ASSIGNMENT_5 DOCUMENTATION

Beneath attached are firstly, the animations and the plots for the 1D and 2D functions respectively, alongwith the output of the gradient descent algorithm. The order of the functions defined have been changed and grouped to the 1D ones first and then the 2D ones.

General pointers regarding the code include that learning rates need to be modified by hit-and-trial approach to get an accurate result. The start points for the gradient descent have to be chosen wisely also.

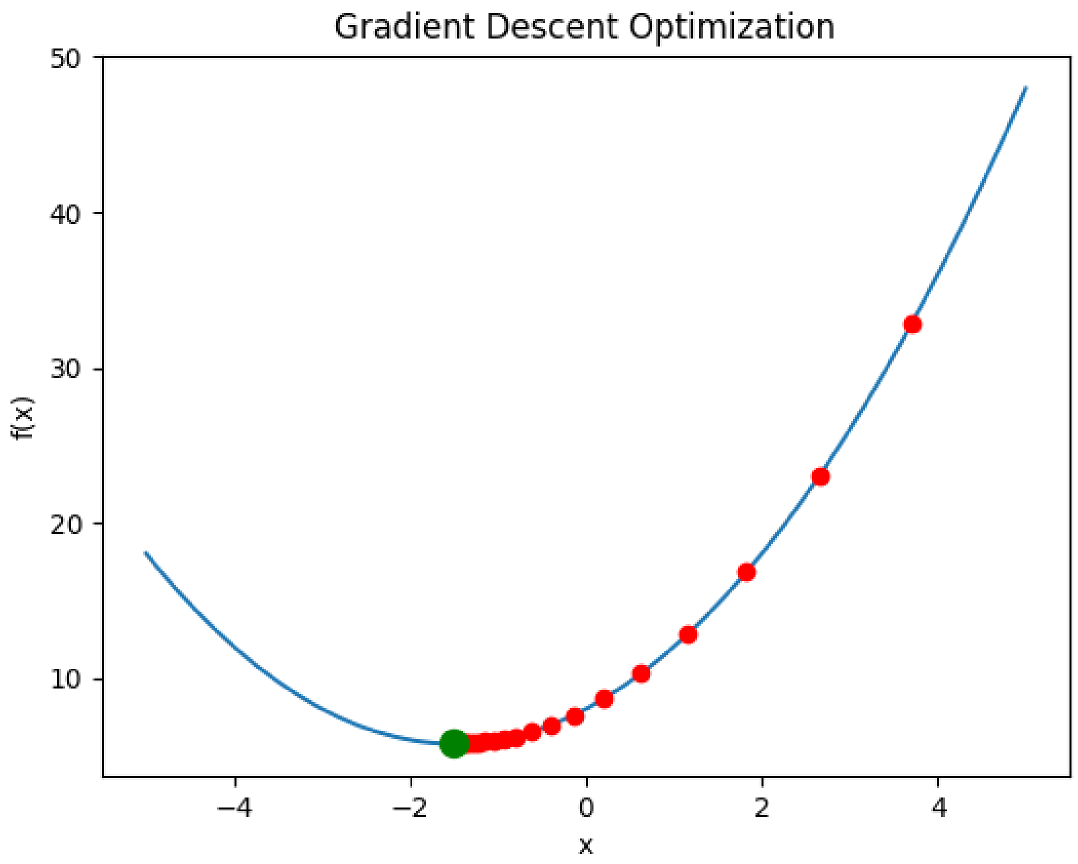
Herewith is attached the animations and the plots obtained for the 1D functions, and the plots for the 2D functions.

Note: The animations were showing valid outputs on the Jupyter terminal until the submission date’s evening. I tried rerunning them post submission but the terminal said the ‘pillow’ extension for saving the animations were not valid, thus I had to attach the .ipnyb file for the animation in the last minute, they give valid outputs.

