МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра систем автоматизированного проектирования

ОТЧЕТ

по лабораторной работе №7

«ВЕКТОРА И МАТРИЦЫ. ГРАДИЕНТ»

Студентка гр. 3353 Карпенко А.Ю.

Преподаватель Копец Е.Е.

Санкт-Петербург

2024

Цель работы

Работа с матрицами, векторами и градиетом в среде sympy

Ход работы:

7.1

- Найти суммы векторов

В этом задании имеют смысл только второй и четвертый примеры, в остальныхразмерность не совпадает

```
from sympy import *
```

```
M1=Matrix([[1,2,3,4]])
M1
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

```
M2=Matrix([[10,11,12,13]])
M2
```

$$\begin{bmatrix} 10 & 11 & 12 & 13 \end{bmatrix}$$

```
M1+M2
```

$$\begin{bmatrix} 11 & 13 & 15 & 17 \end{bmatrix}$$

Рис. 1- Удачная операция сложения второй пары векторов

```
from sympy import *
```

```
M1=Matrix([15,16,17,18,19])
M1
```

$$\begin{bmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \end{bmatrix}$$

```
M2=Matrix([3,3,3,3,3])
M2
```

$$\begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

```
M1+M2
```

$$\begin{bmatrix} 18 \\ 19 \\ 20 \\ 21 \\ 22 \end{bmatrix}$$

Рис.2- Удачная операция сложения четвертой пары векторов

- Найдите значения выражений

```
from sympy import *
```

```
M1=Matrix([[1,2,3,4,5,6]])
M1
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

```
M2=Matrix([[5,6,7,8,9,10]])
M2
```

$$\begin{bmatrix} 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

```
5*M1-3*M2
```

$$\begin{bmatrix} -10 & -8 & -6 & -4 & -2 & 0 \end{bmatrix}$$

Рис. 3- Результат вычисления первого выражения

```
from sympy import *
```

```
M1=Matrix([[7,12,11,14,9,16,21]])
M1
```

$$\begin{bmatrix} 7 & 12 & 11 & 14 & 9 & 16 & 21 \end{bmatrix}$$

```
M2=Matrix([[13,61,24,76,1,3,8]])
M2
```

$$\begin{bmatrix} 13 & 61 & 24 & 76 & 1 & 3 & 8 \end{bmatrix}$$

```
12*M1-3.2*M2
```

$$\begin{bmatrix} 42.4 & -51.2 & 55.2 & -75.2 & 104.8 & 182.4 & 226.4 \end{bmatrix}$$

Рис. 4- Результат вычисления второго выражения

```
from sympy import *
```

```
M1=Matrix([15,16,17,18,19])
M1
```

$$\begin{bmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \end{bmatrix}$$

```
M2=Matrix([3,3,3,3,3])
M2
```

$$\begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

```
7*M1+(1/3)*M2
```

$$\begin{bmatrix} 106.0 \\ 113.0 \\ 120.0 \\ 127.0 \\ 134.0 \end{bmatrix}$$

Рис. 5- Результат вычисления третьего выражения

```
from sympy import *
```

```
M1=Matrix([11,21,78,32,2])
M1
```

$$\begin{bmatrix} 11 \\ 21 \\ 78 \\ 32 \\ 2 \end{bmatrix}$$

```
M2=Matrix([5,6,7,9,3])
M2
```

$$\begin{bmatrix} 5 \\ 6 \\ 7 \\ 9 \\ 3 \end{bmatrix}$$

```
7*M1-(2/5)*M2
```

$$\begin{bmatrix} 75.0 \\ 144.6 \\ 543.2 \\ 220.4 \\ 12.8 \end{bmatrix}$$

Рисунок 6- Результат вычисления четвертого выражения

7.2

- Найдите значения выражений

```
from sympy import *
from sympy import Matrix

vec1 = Matrix([1, 2, 3, 4, 5])
vec2 = Matrix([5, 6, 7, 8, 9])

scalar_product = vec1.dot(vec2)

print(scalar_product)
```

```
115
```

Рисунок 7- Результат вычисления первого выражения

```
from sympy import *
from sympy import Matrix

vec1 = Matrix([4,3,8,12,1])
vec2 = Matrix([3,2,13,8,5])

scalar_product = vec1.dot(vec2)

print(scalar_product)
```

```
223
```

