1. **Using gradient_func_1D:-**

a) Problem 1 (f1 and f2):

- Function: f1(x) = x^2 + 3x + 8 with its derivative f2(x) = 2x + 3

- **Approach:**

1. Choose an initial value for `bestx`, in this case, `bestx = 0.0005`(**any value within the range specified**, the same applies for other bestxs, bestys).

2. Initialise a plot to visualise the function and optimisation process.

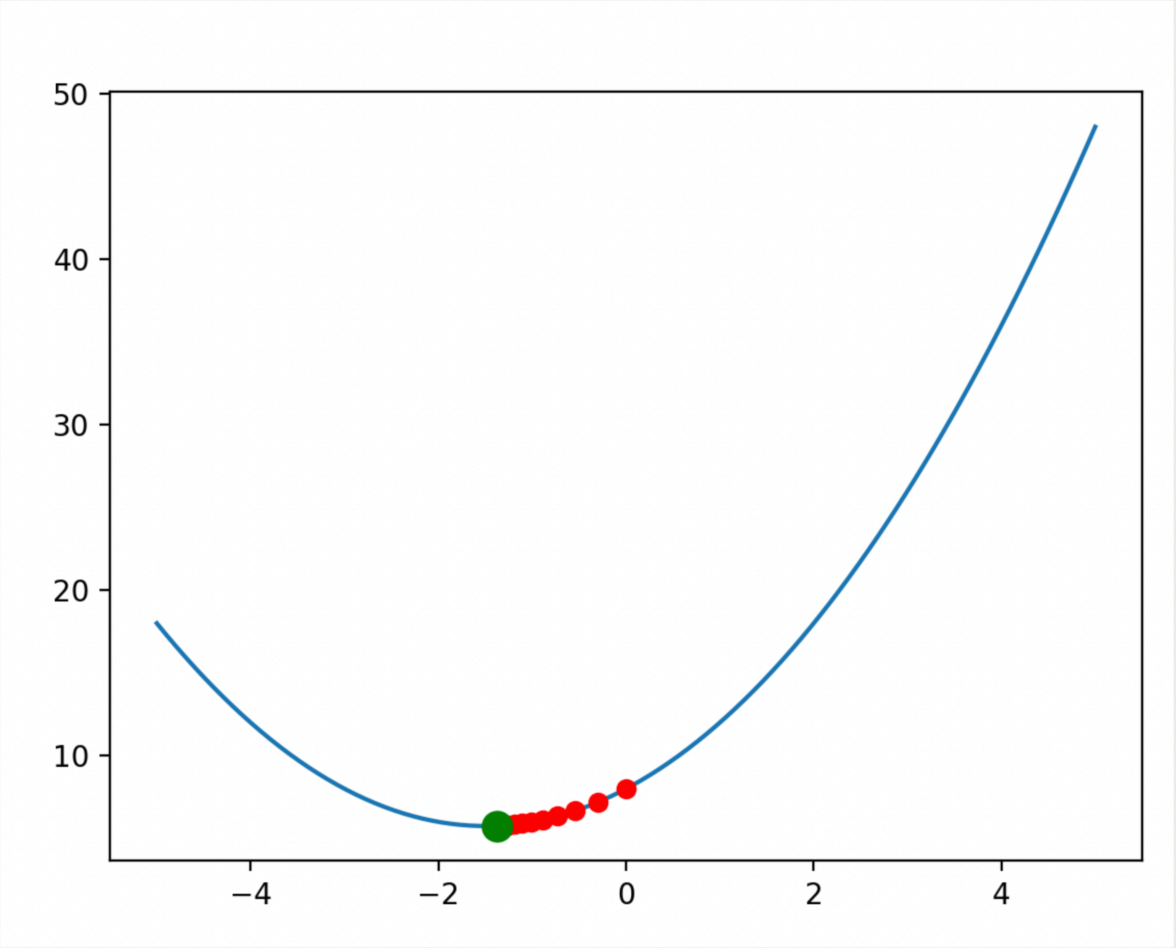
3. Perform gradient descent by iteratively updating `bestx` based on the derivative `f2(x) (the derivative)`.

```
x = bestx - function_derivative(bestx) * lr
```

here, "function_derivative(bestx)" acts as the gradient . The coordinates keep updating until we reach the point where gradient is equal to zero, which is the required minima of the function.

