МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по практической работе №7
по дисциплине «Информационные технологии»
Тема: Вектора и матрицы. Градиент.

Студент гр. 4352 _____________________ Колесникова М. А.

Преподаватель _____________________ Копец Е. Е.

Санкт-Петербург
2025

Цель работы.

Научиться выполнять операции над векторами и матрицами, а также находить градиент.

Основные теоретические положения.

В первом задании нужно сложить векторы, складывать векторы можно только одной размерности.

1.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

– операция невозможна, так как не совпадает количество элементов ($4 \neq 5$);

2.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} + \begin{pmatrix} 10 & 11 & 12 & 13 \end{pmatrix} = \begin{pmatrix} 11 & 13 & 15 & 17 \end{pmatrix};$$

3.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

– операция сложения невозможна, так как вектор-строка и вектор-столбец не совместимы;

4.

$$\begin{pmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 18 \\ 19 \\ 20 \\ 21 \\ 22 \end{pmatrix};$$

5.

$$\begin{pmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 21 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}$$

– операция невозможна, так как не совпадает количество элементов ($6 \neq 5$).

Во втором задании нужно найти значения выражений.

1.

$$5\times\begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6\end{pmatrix}-3\times\begin{pmatrix}5 & 6 & 7 & 8 & 9 & 10\end{pmatrix} = \begin{pmatrix}-10 & -8 & -6 & -4 & -2 & 0\end{pmatrix};$$

2.

$$12 \times \begin{pmatrix}7 & 12 & 11 & 14 & 9 & 16 & 21\end{pmatrix} - 3.2 \times \begin{pmatrix}13 & 61 & 24 & 76 & 1 & 3 & 8\end{pmatrix} =$$

$$= \begin{pmatrix}42.4 & -51.2 & 55.2 & -75.2 & 104.8 & 182.4 & 226.4\end{pmatrix};$$

3.

$$7 \times \begin{pmatrix}15\\16\\17\\18\\19\end{pmatrix} + \frac{1}{3} \times \begin{pmatrix}3\\3\\3\\3\\3\end{pmatrix} = \begin{pmatrix}106\\113\\120\\127\\134\end{pmatrix};$$

4.

$$7 \times \begin{pmatrix}11\\21\\78\\32\\2\end{pmatrix} - \frac{2}{5} \times \begin{pmatrix}5\\6\\7\\9\\3\end{pmatrix} = \begin{pmatrix}75\\144.6\\543.2\\220.4\\12.8\end{pmatrix};$$

В третьем задании нужно найти скалярное произведение векторов, если операция имеет смысл.

1.

$$\begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix} \times \begin{pmatrix}5 & 6 & 7 & 8 & 9\end{pmatrix} = 5 + 12 + 21 + 32 + 45 = 115;$$

2.

$$\begin{pmatrix}4 & 3 & 8 & 12 & 1\end{pmatrix} \times \begin{pmatrix}3 & 2 & 13 & 8 & 5\end{pmatrix} = 12 + 6 + 104 + 96 + 5 = 223;$$

3.
$$\begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix} \times \begin{pmatrix}5 & 6 & 7 & 8 & 9\end{pmatrix}$$
– векторы нельзя скалярно перемножить, так как у них разная размерность $(4 \neq 5)$.

В четвёртом задании нужно транспонировать матрицы. Для нужно их как бы повернуть (поменять строки и столбы местами).

1. Транспонирование вектора-строки:

$$(5\,6\,7\,8\,9)^T = \begin{pmatrix}5\\6\\7\\8\\9\end{pmatrix};$$

2. Транспонирование вектора-столбца:

$$\begin{pmatrix}5\\6\\7\\8\\9\end{pmatrix}^T = (5\,6\,7\,8\,9);$$

3. Транспонирование матрицы 4×4:

$$\begin{pmatrix}1 & 2 & 3 & 4\\5 & 6 & 7 & 8\\9 & 10 & 11 & 12\\13 & 14 & 15 & 16\end{pmatrix}^T = \begin{pmatrix}1 & 5 & 9 & 13\\2 & 6 & 10 & 14\\3 & 7 & 11 & 15\\4 & 8 & 12 & 16\end{pmatrix};$$

