МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №3 по дисциплине технологии распознавания образов

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний

Ввод [1]:
```
import numpy as np
```

Вектор строка

Ввод [6]:
```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np )
```

```
[1 2]
```

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

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

Ввод [8]:
```
>>> v_hor_zeros_v2 = np.zeros((1, 5))
>>> print(v_hor_zeros_v2 )
```

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

Ввод [10]:
```
>>> 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

Ввод [11]:
```
>>> v_vert_np = np.array([[1], [2]])
>>> print(v_vert_np)
```

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

Ввод [12]:
```
>>> v_vert_zeros = np.zeros((5, 1))
>>> print(v_vert_zeros)
```

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

Ввод [13]:
```
>>> v_vert_ones = np.ones((5, 1))
>>> print(v_vert_ones)
```

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

Квадратная матрица

Ввод [14]:
```
>>> 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]]
```

Ввод [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]]
```

Ввод [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]]
```

Ввод [17]:
```
>>> 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

Диагональная матрица

Ввод [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]]
```

Ввод [19]:
```
>>> m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
```

Ввод [20]:
```
>>> diag = np.diag(m_sqr_mx)
>>> print(diag)
```
```
[1 5 9]
```

Ввод [21]:
```
>>> 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

Ввод [22]:
```
>>> 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]]
```

Ввод [23]:
```
>>> m_eye = np.eye(3)
>>> print(m_eye)
```
```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Ввод [24]:
```
>>> m_idnt = np.identity(3)
>>> print(m_idnt)
```

Нулевая матрица

Ввод [25]:
```
>>> m_zeros = np.zeros((3, 3))
>>> print(m_zeros)
```
```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Задание матрицы в общем виде

Ввод [26]:
```
>>> m_mx = np.matrix('1 2 3; 4 5 6')
>>> print(m_mx)
```
```
[[1 2 3]
 [4 5 6]]
```

Ввод [27]:
```
>>> m_var = np.zeros((2, 5))
>>> print(m_var)
```
```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Транспонирование матрицы

Ввод [28]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
```
```
[[1 2 3]
 [4 5 6]]
```

Ввод [29]:
```
>>> A_t = A.transpose()
>>> print(A_t)
```
```
[[1 4]
 [2 5]
 [3 6]]
```

Ввод [30]:
```
print(A.T)
```
```
[[1 4]
```

Действия над матрицами

Умножение матрицы на число

Ввод [38]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
```
```
[[ 3  6  9]
 [12 15 18]]
```

Ввод [39]:
```
>>> A = np.matrix('1 2; 3 4')
>>> L = 1 * A
>>> R = A
>>> print(L)
>>> print(R)
```
```
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]
```

Ввод [40]:
```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = 0 * A
>>> R = Z
>>> print(L)
>>> print(R)
```
```
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
```

Ввод [42]:
```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p + q) * A
>>> R = p * A + q * A
>>> print(L)
>>> print(R)
```
```
[[ 5 10]
 [15 20]]
[[ 5 10]
 [15 20]]
```

Сложение матриц

Ввод [45]:
```
>>> 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]]
```

Ввод [46]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A + B
>>> R = B + A
>>> print(L)
>>> print(R)
```

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

Ввод [47]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('1 7; 9 3')
>>> L = A + (B + C)
>>> R = (A + B) + C
>>> print(L)
>>> print(R)
```

```
[[ 7 15]
 [19 15]]
[[ 7 15]
 [19 15]]
```

Ввод [48]:
```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = A + (-1)*A
>>> print(L)
>>> print(Z)
```

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

Умножение матриц

Ввод [49]:
```
>>> 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]]
```

Ввод [50]:
```
>>> 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)
>>> print(R)
```

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

Ввод [51]:
```
>>> 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 + C)
>>> R = A.dot(B) + A.dot(C)
>>> print(L)
>>> print(R)
```

```
[[35 42]
 [77 94]]
[[35 42]
 [77 94]]
```

Ввод [52]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A.dot(B)
>>> R = B.dot(A)
>>> print(L)
>>> print(R)
```

```
[[19 22]
 [43 50]]
```

