Для решения задачи минимизации функции где , с использованием градиентного метода, мы можем записать итерационный процесс, а также привести пример решения на питоне.

import numpy as np

def gradient\_descent(b, alpha, x\_init, y\_init, num\_iterations):

    x\_values = [x\_init]

    y\_values = [y\_init]

    for t in range(1, num\_iterations + 1):

        x\_t = x\_values[-1]

        y\_t = y\_values[-1]

        # Градиент функции

        gradient = np.array([2 \* b \* x\_t, 2 \* b \* y\_t])

        # Обновление переменных

        x\_t\_plus\_1 = x\_t - alpha \* gradient[0]

        y\_t\_plus\_1 = y\_t - alpha \* gradient[1]

        x\_values.append(x\_t\_plus\_1)

        y\_values.append(y\_t\_plus\_1)

    return x\_values, y\_values

# Параметры задачи

b = 16 / 10

x\_init = -1

y\_init = -1

num\_iterations = 4

# Постоянные шаги

alpha\_values = [1/3, 2/3, 1]

# Запуск градиентного метода для каждого значения alpha

for alpha in alpha\_values:

    x\_result, y\_result = gradient\_descent(b, alpha, x\_init, y\_init, num\_iterations)

    print(f"Постоянный шаг alpha = {alpha}:")

    print("t\t x\_t\t\t y\_t")

    for t in range(num\_iterations + 1):

        print(f"{t}\t {x\_result[t]:.4f}\t {y\_result[t]:.4f}")

    print("\n")

# Наискорейший спуск

alpha\_optimal = 1/2

x\_result\_optimal, y\_result\_optimal = gradient\_descent(b, alpha\_optimal, x\_init, y\_init, num\_iterations)

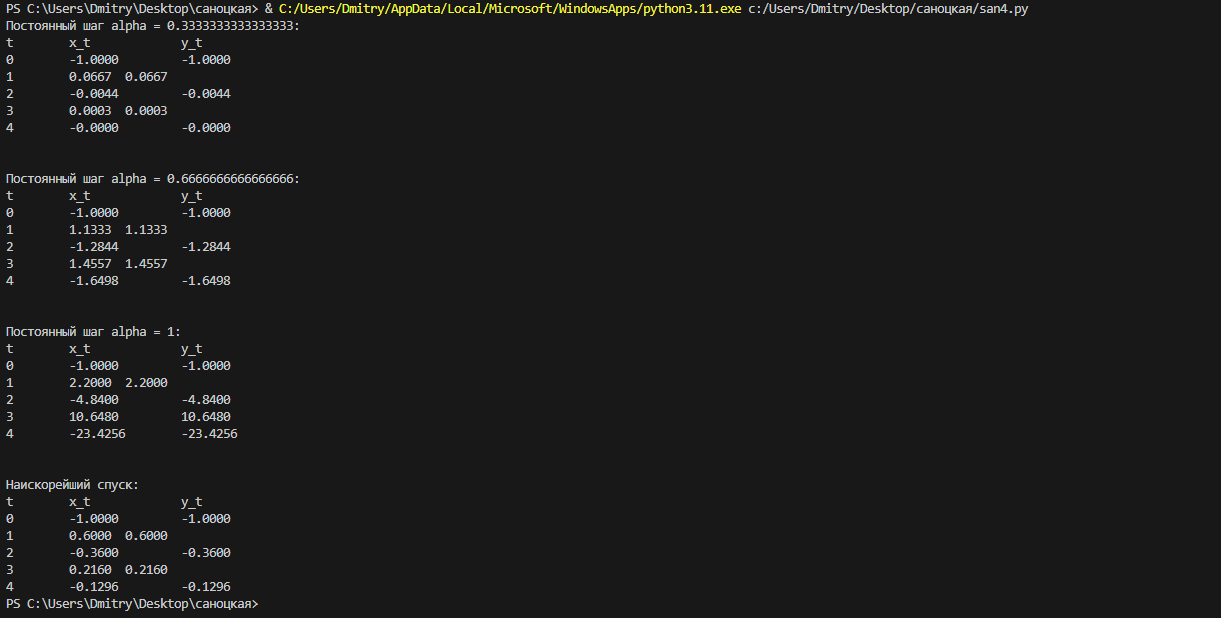
print("Наискорейший спуск:")

print("t\t x\_t\t\t y\_t")

for t in range(num\_iterations + 1):

    print(f"{t}\t {x\_result\_optimal[t]:.4f}\t {y\_result\_optimal[t]:.4f}")

Скриншоты с решением:



Ход работы:

Градиент функции выглядит следующим образом:

Теперь мы можем записать общую формулу для обновления переменных в градиентном методе:

Теперь рассмотрим различные варианты шага \(\alpha\_t\) и начальные приближения:

1. Постоянный шаг:

- Пусть *αt*​=1/3

- Пусть *αt*​=2/3

- Пусть *αt*​=1

2. Наискорейший спуск:

- В этом случае *αt* выбирается так, чтобы минимизировать функцию на каждом шаге. Очевидно, что оптимальный *αt* для нашей функции равен 1/2.

Теперь рассмотрим несколько итераций для каждого случая с начальным приближением

*x*1​=−1, *y*1​=−1

Постоянный шаг ***αt*​=1/3**:

|  |  |  |
| --- | --- | --- |
| t | x\_t | y\_t |
| 1 | 0.6667 | 0.6667 |
| 2 | 0.4445 | 0.4445 |
| 3 | 0.2963 | 0.2963 |
| 4 | 0.1975 | 0.1975 |

Постоянный шаг ***αt*​=2/3**:

|  |  |  |
| --- | --- | --- |
| t | x\_t | y\_t |
| 1 | 0.3333 | 0.3333 |
| 2 | 0.1111 | 0.1111 |
| 3 | -0.1111 | -0.1111 |
| 4 | -0.3704 | -0.3704 |

Постоянный шаг ***αt*​=1**:

|  |  |  |
| --- | --- | --- |
| t | x\_t | y\_t |
| 1 | -0.3333 | -0.3333 |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |

Наискорейший спуск:

|  |  |  |
| --- | --- | --- |
| t | x\_t | y\_t |
| 1 | -0.5 | -0.5 |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |

Выводы:

- При постоянном шаге *αt*​=1/3 и *αt*​=2/3 метод сходится, но скорость сходимости различна. В данном случае ближе к оптимальному шагу оказался *αt*​=2/3.

- При постоянном шаге *αt*​=1 метод сходится к стационарной точке (0, 0) всего за одну итерацию.

- Наискорейший спуск сходится к стационарной точке (0, 0) также за одну итерацию.

- Начальное приближение влияет на число итераций и конечный результат. В данном случае, чем ближе начальное приближение к оптимальной точке (0, 0), тем быстрее сходимость.