Matlab script(hw4.m):

```
%Combined the code from hw2 and hw3 with alterations
clear
clc
close all

% Function from last time
f = @(x, y) (x.^4 + y.^4) / 4 - (x.^3 + y.^3) / 3 - (x.^2 + y.^2) + 4;
% Search range as described on hw pdf(-2 to 3)
xmin = -2;
xmaxim = 3;
ymin = -2;
ymaxim = 3;

% Initialize the starting points
x_start = (xmaxim - xmin) * rand() + xmin;
y_start = (ymaxim - ymin) * rand() + ymin;
learning_rate = 0.1;

% Maximum number of iterations (changed from 50 to 100 for max number)
iterations = randi([10, 100]);
% Arrays to store the data for optimization
x_data = zeros(iterations, 1);
y_data = zeros(iterations, 1);
f_data = zeros(iterations, 1);

% Gradient Descent Optimization method being used
x_current = x_start;
y_current = y_start;
for k = 1:iterations
% Ensure that x and y stay within the search range or -2 and 3 and not
% past it
x_current = max(min(x_current, xmaxim), xmin);
y_current = max(min(y_current, ymaxim), ymin);
% Calculate the gradient using finite differences
grad_x = (f(x_current + 0.01, y_current) - f(x_current, y_current)) / 0.01;
grad_y = (f(x_current, y_current + 0.01) - f(x_current, y_current)) / 0.01;
% Update x and y values using the negative gradient
x_current = x_current - learning_rate * grad_x;
y_current = y_current - learning_rate * grad_y;
% Store the data
x_data(k) = x_current;
y_data(k) = y_current;
f_data(k) = f(x_current, y_current);
end

% Plot the optimization path on a contour plot similar to the hill diagram
% from hw3
x = linspace(xmin, xmaxim, 100);
y = linspace(ymin, ymaxim, 100);
[X, Y] = meshgrid(x, y);
Z = f(X, Y);
figure;
contour(X, Y, Z, 50);
hold on;
plot(x_data, y_data, 'ro', 'MarkerSize', 5);
xlabel('x');
ylabel('y');
title('2-D Diagram');
% Output the final result like hw2
fprintf('Optimal Solution:\n');
fprintf('x = %.4f\n', x_current);
fprintf('y = %.4f\n', y_current);
fprintf('Minimum Value: %.4f\n', f(x_current, y_current));
```

Output on Matlab