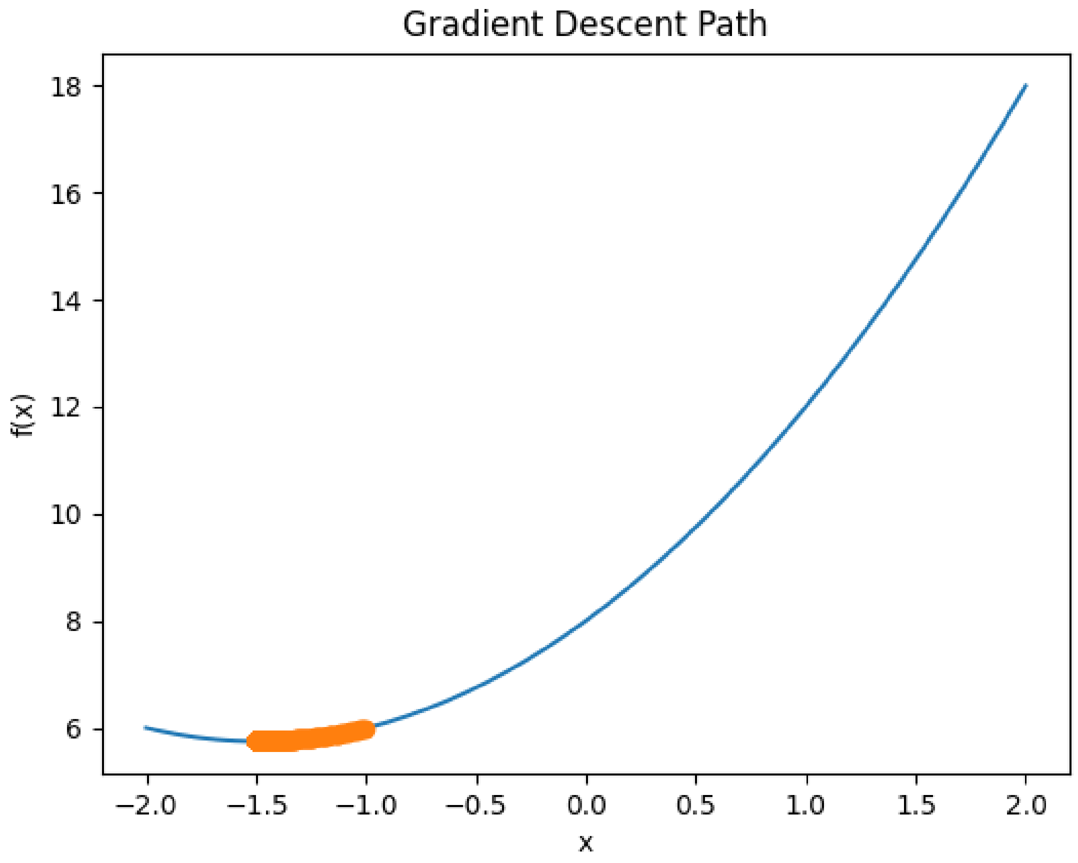
Animations, Plots and Results of the Gradient Descent Algorithm:

Drive link for the animations:
https://drive.google.com/drive/u/0/folders/1FVQaCBLY2uR82ThUYgLi9CXPxjogbjap

Snapshot of the Animation:



Gen_1D figure2:

