Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

ЗВІТ

Про виконання лабораторної роботи №4

Реалізація методу градієнтного спуску

Виконав:

Студент групи ФЕП-31

Сворень Ярослав

Перевірив:

Ас. Рибак А. В.

Львів 2022

Мета роботи: знайти глобальний мінімум довільної функції з використанням методу градієнтного спуску та дослідити результати методу відповідно до різних параметрів alpha, eps.

Хід Роботи:

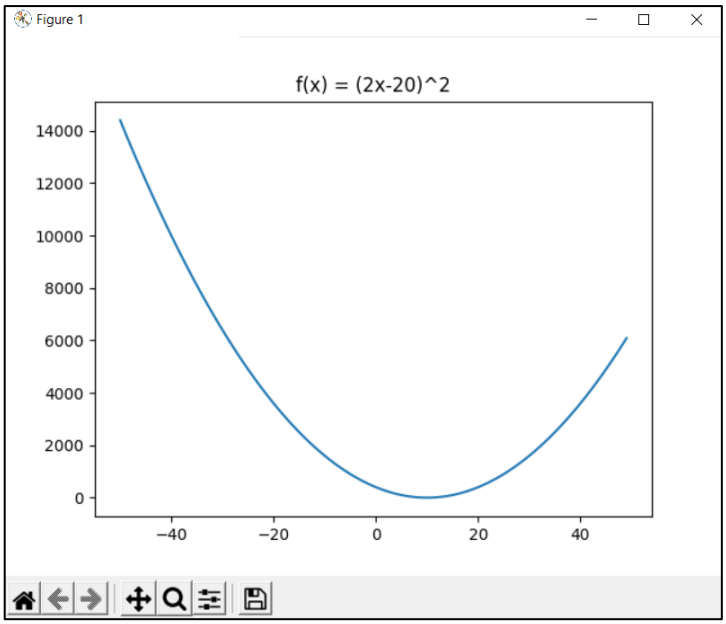
1. Для початку підключаємо необхідні нам бібліотеки, а саме **scipy.misc.derivative()** для другого способу знаходження похідної, **matplotlib.pyplot** для малювання графіків та **pandas** для взаємодії з таблицями. Створюємо зовнішні функції нашої довільної функції **f(x)** та функцію знаходження похідної методом скінченних різниць **d(x)**. Мною було виявлено, що відхилення суттєво зростає при використанні функцій бібліотеки math, тому використовуємо стандартні операнди **Python**.

Lab4ZMN.py > ...

```
1   from scipy.misc import derivative
2   from matplotlib import pyplot as plt
3   import pandas as pd
4
5   def f(x):
6       return (2*x-20)**2
7
8   def df(x):
9       h = 10e-11
10      return (f(x + h) - f(x)) / h
```

2. Далі формуємо масив координат **x,y** та малюємо графік нашої функції **f(x)**. Після того обчислюємо у точці **x=0** похідну **df(x)** та порівнюємо результат з **scipy.misc.derivative()**.

```
28   x_array = []
29   y_array = []
30   for i in range(-50,50):
31       x_array.append(i)
32       y_array.append(f(i))
33   plt.plot(x_array,y_array)
34   plt.title("f(x) = (2x-20)^2")
35   plt.show()
36
37   dy_custom = df(0)
38   dy_lib = derivative(f, 0)
39   print(f"dy_custom = {dy_custom}")
40   print(f"dy_lib = {dy_lib}")
```



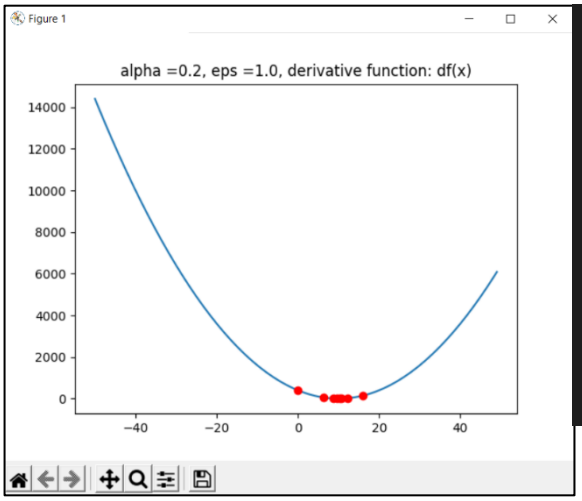

```
PS D:\Programs\ZMN\Lab4ZMN> python -u "d:\Programs\ZMN\Lab4ZMN\Lab4ZMN.py"
dy_custom = -80.00029083632398
dy_lib = -80.0
```

3. **О**писуємо зовнішню функцію **compute_gradient_descent()**, яка допоможе нам знайти координату мінімуму нашої функції. Оскільки ми плануємо порівнювати похідну, знайдену методом скінченних різниць, з похідною бібліотеки **scipy**, то прописуємо параметр **choice**, який буде вказувати який метод знаходження похідної використовувати при виклику функції.

