```
load datasetPS4.mat;

% This is the part of problem 1.
incides = 1:31;
train_X = x(incides(1:19),1);
crossvalidation_X = x(incides(20:25),1);
test_X = x(incides(26:31),1);
train_Y = y(incides(1:19),1);
crossvalidation_Y = y(incides(20:25),1);
test_Y = y(incides(26:31),1);
cost = zeros([10,1]);
for i = 1:10 % perform 10 validation at 10 different degrees.
    train_poly = polynomial_generater(train_X,i);
    weight = linear_regression(train_poly, train_Y);
    validation_poly = polynomial_generater(crossvalidation_X,i);
    validation_y = validation_poly * weight;
    s = sum((validation_y-crossvalidation_Y).^2,'all');
    cost(i,1) = s/(2*length(validation_y));
end
plot(cost)
[C, i] = min(cost);
train_poly = polynomial_generater(train_X,i);
weight = linear_regression(train_poly, train_Y);
test_poly = polynomial_generater(test_X, i);
test_y_prediction = test_poly * weight;
disp(sum((test_y_prediction-test_Y).^2,'all')/ ...
    (2*length(test_y_prediction)));
disp("By observing the graph generated, we can get the" + ...
    " result that, at the sixth degree, we can minimize the cost.");


% This is the part of problem 2.
% p = polyfit(x,y,1);
% disp(p)
% grid on;
% x_min = min(x);
% y_min = min(y);
% x_max = max(x);
% y_max = max(y);
% d_min = polyval(p,x_min);
% d_max = polyval(p,x_max);
% caption = sprintf('y = %f * x + %f', p(1), p(2));
% text(xt, yt, caption, 'FontSize', 16, 'Color', 'r', 'FontWeight', 'bold');
% figure
%
%
% hold on
% scatter(x,y)
% plot([x_min x_max],[d_min d_max],'k--')
% Below are functions we are going to use in the main part.
%
% Use linsolve function to proceed linear regression, and to get weight of
```

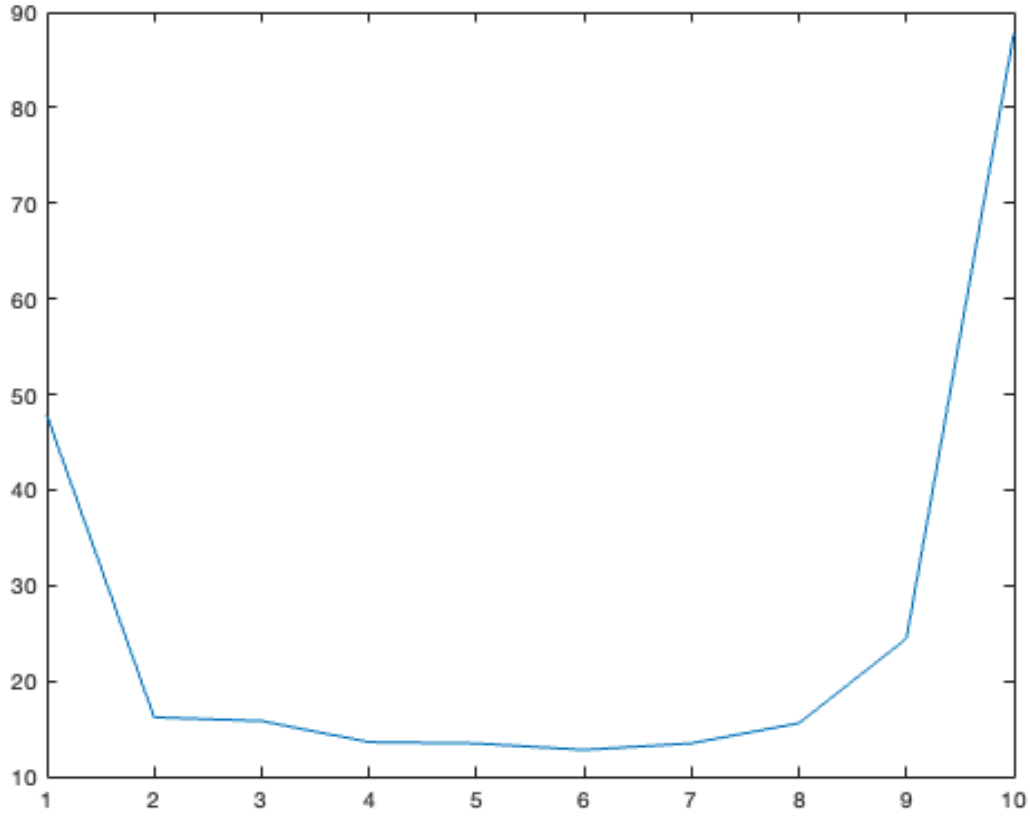
```
% it.
function weight = linear_regression(X,y)
    weight = linsolve(transpose(X)*X, transpose(X)*y);
end

% function weight = ols_regression(X,y,lambda)
%     weight = linsolve(transpose(X)*X, transpose(X)*y);
% end


% This is the function to proceed the given data to a polynpmial
% form. j is the number of parts of the polynomial function, and i is the
% length.
function polynomial_terms = polynomial_generater(X, k)
    len = length(X);
    polynomial_terms = zeros([len,k]);
    for i =  1:len
        for j = 1:k
            polynomial_terms(i,j) = X(i,1)^j;
        end
    end
end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.932315e-18.
   11.6305
```

By observing the graph generated, we can get the result that, at the sixth degree, we can minimize the cost.



Published with MATLAB® R2022a