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

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.3
Исследование методов работы с матрицами и векторами с помощью библиотеки NumPy

Выполнил студент группы ИВТ-б-о-21-1

Пентухов С. А. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь, 2023

Цель работы: Исследовать базовые возможности библиотеки NumPy языка программирования Python

Ссылка https://github.com/Pentuhov/Lab3.3

Ход работы

1. Проработать примеры из лабораторной

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

Вектор строка

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

```
In [3]: v_hor_zeros_v1 = np.zeros((5,))
        print(v_hor_zeros_v1 )
```
```
[0. 0. 0. 0. 0.]
```

```
In [4]: v_hor_zeros_v2 = np.zeros((1, 5))
        print(v_hor_zeros_v2 )
```
```
[[0. 0. 0. 0. 0.]]
```

```
In [5]: 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 [6]: v_vert_np = np.array([[1], [2]])
        print(v_vert_np)
```
```
[[1]
 [2]]
```

```
In [7]: v_vert_zeros = np.zeros((5, 1))
        print(v_vert_zeros)
```
```
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

Рис. 1 Пример Вектора строк и столбца

Квадратная матрица

```
In [9]: 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 [10]: 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 [11]: 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 [12]: 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 [13]: 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 [14]: m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
```

```
In [15]: diag = np.diag(m_sqr_mx)
         print(diag)
```
```
[1 5 9]
```

Рис. 2 Пример матриц

In [16]:
```
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 [17]:
```
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 [18]:
```
m_eye = np.eye(3)
print(m_eye)
```

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

In [19]:
```
m_idnt = np.identity(3)
print(m_idnt)
```

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

Нулевая матрица

In [20]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

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

Рис. 3 Пример нулевой матрицы

Задание матрицы в общем виде

In [21]:
```
m_mx = np.matrix('1 2 3; 4 5 6')
print(m_mx)
```

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

In [22]:
```
m_var = np.zeros((2, 5))
print(m_var)
```

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

Транспонирование матрицы

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

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

In [24]:
```
A_t = A.transpose()
print(A_t)
```

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

In [25]:
```
print(A.T)
```

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

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

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

In [27]:
```
R = (A.T).T
print(R)
```

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

Рис. 4 Пример Транспортирование матрицы

In [30]:
```
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
L = (A.dot(B)).T
R = (B.T).dot(A.T)
print(L)
print(R)
```

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

In [31]:
```
A = np.matrix('1 2 3; 4 5 6')
k = 3
L = (k * A).T
R = k * (A.T)
print(L)
print(R)
```

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

In [32]:
```
A = np.matrix('1 2; 3 4')
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'))
```

```
-2
-2
```

Действия над матрицами

Умножение матрицы на число

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

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

Рис. 5 Пример действия над матрицами

In [34]:
```
A = np.matrix('1 2; 3 4')
L = 1 * A
R = A
print(L)
print(R)
```

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

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

In [36]:
```
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]]
```

In [37]:
```
A = np.matrix('1 2; 3 4')
p = 2
q = 3
L = (p * q) * A
R = p * (q * A)
print(L)
print(R)
```

```
[[ 6 12]
 [18 24]]
[[ 6 12]
 [18 24]]
```

Рис. 6 Пример действия над матрицами

Сложение матриц

In [39]:
```
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 [40]:
```
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]]
```

In [41]:
```
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]]
```

In [42]:
```
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]]
```

Рис. 7 Пример сложений матриц

Умножение матриц

In [43]:
```
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]]
```

In [44]:
```
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]]
```

In [45]:
```
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]]
```

In [46]:
```
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]]
[[23 34]
 [31 46]]
```

In [47]:
```
A = np.matrix('1 2; 3 4')
E = np.matrix('1 0; 0 1')
L = E.dot(A)
```

Рис. 8 Пример умножения матриц

Определитель матрицы

In [49]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
```

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

In [50]:
```
np.linalg.det(A)
```

Out[50]: -14.000000000000009

In [51]:
```
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]]
```

In [52]:
```
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
```

In [53]:
```
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[53]: 0.0

In [54]:
```
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')
print(B)
round(np.linalg.det(A), 3)
round(np.linalg.det(B), 3)
```

Рис. 9 Пример определения матрицы

In [55]:
```
A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
print(A)
np.linalg.det(A)
```