Рисунок 8- Результат вычисления второго выражения

Операция третьего выражения не имеет смысл из-за разной размерности

- Транспонировать матрицы

```
from sympy import *
from sympy import Matrix

matrix = Matrix([[5, 6, 7, 8, 9]])

transposed_matrix = matrix.transpose()

transposed_matrix
```

$$\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

Рисунок 9- Результат транспонирования первой матрицы

```
from sympy import *
from sympy import Matrix

matrix = Matrix([5, 6, 7, 8, 9])

transposed_matrix = matrix.transpose()

transposed_matrix
```

$$\begin{bmatrix} 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

Рисунок 10- Результат транспонирования второй матрицы

```
from sympy import *
from sympy import Matrix

matrix = Matrix([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])

matrix
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

```
transposed_matrix = matrix.transpose()

transposed_matrix
```

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

Рисунок 11- Результат транспонирования третьей матрицы

7.3

- Транспонировать матрицы

```
from sympy import *
from sympy import Matrix
```

```
from sympy import Matrix

def transpose_matrix(matrix):
    if not isinstance(matrix, Matrix):
        raise ValueError("Input must be a SymPy Matrix object")

    transposed_matrix = matrix.transpose()

    return transposed_matrix

input_matrix = Matrix([[5, 6, 7, 8, 9],[10,11,12,13,14]])
transposed_result = transpose_matrix(input_matrix)
transposed_result
```

$$\begin{bmatrix} 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 \end{bmatrix}$$

Рис.12 - Функция способная транспонировать матрицы в sympy

- Графическое, а потом аналитическое нахождение векторов

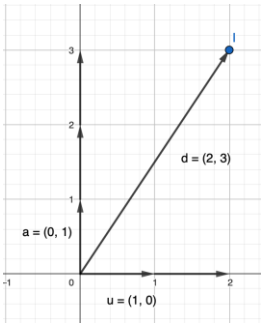

Рис.13- Графические нахождения первого вектора

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[1,0]])
vec1
```

$$\begin{bmatrix} 1 & 0 \end{bmatrix}$$

```
vec2=Matrix([[0,1]])
vec2
```

$$\begin{bmatrix} 0 & 1 \end{bmatrix}$$

```
2*vec1+3*vec2
```

$$\begin{bmatrix} 2 & 3 \end{bmatrix}$$

Рис.14- Аналитическое нахождения первого вектора

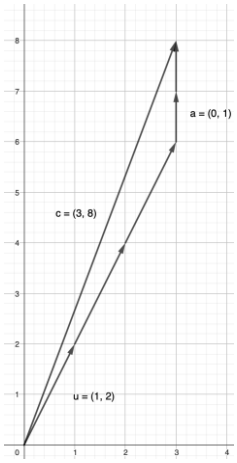

Рис.15- Графические нахождения второго вектора

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[1,2]])
vec1
```

$$\begin{bmatrix}1 & 2\end{bmatrix}$$

```
vec2=Matrix([[0,1]])
vec2
```

$$\begin{bmatrix}0 & 1\end{bmatrix}$$

```
3*vec1+2*vec2
```

$$\begin{bmatrix}3 & 8\end{bmatrix}$$

Рис.16- Аналитическое нахождения второго вектора

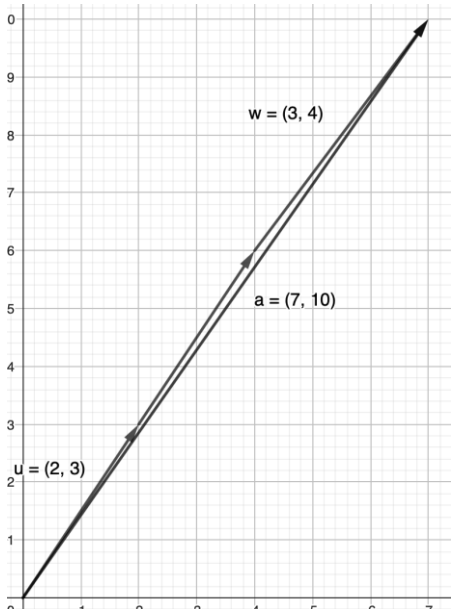

Рис.17- Графическое нахождения третьего вектора

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[2,3]])
vec1
```

$$\begin{bmatrix}2 & 3\end{bmatrix}$$

```
vec2=Matrix([[3,4]])
vec2
```

$$\begin{bmatrix}3 & 4\end{bmatrix}$$

```
2*vec1+vec2
```

$$\begin{bmatrix}7 & 10\end{bmatrix}$$

Рис.18- Аналитическое нахождения третьего вектора

- Найти векторы аналитические и построить их графически

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[1,0,0]])
vec1
```

$$\begin{bmatrix}1 & 0 & 0\end{bmatrix}$$

