
Исходная матрица:
 [[ 5  7]
 [ 2 33]]
 Нулевая матрица:
 [[0 0]
 [0 0]]
Произведение нуля и матрицы A:
 [[0 0]
 [0 0]]
```

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

In [26]:
```
A = np.matrix('5 7 7; 2 33 7; 1 2 3')
print('Исходная матрица:\n', A)
p, q = 5, 7

L = (p + q) * A
print('Произведение матрицы на сумму чисел:\n', L)

R = p * A + q * A
print('Сумма произведений матрицы на каждое из чисел:\n', R)
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]
 [ 1  2  3]]
Произведение матрицы на сумму чисел:
 [[ 60  84  84]
 [ 24 396  84]
 [ 12  24  36]]
Сумма произведений матрицы на каждое из чисел:
 [[ 60  84  84]
 [ 24 396  84]
 [ 12  24  36]]
```

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

In [28]:
```
A = np.matrix('5 7 7; 2 33 7; 1 2 3')
print('Исходная матрица:\n', A)
p, q = 5, 7

L = (p * q) * A
print('Произведение матрицы на произведение 2-ух чисел:\n', L)

R = p * (q * A)
print('Произведение 2 числа и матрицы, умноженное на 1 число:\n', R)
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]
 [ 1  2  3]]
Произведение матрицы на произведение 2-ух чисел:
 [[ 175  245  245]
 [  70 1155  245]
 [  35   70  105]]
Произведение 2 числа и матрицы, умноженное на 1 число:
 [[ 175  245  245]
 [  70 1155  245]
 [  35   70  105]]
```

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

In [29]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)
k = 5

L = k * (A + B)
print('Произведение суммы матриц на число 5:\n', L)

R = k * A + k * B
print('Сумма произведений этих матриц на число 5:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Произведение суммы матриц на число 5:
 [[ 80 145]
 [230 440]]
Сумма произведений этих матриц на число 5:
 [[ 80 145]
 [230 440]]
```

Сложение матриц

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

In [30]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)

L = A + B
print('Сложение матрицы A и B:\n', L)

R = B + A
print('Сложение матрицы B и A:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Сложение матрицы A и B:
 [[16 29]
 [46 88]]
Сложение матрицы B и A:
 [[16 29]
 [46 88]]
```

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

In [32]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)
C = np.matrix('1 3; 3 3')
print('Матрица C:\n', C)

L = A + (B + C)
print('A + (B + C):\n', L)

R = (A + B) + C
print('(A + B) + C:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Матрица C:
 [[1 3]
 [3 3]]
A + (B + C):
 [[17 32]
 [49 91]]
(A + B) + C:
 [[17 32]
 [49 91]]
```

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

In [35]:
```
A = np.matrix('5 7 7; 2 33 7')
print('Исходная матрица:\n', A)
Z = np.matrix('0 0 0; 0 0 0')
print('Нулевая матрица:\n', A)

L = A + (-1)*A
print('Сумма матрицы A и её противоположной матрицы:\n', L)
```

```
Исходная матрица:
 [[ 5  7  7]
 [ 2 33  7]]
Нулевая матрица:
 [[ 5  7  7]
 [ 2 33  7]]
Сумма матрицы A и её противоположной матрицы:
 [[0 0 0]
 [0 0 0]]
```

Умножение матриц

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

In [38]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)
C = np.matrix('1 3; 3 3')
print('Матрица C:\n', C)

L = A.dot(B.dot(C))
print('A*(B*C)=\n', L)

R = (A.dot(B)).dot(C)
print('(A*B)*C=\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Матрица C:
 [[1 3]
 [3 3]]
A*(B*C)=
 [[1848 2574]
 [7051 9999]]
(A*B)*C=
 [[1848 2574]
 [7051 9999]]
```

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

In [39]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)
C = np.matrix('1 3; 3 3')
print('Матрица C:\n', C)

L = A.dot(B + C)
print('Произведение матрицы на сумму матриц:\n', L)

R = A.dot(B) + A.dot(C)
print('Сумма произведений матриц:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Матрица C:
 [[1 3]
 [3 3]]
Произведение матрицы на сумму матриц:
 [[ 389  531]
 [1575 1964]]
Сумма произведений матриц:
 [[ 389  531]
 [1575 1964]]
```

