Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины
«Основы теории электромагнитных полей и волн»
Вариант 13

Выполнил:
Милькевич Александр Александрович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи, направленность (профиль) «Интеллектуальная обработка данных в инфокоммуникационных системах и сетях», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Хацукова А.И

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Основы работы с библиотекой NumPy

Цель: исследовать базовые возможности библеотеки NumPy языка программирования Python.

Ход выполнения работы:

1. Задание 1. Создайте массив NumPy размером 3x3, содержащий числа от 1 до 9. Умножьте все элементы массива на 2, а затем замените все элементы больше 10 на 0. Выведите итоговый массив.

```
import numpy as np
```

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

```
matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
```

```
A*=2
A
```

```
matrix([[ 4,  8, 12],
        [16, 20, 24],
        [28, 32, 36]])
```

```
np.where(A>10, A*0,A)
```

```
array([[4, 8, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

Рисунок 1. Выполнение задания №1

2. Задание №2. Создайте массив NumPy из 20 случайных целых чисел от 1 до 100. Найдите и выведите все элементы, которые делятся на 5 без остатка. Затем замените их на 1 и выведите обновленный массив.

```
import numpy as np
```

```
A = np.random.randint(1,100,size=20)
A
```

```
array([63, 94,  9,  3, 77, 79, 71, 34, 10, 61, 18, 90, 33, 74, 78, 99, 54,
       74, 38, 23])
```

```
B =np.where(A%5==0, A, 0)
B
```

```
array([ 0,  0,  0,  0,  0,  0,  0,  0, 10,  0,  0, 90,  0,  0,  0,  0,  0,
        0,  0,  0])
```

Рисунок 2. Программа для выполнения задания №2

3. Задание №3. Создайте два массива NumPy размером 1×5, заполненные случайными числами от 0 до 50. Объедините эти массивы в один двумерный массив (по строкам). Разделите полученный массив на два массива, каждый из которых содержит 5 элементов.

```
import numpy as np
x = np.random.randint(1,50, size=5)
y = np.random.randint(1,50, size=5)
np.matrix([x,y])
```

```
matrix([[42, 42, 31, 16, 49],
        [21, 43, 41, 31,  1]])
```

```
combined_array = np.concatenate((x, y), axis=0)
print("Объединенный массив:\n", combined_array)
```

```
Объединенный массив:
 [42 42 31 16 49 21 43 41 31  1]
```

```
split_array1 = combined_array[:5]
split_array2 = combined_array[5:]
print("Разделенный массив 2:", split_array1)
print("Разделенный массив 2:", split_array2)
```

```
Разделенный массив 2: [42 42 31 16 49]
Разделенный массив 2: [21 43 41 31  1]
```

Рисунок 3. Программа для выполнения задания №3

4. Задание №4. Создайте массив из 50 чисел, равномерно распределенных от -10 до 10. Вычислите сумму всех элементов, сумму положительных элементов и сумму отрицательных элементов. Выведите результаты.

```
import numpy as np
```

```
m = np.linspace(-10, 10, 50)
print(m)
```

```
[-10.         -9.59183673  -9.18367347  -8.7755102   -8.36734694
  -7.95918367  -7.55102041  -7.14285714  -6.73469388  -6.32653061
  -5.91836735  -5.51020408  -5.10204082  -4.69387755  -4.28571429
  -3.87755102  -3.46938776  -3.06122449  -2.65306122  -2.24489796
  -1.83673469  -1.42857143  -1.02040816  -0.6122449   -0.20408163
   0.20408163   0.6122449    1.02040816   1.42857143   1.83673469
   2.24489796   2.65306122   3.06122449   3.46938776   3.87755102
   4.28571429   4.69387755   5.10204082   5.51020408   5.91836735
   6.32653061   6.73469388   7.14285714   7.55102041   7.95918367
   8.36734694   8.7755102    9.18367347   9.59183673  10.        ]
```