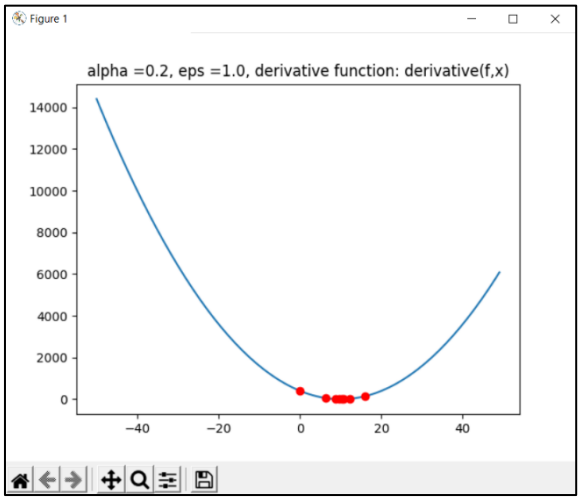
```
12  def compute_gradient_descent(alpha, eps, epoch,choice):
13      x_prev = 0
14      y_prev = f(x_prev)
15      Y = {x_prev: y_prev}
16      for _ in range(epoch):
17          if(choice == 0):
18              x_new = x_prev - alpha * df(x_prev)
19          else:
20              x_new = x_prev - alpha * derivative(f,x_prev)
21          y_new = f(x_new)
22          Y[x_new] = y_new
23          if abs(x_new - x_prev) <= eps:
24              return Y
25          x_prev = x_new
26      return Y
```

4. **В**решті-решт, описуємо масиви значень кроку **alpha**, та точності **eps**, з якими будемо порівнювати результати у циклі знаходження мінімуму, користуючись двома методами знаходження похідних. Згідно з завданням, координати кожної з ітерації будемо наносити на оригінальний графік, щоб візуалізувати процес наближення до мінімуму.

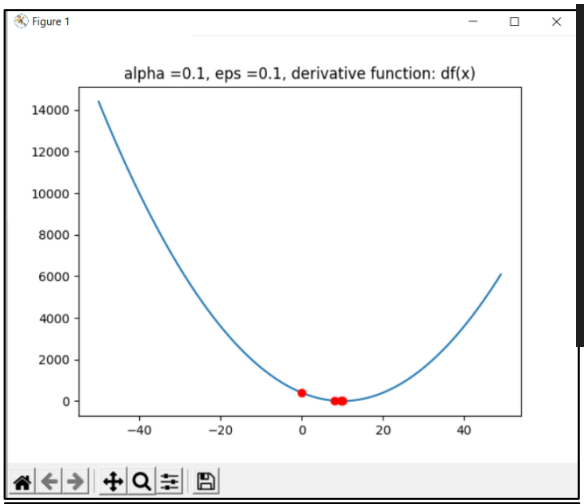
```
45  for i in range(0,4):
46      for j in range(0,2):
47          gradient = compute_gradient_descent(alpha[i], eps[i], 10000,j)
48          actual_iteration = len(gradient)
49          if(j==0):
50              print("\nalpha =",alpha[i],"eps =",eps[i],"derivative function: df(x)")
51              plt.title("alpha ="+str(alpha[i])+", eps ="+str(eps[i])+", derivative function: df(x)")
52          else:
53              print("\nalpha =",alpha[i],"eps =",eps[i],"derivative function: derivative(f,x)")
54              plt.title("alpha ="+str(alpha[i])+", eps ="+str(eps[i])+", derivative function: derivative(f,x)")
55          print(f"actual_iteration = {actual_iteration}")
56          table = pd.DataFrame(gradient.items(), columns=["x", "y"])
57          print(table)
58          result = table.tail(1)
59          print(f"\nGlobal minimum:\n{result}")
60          x = list(gradient.keys())
61          y = list(gradient.values())
62          plt.plot(x_array,y_array)
63          plt.plot(x, y, 'ro')
64          plt.show()
```