4. Visualise the optimisation process in real-time with the plot.

5. Continue this process for a specified number of iterations (e.g., 10).



Minimum value of f1(x):	5.766613139557457

b) Problem 4 (f5 and f6):

- Function: f5(x) = cos(x)^4 - sin(x)^3 - 4(sin(x)^2) + cos(x) + 1 with its derivative f6(x).
- Approach:

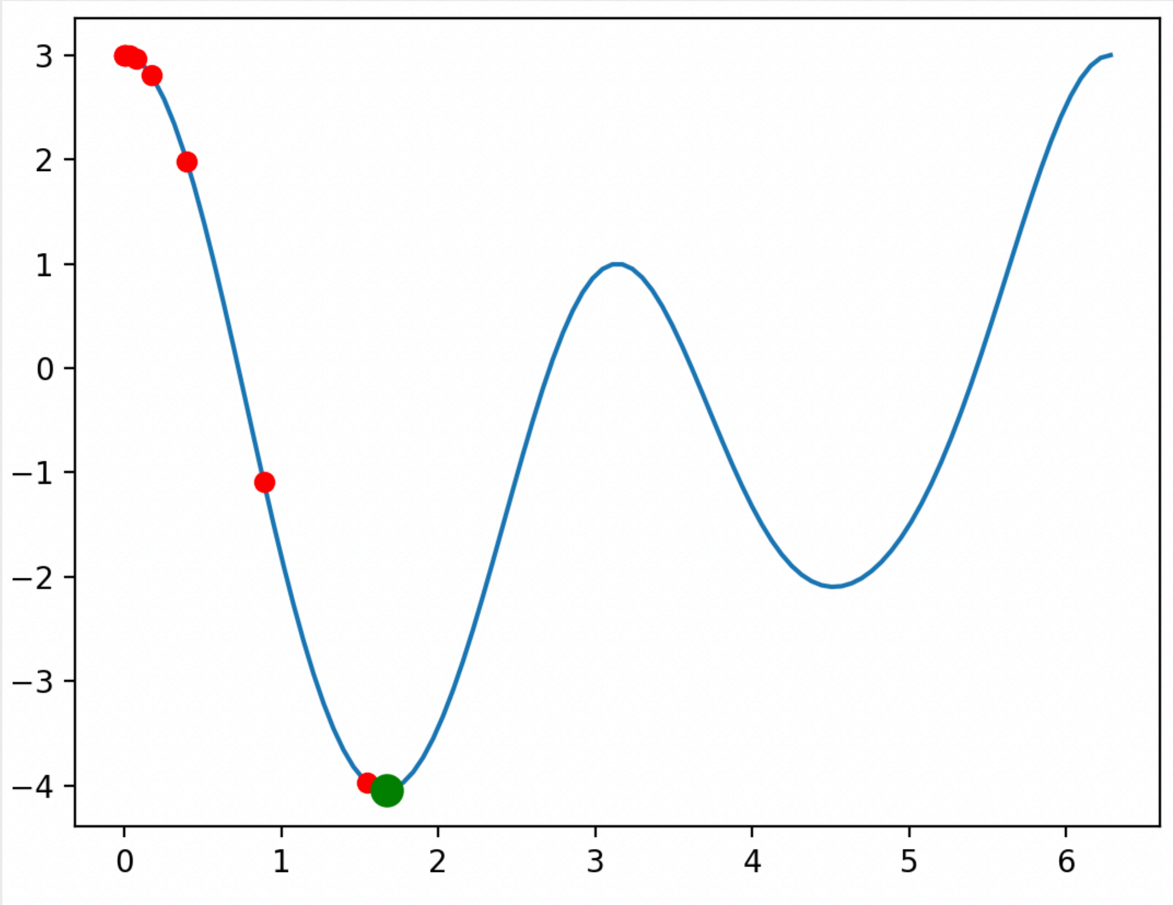
1. Choose an initial value for `bestx`, `bestx = 0.0005`.

