**13)**

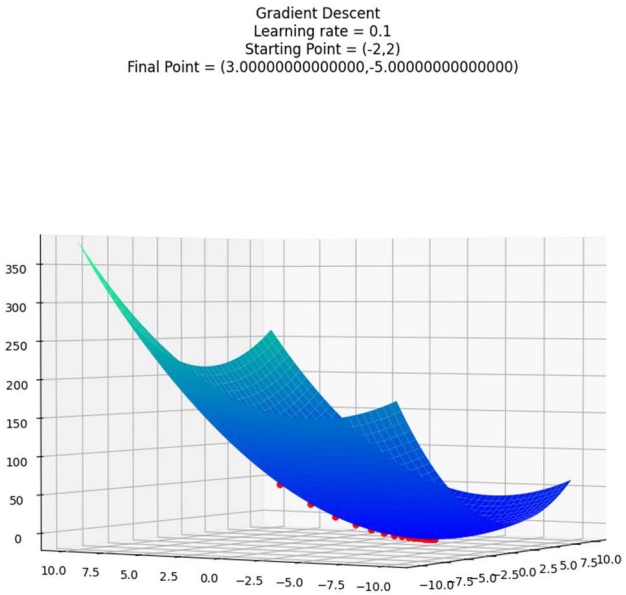
I have experimented with three different learning rates. One thing that I did observe is as the learning rates increases the number of iterations or steps to reach global minimum is decreasing and similarly as the learning rate decreases the number of iterations or steps to reach the global minimum is also increasing.

First, I have tried for only 100 iterations. While with learning rate 0.1, the global minimum has been quickly reached, with learning rate 0.01 it quite did not reach the global minimum but was almost there. However, with learning rate 0.001 it was far away from the global minimum.

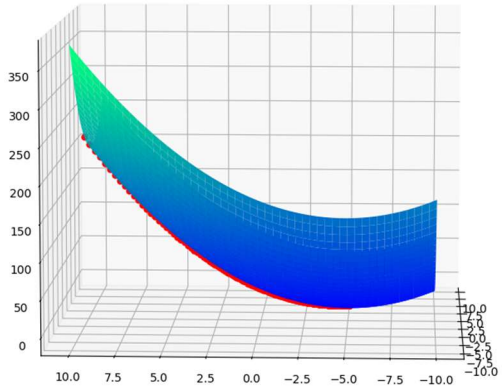
Then, I increased the number of iterations to 500. While with learning rates 0.1 and 0.01 the global minimum was reached, with learning rate 0.001 it still did not reach the global minimum.

Then to see for when the global minimum would be reached with learning rate 0.001, it kept increasing the number of iterations, and found that it reached the global minimum after approximately 4000 iterations.

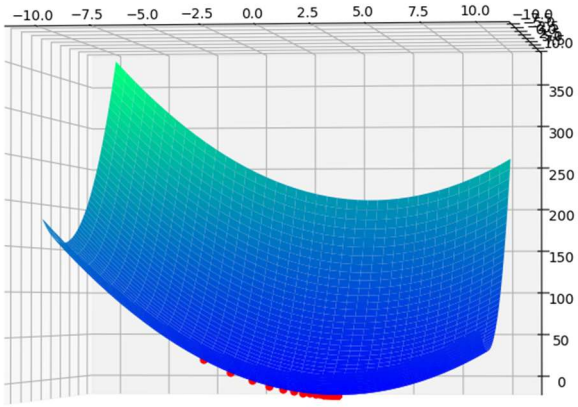
Below are few samples of how with different learning rates, global minimum was reached.



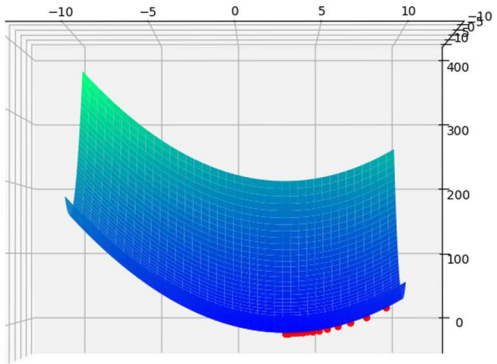
Gradient Descent
Learning rate = 0.01
Starting Point = (-2,10)
Final Point = (2.99979488007427,-4.99938464022282)