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

In [41]:
```
A = np.matrix('5 7; 2 33')
print('Матрица A:\n', A)
B = np.matrix('11 22; 44 55')
print('Матрица B:\n', B)

L = A.dot(B)
print('Произведение A на B:\n', L)

R = B.dot(A)
print('Произведение B на A:\n', R)
```

```
Матрица A:
 [[ 5  7]
 [ 2 33]]
Матрица B:
 [[11 22]
 [44 55]]
Произведение A на B:
 [[ 363  495]
 [1474 1859]]
Произведение B на A:
 [[  99  803]
 [ 330 2123]]
```

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

In [42]:
```
A = np.matrix('5 7; 2 9')
print('Матрица A:\n', A)
E = np.matrix('1 0; 0 1')

L = E.dot(A)
print("Произведение единичной матрицы на матрицу A:\n", L)

R = A.dot(E)
print('Произведение матрицы A на единичную:\n', R)
```

```
Матрица A:
 [[5 7]
 [2 9]]
Произведение единичной матрицы на матрицу A:
 [[5 7]
 [2 9]]
Произведение матрицы A на единичную:
 [[5 7]
 [2 9]]
```

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

In [43]:
```
A = np.matrix('5 7; 2 9')
print('Матрица A:\n', A)
Z = np.zeros((2, 2))

L = Z.dot(A)
print('Произведение матрицы A на нулевую матрицу:\n', L)

R = A.dot(Z)
print('Произведение нулевой матрицы на матрицу A:\n', R)
```

```
Матрица A:
 [[5 7]
 [2 9]]
Произведение матрицы A на нулевую матрицу:
 [[0. 0.]
 [0. 0.]]
Произведение нулевой матрицы на матрицу A:
 [[0. 0.]
 [0. 0.]]
```

Определитель матрицы

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании:

In [45]:
```
A = np.matrix('5 7 4; 2 9 4; 2 1 1 ')
print('Матрица A:\n', A)
print('Транспонированнная матрица A:\n', A.T)

det_A = round(np.linalg.det(A), 3)
print('Определитель исходной матрицы =', det_A)

det_A_t = round(np.linalg.det(A.T), 3)
print('Определитель транспонированной матрицы =', det_A_t)
```

```
Матрица A:
 [[5 7 4]
 [2 9 4]
 [2 1 1]]
Транспонированнная матрица A:
 [[5 2 2]
 [7 9 1]
 [4 4 1]]
Определитель исходной матрицы = 3.0
Определитель транспонированной матрицы = 3.0
```

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

In [46]:
```
A = np.matrix('5 7 4; 0 0 0; 2 1 1 ')
print('Матрица A:\n', A)

print('Определитель матрицы A =', np.linalg.det(A))
```

```
Матрица A:
 [[5 7 4]
 [0 0 0]
 [2 1 1]]
Определитель матрицы A = 0.0
```

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

In [47]:
```
A = np.matrix('9 1 7; 9 6 3; 5 6 1')
print('Матрица A:\n', A)
B = np.matrix('9 1 7; 5 6 1; 9 6 3')
print('Матрица B:\n', B)

print('Определитель матрицы A =', round(np.linalg.det(A), 3))
print('Определитель матрицы B =', round(np.linalg.det(B), 3))
```

```
Матрица A:
 [[9 1 7]
 [9 6 3]
 [5 6 1]]
Матрица B:
 [[9 1 7]
 [5 6 1]
 [9 6 3]]
Определитель матрицы A = 66.0
Определитель матрицы B = -66.0
```

In [48]:
```
A = np.matrix('9 1 7; 9 1 7; 5 6 1')
print('Матрица A:\n', A)

print('Определитель матрицы A =', np.linalg.det(A))
```

```
Матрица A:
 [[9 1 7]
 [9 1 7]
 [5 6 1]]
Определитель матрицы A = 0.0
```