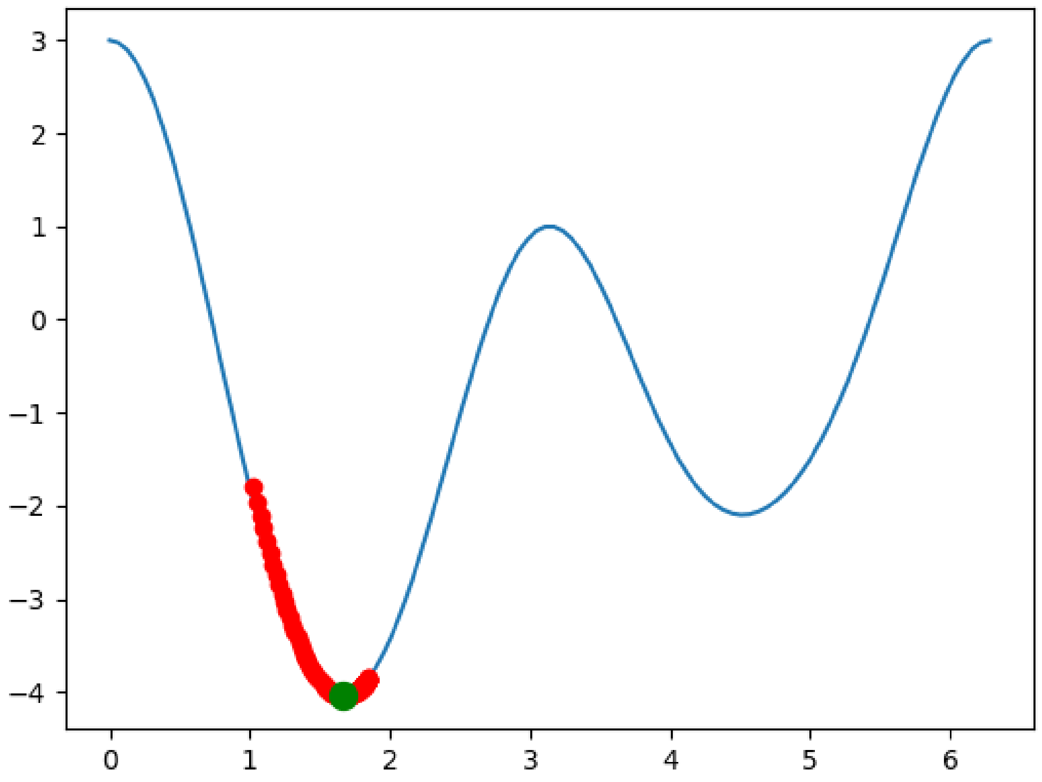


Result:

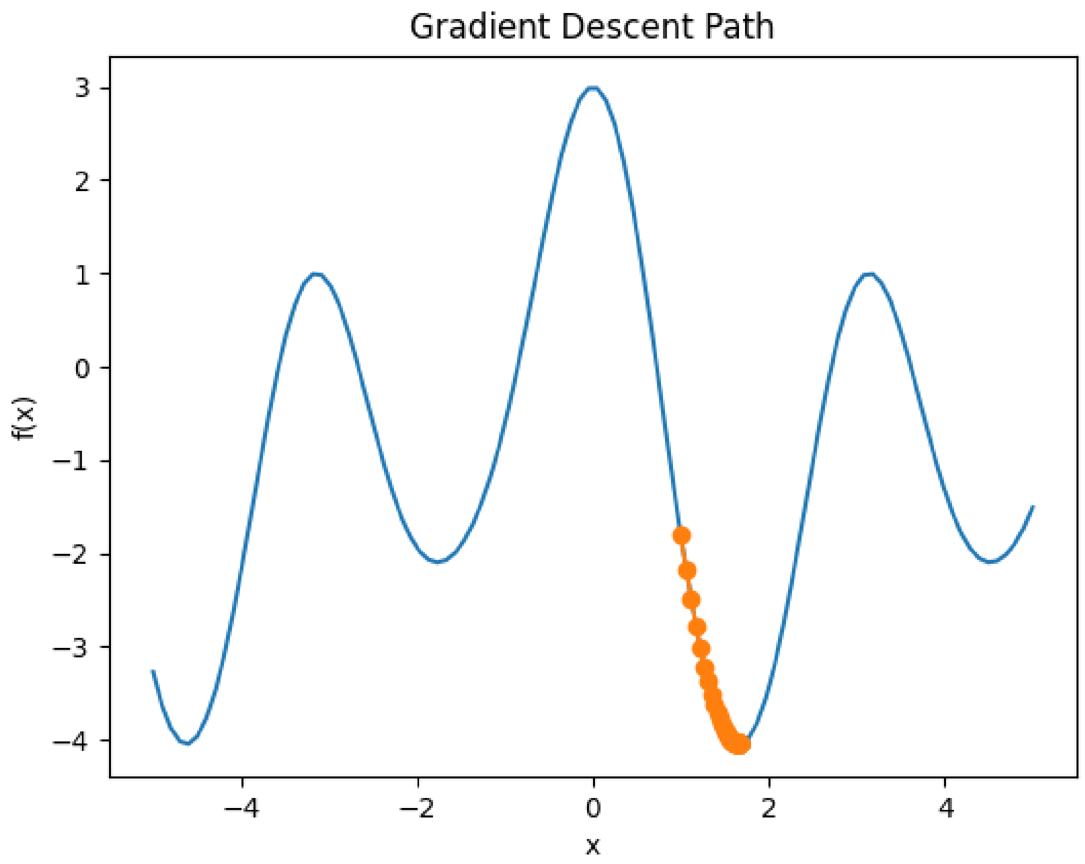
The minima of the given function is 5.750000000024869 at -1.4999950130714144

Animation 2:

Snapshot of the Animation (different coordinates of limits taken in the graph):



Gen_1D figure2:

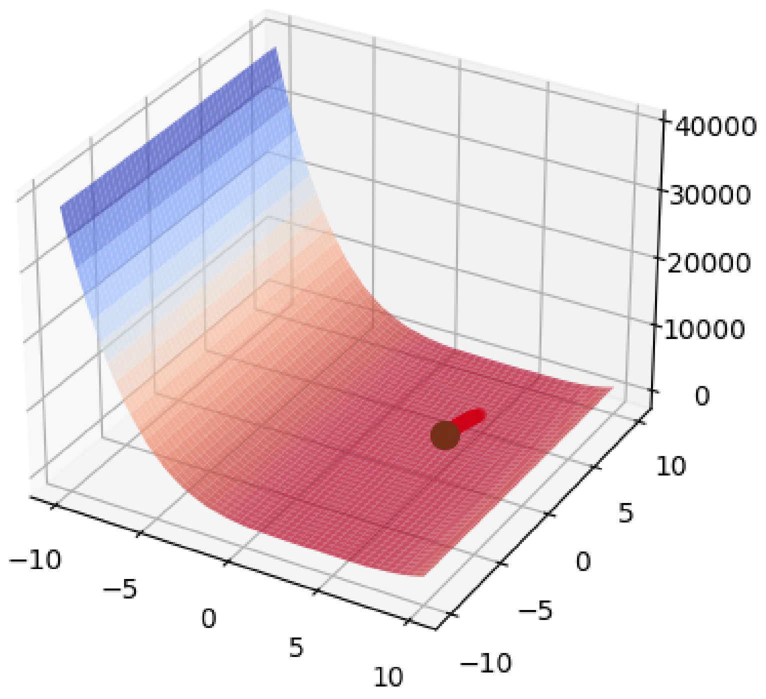


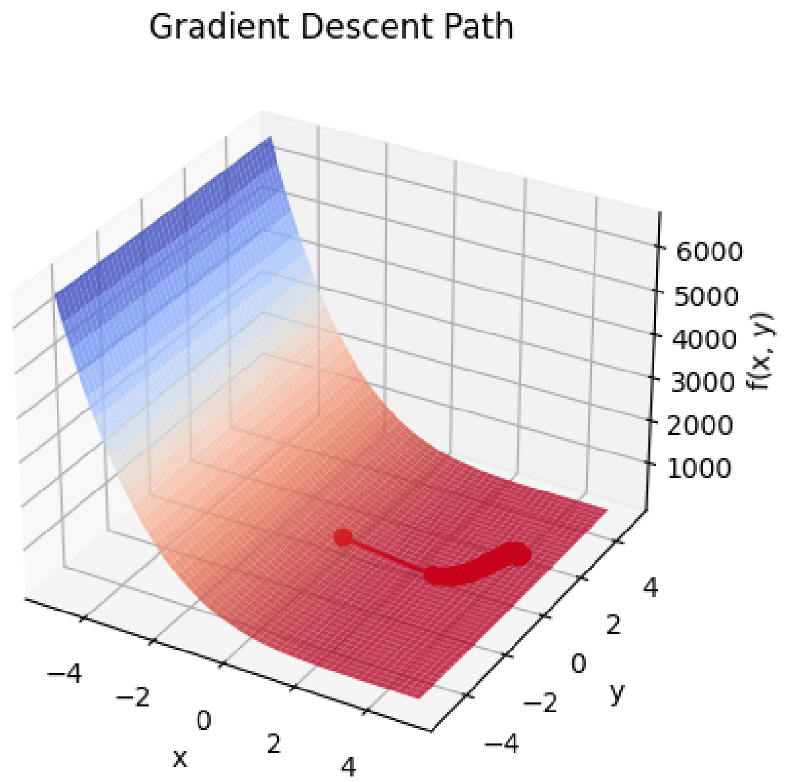
Result:

The minima of the given function is -4.045412051568286 at 1.6616599301916752.

