```
%This Worksheet simulates the polynomial regression model. It also helps
%you to find the optimum order of polynomial to use.
clc
%The following are the inputs to the model
%Enter values for X:
X=[80,40,-40,-120,-200,-280,-340];
%Enter values for Y:
Y=[6.47e-6,6.24e-6,5.72e-6,5.09e-6,4.3e-6,3.33e-6,2.45e-6];
%Enter the order of polynomial regression model:
order_poly=1;
%Enter the lowest order (low_order =>1) of polynomial to check for optimun order:
low_order=1;
%Enter the highest order (hi_order <=n-2) of polynomial to check for optimun order:
hi_order=5;
disp(sprintf('Polynomial Regression Model'))
disp(sprintf('©2007 Fabian Farelo, Autar Kaw'))
disp(sprintf('University of South Florida'))
disp(sprintf('United States of America'))
disp(sprintf('kaw@eng.usf.edu'))

disp(sprintf('\n\nNOTE: This worksheet demonstrates the use of Matlab to illustrate'))
disp(sprintf('the procedure to regress a given data set to a nonlinear model'))
%---------------------------------------------------------------------
disp(sprintf('\n\n***************************** Introduction ******************************'))
disp(sprintf('Given n data points (x1, y1), (x2, y2). . , (xn, yn) least squares'))
disp(sprintf('method can be used to regress the data to a mth order polynomial.'))
disp(sprintf('       y=a0+a1*x+a2*x^2+.....+am*x^m, m<n                    (1)\n'))
disp(sprintf('The residual at each data point is given by'))
disp(sprintf('       Ei=yi-a0-ai*xi........-am*xi^m                        (2)\n'))
disp(sprintf('The sum of the square of the residuals is given by'))
disp(sprintf('       Sr=sum(Ei^2, i=1:n)                                  (3)\n'))
disp(sprintf('To find the constants of the polynomial regression model, we put the'))
disp(sprintf('derivatives with respect to ai to zero, that is,'))
disp(sprintf('d(Sr)/d(a0) =sum[2*(yi-a0-ai*xi........-am*xi^m, i=1:n)(-1)]=0    (4.a)\n'))
disp(sprintf('d(Sr)/d(a1) =sum[2*(yi-a0-ai*xi........-am*xi^m, i=1:n)(-xi)]=0   (4.b)'))
disp(sprintf(' .     .       .      .       .       .      .      .'))
disp(sprintf(' .     .       .      .       .       .      .      .'))
disp(sprintf('d(Sr)/d(am) =sum[2*(yi-a0-ai*xi........-am*xi^m, i=1:n)(-xi^m)]=0 (4.m)\n'))
disp(sprintf('Setting those equations in matrix form gives'))
disp(sprintf('|   n       sum(xi) .....  sum(xi^m)  | |a0| = |sum(yi)     |'))
disp(sprintf('| sum(xi)   sum(xi^2).....  sum(xi^m+1)| |a1| = |sum(xi*yi)  |'))
disp(sprintf('|.....| |..| = |.....  |'))
disp(sprintf('| sum(xi^m)  sum(xi^m+1)....  sum(xi^2m) | |am| = |sum(xi^m*yi)|'))
disp(sprintf('\nThe above simultaneous linear equations are solved\nfor the (m+1) constants a0, a1,. . .,↙
am'))
disp(sprintf('\n\n**************** Polynomial model simulation  ************************'))
disp(sprintf('NOTE:This is the input data that can be modified by the user at the\ntop of the M-File. The↙
order of the polynomial can also be\nselected by the user.'))
disp(sprintf('Once the values are entered, Matlab will generate a polynomial regression model \nfor the↙
given data set.'))