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

Out[55]: 0.0

In [56]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
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)
det_A * k
det_B
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [10  4 -1]
 [16  6  2]]
```

Out[56]: -28.0

In [57]:
```
A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
B = np.matrix('-4 -1 2; 8 3 2; 8 3 1')
C = A.copy()
C[1, :] += B[1, :]
print(C)
print(A)
print(B)
round(np.linalg.det(C), 3)
round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
```

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

Out[57]: 4.0

Рис. 10 Пример определения матрицы

In [58]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
k = 2
B = A.copy()
B[1, :] = B[1, :] + k * B[0, :]
print(A)
print(B)
round(np.linalg.det(A), 3)
round(np.linalg.det(B), 3)
```

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

Out[58]: -14.0

In [59]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
k = 2
A[1, :] = A[0, :] + k * A[2, :]
round(np.linalg.det(A), 3)
```

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

Out[59]: 0.0

In [60]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
k = 2
A[1, :] = k * A[0, :]
print(A)
round(np.linalg.det(A), 3)
```

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

Out[60]: 0.0

Рис. 11 Пример определения матрицы

Обратная матрица

In [61]:
```
A = np.matrix('1 -3; 2 5')
A_inv = np.linalg.inv(A)
print(A_inv)
```

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

In [62]:
```
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.]]
```

In [63]:
```
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]]
```

In [64]:
```
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]]
```

Рис. 12 Пример обратная матрица

Ранг матрицы

In [65]:
```
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
```

In [66]:
```
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
```

Рис. 13 Пример ранга матрицы

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

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

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

Транспонирование суммы матриц равно сумме транспонированных матриц:

In [3]:
```
A = np.random.randint(-20, 100, (3, 3))
B = np.random.randint(-20, 100, (3, 3))
print(f"A\n{A},\nB\n{B}")

print("Транспонирование суммы матриц:")
L = (A + B).T
print(L)

print("Сумма транспонированных матриц:")
R = A.T + B.T
print(R)
```

```
A
[[ 65  94  17]
 [-14  64  47]
 [ 27 -13   1]],
B
[[ 94 -18  44]
 [ 37  47   5]
 [ 36  97  15]]
Транспонирование суммы матриц:
[[159  23  63]
 [ 76 111  84]
 [ 61  52  16]]
Сумма транспонированных матриц:
[[159  23  63]
 [ 76 111  84]
 [ 61  52  16]]
```

Действия над матрицами

Умножение матриц на число

Рис. 14 – Решение заданий 1

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

In [5]:
```
A = np.random.randint(0, 20, (2, 2))
B = np.random.randint(0, 20, (2, 2))
C = np.random.randint(0, 20, (2, 2))
print(f"A\n{A} \nB\n{B} \nC\n{C}")

print("A + (B + C)")
L = A + (B + C)
print(L)
print("(A + B) + C")
R = (A + B) + C
print(R)
```

```
A
[[ 5  2]
 [10 11]]
B
[[5 3]
 [7 4]]
C
[[ 9 19]
 [ 4  7]]
A + (B + C)
[[19 24]
 [21 22]]
(A + B) + C
[[19 24]
 [21 22]]
```

Умножение матриц

Произведение заданной матрицы на единичную равно исходной матрице:

In [6]:
```
A = np.random.randint(0, 10, (3, 3))
E = np.eye(3)
print(f"A \n{A}")

print("E*A")
L = E.dot(A)
print(L)
print("A*E")
R = A.dot(E)
print(R)
```

```
A
[[8 9 3]
 [1 6 1]
```

Рис. 15 – Решение заданий 2

Определитель матрицы

In [28]:
```
A = np.matrix('6 2 4; 2 1 9; 4 3 2')
print(f"A \n{A}")
print(round(np.linalg.det(A), 3))

k = 4
A[1, :] = k * A[0, :]
print(A)
print("△A")
print(round(np.linalg.det(A), 3))
```

```
A
[[6 2 4]
 [2 1 9]
 [4 3 2]]
-78.0
[[ 6  2  4]
 [24  8 16]
 [ 4  3  2]]
△A
0.0
```