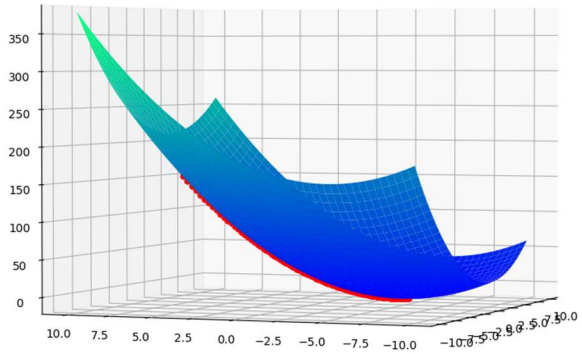

Gradient Descent
Learning rate = 0.1
Starting Point = (-3,-7)
Final Point = (3.00000000000000,-5.00000000000000)



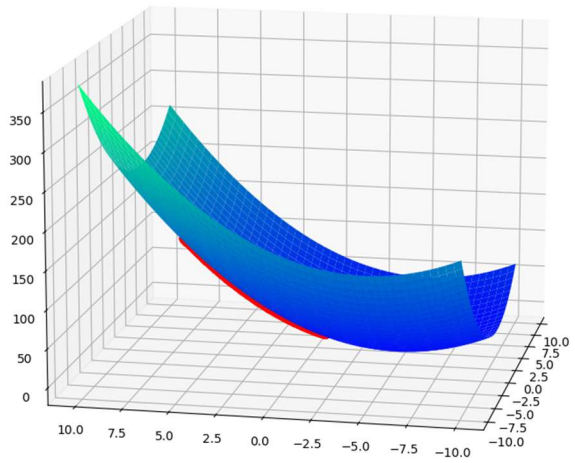

Gradient Descent
Learning rate = 0.1
Starting Point = (9,-5)
Final Point = (3.00000000000000,-5)



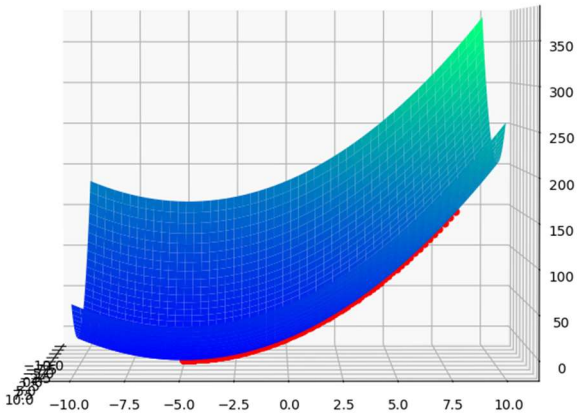

Gradient Descent
Learning rate = 0.01
Starting Point = (-2,7)
Final Point = (2.99979488007427,-4.99950771217825)



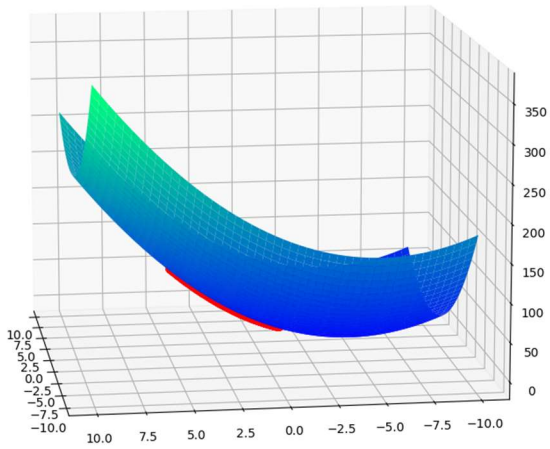
Gradient Descent
Learning rate = 0.001
Starting Point = (-4,6)
Final Point = (0.427421215999887,-0.957376196571252)



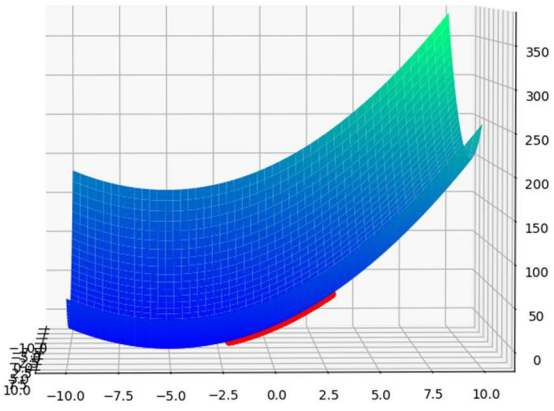

Gradient Descent
Learning rate = 0.01
Starting Point = (3,8)
Final Point = (3,-4.99946668819311)