Определитель матрицы

Ввод [55]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
```
```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
```

Ввод [56]:
```
>>> np.linalg.det(A)
```
Out[56]:
```
-14.000000000000009
```

Ввод [57]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> print(A.T)
```
```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 10  8]
 [-1  4  3]
 [ 2 -1  1]]
```

Ввод [58]:
```
>>> det_A = round(np.linalg.det(A), 3)
>>> det_A_t = round(np.linalg.det(A.T), 3)
>>> print(det_A)
>>> print(det_A_t)
```
```
-14.0
-14.0
```

Ввод [59]:
```
>>> A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
>>> print(A)
>>> np.linalg.det(A)
```
```
[[-4 -1  2]
 [ 0  0  0]
 [ 8  3  1]]
```
Out[59]:
```
0.0
```

Ввод [60]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
```

Обратная матрица

Ввод [67]:
```
>>> A = np.matrix('1 -3; 2 5')
>>> A_inv = np.linalg.inv(A)
>>> print(A_inv)
```
```
[[ 0.45454545  0.27272727]
 [-0.18181818  0.09090909]]
```

Ввод [68]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> A_inv = np.linalg.inv(A)
>>> A_inv_inv = np.linalg.inv(A_inv)
>>> print(A)
>>> print(A_inv_inv)
```
```
[[ 1. -3.]
 [ 2.  5.]]
[[ 1. -3.]
 [ 2.  5.]]
```

Ввод [69]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> L = np.linalg.inv(A.T)
>>> R = (np.linalg.inv(A)).T
>>> print(L)
>>> print(R)
```
```
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
```

Ввод [70]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> B = np.matrix('7. 6.; 1. 8.')
>>> L = np.linalg.inv(A.dot(B))
>>> R = np.linalg.inv(B).dot(np.linalg.inv(A))
>>> print(L)
>>> print(R)
```
```
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
```

Ранг матрицы

Ранг матрицы

Ввод [71]:
```
>>> 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
```

Ввод [72]:
```
>>> m_eye[3][3] = 0
>>> 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. 0.]]
3
```

2. Примеры для свойств матричных вычислений

Дважды транспонированная матрица равна исходной матрице:

Ввод [2]:
```
import numpy as np
```

Ввод [5]:
```
>>> A = np.random.randint(-20, 100, (3, 3))
>>> print(A)
>>> R = (A.T).T
>>> print(R)
```

```
[[ 2  5 28]
 [29 53 94]
 [-5 63 71]]
[[ 2  5 28]
 [29 53 94]
 [-5 63 71]]
```

Транспонирование суммы матриц равно сумме транспонированных матриц:

Ввод [6]:
```
>>> A = np.random.randint(-20, 100, (2, 3))
>>> B = np.random.randint(-20, 100, (2, 3))
>>> L = (A + B).T
>>> R = A.T + B.T
>>> print(L)
>>> print(R)
```

```
[[102  50]
 [ 97 130]
 [122  60]]
[[102  50]
 [ 97 130]
 [122  60]]
```

Транспонирование произведения матриц равно произведению транспонированных матриц расставленных в обратном порядке:

Ввод [7]:
```
>>> A = np.random.randint(-20, 100, (2, 2))
>>> B = np.random.randint(-20, 100, (2, 2))
>>> L = (A.dot(B)).T
>>> R = (B.T).dot(A.T)
>>> print(L)
>>> print(R)
```
```
[[ 727 2809]
 [ 560  335]]
[[ 727 2809]
 [ 560  335]]
```

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

Ввод [8]:
```
>>> A = np.random.randint(-20, 100, (3, 2))
>>> k = 3
>>> L = (k * A).T
>>> R = k * (A.T)
>>> print(L)
>>> print(R)
```
```
[[282  93 132]
 [ -9 -36 183]]
[[282  93 132]
 [ -9 -36 183]]
```

Определители исходной и транспонированной матрицы совпадают:

Ввод [9]:
```
>>> A = np.random.randint(-20, 100, (5, 5))
>>> A_det = np.linalg.det(A)
>>> A_T_det = np.linalg.det(A.T)
>>> print(format(A_det, '.9g'))
>>> print(format(A_T_det, '.9g'))
```
```
125000169
125000169
```

