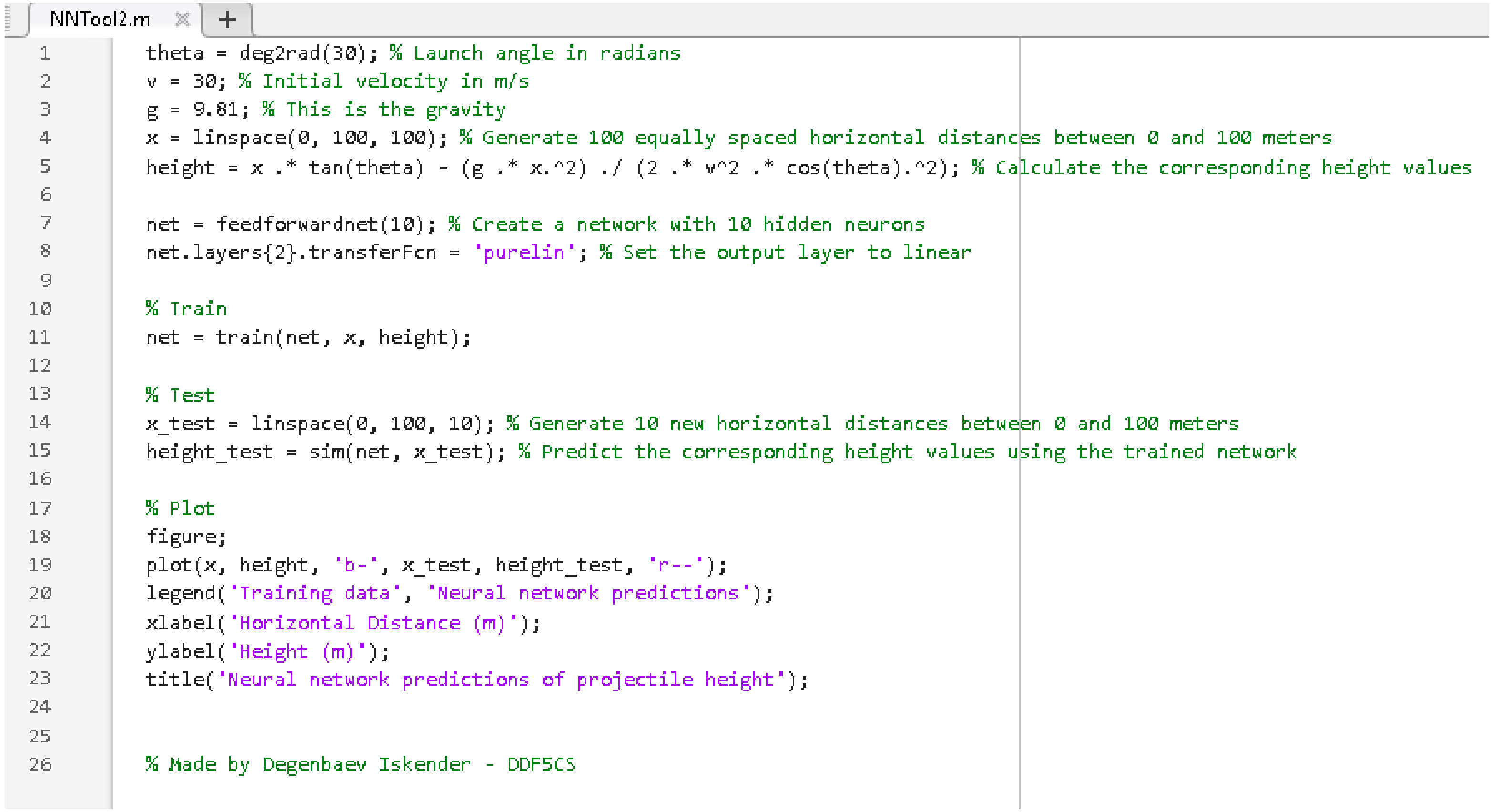
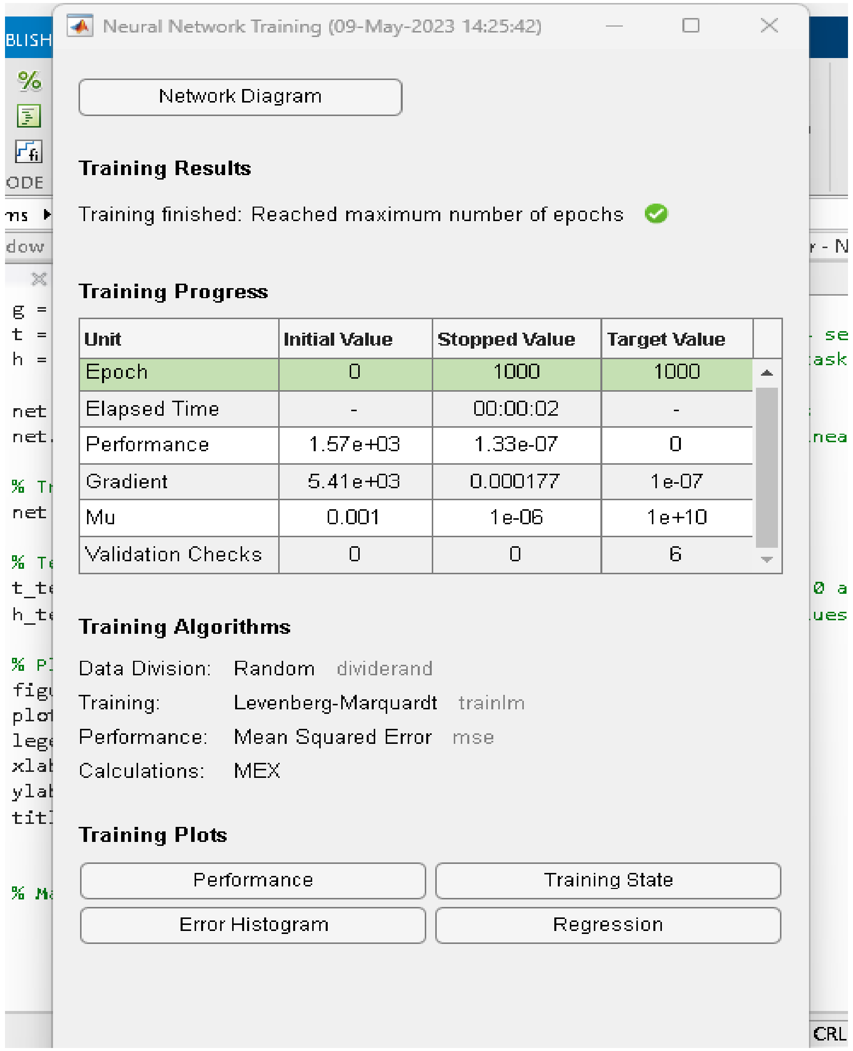
1

