Name: Swapnil De Sarkar ID:231003004031

Assignment-7

```
import numpy as np
import matplotlib.pyplot as plt

def function1(x):
  return x**2 - 4*x

def derivative_function1(x):
  return 2*x - 4

def function2(x):
  return np.sin(3*x) + np.sin(2*x) + np.sin(x)

def derivative_function2(x):
  return 3*np.cos(3*x) + 2*np.cos(2*x) + np.cos(x)

def gradient_descent(start_point, learning_rate, iterations, function, derivative):
  points = [start_point]
  current_point = start_point

  for i in range(iterations):
    gradient = derivative(current_point)
    current_point = current_point - learning_rate * gradient
    points.append(current_point)
  return np.array(points)
def plot_results(function, points, title, x_range):
  plt.figure(figsize=(10, 6))
  x = np.linspace(x_range[0], x_range[1], 1000)
  y = function(x)
  plt.plot(x, y, 'b-', label='Function')
  y_points = function(points)
  plt.plot(points, y_points, 'ro-', label='Gradient Descent Path')
  plt.plot(points[0], y_points[0], 'go', markersize=10, label='Start Point')
  plt.plot(points[-1], y_points[-1], 'mo', markersize=10, label='End Point')
  plt.title(title)
```

```
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True)
    plt.legend()
    plt.show()
if __name__ == "__main__":
    # Part 1: Function y = x^2 - 4x
    print("Part 1: Function y = x^2 - 4x")

    # Initial data points
    initial_points = np.array([-2, -1, 0, 1, 2, 3, 4, 5, 6])

    # Run 1: S = 10, L = 0.01, N = 500
    points1 = gradient_descent(10, 0.01, 500, function1, derivative_function1)
    plot_results(function1, points1, 'Function 1: S=10, L=0.01, N=500', [-3, 11])
    print(f"Final point for Run 1: x = {points1[-1]}, y = {function1(points1[-1])}")

    # Run 2: S = 10, L = 0.1, N = 100
    points2 = gradient_descent(10, 0.1, 100, function1, derivative_function1)
    plot_results(function1, points2, 'Function 1: S=10, L=0.1, N=100', [-3, 11])
    print(f"Final point for Run 2: x = {points2[-1]}, y = {function1(points2[-1])}")

    # Run 3: S = 10, L = 1.0, N = 100
    points3 = gradient_descent(10, 1.0, 100, function1, derivative_function1)
    plot_results(function1, points3, 'Function 1: S=10, L=1.0, N=100', [-3, 11])
    print(f"Final point for Run 3: x = {points3[-1]}, y = {function1(points3[-1])}")

    # Part 2: Function y = sin(3x) + sin(2x) + sin(x)
    print("\nPart 2: Function y = sin(3x) + sin(2x) + sin(x)")

    # Initialize data with 101 evenly spaced points between -4 and 4
    x_data = np.linspace(-4, 4, 101)

    # Run 1: S = 0, L = 0.01, N = 500
    points4 = gradient_descent(0, 0.01, 500, function2, derivative_function2)
    plot_results(function2, points4, 'Function 2: S=0, L=0.01, N=500', [-4, 4])
    print(f"Final point for Run 1: x = {points4[-1]}, y = {function2(points4[-1])}")
```

```
# Run 2: S = 1, L = 0.01, N = 500
points5 = gradient_descent(1, 0.01, 500, function2, derivative_function2)
plot_results(function2, points5, 'Function 2: S=1, L=0.01, N=500', [-4, 4])
print(f"Final point for Run 2: x = {points5[-1]}, y = {function2(points5[-1])}")
```

Output:

Part 1: Function y = x^2 - 4x

Final point for Run 1: x = 2.0003281918811644, y = -3.999999892290089

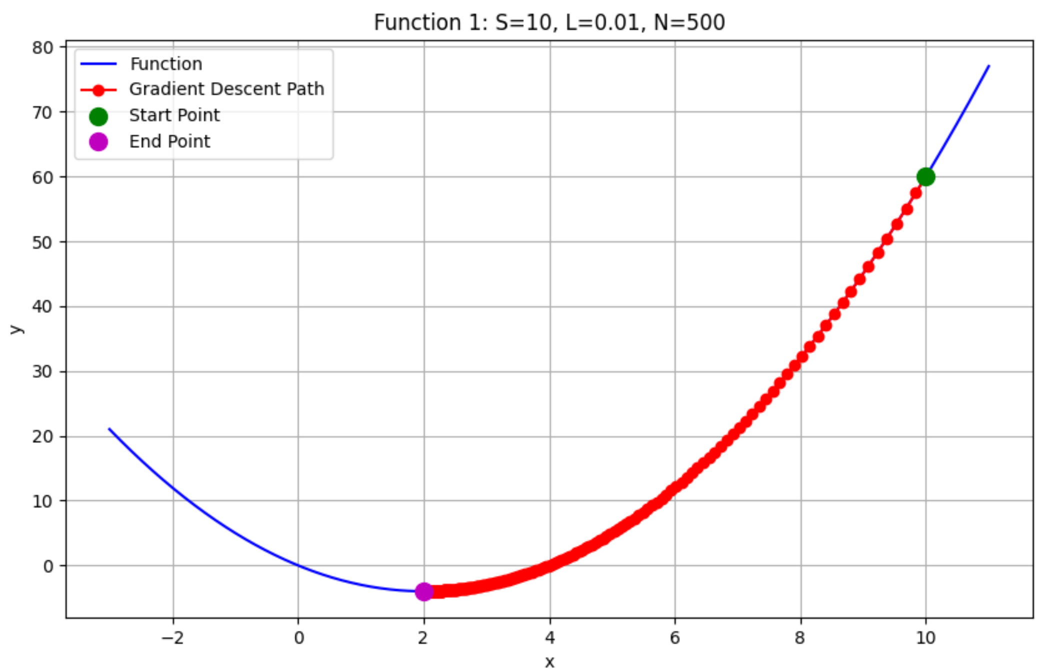
Final point for Run 2: x = 2.000000001629629, y = -4.0