```
vec2=Matrix([[0,1,0]])
vec2
```

$$\begin{bmatrix}0 & 1 & 0\end{bmatrix}$$

```
vec3=Matrix([[0,0,1]])
vec3
```

$$\begin{bmatrix}0 & 0 & 1\end{bmatrix}$$

```
2*vec1+3*vec2+vec3
```

$$\begin{bmatrix}2 & 3 & 1\end{bmatrix}$$

Рис.19- Аналитическое нахождение первого вектора

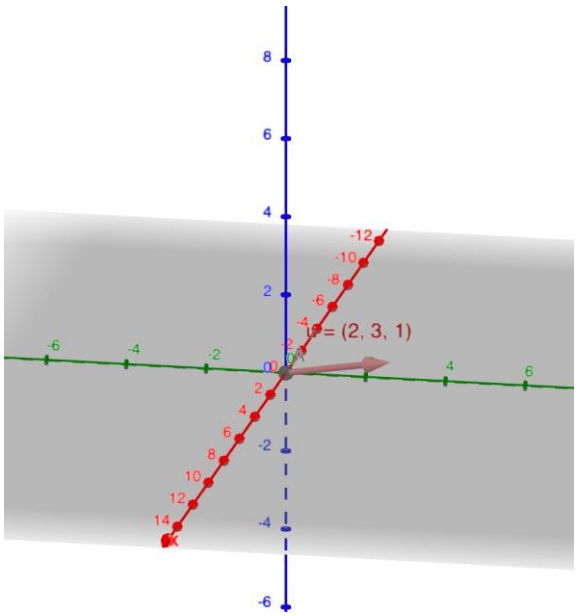

Рис.20- Графическое построение первого вектора

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[2,3,4]])
vec1
```

$$\begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$$

```
vec2=Matrix([[1,1,1]])
vec2
```

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

```
vec1+2*vec2
```

$$\begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

Рис.21- Аналитическое нахождение второго вектора

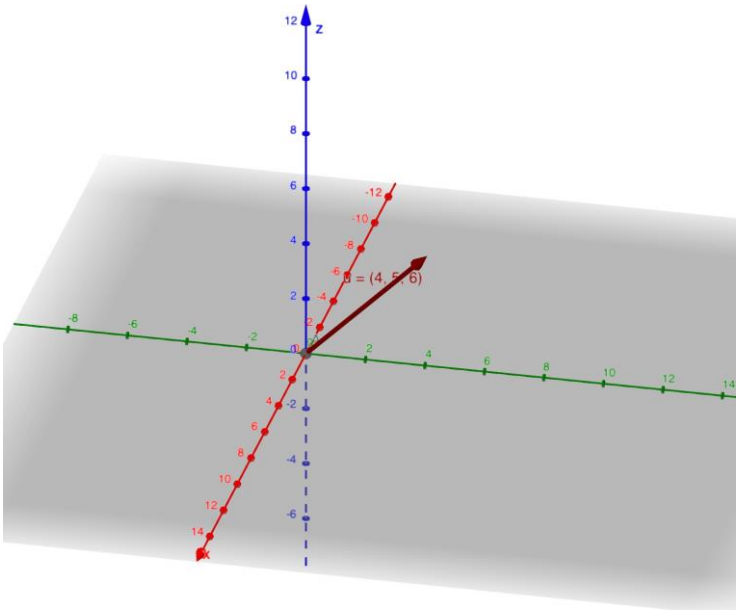

Рис.22- Графическое построение второго вектора

7.4

- Найдем значения выражений

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[2,5, 0, 6, 8, 10, 6, 7, 5, 7]])
vec1
```

```
[2  5  0  6  8  10  6  7  5  7]
```

```
vec2=Matrix([[8, 6, 7, 9, 6, 6, 2, 3, 2, 3]])
vec2
```

```
[8  6  7  9  6  6  2  3  2  3]
```

```
vec1=3*vec1
vec1
```

```
[6  15  0  18  24  30  18  21  15  21]
```

```
vec2=2*vec2
vec2
```

```
[80  60  70  90  60  60  20  30  20  30]
```

```
scalar_product=vec1.dot(vec2)
scalar_product
```

```
8160
```

Рис.23-Нахождения значения первого выражения

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[9, 6, 7, 7, 0, 1, 6, 8, 1, 2]])
vec1
```

```
[9  6  7  7  0  1  6  8  1  2]
```

```
vec1=vec1*6
```

```
vec2=Matrix([[0, 2, 2, 6, 7, 8, 8, 3, 1, 8]])
vec2
```

```
[0  2  2  6  7  8  8  3  1  8]
```

```
vec2=vec2*5
```

```
scalar_product=vec1.dot(vec2)
scalar_product
```

```
4950
```

