```
% ********************************************
% Suppose f is a function of two design variables x1 and x2 as follows:
%             f = 3*x1^3+10*x2^2-4*x1*x2-6*x1
%
% Generate a response surface with all of the linear and quadratic terms
% using the least squares method. (number of sample points = 9)
% ********************************************

clc
clear

% We want to approximate evaluations of our problem via the form:
% f_hat = d1 + d[2:n]*z
% where d is a coefficient vector and z is in this case, the linear and
% quadratic terms of x
%
% We solve for the d coefficients though the linear system of equations:
% A*d = b
% We Initialize A and b as empty vectors/matrices of proper size
% For a two variable linear and quadratic term approximation, we will have
% 6 terms in the b vector, and a 6x6 A matrix (yielding a vector of size 6
% for d)
A=zeros(6);
b=zeros(6,1);
% k: number of sample points
k=9;
% sample points X=(x1,x2)
x1=[-1.5,-1.5,-1.5,1.25,1.25,1.25,4,4,4];
x2=[-3,0,3,-3,0,3,-3,0,3];
X=[x1;x2];

% Define the linear and quadratic terms based on sampled points (x1, x2)
% z1 = x1, z2 = x2, z3 = x1^2, z4 = x2^2, z5 = x1*x2
z1=x1;
z2=x2;
z3=x1.^2;
z4=x2.^2;
z5=x1.*x2;
z=[z1;z2;z3;z4;z5]';
% Show all Linear and Quadratic Terms
z

%Calulate A
A(1,1)=k;
A(1,2:6)=sum(z);
A(2:6,1)=sum(z)';
for i=2:6
    for j=2:6
        A(i,j)=sum(z(:,i-1).*z(:,j-1));
    end
end
%Show A matrix
```

```
A

% Calculate b
%f = @(x) 3*x(1)^3+10*x(2)^2-4*x(1)*x(2)-6*x(1);

f = @(x) 3*x(1)^2-10*x(2)^2+2*x(1)*x(2)-6*x(1)+8; % Changed to solve problem
 in lab

for i=1:k
    F(i)=f(X(:,i));
end
b(1)=sum(F);
for i=1:5
    b(i+1)=sum(F'.*z(:,i));
end
% Show b coefficients
b
% solve linear equation A*d=b
d = linsolve(A,b);
% Show d coefficients
d
%Response Surface Model is an approximation for f that uses d as the
%coefficients for the linear and quadratic order terms (z vector) above.
% You could use a cubic or higher order RSM could use higher order terms as
 well
% NOTE: d1 is an intercept, not a coefficients

% Compare approximation for sample points with true values
approx = d(1) +z* d(2:6)
table(F',approx)
% With a limited number of points (such as in this case), the approximation
% should very closely or exactly fit the true values. However, in general
% we are using a least squares approximation, so this is not generally:

%%%% ANSWER %%%%
% As seen in the table the approximated values are almost spot in if not
% exact to the actual solution as stated in the comment above.


% Try varying x1 and x2 above, and then comparing the true and approximate
% values again

% Try with test points
x1_new = 8;
x2_new = 8;
z_test = [x1_new x2_new x1_new.^2 x2_new.^2 x1_new.*x2_new];
approx_check = d(1) +z_test*d(2:6);
% Evaluate the true function at this point. Is it the same

% the true value and the approximation are the same.
x = [x1_new,x2_new];
display('Test Point Value then Approx. Value:')
f(x)
approx_check
```

```
z =

   -1.5000   -3.0000    2.2500    9.0000    4.5000
   -1.5000         0    2.2500         0         0
   -1.5000    3.0000    2.2500    9.0000   -4.5000
    1.2500   -3.0000    1.5625    9.0000   -3.7500
    1.2500         0    1.5625         0         0
    1.2500    3.0000    1.5625    9.0000    3.7500
    4.0000   -3.0000   16.0000    9.0000  -12.0000
    4.0000         0   16.0000         0         0
    4.0000    3.0000   16.0000    9.0000   12.0000


A =

    9.0000   11.2500         0   59.4375   54.0000         0
   11.2500   59.4375         0  187.7344   67.5000         0
         0         0   54.0000         0         0   67.5000
   59.4375  187.7344         0  790.5117  356.6250         0
   54.0000   67.5000         0  356.6250  486.0000         0
         0         0   67.5000         0         0  356.6250


b =

   1.0e+03 *

   -0.3572
   -0.3784
    0.1350
   -1.8456
   -3.7631
    0.7133


d =

    8.0000
   -6.0000
         0
    3.0000
  -10.0000
    2.0000


approx =

  -57.2500
   23.7500
  -75.2500
  -92.3125
    5.1875
  -77.3125
```

```
  -82.0000
   32.0000
  -34.0000


ans =

  9×2 table

     Var1      approx
    ______    ______

    -57.25    -57.25
     23.75     23.75
    -75.25    -75.25
   -92.312   -92.312
    5.1875    5.1875
   -77.312   -77.312
       -82       -82
        32        32
       -34       -34

Test Point Value then Approx. Value:

ans =

  -360


approx_check =

 -360.0000
```

Published with MATLAB® R2021b