2. Initialise a plot to visualise the function and optimisation process.

3. Perform gradient descent by iteratively updating `bestx`as discussed in 1.a) based on the derivative `f6(x)`.

4. Visualise the optimisation process in real-time with the plot.

5. Continue this process for a specified number of iterations (e.g., 10).



Minimum value of f5(x):	**-4.044717280848179**

2. **Using gradient_func_2D:-**

a) Problem 2 (f3, df3_dx, and df3_dy):

- Function: f3(x, y) = x^4 - 16x^3 + 96x^2 - 256x + y^2 - 4y + 262 with its partial derivatives `df3_dx` and `df3_dy`.

- **Approach:**

1. Choose initial values for `bestx` and `besty`, in this case, `(bestx, besty) = (3.52, 0.91)`{these values were guessed intuitively by inferring the 3-d plot provided that it lies in the specified range }

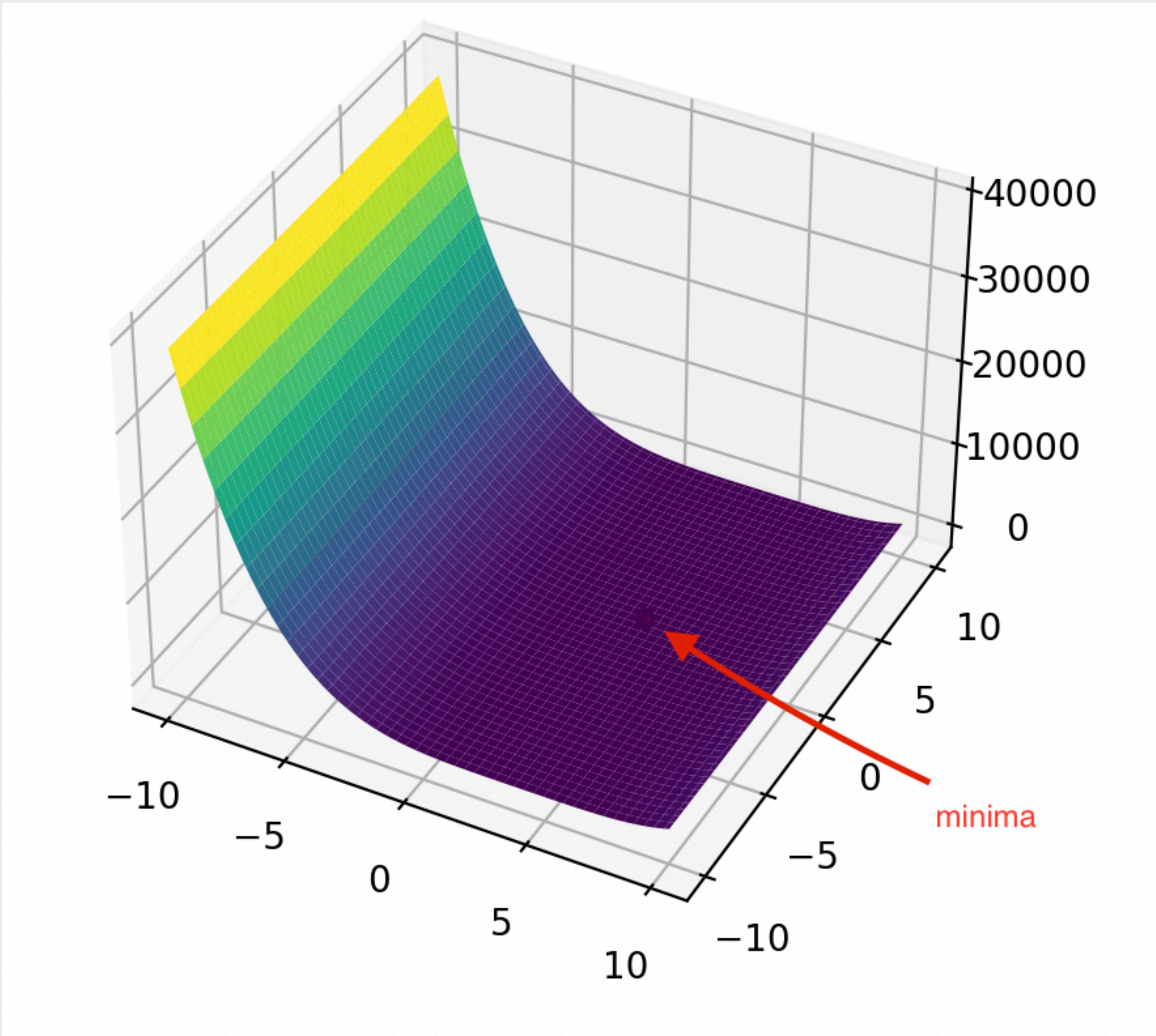
2. Initialise a 3D plot to visualise the function and optimisation process.