Final point for Run 3: x = 10.0, y = 60.0

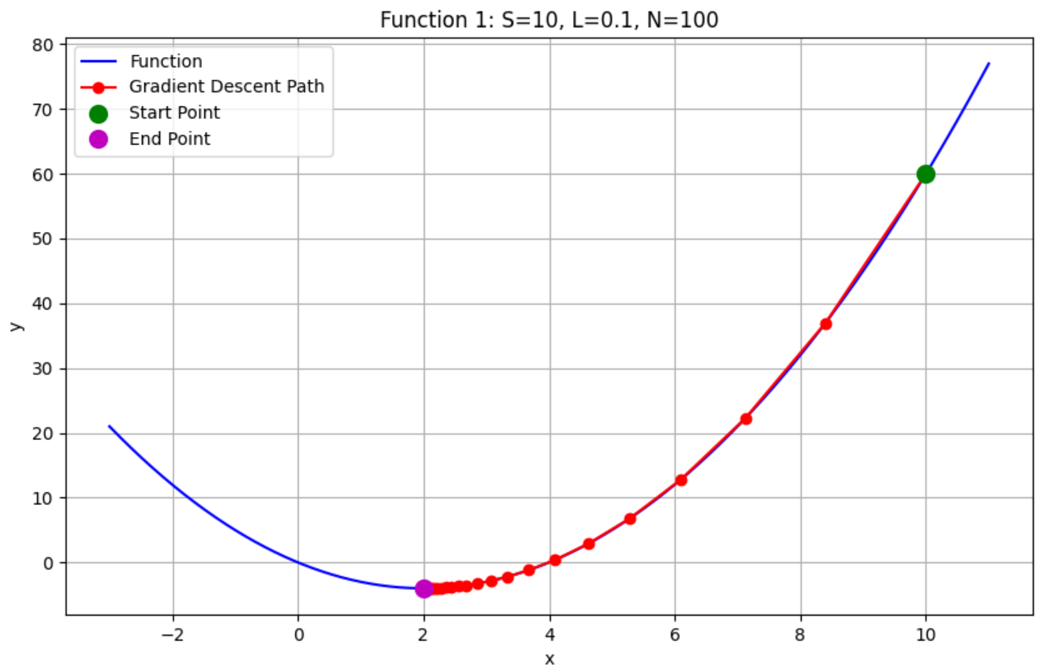
Part 2: Function y = sin(3x) + sin(2x) + sin(x)

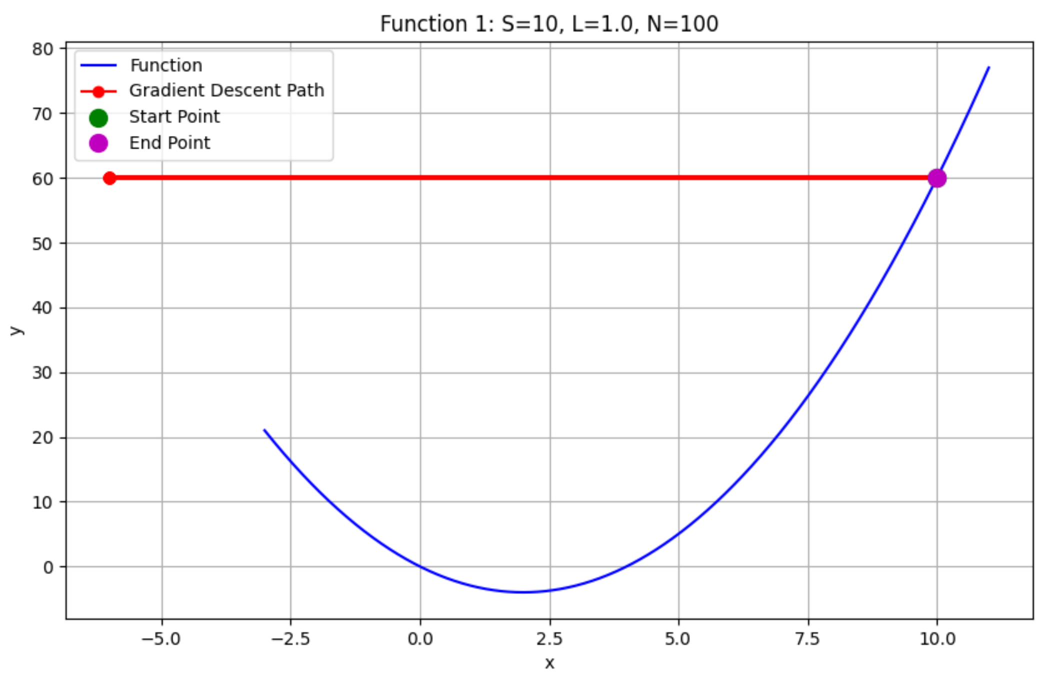
Final point for Run 1: x = -0.6672910715244723, y = -2.499607604320057

Final point for Run 2: x = 1.815198410100203, y = -0.24232067389192802



Name: Swapnil De Sarkar ID:231003004031





Name: Swapnil De Sarkar ID:231003004031