X=X'
Y=Y'
lx=length(X);
ly=length(Y);
if (lx)<(ly)
```

```
   disp(sprintf('The number of elements for the input arrays X and Y must be the same'))
else
    if (lx)>(ly)
    disp(sprintf('The number of elements for the input arrays X and Y must be the same'))
    else
    disp(sprintf('Using the matrix form above, we have the coefficient matrix "M"'))
%Creating the matrix size according to the order of the polynomial model
M=zeros(order_poly+1);
%Finding each value of the matrix
M(1,1)=length(X);
for i=2:order_poly+1
   for j=1:length(X)
   M(1,i)=M(1,i)+X(j)^(i-1);
end
end
for i=1:order_poly+1
      for k=2:order_poly+1
      for j=1:length(X)

         M(k,i)=M(k,i)+X(j)^(i+k-2);

      end
      end
end
format short g
M
disp(sprintf('Now, the right hand side vector "C"'))
%Creating the vector size according to the order of the polynomial model
C=zeros(order_poly+1,1);
%Finding each value of the array
for i=1:length(X)
   C(1,1)=C(1,1)+Y(i);
end
for i=2:order_poly+1
   for j=1:length(X)
      C(i,1)=C(i,1)+(X(j)^(i-1))*Y(j);
   end
end
format short g
C
%Merging the arrays M and C to use Matlab to reduce it
E=horzcat(M,C);
a=rref(E);
%Selecting the last columm with the coefficients values
a=a(:,order_poly+2);
format short g
disp(sprintf('Solving the system of simultaneous linear equations M*a=C,\nfor (a) we get'))
a
disp(sprintf('This array contains the coefficients of the model as'))
disp(sprintf('a=[a0, a1,.....,am]'))
disp(sprintf('The plot shows the data points as well as the regression model.'))

%------- Plot --------

xp=(min(X):0.001:max(X));
yp=zeros(1,length(xp));
```

```
for i=1:length(xp)
   yp(i)=a(1);
   for j=1:order_poly
   yp(i)=yp(i)+a(j+1)*xp(i)^(j);
   end
end
plot(xp,yp)
title(['Polynomial Regression of order ',num2str(order_poly)])
xlabel('x')
ylabel('y = a_0+a_1*x+a_2*x^2+.....+a_m*x^m')
hold on
plot(X,Y,'bo','MarkerFaceColor','b')
hold off
end
end
 %-----------------------------
disp(sprintf('\n\n******************* Optimum order************************'))
disp(sprintf('In this section the user can determine the optimum order of the polynomial model by '))
disp(sprintf('plotting the variance defined as  \n       Sr/(n-(m+1)) '))
disp(sprintf('as a function of m, where n is the number of data points, Sr is the sum of the square of '))
disp(sprintf('residuals and m is the order of the polynomial. The optimum order is considered as to be'))
disp(sprintf('the one where the value of the variance Sr/[n-(m+1)] is minimum or where its value is'))
disp(sprintf('significantly decreasing.'))
disp(sprintf('\nIn the following procedure, an mth order polynomial regression model is calculated for↙
each'))
disp(sprintf('order specified in the Low_order to High_order range. Matlab then returns the variance of↙
each'))
disp(sprintf('model.'))

%Doing the polynomial regression and storing the Sr values
Sr=zeros(low_order,hi_order);
for order_poly=low_order:hi_order
M=zeros(order_poly+1);
M(1,1)=length(X);
% n=length(X)
for i=2:order_poly+1
   for j=1:length(X)
   M(1,i)=M(1,i)+X(j)^(i-1);
end
end
for i=1:order_poly+1
      for k=2:order_poly+1
      for j=1:length(X)
         M(k,i)=M(k,i)+X(j)^(i+k-2);
      end
      end
end
C=zeros(order_poly+1,1);
for i=1:length(X)
   C(1,1)=C(1,1)+Y(i);
end
for i=2:order_poly+1
   for j=1:length(X)
      C(i,1)=C(i,1)+(X(j)^(i-1))*Y(j);
   end
end
```

```
E=horzcat(M,C);
a=rref(E);
a=a(:,order_poly+2);


for i=1:length(X)
   summ=0;
      for j=1:order_poly
      summ=summ+a(j+1)*X(i)^(j);
      end
   Sr(order_poly)=Sr(order_poly)+(Y(i)-(a(1)+summ))^2;
end
end
Sr2=zeros(low_order,hi_order);
for i=low_order:hi_order
Sr2(i)=Sr(i)/(length(X)-(i+1));
end

%_____________________________________________________________
%Ploting the results
pl=(1:5);
figure
plot(pl,Sr2,'LineWidth',2)
axis([0 6 0 max(Sr2)])
title('Optimum Order of Polynomial')
xlabel('Order of Polynomial, m')
ylabel('Sr\div[n-(m+1)]')
for i=low_order:hi_order
   if Sr2(i)==min(Sr2)
      optimum_ord=i;
   end
end

disp(sprintf('The optimum order, as seen in the plot, is then:'))
optimum_ord
```