Using Newtonian physics, please train a neural network by employing the appropriate equation (h(t) = position in x) for the case of free falling from a drop height of 20 meters.

So, below, you can see a simple script for this problem:



```
theta = deg2rad(30); % Launch angle in radians
v = 30; % Initial velocity in m/s
g = 9.81; % This is the gravity
x = linspace(0, 100, 100); % Generate 100 equally spaced horizontal distances between 0 and 100 meters
height = x .* tan(theta) - (g .* x.^2) ./ (2 .* v^2 .* cos(theta).^2); % Calculate the corresponding height values

net = feedforwardnet(10); % Create a network with 10 hidden neurons
net.layers{2}.transferFcn = 'purelin'; % Set the output layer to linear

% Train
net = train(net, x, height);

% Test
x_test = linspace(0, 100, 10); % Generate 10 new horizontal distances between 0 and 100 meters
height_test = sim(net, x_test); % Predict the corresponding height values using the trained network

% Plot
figure;
plot(x, height, 'b-', x_test, height_test, 'r--');
legend('Training data', 'Neural network predictions');
xlabel('Horizontal Distance (m)');
ylabel('Height (m)');
title('Neural network predictions of projectile height');


% Made by Degenbaev Iskender - DDF5CS
```



Neural Network Training (09-May-2023 14:25:42)