```
all_sum= np.sum(m)
print(all_sum)
```

```
7.105427357601002e-15
```

```
plus_sum=np.sum(m[m>0])
print(plus_sum)
```

```
127.55102040816328
```

```
minus_sum=np.sum(m[m<0])
print(minus_sum)
```

```
-127.55102040816327
```

Рисунок 4. Программа для выполнения задания №4

5. Задание №5. Создать единичную матрицу 4x4. Диагональную матрицу размером 4x4 с диагональными элементами [5,10,15,20] (не используя циклы). Найти сумму всех элементов каждой из этих матриц и сравните результаты.

```
import numpy as np
```

```
eye = np.eye(4)
diag = np.matrix([[5,0,0,0],[0,10,0,0],[0,0,15,0],[0,0,0,20]])

print(eye,"\n\n",diag)
```

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

 [[ 5  0  0  0]
 [ 0 10  0  0]
 [ 0  0 15  0]
 [ 0  0  0 20]]
```

```
sum_eye = np.sum(eye)
sum_diag = np.sum(diag)
print('сумма единичной матрицы:', sum_eye, "\nсумма диагональной матрицы:", sum_diag,'\nsum_eye >sum_diag?:',sum_eye < sum_diag)
```

```
сумма единичной матрицы: 4.0
сумма диагональной матрицы: 50
sum_eye >sum_diag?: True
```

Рисунок 5. Программа для выполнения задания №5

6. Задание 6: Создание и базовые операции с матрицами. Создайте две квадратные матрицы NumPy размером 3×3, заполненные случайными целыми числами от 1 до 20. Вычислите и выведите:

- их сумму;
- их разность;
- их поэлементное произведение.

```
import numpy as np
```

```
A = np.random.randint(1, 21, size=(3, 3))
B = np.random.randint(1, 21, size=(3, 3))
print(A)
print()
print(B)
```

```
[[20 12 14]
 [18 10 11]
 [18 18 16]]

[[20 11 20]
 [ 6 15 19]
 [17 18 11]]
```

```
Sum = A+B
Raz = A-B
Umn = A*B
```

```
print("Сумма:\n",Sum, "\n")
print("Разность:\n",Raz, "\n")
print("Произведение:\n",Umn, "\n")
```

```
Сумма:
 [[28 20 23]
 [ 8 27 22]
 [26  9 11]]

Разность:
 [[-4 16  9]
 [-2  9  2]
 [12  1  9]]

Произведение:
 [[192  36 112]
 [ 15 162 120]
 [133  20  10]]
```

Рисунок 6. Программа для выполнения задания№6

7. Задание 7. Умножение матриц. Создайте две матрицы NumPy: Первую размером 2×3, заполненную случайными числами от 1 до 10. Вторую размером 3×2, заполненную случайными числами от 1 до 10. Выполните матричное умножение (@ или np.dot) и выведите результат.

```
import numpy as np
```

```
A = np.random.randint(0, 20, size=(2, 3))
B = np.random.randint(0, 20, size=(3, 2))
print(A)
print(B)
```

```
[[ 9 10 15]
 [19 11 19]]
[[ 6  2]
 [10 18]
 [13  7]]
```

```
print(A.dot(B))
```

```
[[349 303]
 [471 369]]
```

Рисунок 7. Программа для выполнения задания №7

8. Задание 8. Определитель и обратная матрица. Создайте случайную квадратную матрицу 3×3. Найдите и выведите: Определитель этой матрицы Обратную матрицу (если существует, иначе выведите сообщение, что матрица вырождена) Используйте функции np.linalg.det и np.linalg.inv .

```
import numpy as np
```

```
A = np.random.randint(1, 21, size=(3, 3))
print("Матрица:\n", A)
```

```
Матрица:
 [[10 18 18]
 [ 9 11  7]
 [14 11  7]]
```

```
det = np.linalg.det(A)
print("Определитель матрицы:{:.0f}".format(det))
```

```
Определитель матрицы:1200
```

```
try:
    inverse_matrix = np.linalg.inv(A)
    print("Обратная матрица:\n", inverse_matrix)
except np.linalg.LinAlgError:
    print("Матрица вырождена и не имеет обратной матрицы.")
```

```
Обратная матрица:
 [[-0.145      -0.115       0.215     ]
 [-0.01666667 -0.11666667  0.11666667]
 [ 0.16416667  0.19916667 -0.24916667]]
```