Рис.24-Нахождения значения второго выражения

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[2, 5, 0, 6, 8, 10, 6, 7, 5, 7]])
vec1
```

```
[2  5  0  6  8  10  6  7  5  7]
```

```
vec2=Matrix([[8, 6, 7, 9, 6, 6, 2, 3, 2, 3]])
vec2
```

```
[8  6  7  9  6  6  2  3  2  3]
```

```
scalar_product=vec1.dot(vec2)
scalar_product
```

```
272
```

Рис.25-Нахождения значения третьего выражения

```
from sympy import *
from sympy import Matrix
```

```
vec1=Matrix([[9, 6, 7, 7, 0, 1, 6, 8, 1, 2]])
vec1
```

```
[9  6  7  7  0  1  6  8  1  2]
```

```
vec2=Matrix([[0, 2, 2, 6, 7, 8, 8, 3, 1, 8]])
vec2
```

```
[0  2  2  6  7  8  8  3  1  8]
```

```
scalar_product=vec1.dot(vec2)
scalar_product
```

```
165
```

Рис.26-Нахождения значения четвертого выражения

- Транспонировать матрицы

```
from sympy import *
from sympy import Matrix
```

```
def transpose_matrix(matrix):

    if not isinstance(matrix, Matrix):
        raise ValueError("Input must be a SymPy Matrix object")

    transposed_matrix = matrix.transpose()

    return transposed_matrix

input_matrix = Matrix([[2,1,7,4],[5,6,7,3],[9,8,2,12],[11,14,15,15]])
transposed_result = transpose_matrix(input_matrix)

transposed_result
```

$$\begin{bmatrix} 2 & 5 & 9 & 11 \\ 1 & 6 & 8 & 14 \\ 7 & 7 & 2 & 15 \\ 4 & 3 & 12 & 15 \end{bmatrix}$$

Рис.27-Транспонирование первой матрицы

```
from sympy import *
from sympy import Matrix
```

```
def transpose_matrix(matrix):

    if not isinstance(matrix, Matrix):
        raise ValueError("Input must be a SymPy Matrix object")

    transposed_matrix = matrix.transpose()

    return transposed_matrix

input_matrix = Matrix([[3,7,8,3,6],[2,5,9,4,13]])
transposed_result = transpose_matrix(input_matrix)

transposed_result
```

$$\begin{bmatrix} 3 & 2 \\ 7 & 5 \\ 8 & 9 \\ 3 & 4 \\ 6 & 13 \end{bmatrix}$$

Рис.28-Транспонирование второй матрицы

- Нужно завершить градиентный спуск до MSE меньше 6,36

```
from sympy import *
from sympy import Matrix
```

```
a2, a1 = symbols('a2,a1')

f = (1/4)*((a1+2*a2-5)**2+(5*a1+3*a2-6)**2+(2*a1+4*a2-10)**2+(3*a1+7*a2-8)**2)

delta_MSE_x=19.5*a1+23*a2-39.5

delta_MSE_y=23*a1+39*a2-62

f_func = lambdify((a2, a1), f)

delta_MSE_x_func = lambdify((a2, a1), delta_MSE_x)

delta_MSE_y_func = lambdify((a2, a1), delta_MSE_y)


value_x = 0.57
value_y = 0.91


count = 0


while f_func(value_y, value_x)>=6.36:
    value_x_num = delta_MSE_x_func(value_y, value_x)
    value_y_num = delta_MSE_y_func(value_y, value_x)

    value_x=value_x+(0.01*(-value_x_num))
    value_y=value_y+(0.01*(-value_y_num))


    count += 1

print(f"Значение стало меньше 6.36 после {count} итераций.")
print(f"Значение a1 {value_x}")
print(f"Значение a2 {value_y}")
print(f"Значение среднеквадратичной ошибки {f_func(value_y, value_x)}")
```

```
Значение стало меньше 6.36 после 5 итераций.
Значение a1 0.6772467261994688
Значение a2 1.1703118543348126
Значение среднеквадратичной ошибки 6.348733711093741
```

Рис.29-Результат работы функции

Вывод

В процессе работы мы научились выполнять матричные вычисления, поработали свекторами и завершили вычисление градиентного спуска