Network Diagram

Training Results

Training finished: Reached maximum number of epochs ✓

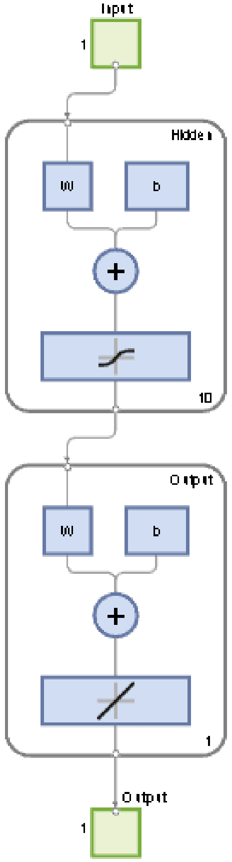
Training Progress

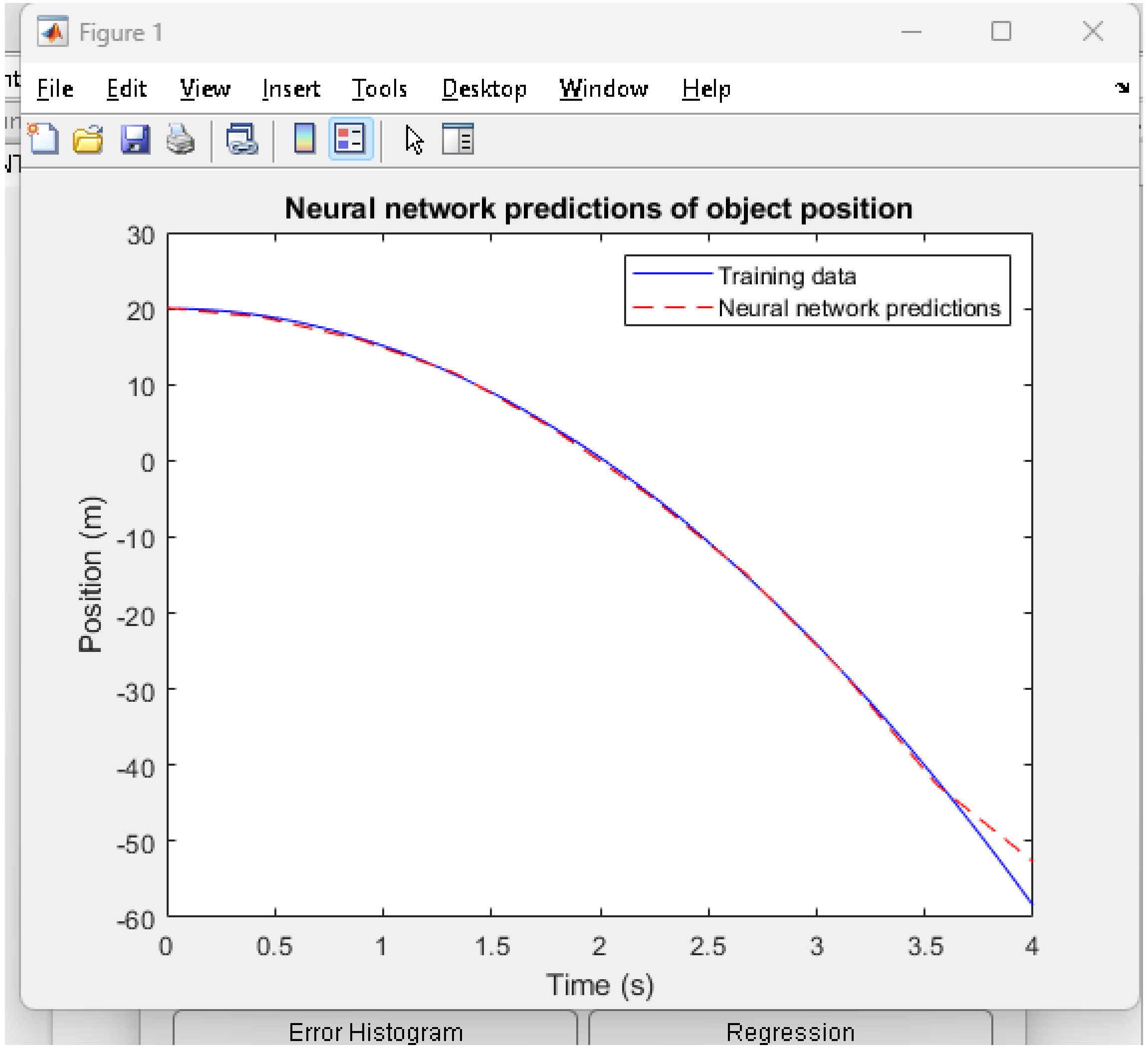
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	1000	1000
Elapsed Time	-	00:00:02	-
Performance	1.57e+03	1.33e-07	0
Gradient	5.41e+03	0.000177	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	0	6