Рисунок 8. Программа для выполненная задания №8

9. Задание 9. Транспортирование и след матрицы. Создайте матрицу NumPy размером 4×4, содержащую случайные целые числа от 1 до 50. Выведите: Исходную матрицу Транспонированную матрицу След матрицы (сумму элементов на главной диагонали) Используйте np.trace для нахождения следа.

```
import numpy as np
```

```
matrix = np.random.randint(1, 51, size=(4, 4))
print("Исходная матрица:\n", matrix)
```

```
Исходная матрица:
 [[11 34 24 50]
 [20 33  7 47]
 [13 45 20 27]
 [28 43 12 15]]
```

```
transposed_matrix = matrix.T
print("Транспонированная матрица:\n", transposed_matrix)
```

```
Транспонированная матрица:
 [[11 20 13 28]
 [34 33 45 43]
 [24  7 20 12]
 [50 47 27 15]]
```

```
trace = np.trace(matrix)
print("След матрицы:", trace)
```

```
След матрицы: 79
```

Рисунок 9. Программа для выполнения задания №9

10. Задание 10. Системы линейных уравнений. Решить систему вида: (см.Рисунок 10.1). Используйте матричное представление Ax = B , где A – матрица коэффициентов, x – вектор неизвестных, B – вектор правой части. Решите систему с помощью np.linalg.solve и выведите результат.

$$\begin{cases} 2x + 3y - z = 5 \\ 4x - y + 2z = 6 \\ -3x + 5y + 4z = -2 \end{cases}$$

Рисунок 10.1. Система уравнений

```
import numpy as np
```

```
A = np.matrix([[2,3,-1],[4,-1,2],[-3,5,4]])
b = np.array([5,6,-2])
x = np.linalg.solve(A,b)
print(x)
```

```
[1.63963964 0.57657658 0.00900901]
```

```
np.allclose(A.dot(x),b)
```

```
True
```

Рисунок 10.2. Программа для выполнения задания№10

11. Индивидуальное задание. Оптимизация рабочего времени. Три сотрудника должны выполнить работу за минимальное время. Первый выполняет задачу за 5 часов, второй — за 3 часа, а третий — за 2 часа. Как распределить работу между ними, если всего 20 задач?

```
import numpy as np
```

```
A = np.array([[1,1,1],[3,-5,0],[2,0,-5]])
B = np.array([20,0,0])
```

```
dA = np.linalg.det(A)
print(dA)
```

```
50.000000000000014
```

```
Ax = np.copy(A)
Ay = np.copy(A)
Az = np.copy(A)

Ax[:, 0] = B
Ay[:, 1] = B
Az[:, 2] = B

d1 = np.linalg.det(Ax)
d2 = np.linalg.det(Ay)
d3 = np.linalg.det(Az)

print("{} \n\n{} \n{}".format(Ax,Ay,Az))
print("d1={:.0f} \nd2={:.0f} \nd3={:.0f}".format(d1,d2,d3))
```

```
[[20  1  1]
 [ 0 -5  0]
 [ 0  0 -5]]

[[ 1 20  1]
 [ 3  0  0]
 [ 2  0 -5]]
[[ 1  1 20]
 [ 3 -5  0]
 [ 2  0  0]]
d1=500
d2=300
d3=200
```

```
x = d1/dA
y = d2/dA
z = d3/dA
print("Ответ\nx={:.0f} \ny={:.0f} \nz={:.0f}".format(x,y,z))
```

```
Ответ
x=10
y=6
z=4
```

```
np.linalg.solve(A,B) #проверка
```

```
array([10.,  6.,  4.])
```