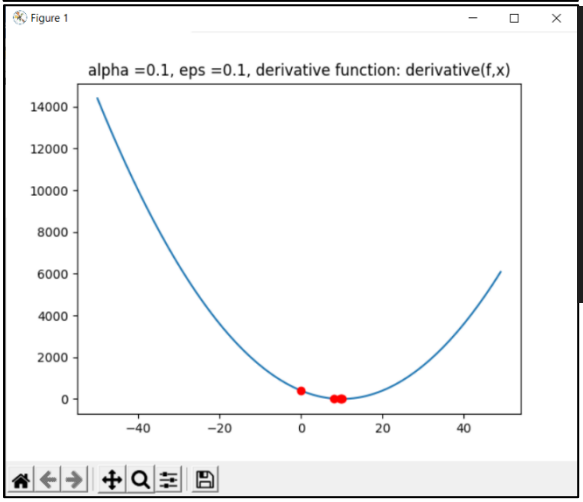

```
alpha = 0.2 eps = 1.0 derivative function: df(x)
actual_iteration = 8
           x           y
0   0.000000  400.000000
1  16.000058  144.002792
2   6.400171   51.835074
3  12.159887   18.660453
4   8.704063    6.717808
5  10.777562    2.418410
6   9.533462    0.870631
7  10.279923    0.313427

Global minimum:
           x         y
7  10.279923  0.313427
```



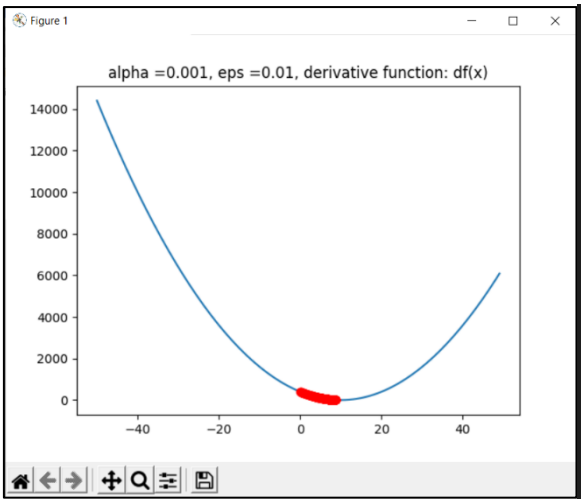

```
alpha = 0.2 eps = 1.0 derivative function: derivative(f,x)
actual_iteration = 8
           x           y
0   0.000000  400.000000
1  16.000000  144.000000
2   6.400000   51.840000
3  12.160000   18.662400
4   8.704000    6.718464
5  10.777600    2.418647
6   9.533440    0.870713
7  10.279936    0.313457

Global minimum:
           x         y
7  10.279936  0.313457
```



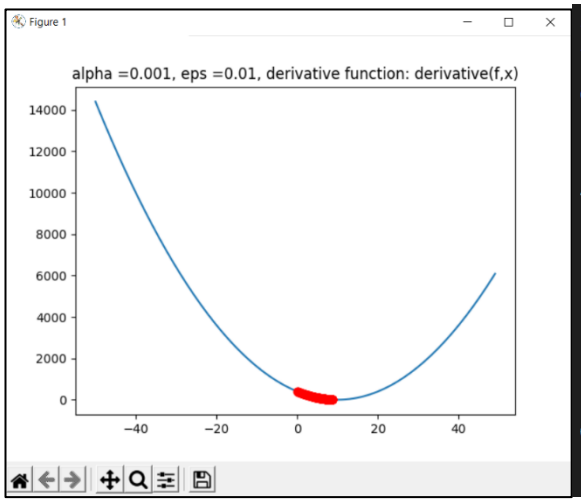

```
alpha = 0.1 eps = 0.1 derivative function: df(x)
actual_iteration = 5
           x           y
0   0.000000  400.000000
1   8.000029   15.999535
2   9.600006    0.639980
3   9.920001    0.025599
4   9.984000    0.001024

Global minimum:
       x         y
4  9.984  0.001024
```



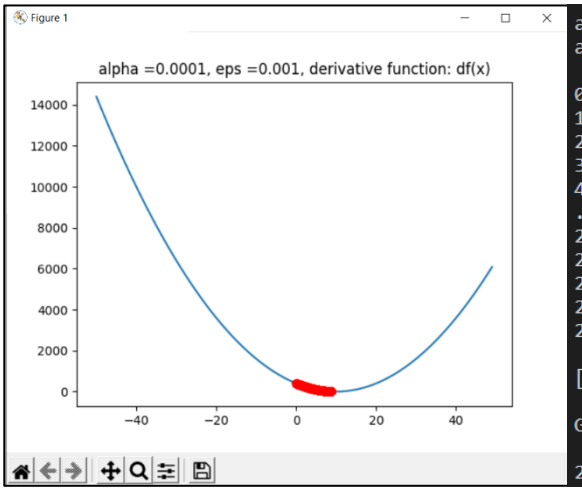

```
alpha = 0.1 eps = 0.1 derivative function: derivative(f,x)
actual_iteration = 5
       x           y
0  0.000  400.000000
1  8.000   16.000000
2  9.600    0.640000
3  9.920    0.025600
4  9.984    0.001024

Global minimum:
       x         y
4  9.984  0.001024
```