Транспонирование матрицы можно запрограммировать с помощью sympy (рис. 1).

```python
from sympy import*
def transpose(matrix):
    print("Исходная матрица:\n")
    pprint(matrix)
    print("\nТранспонированная:\n")
    return matrix.T
```

✓ 0.0s

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

✓ 0.0s

Исходная матрица:

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

Транспонированная:

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

Рисунок 1 – Нахождение транспонированной матрицы

Складывать векторы можно по правилу треугольника: начало одного из векторов переносится в конец другого (на рисунках выделен красным пунктиром), затем рисуется вектор из начала первого вектора (на рисунках выделен фиолетовым). Векторы с рисунков можно проверить посчитав выражения:

a. $2 \cdot (1\ \ 0) + 3 \cdot (0\ \ 1) = (2\ \ 0) + (0\ \ 3) = (2\ \ 3)$. Значение совпадает с геометрическим (рис. 2).

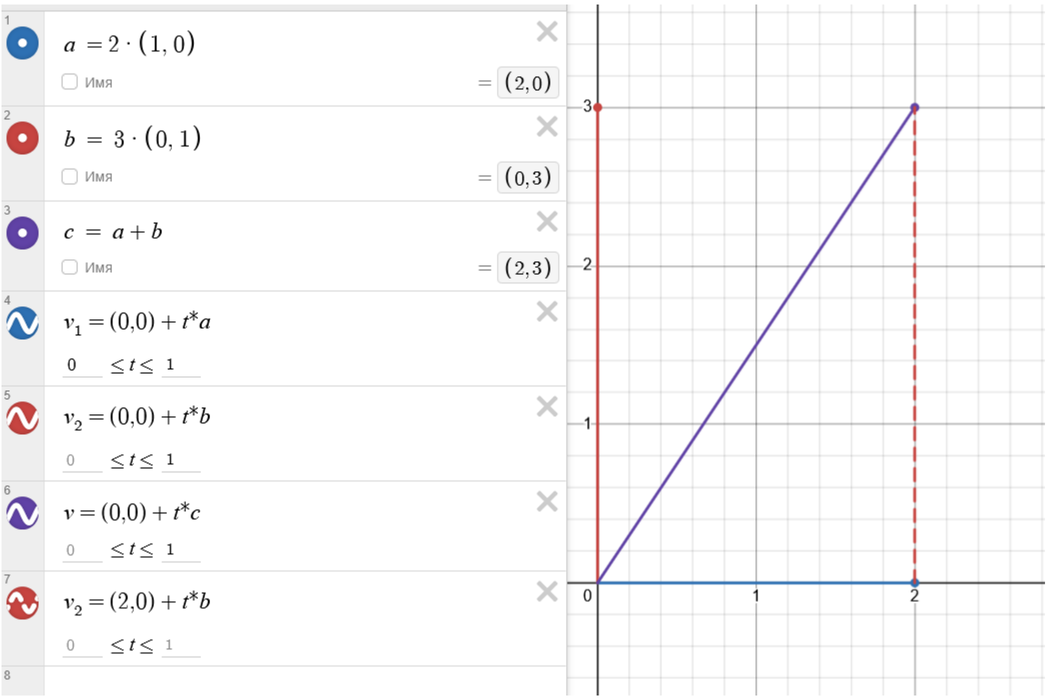

Рисунок 2 – Сложение векторов $2 \cdot (1\ 0) + 3 \cdot (0\ 1)$

b. $3 \cdot (1\ 2) + 2 \cdot (0\ 1) = (3\ 6) + (0\ 2) = (3\ 8)$. Значение совпадает с геометрическим (рис. 3).