Рисунок 11. Решение задачи методом крамера

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

```
[7]: A = np.array([[1,1,1],[3,-5,0],[2,0,-5]])
     B = np.array([20,0,0])
```

```
[9]: A_inv=np.linalg.inv(A)
     X = np.dot(A_inv, B)
     print(X)

     [10.  6.  4.]
```

```
[11]: np.linalg.solve(A,B)
```

```
[11]: array([10.,  6.,  4.])
```

Рисунок 12. Решение задачи матричным методом

Вывод: в ходе лабораторной работы были изучены базовые функции библиотеки NumPy для создания массивов, выполнения поэлементных и матричных операций, индексации и вычисления определителей/обратных матриц, а также решение задач с использованием, изученных технологий.

Контрольные вопросы:

1. Каково назначение библиотеки NumPy? Это открытая библиотека для языка Python, предназначенная для эффективных вычислений с многомерными массивами и матрицами. NumPy предоставляет множество высокоуровневых математических функций для работы с этими структурами данных и выполнения математических операций.

2. Что такое массивы ndarray? Это многомерный массив — основная структура данных в NumPy . Объект, который представляет собой таблицу данных (например,вектор или матрицу) и позволяет выполнять операции с ними с высокой эффективностью.

3. Как осуществляется доступ к частям многомерного массива? Доступ осуществляется с помощью обращения к индексам элемента(например m[0,1], где 0 - индекс строки, а 1 - индекс столбца). Если вместо индекса строки или столбца ввести «:», то будут выведены все элементы строки или столбца соотвественно. Если постоавить знак «:» перед или после индекса, то будут выедены все элементы до или после этого элемента соотвествено.

4. Как осуществляется расчет статистик по данным? NumPy предоставляет множество функций для вычисления статистических показателей:

- np.mean(arr): Среднее арифметическое.
- np.median(arr): Медиана.
- np.std(arr): Стандартное отклонение.
- np.var(arr): Дисперсия.
- np.min(arr): Минимальное значение.
- np.max(arr): Максимальное значение.
- np.sum(arr): Сумма всех элементов.

– np.percentile(arr, q): q-й процентиль.

– np.corrcoef(arr): Матрица корреляции.

5. Как выполняется выборка данных из массивов ndarray? Булева индексация (маскирование): Создание булева массива (маски) той же формы, что и исходный массив. Элементы, соответствующие True в маске, выбираются. Целочисленная индексация: Выбор элементов на основе списка или массива индексов.

6. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python. Вектор-строка: Массив с одним измерением. Вектор-столбец: Матрица с одним столбцом. В NumPy часто представляется как двумерный массив. Квадратная матрица: Число строк равно числу столбцов. Диагональная матрица: Все элементы вне главной диагонали равны нулю. Единичная матрица: Диагональная матрица, где все элементы на главной диагонали равны 1. Нулевая матрица: Все элементы равны нулю. Треугольная матрица.

7. Как выполняется транспонирование матриц? Меняются местами строки и столбцы

8. Приведите свойства операции транспонирования матриц.Дважды транспонированная матрица равна исходной матрице. Транспонирование суммы матриц равно сумме транспонированных матриц. Транспонирование произведения матриц равно произведению транспонированных матриц расставленных в обратном порядке. Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу.

9. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц? arr.T и np.transpose(arr).

10. Какие существуют основные действия над матрицами? Умножение матрицы на число (скалярное умножение). Сложение и вычитание матриц. Умножение матриц. Нахождение определителя матрицы. Нахождение

обратной матрицы. Определители исходной и транспонированной матрицы совпадают

11. Как осуществляется умножение матрицы на число? Умножение всех элементов на число.