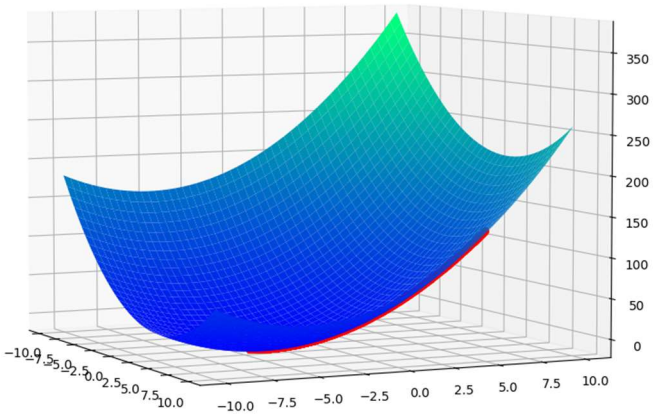

Gradient Descent
Learning rate = 0.001
Starting Point = (0,5)
Final Point = (1.89746623542852,-1.32488745142841)




Gradient Descent
Learning rate = 0.001
Starting Point = (5,3)
Final Point = (3.73502250971432,-2.05990996114273)



Gradient Descent
Learning rate = 0.001
Starting Point = (8,6)
Final Point = (3.00166393039022,-4.99633935314151)
Number of iterations = 4000




Below are the code snippets(screenshots) for the above-described experiments.

GD_nonlinear_polynomial_regression.py

```python
from sympy import lambdify, symbols, diff
import numpy as np
import random
from matplotlib import pyplot as plt

def apply_GD(function_equation, x, y, learning_rate = 0.1):
    partial_derivative_x = diff(function_equation,x)
    partial_derivative_y = diff(function_equation,y)

    print(partial_derivative_x, partial_derivative_y)

    x_list = list()
    y_list = list()

    x0 = random.randint(-10,10)
    y0 = random.randint(-10,10)

    # print(x0, y0, end='\n')
    x_list.append(x0)
    y_list.append(y0)

    for i in range(4000):
        eval_partial_derivative_at_x = partial_derivative_x.subs(x, x0)
        eval_partial_derivative_at_y = partial_derivative_y.subs(y, y0)

        x0 = x0 - (learning_rate * eval_partial_derivative_at_x)
        y0 = y0 - (learning_rate * eval_partial_derivative_at_y)

        x_list.append(x0)
        y_list.append(y0)

    # print(eval_test_x, eval_test_y)
    # plt.scatter(x_list, y_list)
    # plt.show()

    return x_list, y_list
```

```python
def main():

    x = symbols('x')
    y = symbols('y')

    function_equation = x**2 - 6*x + y**2 +10*y +20

    learning_rate = 0.001

    x_coordinates_list, y_coordinates_list = apply_GD(function_equation, x, y, learning_rate)

    print(x_coordinates_list[len(x_coordinates_list)-1], y_coordinates_list[len(y_coordinates_list)-1])

    numpy_function = lambdify([x,y], function_equation, 'numpy')

    x_grid = np.linspace(-10, 10, 40)
    y_grid = np.linspace(-10, 10)

    X, Y = np.meshgrid(x_grid, y_grid)
    Z = numpy_function(X, Y)

    x_array = np.array(x_coordinates_list, dtype="float64")
    y_array = np.array(y_coordinates_list, dtype="float64")
    z_coordinates_list = list(numpy_function(x_array, y_array))

    ax = plt.axes(projection = '3d')
    ax.plot_surface(X, Y, Z, cstride = 1, rstride = 1, cmap = "winter", edgecolor = None)
    ax.plot(x_coordinates_list, y_coordinates_list, z_coordinates_list, 'ro', markersize = 5)
    ax.set_title("Gradient Descent \n Learning rate = {0}\n Starting Point = ({1},{2})\n Final Point = ({3},{4})\n Number of iterations = 4

    plt.show()

if __name__ == "__main__":
    main()
```