Training Algorithms

Data Division: Random dividerand
Training: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

Training Plots

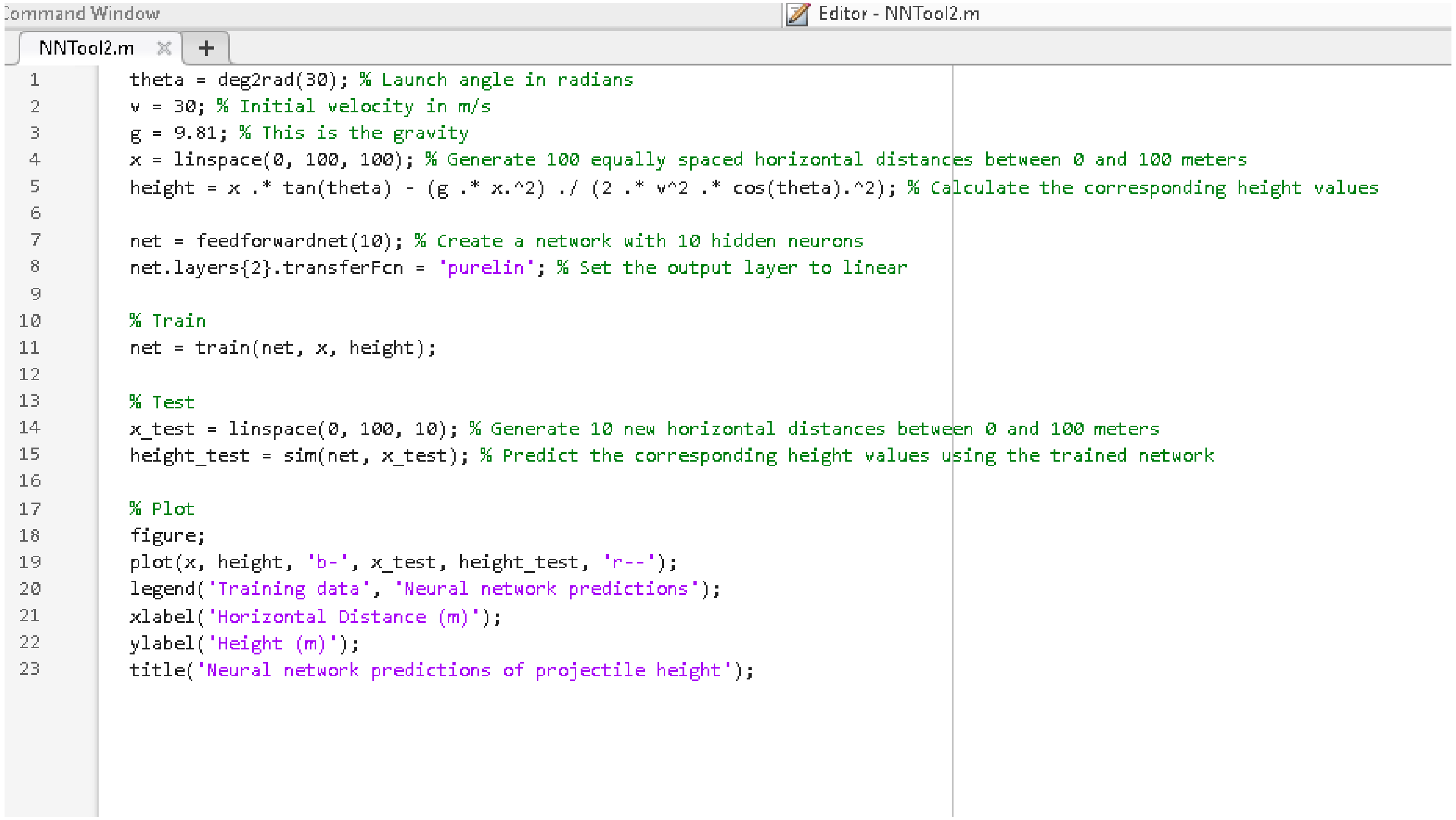




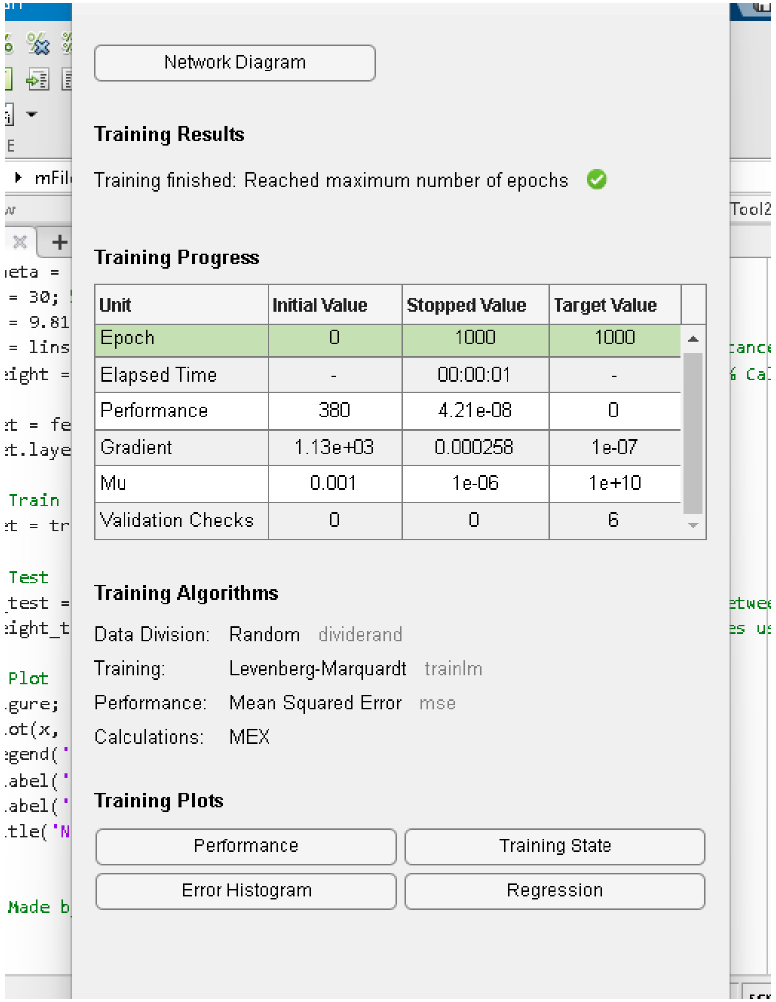
So, here you can see training results, network diagram and plot.

2

Similar to the first problem, please train a neural network by employing the appropriate equation (g(x) = position in height) for a projectile where the object is launched at an elevation angles 30° upwards with an initial velocity of 30 m/s.



```
theta = deg2rad(30); % Launch angle in radians
v = 30; % Initial velocity in m/s
g = 9.81; % This is the gravity
x = linspace(0, 100, 100); % Generate 100 equally spaced horizontal distances between 0 and 100 meters
height = x .* tan(theta) - (g .* x.^2) ./ (2 .* v^2 .* cos(theta).^2); % Calculate the corresponding height values

net = feedforwardnet(10); % Create a network with 10 hidden neurons
net.layers{2}.transferFcn = 'purelin'; % Set the output layer to linear

% Train
net = train(net, x, height);

% Test
x_test = linspace(0, 100, 10); % Generate 10 new horizontal distances between 0 and 100 meters
height_test = sim(net, x_test); % Predict the corresponding height values using the trained network

% Plot
figure;
plot(x, height, 'b-', x_test, height_test, 'r--');
legend('Training data', 'Neural network predictions');
xlabel('Horizontal Distance (m)');
ylabel('Height (m)');
title('Neural network predictions of projectile height');
```



Training Results

Training finished: Reached maximum number of epochs ✓

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	1000	1000
Elapsed Time	-	00:00:01	-
Performance	380	4.21e-08	0
Gradient	1.13e+03	0.000258	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	0	6