Произведение единицы и любой заданной матрицы равно заданной матрице:

Ввод [10]:
```
>>> A = np.random.randint(5, 100, (2, 2))
>>> L = 1 * A
>>> R = A
>>> print(L)
>>> print(R)
```
```
[[93 59]
 [88  5]]
[[93 59]
 [88  5]]
```

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

Ввод [12]:
```
>>> A = np.random.randint(-20, 100, (2, 2))
>>> Z = np.matrix('0 0; 0 0')
>>> L = 0 * A
>>> R = Z
>>> print(L)
>>> print(R)
```
```
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
```

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

Ввод [13]:
```
>>> A = np.random.randint(-20, 100, (4, 7))
>>> p = 2
>>> q = 3
>>> L = (p + q) * A
>>> R = p * A + q * A
>>> print(L)
>>> print(R)
```
```
[[145 465 250  80 415 230 400]
```

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

Ввод [14]:
```
>>> A = np.random.randint(-20, 100, (2, 7))
>>> p = 2
>>> q = 3
>>> L = (p * q) * A
>>> R = p * (q * A)
>>> print(L)
>>> print(R)
```

```
[[  60  474  330  378  210   54  258]
 [ 426   72  432  384 -108  -84  246]]
[[  60  474  330  378  210   54  258]
 [ 426   72  432  384 -108  -84  246]]
```

Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число:

Ввод [16]:
```
>>> A = np.random.randint(50, 100, (5, 3))
>>> B = np.random.randint(30, 100, (5, 3))
>>> k = 3
>>> L = k * (A + B)
>>> R = k * A + k * B
>>> print(L)
>>> print(R)
```

```
[[459 378 348]
 [321 393 483]
 [342 447 351]
 [351 366 435]
 [474 333 396]]
[[459 378 348]
 [321 393 483]
 [342 447 351]
 [351 366 435]
 [474 333 396]]
```

Коммутативность сложения. От перестановки матриц их сумма не изменяется:

Ввод [17]:
```
>>> A = np.random.randint(-10, 100, (3, 3))
>>> B = np.random.randint(-19, 100, (3, 3))
>>> L = A + B
>>> R = B + A
>>> print(L)
>>> print(R)
```

```
[[120 150  51]
 [ 44  66  72]
 [153  37 112]]
[[120 150  51]
 [ 44  66  72]
 [153  37 112]]
```

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

Ввод [18]:
```
>>> A = np.random.randint(-20, 50, (5, 2))
>>> B = np.random.randint(0, 50, (5, 2))
>>> C = np.random.randint(-20, 50, (5, 2))
>>> L = A + (B + C)
>>> R = (A + B) + C
>>> print(L)
>>> print(R)
```

```
[[97  9]
 [78 88]
 [21 96]
 [76 84]
 [36 56]]
[[97  9]
 [78 88]
 [21 96]
 [76 84]
 [36 56]]
```

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

Ввод [19]:
```
>>> A = np.random.randint(0, 50, (2, 2))
>>> Z = np.matrix('0 0; 0 0')
>>> L = A + (-1)*A
>>> print(L)
>>> print(Z)
```

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

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

Ввод [20]:
```
>>> A = np.random.randint(-7, 50, (3, 3))
>>> B = np.random.randint(-7, 50, (3, 3))
>>> C = np.random.randint(-7, 50, (3, 3))
>>> L = A.dot(B.dot(C))
>>> R = (A.dot(B)).dot(C)
>>> print(L)
>>> print(R)
```

```
[[121648  27334  63100]
 [ 51026  11158  24370]
 [ 56020  12568  28552]]
[[121648  27334  63100]
 [ 51026  11158  24370]
 [ 56020  12568  28552]]
```

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

Ввод [21]:
```
>>> A = np.random.randint(-20, 50, (2, 2))
>>> B = np.random.randint(-20, 50, (2, 2))
>>> C = np.random.randint(-20, 50, (2, 2))
>>> L = A.dot(B + C)
>>> R = A.dot(B) + A.dot(C)
>>> print(L)
>>> print(R)
```

```
[[ 500  700]
 [-292 -308]]
[[ 500  700]
 [-292 -308]]
```