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

In [50]:
```
A = np.matrix('5 7 2; 2 33 1; 7 7 8')
print('Матрица A:\n', A)

k = 5
B = A.copy()
B[2, :] = k * B[2, :]
print('Матрица B, элементы 3 строки которой умножены на число:\n', B)

det_A = round(np.linalg.det(A), 3)
print('Определитель A, умноженный на число =', det_A * k)

det_B = round(np.linalg.det(B), 3)
print('Определитель B', det_B)
```

```
Матрица A:
 [[ 5  7  2]
 [ 2 33  1]
 [ 7  7  8]]
Матрица B, элементы 3 строки которой умножены на число:
 [[ 5  7  2]
 [ 2 33  1]
 [35 35 40]]
Определитель A, умноженный на число = 3940.0
Определитель B 3940.0
```

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

In [57]:
```
A = np.matrix('3 1 1; 5 4 2; 1 1 1')
print('Матрица A:\n', A)
B = np.matrix('3 1 1; 1 2 2; 1 1 1')
print('Матрица B:\n', B)

C = A.copy()
C[1, :] += B[1, :]
print('Матрица C, у которой элементы 2 строки равны'
      'сумме элементов вторых строк матриц A и B', C)

print('Определитель матрицы C =', round(np.linalg.det(C), 2))

print('Сумма определителей матриц A и B =', round(np.linalg.det(A), 2) +
```

```
Матрица A:
 [[3 1 1]
 [5 4 2]
 [1 1 1]]
Матрица B:
 [[3 1 1]
 [1 2 2]
 [1 1 1]]
Матрица C, у которой элементы 2 строки равнысумме элементов вторых стро
к матриц A и B [[3 1 1]
 [6 6 4]
 [1 1 1]]
Определитель матрицы C = 4.0
Сумма определителей матриц A и B = 4.0
```

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

In [58]:
```
A = np.matrix('3 1 1; 5 4 2; 1 1 1')
print('Матрица A:\n', A)

k = 3
B = A.copy()
B[1, :] = B[1, :] + k * B[0, :]
print('Матрица B:\n', B)

print('Определитель матрицы A =', round(np.linalg.det(A), 3))
print('Определитель матрицы B =', round(np.linalg.det(B), 3))
```

```
Матрица A:
 [[3 1 1]
 [5 4 2]
 [1 1 1]]
Матрица B:
 [[ 3  1  1]
 [14  7  5]
 [ 1  1  1]]
Определитель матрицы A = 4.0
Определитель матрицы B = 4.0
```

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

In [59]:
```
A = np.matrix('3 1 1; 5 4 2; 1 1 1')
print('Матрица A:\n', A)

k = 3
A[1, :] = A[0, :] + k * A[2, :]
print('Определитель матрицы A =', round(np.linalg.det(A), 3))
```

```
Матрица A:
 [[3 1 1]
 [5 4 2]
 [1 1 1]]
Определитель матрицы A = 0.0
```

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

In [60]:
```
A = np.matrix('3 1 1; 5 4 2; 1 1 1')
print('Матрица A:\n', A)

k = 3
A[1, :] = k * A[0, :]
print('Матрица A после умножения на k:\n', A)

print('Определитель матрицы A =', round(np.linalg.det(A), 3))
```

```
Матрица A:
 [[3 1 1]
 [5 4 2]
 [1 1 1]]
Матрица A после умножения на k:
 [[3 1 1]
 [9 3 3]
 [1 1 1]]
Определитель матрицы A = 0.0
```

Обратная матрица

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица:

In [61]:
```
A = np.matrix('3. -1.; 11. 1.')
print('Матрица A:\n', A)

A_inv = np.linalg.inv(A)
print('Обратная матрица:\n', A)

A_inv_inv = np.linalg.inv(A_inv)
print('Обратная матрица обратной матрицы:\n', A_inv_inv)
```

```
Матрица A:
 [[ 3. -1.]
 [11.  1.]]
Обратная матрица:
 [[ 3. -1.]
 [11.  1.]]
Обратная матрица обратной матрицы:
 [[ 3. -1.]
 [11.  1.]]
```