Gen_2D figure1:

Gradient Descent Optimization in 3D



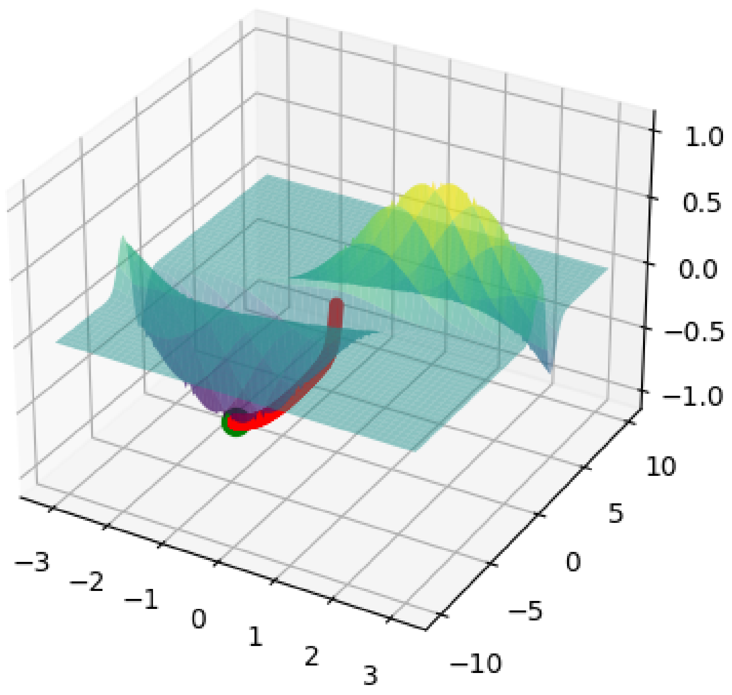


Result:

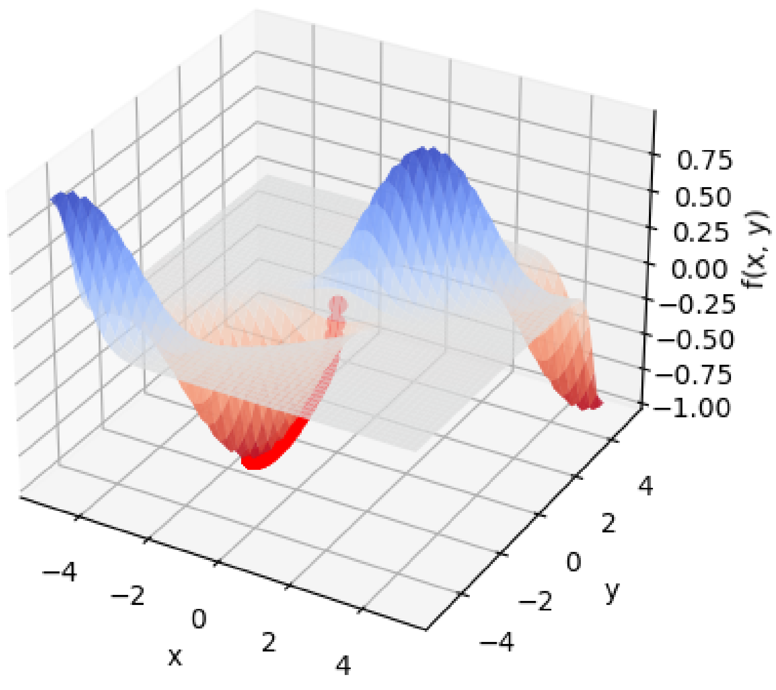
The minima of the given function is 2.000000000003297 and it occurs at (3.9986427901506625, 1.9999999999999947)

Gen_2D figure2:

Gradient Descent Optimization in 3D



Gradient Descent Path



Result:

The minima of the given function is -1.0 and it occurs at (-1.5707963267931242, -1.570796326793513)

Code explanation for the 1D functions, plots and animations:

Code explanation for the 1D Plots:

1. Importing Libraries

2. Initializing Variables:

start: The initial value of x from which optimization begins.
lr (learning rate): The step size used for gradient descent.

3. Defining the One-Dimensional Function and Derivative Function:

func(x): This function defines a one-dimensional function of x.
deriv(x): Calculates the derivative of the function at a given point x. The derivative represents the slope of the function at that point and is used to update the value of x during gradient descent.

3. Creating a Plot:

The code initializes a Matplotlib figure and axes for plotting.

4. Generating Base Data:

xbase: An array of values used for plotting the base function.
ybase: Represents the function values corresponding to xbase and is plotted as a blue curve.

5. Gradient Descent Function:

gradient_descent_1d(func, deriv, start, lr): This function performs gradient descent optimization for a one-dimensional function.
It takes the following arguments:
func: The one-dimensional function to be optimized.
deriv: The derivative of the function.
start: The initial value of x.
lr: The learning rate for gradient descent.
The function iteratively updates x using gradient descent until the gradient magnitude becomes very small (converges).
It keeps track of the optimization path in path_x and path_y.

