ASSIGNMENT 5 : GRADIENT DESCENT

Problem 1

- I have defined a generic gradient descent function *generic*. It can take five arguments
  - Function: It takes the function defined in the starting. It varies from problem to problem.
  - Derivative: It takes the derivative of the function.
  - frange: The range within which the minimum should be obtained.
  - bestx: Starting point for gradient descent.
  - learningrate: It decides by how much the bestx should change each time.
- This gradient function can be used for any single variable function. But, one has to care in defining the function and derivatives correctly, before calling them in the gradient descent function.
- I have observed the plot and changed the bestx accordingly, such that optimum is found.

Parameters taken:

- bestx : 4
- learningrate : 0.05
- frame : 50
- frange : [-5,5]

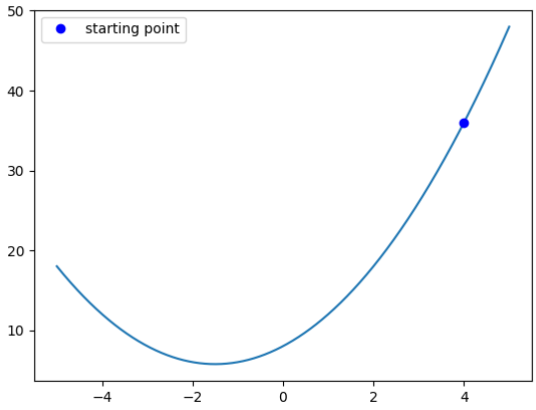


Figure 1: Prob 1: Starting point

Problem 2

- For the 3D plot `from mpl_toolkits.mplot3d import Axes3D` was im-

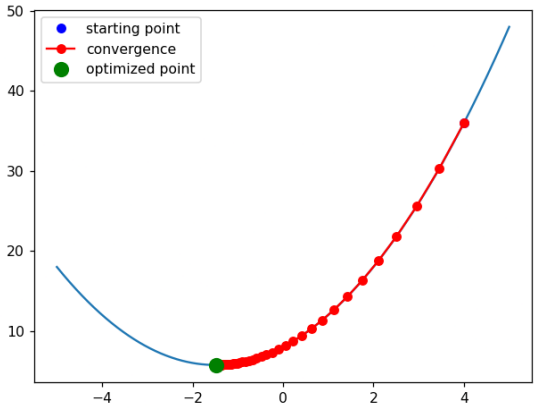


Figure 2: Prob 1: Getting to the optimized point

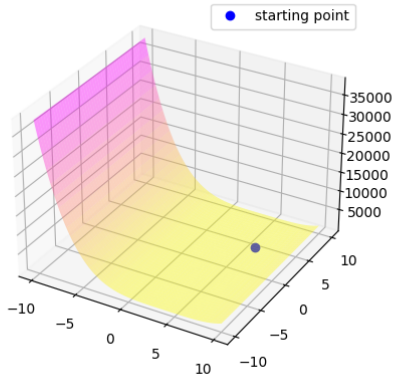
ported.

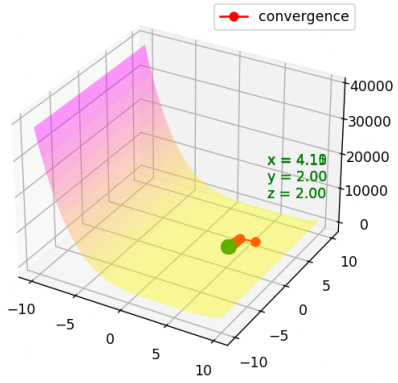
- Function derrivatives were manually written.
- np.meshgrid is used to create coordinate grid arrays from two or more coordinate vectors. Here such grid is created for xbase and ybase.
- Tried to get the initial values from the graph, but couldn't make proper guess, so I have calculated the minimum value and took the nearest values as initial values.
- Graph was plotted using plot_surface.(alpha is used to create transperancy)
- For 3D plots, lnall can't take empty lists as arguments, so xall, yall, zall are initialised to bestx, besty, bestcost.
- Here, generic gradient descent function *generic* takes only seven arguments:
  - Function: It takes the function defined in the starting. It varies from problem to problem.
  - Derivative_x: It takes the derivative of the function with respect to x.
  - Derivative_y: It takes the derivative of the function with respect to y.
  - frange: The range within which the minimum should be obtained.
  - bestx, besty: Starting point for gradient descent.
  - learningrate: It decides by how much the bestx should change each time.