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

In [62]:
```
A = np.matrix('3. -1.; 11. 1.')
print('Матрица A:\n', A)

L = np.linalg.inv(A.T)
print('Обратная матрица транспонированной матрицы:\n', L)

R = (np.linalg.inv(A)).T
print('Транспонированная матрица от обратной матрицы:\n', R)
```

```
Матрица A:
 [[ 3. -1.]
 [11.  1.]]
Обратная матрица транспонированной матрицы:
 [[ 0.07142857 -0.78571429]
 [ 0.07142857  0.21428571]]
Транспонированная матрица от обратной матрицы:
 [[ 0.07142857 -0.78571429]
 [ 0.07142857  0.21428571]]
```

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

In [63]:
```
A = np.matrix('3 1; 5 4')
print('Матрица A:\n', A)
B = np.matrix('1 1; 1 2')
print('Матрица B:\n', B)

L = np.linalg.inv(A.dot(B))
print('Обратная матрица произведения матриц:\n', L)

R = np.linalg.inv(B).dot(np.linalg.inv(A))
print('Произведение обратных матриц:\n', R)
```

```
Матрица A:
 [[3 1]
 [5 4]]
Матрица B:
 [[1 1]
 [1 2]]
Обратная матрица произведения матриц:
 [[ 1.85714286 -0.71428571]
 [-1.28571429  0.57142857]]
Произведение обратных матриц:
 [[ 1.85714286 -0.71428571]
 [-1.28571429  0.57142857]]
```

Рисунки 8 – 35 - Собственные примеры для каждого из представленных свойств матричных вычислений

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

Собственные примеры решения систем линейных уравнений матричным методом и методом Крамера.

Матричный метод

Найти решение СЛАУ матричным методом:

$$\begin{cases} 3x + 2y - 5z = -1 \\ 2x - y + 3z = 13 \\ x + 2y - z = 9 \end{cases}$$

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

In [2]:
```
A = np.matrix('3 2 -5; 2 -1 3; 1 2 -1')
B = np.matrix('-1; 13; 9')
```

In [3]:
```
if round(np.linalg.det(A), 3) != 0:
    A_inv = np.linalg.inv(A)
    R = A_inv.dot(B)
    print('X =\n', R)
else:
    print("Определитель равен 0, СЛАУ решить нельзя!")
```

```
X =
 [[3.]
 [5.]
```

Рисунок 36 – Решение СЛАУ матричным методом

Метод Крамера

In [4]:
```
if round(np.linalg.det(A), 3) != 0:
    C = A.copy()
    C[:, 0] = B[:, 0]
    x1 = round(np.linalg.det(C), 3) / round(np.linalg.det(A), 3)

    D = A.copy()
    D[:, 1] = B[:, 0]
    x2 = round(np.linalg.det(D), 3) / round(np.linalg.det(A), 3)

    E = A.copy()
    E[:, 2] = B[:, 0]
    x3 = round(np.linalg.det(E), 3) / round(np.linalg.det(A), 3)

    print(f"x1 = {x1}\n", f"x2 = {x2}\n", f"x3 = {x3}\n")
```

```
x1 = 3.0
 x2 = 5.0
 x3 = 4.0
```

Рисунок 37 – Решение СЛАУ методом Крамера

Ответы на вопросы:

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

Вектор строка

In [4]:
```
v_hor_np = np.array([1, 2])
print(v_hor_np)
```

```
[1 2]
```

In [7]:
```
v_hor_zeros_v1 = np.zeros((5,))
print(v_hor_zeros_v1 )
v_hor_zeros_v2 = np.zeros((1, 5))
print(v_hor_zeros_v2 )
```

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

In [8]:
```
v_hor_one_v1 = np.ones((5,))
print(v_hor_one_v1)
v_hor_one_v2 = np.ones((1, 5))
print(v_hor_one_v2)
```