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

Ввод [22]:
```
>>> A = np.random.randint(-20, 100, (5, 5))
>>> B = np.random.randint(-20, 100, (5, 5))
>>> L = A.dot(B)
>>> R = B.dot(A)
>>> print(L)
>>> print(R)
```

```
[[ 6528  9242  7567   423 11170]
 [13833 16291  3156  7395 17196]
 [ 3329  9679  3463  4272  9356]
 [ 6638 11207  7364   564  9548]
 [10496 15344  7517  2385 14334]]
[[ 9328  5956 10972  9496 12654]
 [ 9567  8068  4806  7618  9016]
 [ 9679 10040  5783  8621 13522]
 [ 7435  4072 14676  6350 11828]
 [10045  6686 11107  6751 11651]]
```

Произведение заданной матрицы на единичную равно исходной матрице:

Ввод [23]:
```
>>> A = np.random.randint(-20, 100, (2, 2))
>>> E = np.matrix('1 0; 0 1')
>>> L = E.dot(A)
>>> R = A.dot(E)
>>> print(L)
>>> print(R)
>>> print(A)
```

```
[[44 47]
 [76 55]]
[[44 47]
 [76 55]]
[[44 47]
 [76 55]]
```

Произведение заданной матрицы на нулевую матрицу равно нулевой матрице:

Ввод [24]:
```
>>> A = np.random.randint(50, 100, (4, 4))
>>> Z = np.zeros((4, 4))
>>> L = Z.dot(A)
>>> R = A.dot(Z)
>>> print(L)
>>> print(R)
>>> print(Z)
```

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

Определитель матрицы остается неизменным при ее транспонировании:

Ввод [26]:
```
>>> A = np.random.randint(-6, 100, (5, 5))
>>> print(A)
>>> print(A.T)
>>> det_A = round(np.linalg.det(A), 3)
>>> det_A_t = round(np.linalg.det(A.T), 3)
>>> print(det_A)
>>> print(det_A_t)
```

```
[[67 60 80 83 45]
 [-6 69 60 47 51]
 [91 18 45 48 87]
 [11 89 34  5 13]
 [10 13 42  9 86]]
[[67 -6 91 11 10]
 [60 69 18 89 13]
 [80 60 45 34 42]
 [83 47 48  5  9]
 [45 51 87 13 86]]
701067603.0
701067603.0
```

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

Ввод [28]:
```
>>> A = np.matrix('243 22 1512 4242; 0 0 0 0; 32 7 9 4; 42 1214 455 522')
>>> print(A)
>>> np.linalg.det(A)
```

```
[[ 243   22 1512 4242]
 [   0    0    0    0]
 [  32    7    9    4]
 [  42 1214  455  522]]
```

Out[28]:
```
0.0
```

При перестановке строк матрицы знак ее определителя меняется на противоположный:

Ввод [34]:
```
>>> A = np.matrix('-90 134; 1141 -1')
>>> print(A)
>>> B = np.matrix('1141 -1; -90 134')
>>> print(B)
>>> print(round(np.linalg.det(A), 3))
>>> round(np.linalg.det(B), 3)
```

```
[[ -90  134]
 [1141   -1]]
[[1141   -1]
 [ -90  134]]
-152804.0
```

Out[34]:
```
152804.0
```

Если у матрицы есть две одинаковые строки, то ее определитель равен нулю:

Ввод [35]:
```
>>> A = np.matrix('-4 -1 2 453; -4 -1 2 363; 442 22 11 32; 442 22 11 32')
>>> print(A)
>>> np.linalg.det(A)
```

```
[[ -4  -1   2 453]
 [ -4  -1   2 363]
 [442  22  11  32]
 [442  22  11  32]]
```

Out[35]:
```
0.0
```

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

Ввод [37]:
```
>>> A = np.random.randint(-15, 90, (7, 7))
>>> print(A)
>>> k = 2
>>> B = A.copy()
```