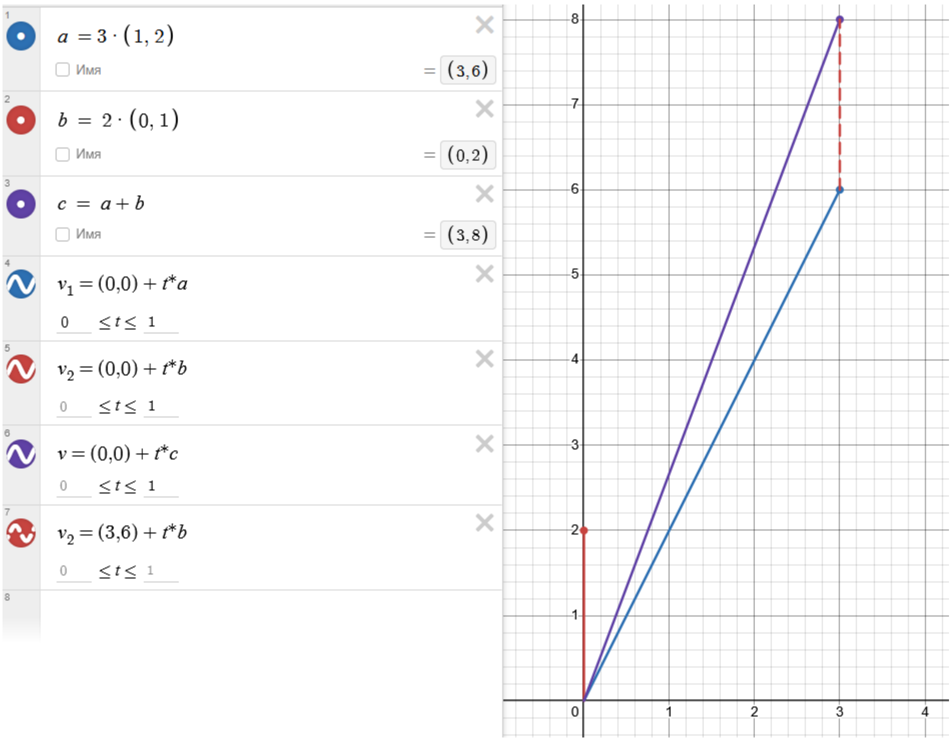

Рисунок 3 – Сложение векторов $3 \cdot (1\ 2) + 2 \cdot (0\ 1)$

c. $2 \cdot (2\ 3) + (3\ 4) = (4\ 6) + (3\ 4) = (7\ 10)$. Значение совпадает с геометрическим (рис. 4).

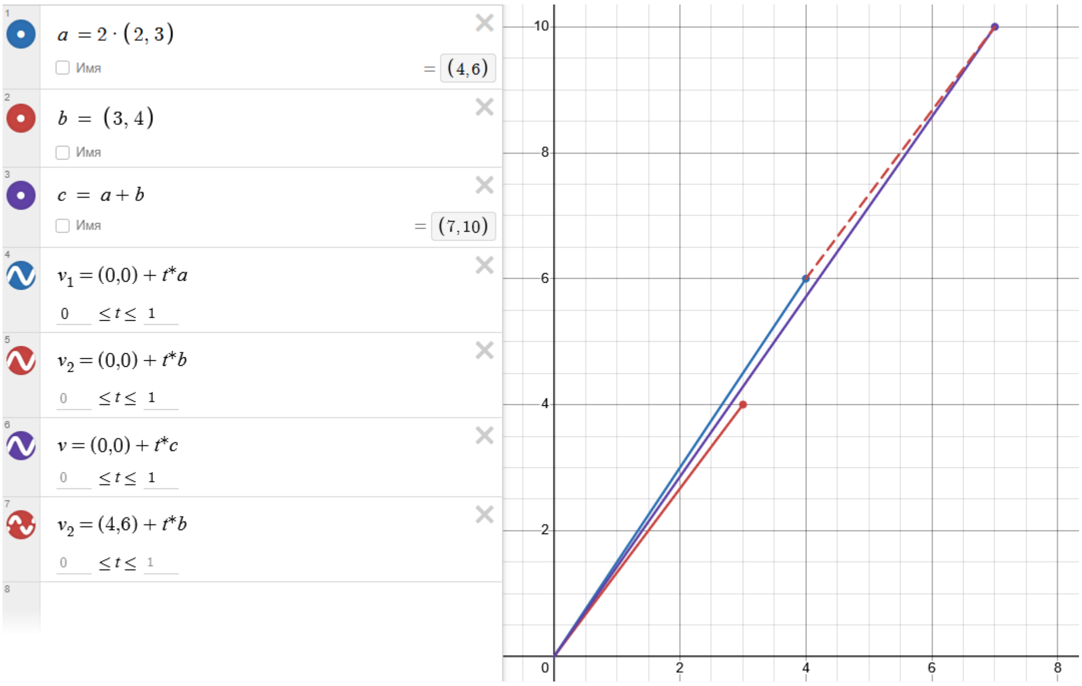

Рисунок 4 – Сложение векторов $2 \cdot (2\ 3) + (3\ 4)$

Для следующего задания найдём аналитически значение, а затем построим график.