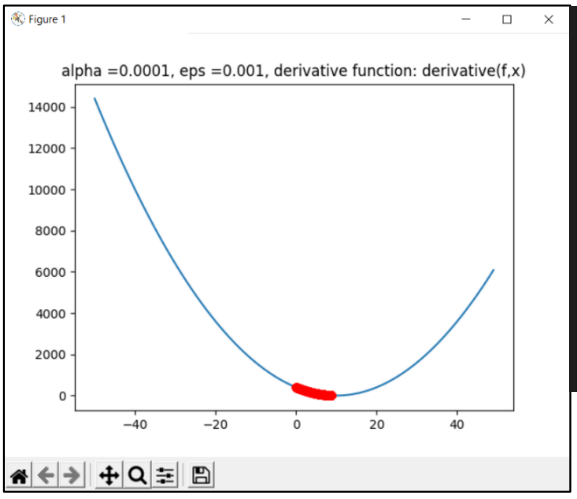

```
alpha = 0.001 eps = 0.01 derivative function: df(x)
actual_iteration = 261
            x           y
0    0.000000  400.000000
1    0.080000  393.625577
2    0.159361  387.352741
3    0.238086  381.179874
4    0.316181  375.105380
..        ...         ...
256  8.720654    6.546905
257  8.730889    6.442574
258  8.741042    6.339905
259  8.751113    6.238872
260  8.761104    6.139449

[261 rows x 2 columns]

Global minimum:
            x         y
260  8.761104  6.139449
```




```
alpha = 0.001 eps = 0.01 derivative function: derivative(f,x)
actual_iteration = 261
            x           y
0    0.000000  400.000000
1    0.080000  393.625600
2    0.159360  387.352782
3    0.238085  381.179928
4    0.316180  375.105445
..        ...         ...
256  8.720654    6.546906
257  8.730889    6.442575
258  8.741041    6.339906
259  8.751113    6.238873
260  8.761104    6.139451

[261 rows x 2 columns]

Global minimum:
            x         y
260  8.761104  6.139451
```




```
alpha = 0.0001 eps = 0.001 derivative function: df(x)
actual_iteration = 2601
            x           y
0    0.000000  400.000000
1    0.008000  399.360254
2    0.015994  398.721532
3    0.023981  398.083830
4    0.031962  397.447151
...       ...         ...
2596  8.747735    6.272668
2597  8.748737    6.262636
2598  8.749738    6.252620
2599  8.750738    6.242619
2600  8.751738    6.232635

[2601 rows x 2 columns]

Global minimum:
            x         y
2600  8.751738  6.232635
```




```
alpha = 0.0001 eps = 0.001 derivative function: derivative(f,x)
actual_iteration = 2601
            x           y
0    0.000000  400.000000
1    0.008000  399.360256
2    0.015994  398.721535
3    0.023981  398.083836
4    0.031962  397.447157
...       ...         ...
2596  8.747735    6.272671
2597  8.748737    6.262638
2598  8.749738    6.252622
2599  8.750738    6.242622
2600  8.751737    6.232638

[2601 rows x 2 columns]

Global minimum:
            x         y
2600  8.751737  6.232638
```

Висновок: на цій лабораторній роботі було розглянуто знаходження похідної методом скінченних різниць та функцією scipy.misc.derivative(), після чого було порівняно їх результати та виявлено, що derivative() не є хорошим варіантом для знаходження похідних, оскільки при використанні методів бібліотеки math вона не спроможна працювати з тригонометричними та іншими формулами, коли метод скінченних різниць не мав жодних проблем у роботі з цими функціями і давав чітку, правильну відповідь. Також було розглянуто метод градієнтного спуску, який показав, що для отримання чіткого мінімуму функції варто ставити крок alpha не менше, чим на 1 тисячну. Вивід з терміналу показав помилкові результати з кроком 0.2 – 10.279, з кроком 0.1 – 9.984, з кроком 0.001 – 8.761, з кроком 0.0001 – 8.751.