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

Ввод [37]:
```
>>> A = np.random.randint(-15, 90, (7, 7))
>>> print(A)
>>> k = 2
>>> B = A.copy()
>>> B[2, :] = k * B[2, :]
>>> print(B)
>>> det_A = round(np.linalg.det(A), 3)
>>> det_B = round(np.linalg.det(B), 3)
>>> print(det_A * k)
>>> det_B
```

```
[[ 49  40  70  26  60  -6  25]
 [ -3  68  88  83  39  -1  62]
 [ 23 -11  54   6 -11 -10  -7]
 [ 48  55  76  20  -2 -11  52]
 [ 84  44  15  65  39  20  15]
 [ 73  42  66  78  76  41  62]
 [ 61  80  31   4  77  64   4]]
[[ 49  40  70  26  60  -6  25]
 [ -3  68  88  83  39  -1  62]
 [ 46 -22 108  12 -22 -20 -14]
 [ 48  55  76  20  -2 -11  52]
 [ 84  44  15  65  39  20  15]
 [ 73  42  66  78  76  41  62]
 [ 61  80  31   4  77  64   4]]
-15279913856789.982
```

Out[37]:
```
-15279913856790.01
```

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

Ввод [39]:
```
>>> A = np.random.randint(-15, 90, (4, 4))
>>> B = np.random.randint(-15, 90, (4, 4))
>>> C = A.copy()
>>> C[1, :] += B[1, :]
>>> print(C)
>>> print(A)
>>> print(B)
>>> print(round(np.linalg.det(C), 3))
>>> round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
```

```
[[83 20 70 72]
 [75 33 35 83]
 [ 2 72 61 11]
 [72 62 18  6]]
[[83 20 70 72]
 [75 16 29 77]
 [ 2 72 61 11]
 [72 62 18  6]]
[[-5  0 68 85]
 [ 0 17  6  6]
 [60 23 87 74]
 [12 43 17 51]]
-22590248.0
```

Out[39]: `-20810569.0`

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

Ввод [40]:
```
>>> A = np.random.randint(7, 90, (3, 3))
>>> k = 2
>>> B = A.copy()
>>> B[1, :] = B[1, :] + k * B[0, :]
>>> print(A)
>>> print(B)
>>> print(round(np.linalg.det(A), 3))
>>> round(np.linalg.det(B), 3)
```

```
[[49 35 21]
```

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

Ввод [41]:
```
>>> A = np.matrix('535 -241 22; 1 24 -1; 526 5 1')
>>> print(A)
>>> k = 2
>>> A[1, :] = A[0, :] + k * A[2, :]
>>> round(np.linalg.det(A), 3)
```

```
[[ 535 -241   22]
 [   1   24   -1]
 [ 526    5    1]]
```

Out[41]: `0.0`

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

Ввод [42]:
```
>>> A = np.random.randint(-87, 90, (6, 6))
>>> print(A)
>>> k = 2
>>> A[1, :] = k * A[0, :]
>>> print(A)
>>> round(np.linalg.det(A), 3)
```

```
[[ 32  35 -79 -13  -2  24]
 [-25  56 -10 -84  42 -26]
 [ 63 -61  -3  47 -86  85]
 [ 22 -43 -69  -2  57  34]
 [-54  -2  23 -56 -32   9]
 [-80  11  16 -87  84  -2]]
[[  32   35  -79  -13   -2   24]
 [  64   70 -158  -26   -4   48]
 [  63  -61   -3   47  -86   85]
 [  22  -43  -69   -2   57   34]
 [ -54   -2   23  -56  -32    9]
 [ -80   11   16  -87   84   -2]]
```

Out[42]: `0.0`

Обратная матрица обратной матрицы есть исходная матрица:

Ввод [44]:
```
>>> A = np.random.uniform(-87., 90., (4, 4))
>>> A_inv = np.linalg.inv(A)
>>> A_inv_inv = np.linalg.inv(A_inv)
>>> print(A)
>>> print(A_inv_inv)
```

```
[[-77.51297809  36.9903236   37.91831909 -57.66060159]
 [ -1.49275301  48.38315589  68.34564781  -5.42582336]
 [-71.295893   -56.50234838  47.53848087 -41.08487581]
 [ 69.18670277 -28.66419052 -14.3503396   51.63565556]]
[[-77.51297809  36.9903236   37.91831909 -57.66060159]
 [ -1.49275301  48.38315589  68.34564781  -5.42582336]
 [-71.295893   -56.50234838  47.53848087 -41.08487581]
 [ 69.18670277 -28.66419052 -14.3503396   51.63565556]]
```

Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы:

Ввод [46]:
```
>>> A = np.random.uniform(-7., 90., (3, 3))
>>> L = np.linalg.inv(A.T)
>>> R = (np.linalg.inv(A)).T
>>> print(L)
>>> print(R)
```

```
[[-0.01803632  0.07557992 -0.06375944]
 [ 0.05061056 -0.06537994  0.01589703]
 [-0.0238444  -0.03909121  0.10282702]]
[[-0.01803632  0.07557992 -0.06375944]
 [ 0.05061056 -0.06537994  0.01589703]
 [-0.0238444  -0.03909121  0.10282702]]
```

Обратная матрица произведения матриц равна произведению обратных матриц:

Ввод [47]:
```
>>> A = np.random.uniform(3., 90., (3, 3))
>>> B = np.random.uniform(50., 90., (3, 3))
>>> L = np.linalg.inv(A.dot(B))
>>> R = np.linalg.inv(B).dot(np.linalg.inv(A))
>>> print(L)
>>> print(R)
```

```
[[ 0.00539561 -0.007298    0.00118221]
 [-0.00441834  0.00759437 -0.00227042]
 [-0.00096821 -0.00051423  0.0013198 ]]
[[ 0.00539561 -0.007298    0.00118221]
 [-0.00441834  0.00759437 -0.00227042]
 [-0.00096821 -0.00051423  0.0013198 ]]
```

3. Примеры решения СЛУ матричным методом и методом Крамера

Матричный метод: (рис 1-2).

Нахождение уравнения плоскости по точкам, через которые она проходит.

Уравнение плоскости в 3-х мерном пространстве задаётся уравнением:

$$z = ax + by + c$$

Уравнение плоскости однозначно задаётся 3 точками через которые она проходит. Таким образом легко понять, что если мы знаем координаты точек, через которые проходит плоскость, то в уравнении выше у вас 3 переменных: a, b, c. А значения x, y, z нам известны для 3 точек. Если плоскость проходит через точки (1;-6;1), (0;-3;2) и (-3;0;-1), то мы легко можем найти коэффициенты, подставив значения соответствующих координат для всех 3 точек в уравнение выше и получить систему из 3 уравнений, которая затем используется для составления матрицы:

$$\begin{pmatrix} 1 & -6 & 1 \\ 0 & -3 & 1 \\ -3 & 0 & 1 \end{pmatrix}$$

А также вектор свободных членов:

$$\begin{pmatrix} 1 & 2 & -1 \end{pmatrix}$$

Ввод [5]:
```python
import numpy as np

m = np.array([[1., -6., 1.], [0., -3., 1], [-3, 0, 1]])
v = np.array([1., 2., -1.])

np.linalg.solve(m, v)
```

Out[5]:
```
array([2., 1., 5.])
```

Ответ: уравнение искомой плоскости в пространстве задаётся уравнением z = 2x + y + 5

Рисунок 1 – Матричный метод для нахождения уравнения плоскости

Произвольные матрицы уравнений и решений

Ввод [8]:
```python
matrix = np.random.uniform(-20., 100., (4, 4))
print(matrix)
vec = np.random.uniform(-20., 100., (4, 1))
print(vec)
np.linalg.solve(matrix, vec)
```

```
[[ 43.34951311 -19.94224474  54.21055599  28.54922287]
 [ 47.77493978   1.81376561  53.90381902  31.31676901]
 [ 91.31541817  16.3778718   -6.55630636  26.26389007]
 [-10.90239254   8.5651597   71.74041152  52.82606331]]
[[43.6299613 ]
 [-2.70685347]
 [13.95645529]
 [-0.44322837]]
```

Out[8]:
```
array([[ 0.01714457],
       [-2.36362512],
       [-0.9749016 ],
       [ 1.70234826]])
```

Рисунок 2 – Решение произвольных матриц матричным методом

Метод Крамера: (рис 3-4).