a. $2 \cdot (1\ 0\ 0) + 3 \cdot (0\ 1\ 0) + (0\ 0\ 1) = (2\ 3\ 1)$;

b. $(2\ 3\ 4) + 2 \cdot (1\ 1\ 1) = (4\ 5\ 6)$

Для сложения также используется правило треугольника, но также нужно поднять вектор на ещё одно измерение, которое на рисунках выделено серым (рис. 5, 6).

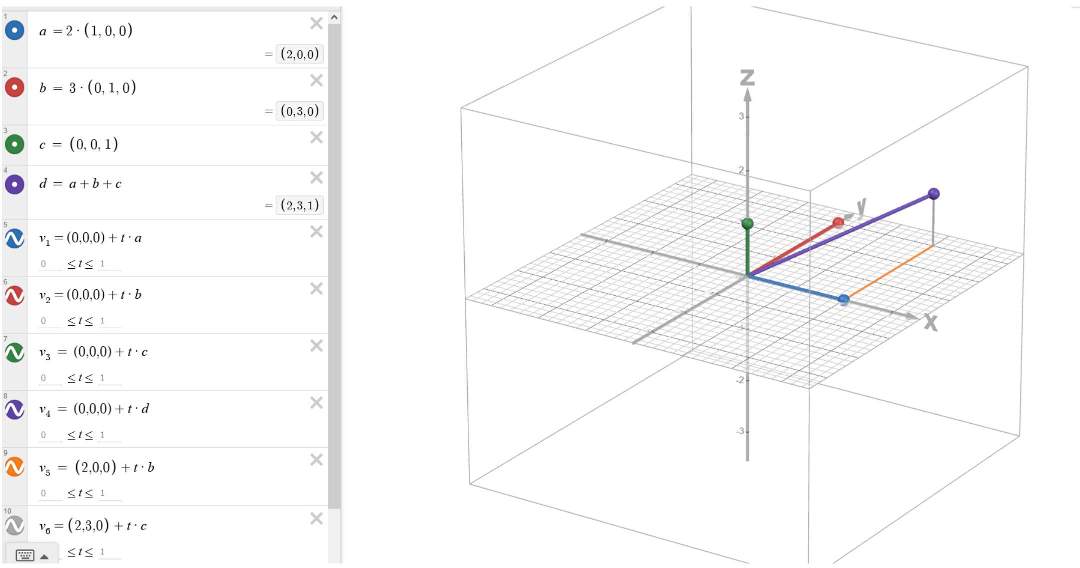

Рисунок 5 – Сложение векторов $2 \cdot (1\ 0\ 0) + 3 \cdot (0\ 1\ 0) + (0\ 0\ 1)$

