

function J = computeCost(X, y, theta)

%COMPUTECOST Compute cost for linear regression

% J = COMPUTECOST(X, y, theta) computes the cost of using theta as the

% parameter for linear regression to fit the data points in X and y

% Initialize some useful values

m = length(y); % number of training examples

% You need to return the following variables correctly

J = 0;

% ====================== YOUR CODE HERE ======================

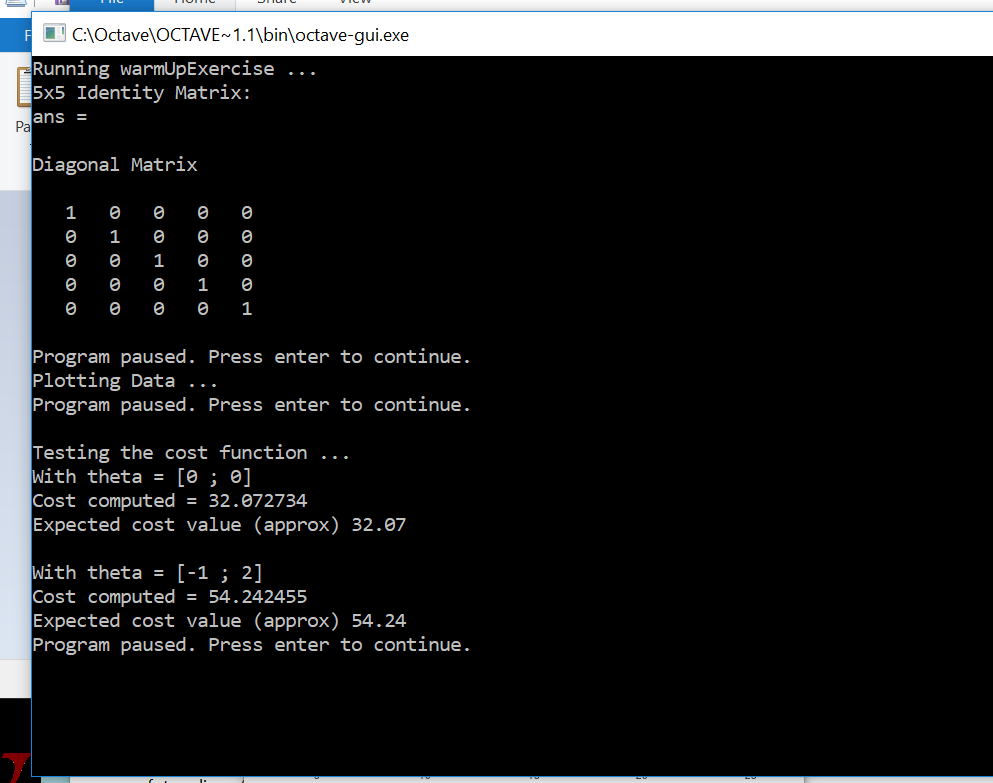
% Instructions: Compute the cost of a particular choice of theta

% You should set J to the cost.

J=(1/(2\*m))\*sum((((X\*theta)-y).^2));

% =========================================================================

end



function [theta, J\_history] = gradientDescent(X, y, theta, alpha, num\_iters)

%GRADIENTDESCENT Performs gradient descent to learn theta

% theta = GRADIENTDESCENT(X, y, theta, alpha, num\_iters) updates theta by

% taking num\_iters gradient steps with learning rate alpha

% Initialize some useful values

m = length(y); % number of training examples

J\_history = zeros(num\_iters, 1);

for iter = 1:num\_iters

theta=theta-(alpha/m)\*(X')\*(((X\*theta)-y));

% ====================== YOUR CODE HERE ======================

% Instructions: Perform a single gradient step on the parameter vector

% theta.

%

% Hint: While debugging, it can be useful to print out the values

% of the cost function (computeCost) and gradient here.

%

% ============================================================

% Save the cost J in every iteration

J\_history(iter) = computeCost(X, y, theta);

end

end