3. Perform gradient descent by iteratively updating both `bestx` and `besty` based on their respective derivatives.

4. Visualise the optimisation process in the 3D plot.

5. Continue this process for a specified number of iterations or until

convergence.



Minimum value of f3(x):	2.000001533507202

b) Problem 3 (f4, df4_dx, and df4_dy):

- Function: f4(x, y) = exp(-(x - y)^2) * sin(y) with its partial derivatives `df4_dx` and `df4_dy`.

- Approach:

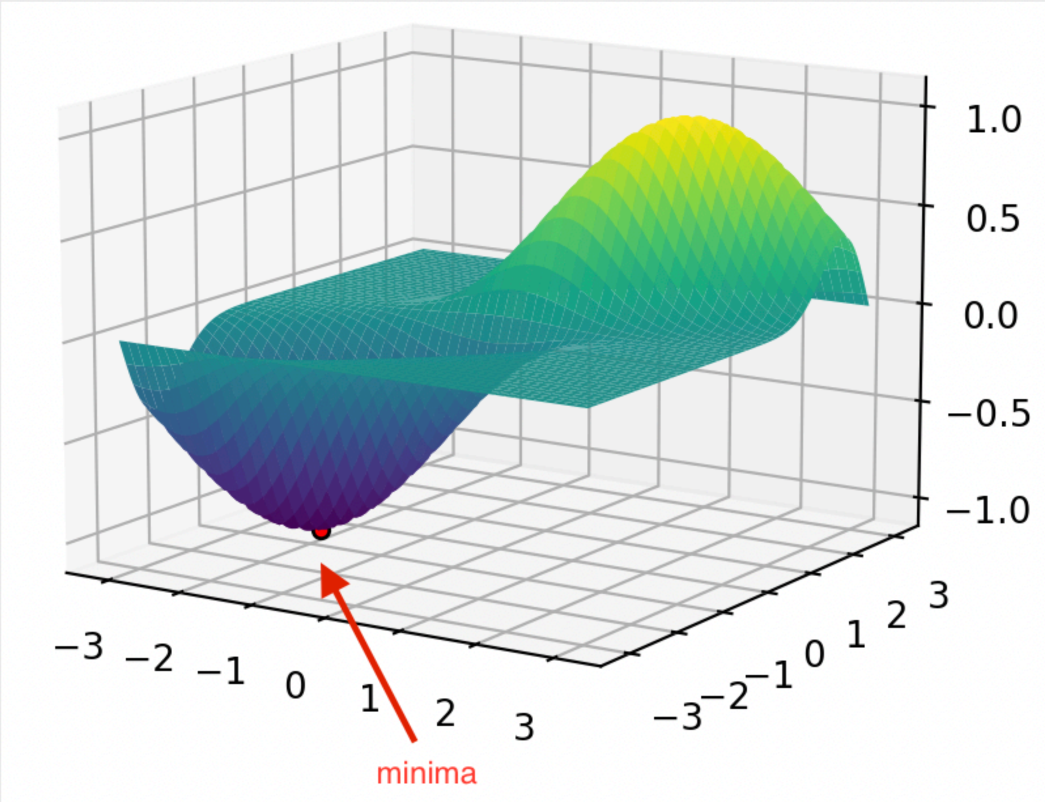
1. Choose initial values for `bestx` and `besty`, in this case, `(bestx, besty) = (-1.5, -1.5)`{the values here are chosen the same way as done in 2.a)}.

2. Initialise a 3D plot to visualise the function and optimisation process.

3. Perform gradient descent by iteratively updating both `bestx` and `besty` based on their respective derivatives.

4. Visualise the optimisation process in the 3D plot.

5. Continue this process for a specified number of iterations or until convergence.



Minimum value of f4(x):	-1.0

In all cases, the optimisation process involves updating the input variable(s) using the gradient and a fixed learning rate. Visualisations help track the progress and illustrate how the algorithm converges to the minimum value. The choice of the initial values can affect the optimisation process, and the number of iterations can be adjusted to achieve the desired level of convergence.