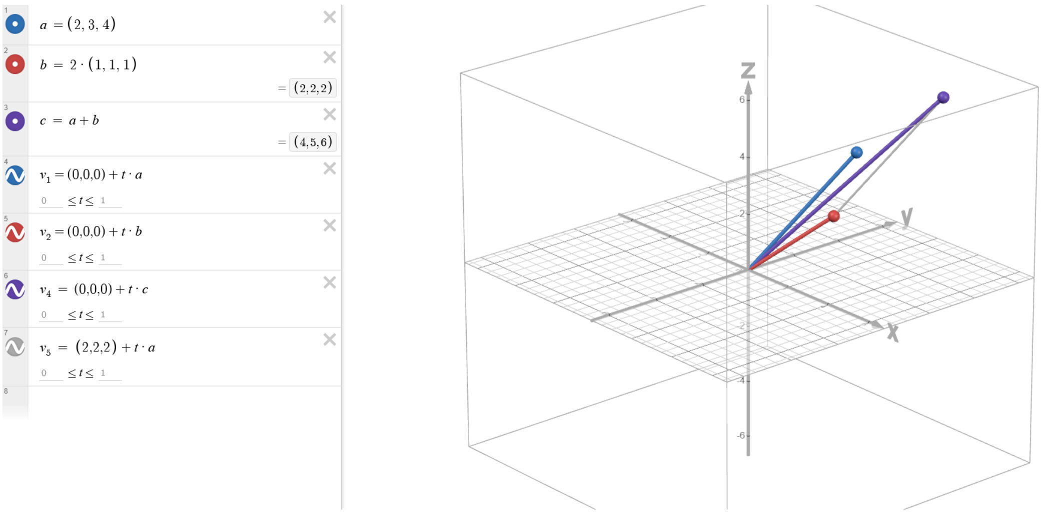

Рисунок 6 – Сложение векторов $(2\ 3\ 4) + 2 \cdot (1\ 1\ 1)$

В следующем задании нужно умножить векторы на константы, а затем скалярно перемножить полученные векторы.

1.

$$3 \times (2\ 5\ 0\ 6\ 8\ 10\ 6\ 7\ 5\ 7) \cdot 2 \times (8\ 6\ 7\ 9\ 6\ 6\ 2\ 3\ 2\ 3) =$$

$$= (6\ 15\ 0\ 18\ 24\ 30\ 18\ 21\ 15\ 21) \cdot (16\ 12\ 14\ 18\ 12\ 12\ 4\ 6\ 4\ 6) =$$

$$= 96 + 180 + 0 + 324 + 288 + 360 + 72 + 126 + 60 + 126 = 1632;$$

2.

$$6 \times (9\ 6\ 7\ 7\ 0\ 1\ 6\ 8\ 1\ 2) \cdot 5 \times (0\ 2\ 2\ 6\ 7\ 8\ 8\ 3\ 1\ 8) =$$

$$= (54\ 36\ 42\ 42\ 0\ 6\ 36\ 48\ 6\ 12) \cdot (0\ 10\ 10\ 30\ 35\ 40\ 40\ 15\ 5\ 40) =$$

$$= 0 + 360 + 420 + 1260 + 0 + 240 + 1440 + 720 + 30 + 480 = 4950;$$

3.

$$(2\ 5\ 0\ 6\ 8\ 10\ 6\ 7\ 5\ 7) \cdot (8\ 6\ 7\ 9\ 6\ 6\ 2\ 3\ 2\ 3) =$$

$$= 16 + 30 + 0 + 54 + 48 + 60 + 12 + 21 + 10 + 21 = 272;$$

4.

$$(9\ 6\ 7\ 7\ 0\ 1\ 6\ 8\ 1\ 2) \cdot (0\ 2\ 2\ 6\ 7\ 8\ 8\ 3\ 1\ 8) =$$

$$= 0 + 12 + 14 + 42 + 0 + 8 + 48 + 24 + 1 + 16 = 165.$$

Для транспонирования матрицы нужно написать код, в котором функция будет принимать и возвращать либо numpy.array, либо list (рис. 7). Результат его работы представлен (рис. 8).

```
import numpy as np
def transpose_matrix(matrix):
    if isinstance(matrix, np.ndarray):
        return matrix.T
    elif isinstance(matrix, list):
        return [list(row) for row in zip(*matrix)]
    else:
        raise ValueError("Неподдерживаемый тип данных.")
matrix1 = [
    [2, 1, 7, 4],
    [5, 6, 7, 3],
    [9, 8, 2, 12],
    [11, 14, 15, 15]
]
matrix2 = [
    [3, 7, 8, 3, 6],
    [2, 5, 9, 4, 13]
]
transposed_matrix1 = transpose_matrix(matrix1)
transposed_matrix2 = transpose_matrix(matrix2)
print("Транспонированная матрица 1 (list):")
for row in transposed_matrix1:
    print(row)

print("\nТранспонированная матрица 2 (list):")
for row in transposed_matrix2:
    print(row)

np_matrix1 = np.array(matrix1)
np_matrix2 = np.array(matrix2)

transposed_np_matrix1 = transpose_matrix(np_matrix1)
transposed_np_matrix2 = transpose_matrix(np_matrix2)

print("\nТранспонированная матрица 1 (numpy.array):")
print(transposed_np_matrix1)

print("\nТранспонированная матрица 2 (numpy.array):")
print(transposed_np_matrix2)
```
✓ 0.0s