12. Какие свойства операции умножения матрицы на число? Произведение единицы и любой заданной матрицы равно заданной матрице. Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы.Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел. Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число.Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число.

13. Как осуществляется операции сложения и вычитания матриц? Сложение и вычитание матриц выполняются поэлементно. Матрицы должны иметь одинаковую размерность.

14. Каковы свойства операций сложения и вычитания матриц? Коммутативность, ассоциативность, сложение с нулевой матрицей равно изначальной матрице. Вычетание матрицы самой себя равно нулевой матрице.

15. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц? Операторы + и -.

16. Как осуществляется операция умножения матриц? Умножать можно только матрицы, отвечающие следующему требованию: количество столбцов первой матрицы должно быть равно числу строк второй матрицы.

17. Каковы свойства операции умножения матриц? Ассоциативность умножения. Дистрибутивность умножения. Умножение матриц в общем виде не коммутативно. Произведение заданной матрицы на единичную равно исходной матрице. Произведение заданной матрицы на нулевую матрицу равно нулевой матрице.

18. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц? np.dot(A, B), A @ B, np.matmul(A, B).

19. Что такое определитель матрицы? Каковы свойства определителя матрицы? Определитель (determinant) — это скалярная величина, которая может быть вычислена только для квадратных матриц. Определитель матрицы остается неизменным при ее транспонировании. Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю. При перестановке строк матрицы знак ее определителя меняется на противоположный.Если у матрицы есть две одинаковые строки, то ее определитель равен нулю. Если все элементы строки или столбца матрицы умножить на какое- то число, то и определитель будет умножен на это число. Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц. Если к элементам одной строки прибавить элементы другой строки,умноженные на одно и тоже число, то определитель матрицы не изменится. Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю. Если матрица содержит пропорциональные строки, то ее определитель равен нулю.

20. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы? np.linalg.det(A)

21. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы? Обратная матрица - это такая матрица (обозначается A-1, которая при умножении на исходную матрицу A дает единичную матрицу I. Обратная матрица существует только для квадратных *невырожденных* матриц (то есть, для матриц, у которых определитель не равен нулю). Алгоритм нахождения обратной матрицы (метод Гаусса-Жордана): 1. Создать расширенную матрицу: Справа от исходной матрицы A приписать единичную матрицу I той же размерности. Получится расширенная матрица [A | I]. 2. Выполнить элементарные преобразования строк над расширенной

матрицей: Цель - привести матрицу A к единичной матрице. При этом все те же преобразования применяются и к матрице I. 3. Получить обратную матрицу: После того как матрица A преобразована в единичную матрицу, на месте исходной единичной матрицы I окажется обратная матрица A-1. Теперь расширенная матрица будет иметь вид [I | A-1].

22. Каковы свойства обратной матрицы? 1. A * A-1 = A-1 * A = I, где A - исходная матрица, A-1 - ее обратная, а I - единичная матрица. Это основное определение обратной матрицы 2. (A-1) -1> = A. Обратная к обратной матрице - это исходная матрица. 3. (A * B-1 = B-1 * A-1. Обратная произведения матриц равна произведению обратных матриц в обратном порядке. Это свойство применимо, если и A, и B - квадратные невырожденные матрицы. 4. (AT)-1 = (A-1)T. Обратная транспонированной матрицы равна транспонированной обратной матрице. 5. det(A-1) = 1 / det(A). Определитель обратной матрицы равен обратной величине определителя исходной матрицы. 6. Если A - диагональная матрица, то A-1 также является диагональной матрицей, и ее диагональные элементы являются обратными диагональным элементам матрицы A (при условии, что все диагональные элементы A ненулевые). 7. Если A - ортогональная матрица (AT * A = I), то A-1 = AT. То есть, обратная матрица ортогональной матрицы равна ее транспонированной матрице. 8. Умножение системы уравнений на обратную матрицу позволяет решить эту систему. Если Ax = b, то x = A-1b.

23. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы? numpy.linalg.inv()

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

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