6. Display and Save the Plot:

The plot is displayed using plt.show().
The code also saves the plot as "figure1" in the working directory using plt.savefig.

7. Returning Optimization Result:

The function returns the (x, y) point and the corresponding function value, which is the minimum value found during optimization.

The learning rate (lr) can significantly affect the convergence of the optimization. We need to experiment with different learning rates.

Code explanation for the 1D Animations:

1. Importing Libraries

2. Initializing Variables:

start: The initial value of x from which optimization begins.
lr (learning rate): The step size used for gradient descent.

3. Defining the One-Dimensional Function and the Derivative Function:

func(x): This function defines a one-dimensional function of x.
deriv(x): Calculates the derivative of the function at a given point x. The derivative represents the slope of the function at that point and is used to update the value of x during gradient descent.

4. Creating a Plot:

The code initializes a Matplotlib figure and axes for plotting.

5. Generating Base Data:

xbase: An array of values used for plotting the base function.
ybase: Represents the function values corresponding to xbase and is plotted as a blue curve.

6. Gradient Descent Function:

gradient_descent_1d(func, deriv, start, lr): This function performs gradient descent optimization for a one-dimensional function.
It takes the following arguments:
func: The one-dimensional function to be optimized.
deriv: The derivative of the function.
start: The initial value of x.
lr: The learning rate for gradient descent.
The function iteratively updates x using gradient descent until the gradient magnitude becomes very small (converges).
It keeps track of the optimization path in path_x and path_y.

7. Plot Optimization Path:

The code plots the optimization path as a red curve.
Labels and a title are added to the plot for clarity.

8. Display and Save the Plot:

The plot is displayed using plt.show().
The code also saves the plot as "animation1" and "animation2" in the working directory.

9. Return Optimization Result:
The function returns the (x, y) point and the corresponding function value, which is the minimum value found during optimization.

10. Optimization and Output:

The gradient_descent_1d function is called with the provided function, derivative, initial value, and learning rate.
The code prints the minimum value and the corresponding x point where it occurs.

The learning rate (lr) can significantly affect the convergence of the optimization. We need to experiment with different learning rates.

Some important pointers regarding the code:

bestcost is a variable that could be used to keep track of the best cost found during optimization, but it's not used in this code.

fig, ax = plt.subplots() initializes a Matplotlib figure and axes for plotting.

xall and yall are lists that will be used to keep track of x and y values during the optimization.

lnall is a red dot that represents the current position during optimization.

lngood is a green dot that represents the best position found during optimization (in this code, it always shows the current position).

Code explanation for the 2D functions:

1. Importing the libraries

2. Defining the Two-Variable Function and Partial Derivatives

3. Initializing Variables:

   bestcost: A large initial value for the best cost found during optimization.
   startx and starty: Initial values for x and y from which optimization begins.

lr (learning rate): The step size used for gradient descent.
These are adjusted using hit and trial approaches.

4. Gradient Descent Function:

   twovariable(f, df_dx, df_dy, startx, starty, lr): This function performs gradient descent optimization for a two-variable function.
   It takes the following arguments:
   f: The two-variable function to be optimized.
   df_dx and df_dy: The partial derivatives of the function with respect to x and y.
   startx and starty: The initial values for x and y.
   lr: The learning rate for gradient descent.
   The function iteratively updates x and y using gradient descent until the gradient magnitude becomes very small (converges).
   It keeps track of the optimization path in path_x and path_y.

5. 3D Plotting and Visualization:

   The code creates a 3D plot to visualize the optimization path.

It defines a range of values for x and y, computes the corresponding function values, and plots the function surface.
Labels and titles are added for clarity.

6. Saving and Displaying the Plot:

   The code saves the 3D plot as "figure3" and "figure4" using plt.savefig.

7. Return Optimization Result, Optimization and Output:

   The function returns the (x, y) point and the corresponding function value, which is the minimum value found during optimization.

   The twovariable function is called with the provided function, derivatives, initial values, and learning rate. The code prints the minimum value and the corresponding (x, y) point where it occurs.

Collaborations:

Confirmed answers expected to various problems from a few people, and verified the graphs. Did not get the approach to one section of the code, especially was stuck with the plotting of the 3D graph, so took help from a senior also.