**EX1.m – Basic Gradient Descent**

data = load('ex1data1.txt');

X = data(:, 1); y = data(:, 2);

X = [ones(m, 1), data(:,1)]; % Add a column of ones to x

theta = zeros(2, 1); % initialize fitting parameters

% Some gradient descent settings

iterations = 1500;

alpha = 0.01;

theta = gradientDescent(X, y, theta, alpha, iterations);

fprintf('Theta found by gradient descent: ');

fprintf('%f %f \n', theta(1), theta(2));

fprintf('Visualizing J(theta\_0, theta\_1) ...\n')

% Grid over which we will calculate J; Draw Surface & Contour plots

theta0\_vals = linspace(-10, 10, 100);

theta1\_vals = linspace(-1, 4, 100);

J\_vals = zeros(length(theta0\_vals), length(theta1\_vals));

for i = 1:length(theta0\_vals)

for j = 1:length(theta1\_vals)

t = [theta0\_vals(i); theta1\_vals(j)];

J\_vals(i,j) = computeCost(X, y, t);

end

end

% Because of the way meshgrids work in the surf command, we need to

% transpose J\_vals before calling surf, or else the axes will be flipped

J\_vals = J\_vals';

figure;

surf(theta0\_vals, theta1\_vals, J\_vals)

xlabel('\theta\_0'); ylabel('\theta\_1');

figure;

% Plot J\_vals as 15 contours spaced logarithmically between 0.01 and 100

contour(theta0\_vals, theta1\_vals, J\_vals, logspace(-2, 3, 20))

xlabel('\theta\_0'); ylabel('\theta\_1');

hold on;

plot(theta(1), theta(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);

**COMPUTE COST**

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

m = length(y);

J = 0;

S = y-X\*theta;

V = sum(S.^2);

J = V/(2\*m);

end

**GRADIENT DESCENT**

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

%GRADIENTDESCENT Performs gradient descent to learn theta

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

% taking num\_iters gradient steps with learning rate alpha

m = length(y);

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

for iter = 1:num\_iters

temp = theta;

theta(1) = temp(1) - (alpha/m)\*(sum(X\*temp - y));

theta(2) = temp(2) - (alpha/m)\*(sum((X\*temp - y).\*(X(:,2))));

% Save the cost J in every iteration

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

end

end