Обратная матрица

Обратная матрица обратной матрицы

In [7]:
```
A = np.random.uniform(0, 10, (2, 2))
print(A)
A_inv = np.linalg.inv(A)
A_inv_inv = np.linalg.inv(A_inv)
print(A_inv_inv)
```

```
[[3.84695618 8.32466996]
 [8.03941452 5.17660158]]
[[3.84695618 8.32466996]
 [8.03941452 5.17660158]]
```

In [ ]:

Рис. 16 – Решение заданий 3

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

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

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

Метод решения матричным методом
Записываем уравнение в виде $AX = B$. Выражаем X: $X = A ^{-1} \times B$. Если определитель матрицы A больше 0, то находим обратную матрицу. Затем умножаем обратную матрицу A на матрицу свободных членов B, получаем ответ.

$$X = A^{-1} \cdot B$$

Пусть задана система уравнений:

$$\begin{cases} 2x_1 - 4x_2 + 3x_3 = 1 \\ 2x_1 - 2x_2 + 4x_3 = 3 \\ 3x_1 - x_2 + 5x_3 = 2 \end{cases}$$

Приведем систему к матричному виду:

$$\left(\begin{array}{ccc|c} 2 & -4 & 3 & 1 \\ 2 & -2 & 4 & 3 \\ 3 & -1 & 5 & 2 \end{array}\right)$$

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

In [2]:
```
A = np.array([[2., 2., 3.], [-4., -2., -1], [3, 4, 5]])
B = np.array([1., 3., 2.])

# Определитель
det_A = np.linalg.det(A)

if det_A != 0:
    # Обратная матрица
    inv_A = np.linalg.inv(A)
    # Присоединенная матрица, решения
    result = inv_A.dot(B)
    print(result)
else:
    print("Матричный метод не подходит для решения этой системы!")
```

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

Метод решения СЛАУ методом Крамера

Рис. 17 – Решение домашнего задания 1

Теорема Крамера: Если определитель матрицы $\triangle$ квадратной системы не равен нулю, то система совместна и имеет единственное решение, которое находится по формулам Крамера: $x_1 = \frac{\triangle_1}{\triangle}; x_2 = \frac{\triangle_2}{\triangle}; x_3 = \frac{\triangle_3}{\triangle}$, где $\triangle_i$ - определитель матрицы системы, где вместо i-го столбца стоит столбец правых частей.

В качестве примера возьмем СЛАУ, решенную выше матричным методом:

$$\begin{cases} 2x_1 - 4x_2 + 3x_3 = 1 \\ 2x_1 - 2x_2 + 4x_3 = 3 \\ 3x_1 - x_2 + 5x_3 = 2 \end{cases}$$

Приведем систему к матричному виду:

$$\left(\begin{array}{ccc|c} 2 & -4 & 3 & 1 \\ 2 & -2 & 4 & 3 \\ 3 & -1 & 5 & 2 \end{array}\right)$$

Найдем определитель $\triangle A$, побочные определители для каждого столбца.

In [3]:
```
A = np.array([[2., 2., 3.], [-4., -2., -1], [3, 4, 5]])
B = np.array([1., 3., 2.])

# считаем главный определитель
det_A = np.linalg.det(A)

if (det_A != 0):
    # Побочные определители
    kram_dets = []
    for i in range(len(A)):
        copied_A = np.array(A)
        copied_A[:, i] = B
        kram_dets.append(np.linalg.det(copied_A))
    # Решения
    for det_i in kram_dets:
        print(float(det_i) / det_A)
else:
    print("Матричный метод не подходит для решения этой системы!")
```

```
-1.0000000000000004
-0.0
1.0
```

Результаты решений СЛАУ обоими способами совпали

Рис. 18 – Решение домашнего задания 2

Вывод: В результате выполнения работы были исследованы методы работы с матрицами и векторами с помощью библиотеки NumPy.

Контрольные 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_hor_np = np.array([1, 2])

Вектор-столбец

Вектор-столбец имеет следующую математическую запись. v_vert_np = np.array([[1], [2]])

Квадратная матрица

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 Т.

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?

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?

np.add, np.subtract или знаки +, -.

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?

np.dot(), np.multiply().

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?

Функция det() из пакета linalg.

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

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству:

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.

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.

Алгоритм и решение приведено в лабораторной работе.

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

Алгоритм и решение приведено в лабораторной работе.