- The gradient descent function tries to optimize the value simultaneously for both x and y coordinates.
- `set_data`, `set_3d_properties` are used for updating values for the 3D line plot.

Parameters taken:

- bestx, besty : 6, 4
- learningrate : 0.05
- frame : 100
- frange : [-10,10]





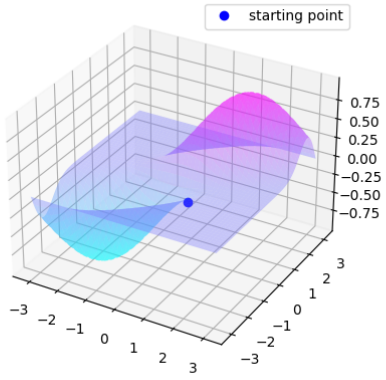
Problem 3

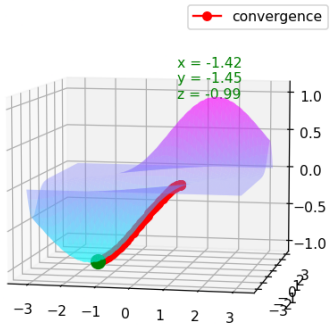
- It's analysis was similar to the Problem 2.
- Initial values were obtained from the graph plotted using plot_surface.

Parameters taken:

- bestx, besty : -1, 1
- learningrate : 0.1

- frame : 100
- frange : [-pi,pi]



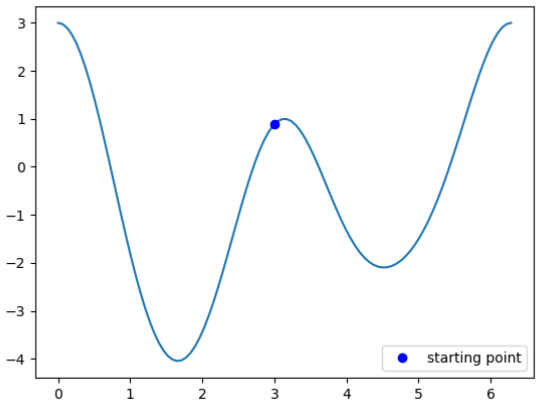


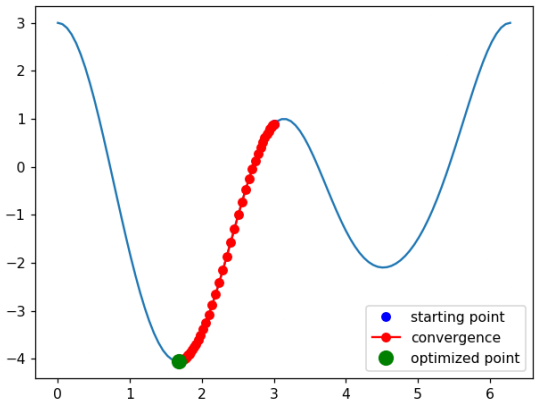
Problem 4

- It's analysis was similar to the Problem 1.
- Same gradient descent function is used.
- Initial values were taken from the graph plotted.

Parameters taken:

- bestx : 3
- learningrate : 0.01
- frame : 100
- frange : [0,2pi]





Running the code:

I have submitted two python files. One for 2D generic function and other for 3D generic function. Function and their derivatives have to be pre-defined by the user before plotting, in the code, manually. One has to take care in giving arguments like bestx/besty, frange, learning rate which decides the result.