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

Вектор-столбец

In [10]:
```
v_vert_np = np.array([[1], [2]])
print(v_vert_np)
```

```
[[1]
 [2]]
```

In [22]:
```
v_vert_zeros = np.zeros((5, 1))
print(v_vert_zeros)
v_vert_ones = np.ones((5, 1))
print(v_vert_ones)
```

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

Квадратная матрица

In [13]:
```
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]]
```

In [15]:
```
m_sqr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
m_sqr_arr = np.array(m_sqr)
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [16]:
```
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]]
```

In [28]:
```
#Matlab
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

In [18]:
```
m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
m_diag_np = np.matrix(m_diag)
print(m_diag_np)
```

```
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

In [21]:
```
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
diag = np.diag(m_sqr_mx)
print(diag)
print()

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

```
[1 5 9]

[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

Единичная матрица

In [24]:
```
m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
m_e_np = np.matrix(m_e)
print(m_e_np)
```

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

In [26]:
```
m_eye = np.eye(3)
print(m_eye)
```

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

In [27]:
```
m_idnt = np.identity(3)
print(m_idnt)
```

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

Нулевая матрица

In [29]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

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

Задание матрицы в общем виде

In [32]:
```
m_mx = np.matrix('1 2 3; 4 5 6')
print(m_mx)
print()

m_var = np.zeros((2, 5))
print(m_var)
```

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

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

2. Как выполняется транспонирование матриц?

С помощью transpose()

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

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?

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?

Сложение матриц

```
A = np.matrix('1 6 3; 8 2 7')
B = np.matrix('8 1 5; 6 9 12')
C = A + B
print(C)
```

```
[[ 9  7  8]
 [14 11 19]]
```

Аналогично вычитание

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?

Умножение матриц

```
: 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]]
```

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.

Матричным метод

In [56]:
```
import numpy as np
slau = np.random.randint(0, 15, size = (3, 4))
slau
```

Out[56]:
```
array([[ 7, 14,  4,  8],
       [ 1,  0,  5,  6],
       [ 7,  3, 10,  5]])
```

In [62]:
```
slau_a = slau[:, 0:3]
slau_b = slau[:, 3]
x = np.ones((3, 1))

print("slau_a")
print(slau_a)
print("slau_b")
print(slau_b)
```

```
slau_a
[[ 7 14  4]
 [ 1  0  5]
 [ 7  3 10]]
slau_b
[8 6 5]
```

In [63]:
```
#Найдем определитель
A_det = np.linalg.det(slau_a)

if A_det != 0:
    #Найдем обратную матрицу
    A_inv = np.linalg.inv(slau_a)
    x = A_inv.dot(slau_b)
    print("Решения:")
    print(x)
else:
    print("Матрица вырожденная, нельзя продолжить")
```

```
Решения:
[-2.09338521  1.15564202  1.61867704]
```

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

Метод Крамера

In [46]:
```
slau = np.random.randint(0, 15, size = (3, 4))
```

In [60]:
```
slau_a = slau[:, 0:3]
slau_b = np.ones((3, 1))
x = np.ones((3, 1))
slau_b[0, 0] = slau[0, 3]
slau_b[1, 0] = slau[1, 3]
slau_b[2, 0] = slau[2, 3]

print("slau_a")
print(slau_a)
print("slau_b")
print(slau_b)
```

```
slau_a
[[ 7 14  4]
 [ 1  0  5]
 [ 7  3 10]]
slau_b
[[8.]
 [6.]
 [5.]]
```

In [61]:
```
#Найдем определитель
A_det = np.linalg.det(slau_a)

if A_det != 0:
    #Найдем дополнительные определители
    for i in range(3):
        A_dop = slau_a.copy()
        A_dop[:, i] = slau_b[:, 0]
        x[i,0] = round(np.linalg.det(A_dop), 3) / round(np.linalg.det(slau_a), 3)
    print("Решения:")
    print(x)
else:
    print("Матрица вырожденная, нельзя продолжить")
```

```
Решения:
[[-2.09338521]
 [ 1.15564202]
 [ 1.61867704]]
```