Метод Крамера

Ввод [1]:
```
import numpy as np
```

Ввод [5]:
```
n = np.random.randint(1, 7)
a = np.random.uniform(-20., 100., (n, n))
b = np.random.uniform(-20., 100., (n, 1))
print(f"The matrix of the equation:\n {a}")
print(f"The matrix of the free terms of the equation:\n {b}")
det = np.linalg.det(a)
print(f"The main determinant: {det}")
print("The Kramer algorithm: ")
for j in range(n):
    a_sav = a.copy()
    a_sav[::, j] = b[::, 0]
    print(a_sav)
    det_x = np.linalg.det(a_sav)
    print(f"The minor determinant number {j + 1}: {det_x}")
    x = det_x / det
    print(f"The {j + 1} root of the equation: {x}")
```

```
The matrix of the equation:
 [[ 24.88253275  82.74066442  81.06812728  54.88462965  74.40965796]
 [ 36.17160166  25.02902067 -10.45753698  97.6267258   77.76361442]
 [ 81.77580652  69.50578164   0.23114888  62.2449725   18.75661774]
 [ 20.48483616  36.51403248  49.22799368  59.06105178 -12.93240583]
 [ 37.51011042  77.46890253  79.2949131   -2.60142186  -8.15088509]]
The matrix of the free terms of the equation:
 [[  3.78095474]
 [-19.574148  ]
 [ 75.57543813]
 [ 33.53440075]
 [ 44.12447613]]
The main determinant: -375901204.6272254
The Kramer algorithm:
[[  3.78095474  82.74066442  81.06812728  54.88462965  74.40965796]
 [-19.574148    25.02902067 -10.45753698  97.6267258   77.76361442]
 [ 75.57543813  69.50578164   0.23114888  62.2449725   18.75661774]
 [ 33.53440075  36.51403248  49.22799368  59.06105178 -12.93240583]
 [ 44.12447613  77.46890253  79.2949131   -2.60142186  -8.15088509]]
The minor determinant number 1: 653958230.7048887
The 1 root of the equation: -1.7397077281340654
[[ 24.88253275   3.78095474  81.06812728  54.88462965  74.40965796]
 [ 36.17160166 -19.574148   -10.45753698  97.6267258   77.76361442]
 [ 81.77580652  75.57543813   0.23114888  62.2449725   18.75661774]
```

Рисунок 3 – Решение СЛУ методом Крамера, алгоритм программы

```
The matrix of the equation:
 [[ 24.88253275  82.74066442  81.06812728  54.88462965  74.40965796]
 [ 36.17160166  25.02902067 -10.45753698  97.6267258   77.76361442]
 [ 81.77580652  69.50578164   0.23114888  62.2449725   18.75661774]
 [ 20.48483616  36.51403248  49.22799368  59.06105178 -12.93240583]
 [ 37.51011042  77.46890253  79.2949131   -2.60142186  -8.15088509]]
The matrix of the free terms of the equation:
 [[  3.78095474]
 [-19.574148  ]
 [ 75.57543813]
 [ 33.53440075]
 [ 44.12447613]]
The main determinant: -375901204.6272254
The Kramer algorithm:
[[  3.78095474  82.74066442  81.06812728  54.88462965  74.40965796]
 [-19.574148    25.02902067 -10.45753698  97.6267258   77.76361442]
 [ 75.57543813  69.50578164   0.23114888  62.2449725   18.75661774]
 [ 33.53440075  36.51403248  49.22799368  59.06105178 -12.93240583]
 [ 44.12447613  77.46890253  79.2949131   -2.60142186  -8.15088509]]
The minor determinant number 1: 653958230.7048887
The 1 root of the equation: -1.7397077281340654
[[ 24.88253275   3.78095474  81.06812728  54.88462965  74.40965796]
 [ 36.17160166 -19.574148   -10.45753698  97.6267258   77.76361442]
 [ 81.77580652  75.57543813   0.23114888  62.2449725   18.75661774]
 [ 20.48483616  33.53440075  49.22799368  59.06105178 -12.93240583]
 [ 37.51011042  44.12447613  79.2949131   -2.60142186  -8.15088509]]
The minor determinant number 2: -1153867214.5544217
The 2 root of the equation: 3.0696023326094193
[[ 24.88253275  82.74066442   3.78095474  54.88462965  74.40965796]
 [ 36.17160166  25.02902067 -19.574148    97.6267258   77.76361442]
 [ 81.77580652  69.50578164  75.57543813  62.2449725   18.75661774]
 [ 20.48483616  36.51403248  33.53440075  59.06105178 -12.93240583]
 [ 37.51011042  77.46890253  44.12447613  -2.60142186  -8.15088509]]
The minor determinant number 3: 650594301.1322994
The 3 root of the equation: -1.7307587555551527
[[ 24.88253275  82.74066442  81.06812728   3.78095474  74.40965796]
 [ 36.17160166  25.02902067 -10.45753698 -19.574148    77.76361442]
 [ 81.77580652  69.50578164   0.23114888  75.57543813  18.75661774]
 [ 20.48483616  36.51403248  49.22799368  33.53440075 -12.93240583]
 [ 37.51011042  77.46890253  79.2949131   44.12447613  -8.15088509]]
The minor determinant number 4: -168304918.31256792
The 4 root of the equation: 0.4477371081571099
[[ 24.88253275  82.74066442  81.06812728  54.88462965   3.78095474]
 [ 36.17160166  25.02902067 -10.45753698  97.6267258  -19.574148  ]
 [ 81.77580652  69.50578164   0.23114888  62.2449725   75.57543813]
 [ 20.48483616  36.51403248  49.22799368  59.06105178  33.53440075]
 [ 37.51011042  77.46890253  79.2949131   -2.60142186  44.12447613]]
The minor determinant number 5: 460601888.12900585
The 5 root of the equation: -1.2253269807575546
```