Рисунок 7 – Код для транспонирование матриц

```
Транспонированная матрица 1 (list):
[2, 5, 9, 11]
[1, 6, 8, 14]
[7, 7, 2, 15]
[4, 3, 12, 15]

Транспонированная матрица 2 (list):
[3, 2]
[7, 5]
[8, 9]
[3, 4]
[6, 13]

Транспонированная матрица 1 (numpy.array):
[[ 2  5  9 11]
 [ 1  6  8 14]
 [ 7  7  2 15]
 [ 4  3 12 15]]

Транспонированная матрица 2 (numpy.array):
[[ 3  2]
 [ 7  5]
 [ 8  9]
 [ 3  4]
 [ 6 13]]
```

Рисунок 8 – Транспонированные матрицы

В последнем задании нужно завершить градиентный спуск, начатый в методических материалах. Движение будет происходить в направлении антиградиента, размер шага равен 0.01. В каждой новой точке антиградиент будет разным. Нужно добиться величины MSE меньше 6,36, а также посчитать количество шагов градиентного спуска.

Функция потерь MSE:

$$MSE(a_1, a_2) = \frac{1}{4}((a_1 + 2a_2 - 5)^2 + (5a_1 + 3a_2 - 6)^2 +$$

$$+(2a_1 + 4a_2 - 10)^2 + (3a_1 + 7a_2 - 8)^2).$$

Градиент MSE:

$$\nabla \mathrm{MSE}(a_1, a_2) = \begin{pmatrix} 19.5a_1 + 23a_2 - 39.5 \\ 23a_1 + 39a_2 - 62 \end{pmatrix}.$$

Начальная точка: $(a_1, a_2) = (0.57, 0.91)$.

Начальное значение MSE: 8.56.

Вычисляем новое значение MSE (рис. 9). Значение меньше 6.36 было достигнуто за 6 шагов.

```python
def mse(a1, a2):
    return 0.25 * ((a1 + 2*a2 - 5)**2 + (5*a1 + 3*a2 - 6)**2 + (2*a1 + 4*a2 - 10)**2 + (3*a1 + 7*a2 - 8)**2)
def gradient(a1, a2):
    df_da1 = 19.5 * a1 + 23 * a2 - 39.5
    df_da2 = 23 * a1 + 39 * a2 - 62
    return np.array([df_da1, df_da2])
a1, a2 = 0.57, 0.91
alpha = 0.01
steps = 0
while True:
    current_mse = mse(a1, a2)
    grad = gradient(a1, a2)
    a1 -= alpha * grad[0]
    a2 -= alpha * grad[1]
    steps += 1
    print(f"Шаг {steps}: (a1, a2) = ({a1:.4f}, {a2:.4f}), MSE = {current_mse:.4f}")
    if current_mse < 6.36:
        break

print(f"\nИтог: достигнуто MSE < 6.36 за {steps} шагов.")
```

✓ 0.0s

```
Шаг 1: (a1, a2) = (0.6445, 1.0440), MSE = 8.5608
Шаг 2: (a1, a2) = (0.6737, 1.1086), MSE = 6.8435
Шаг 3: (a1, a2) = (0.6824, 1.1413), MSE = 6.4741
Шаг 4: (a1, a2) = (0.6818, 1.1592), MSE = 6.3879
Шаг 5: (a1, a2) = (0.6772, 1.1703), MSE = 6.3617
Шаг 6: (a1, a2) = (0.6710, 1.1781), MSE = 6.3487

Итог: достигнуто MSE < 6.36 за 6 шагов.
```

Рисунок 9 – Поиск градиента меньшего 6.36

Вывод.

В ходе работы было изучено сложение векторов, умножение вектора на число, транспонирование матриц. А также нахождение скалярного произведение векторов как аналитически, так и геометрически. Помимо этого было изучено нахождение функции векторного аргумента и градиента.