Training Algorithms

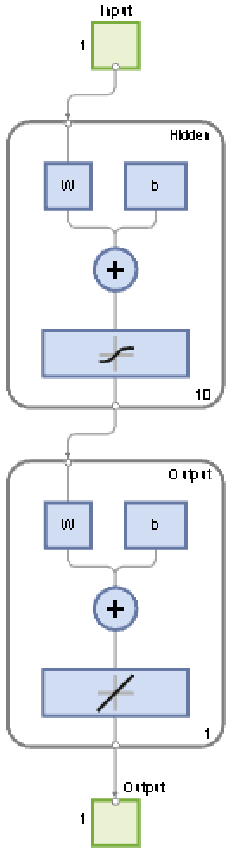
Data Division: Random dividerand

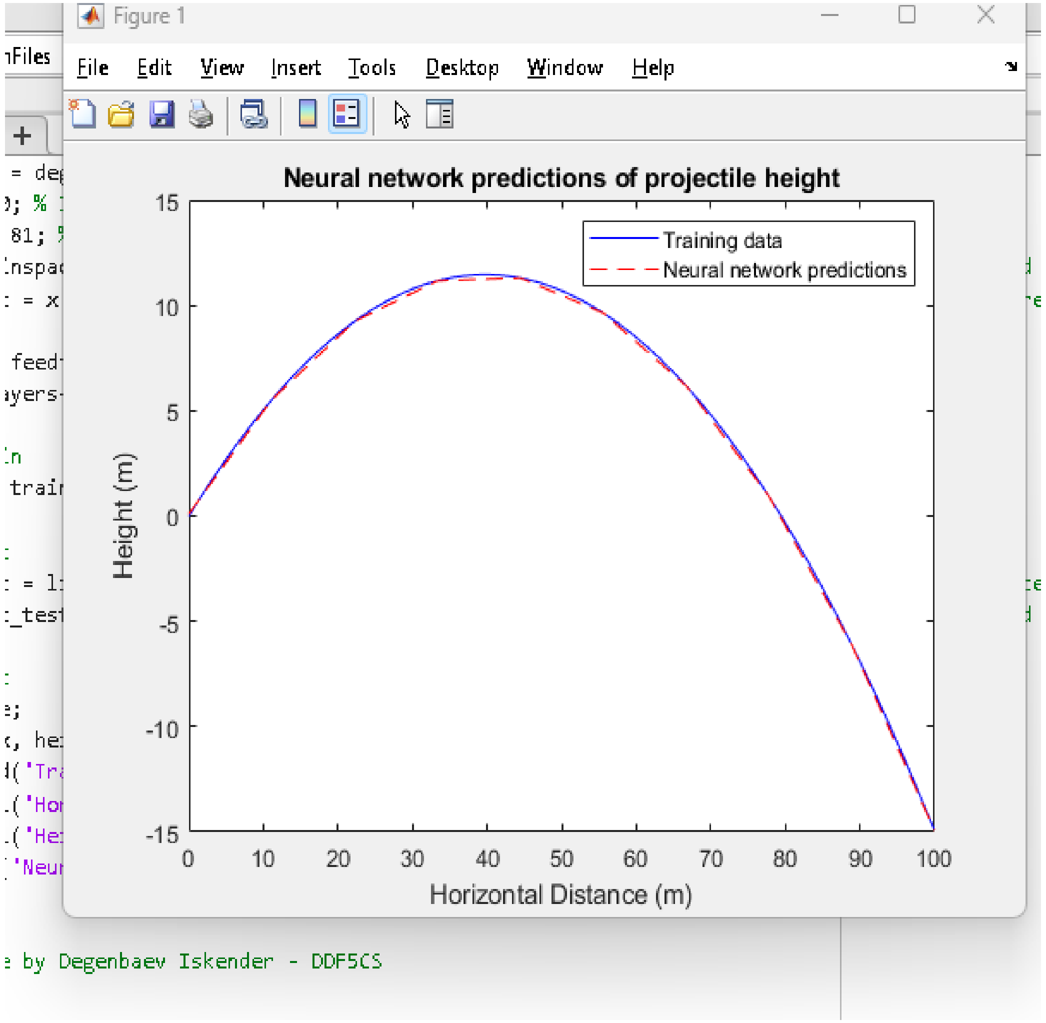
Training: Levenberg-Marquardt trainlm

Performance: Mean Squared Error mse

Calculations: MEX

Training Plots





e by Degenbaev Iskender - DDF5CS

So, here you can see training results, network diagram and plot.