Рисунок 4 – Результат работы программы для случайного числа n

4. Ответы на вопросы

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

Вектор

Вектором называется матрица, у которой есть только один столбец или одна строка.

Вектор-строка

Вектор-строка имеет следующую математическую запись.

$$v = (1\ 2)$$

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

Вектор-столбец

Вектор-столбец имеет следующую математическую запись.

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

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

Квадратная матрица

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]])

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

m_sqr_mx = np.matrix('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.

m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]

m_diag_np = np.matrix(m_diag)

diag = np.diag(m_sqr_mx)

Единичная матрица

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

m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

m_e_np = np.matrix(m_e)

m_eye = np.eye(3)

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)

Нулевая матрица

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. Полученная в результате матрица называется транспонированной. Символ операции транспонирования – буква T.

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?

Функция transpose() или A.T

5. Какие существуют основные действия над матрицами?

Умножение матрицы на число

Сложение матриц

Умножение матриц

Определитель матрицы

Обратная матрица

Ранг матрицы

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?

$$A = \begin{pmatrix} 1 & 6 & 3 \\ 8 & 2 & 7 \end{pmatrix},\ B = \begin{pmatrix} 8 & 1 & 5 \\ 6 & 9 & 12 \end{pmatrix},$$

$$C = A + B,$$

$$C = \begin{pmatrix} 1+8 & 6+1 & 3+5 \\ 8+6 & 2+9 & 7+12 \end{pmatrix} = \begin{pmatrix} 9 & 7 & 8 \\ 14 & 11 & 19 \end{pmatrix}.$$

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?

Знак +

11. Как осуществляется операция умножения матриц?

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?

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 А 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?

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 – 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 строится на базе исходной A путем замены всех элементов матрицы A на их алгебраические дополнения.

Транспонируя союзную, мы получим так называемую присоединенную матрицу.

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

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?

Функция inv()

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений.

Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Создаём матрицу СЛАУ и матрицу свободных коэффициентов, находим главный определитель матрицы (det()), в цикле от 1 до n меняем столбцы матрицы СЛАУ на столбец из матрицы свободных коэффициентов, находим определители полученных матрицы, находим корни уравнения путём деления определителей полученных матриц на главный определитель.

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений.

Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy.

Создаём матрицу СЛАУ и матрицу свободных коэффициентов. Для решения применяется средство numpy: numpy.linalg.solve().