I. Background. Up to an $(n-1)^{th}$-degree polynomial can be fitted to *n* data points. A polynomial of degree $n-1$ will produce a "perfect" fit. For example consider a set of four data points $\{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$. A cubic will satisfy the equations:

$$\begin{cases} y_1 = a_3x_1^3 + a_2x_1^2 + a_1x_1^1 + a_0 \\ y_2 = a_3x_2^3 + a_2x_2^2 + a_1x_2^1 + a_0 \\ y_3 = a_3x_3^3 + a_2x_3^2 + a_1x_3^1 + a_0 \\ y_4 = a_3x_4^3 + a_2x_4^2 + a_1x_4^1 + a_0 \end{cases},$$

where a_i are undetermined constants. This equation can be written in matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ x_3^3 & x_3^2 & x_3 & 1 \\ x_4^3 & x_4^2 & x_4 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix},$$

or in shorthand form: $Y = XA$. Using matrix operations we find the solution for A by left multiplying both sides of the equation by the inverse of the X, i.e.:

$$X^{-1}Y = X^{-1}XA = IA \Rightarrow A = X^{-1}Y,$$

where A contains the coefficients of a cubic that fits the four data points.

II. For a polynomial of degree $< n-1$ the fit will least squares approximation. Suppose we wish to fit the same four data points to a quadratic, i.e.

$$\begin{cases} y_1 = a_2x_1^2 + a_1x_1^1 + a_0 \\ y_2 = a_2x_2^2 + a_1x_2^1 + a_0 \\ y_3 = a_2x_3^2 + a_1x_3^1 + a_0 \\ y_4 = a_2x_4^2 + a_1x_4^1 + a_0 \end{cases} \Rightarrow\Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} \Rightarrow\Rightarrow Y = XA$$

In order to get a least squares fit for A we first left multiply both sides by X^T:

$$X^TY = X^TXA$$

We than multiply both sides by the inverse of $X^T X$, i.e.

$$\left(X^T X\right)^{-1} X^T Y = \left(X^T X\right)^{-1}\left(X^T X\right) A = IA \Rightarrow A = \left(X^T X\right)^{-1} X^T Y$$

We now have solved for the coefficients in A such that the resulting parabola will be a least squares best fit for the data points.

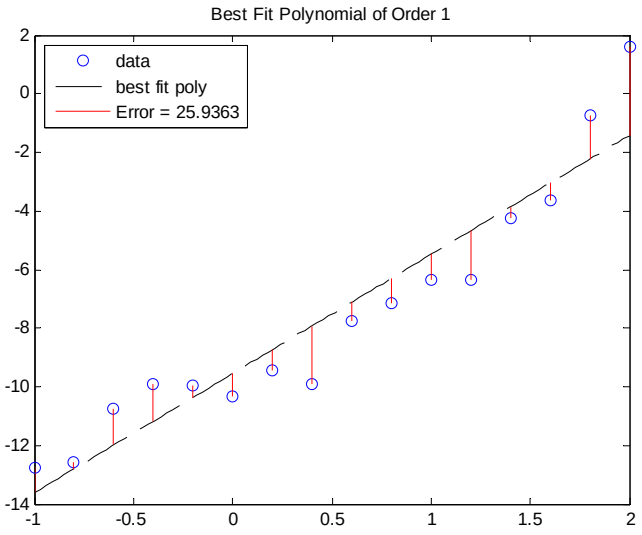
Note: 1. You studied the theory of why this minimizes the error between the curve and the date in Intro to Matrix Algebra.

Note: 2. $A = (X^T X)^{-1} X^T Y$ is not the best way to do this numerically. Methods have been devised to minimize the steps and round-off error. Any linear algebraist worth his salt would hang you for using this method (i.e. my lin alg professor at NPS)

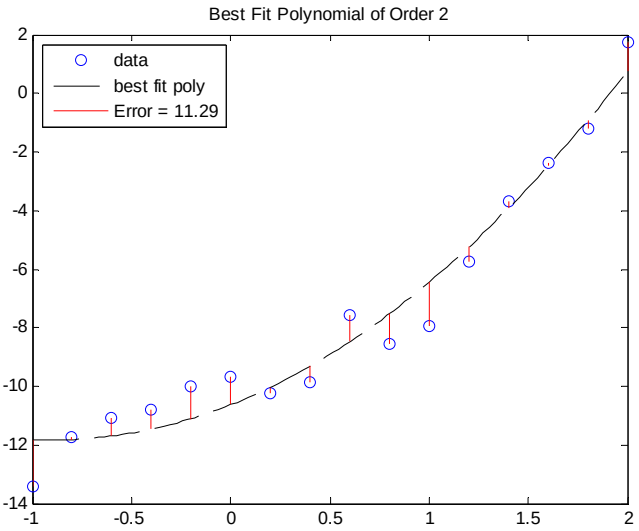
Sources. For more information see:
http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html

III. Example [**go3**] : Created data for: $y = x^3 - 3x^2 + 2x - 1$ with random noise added to the y values.

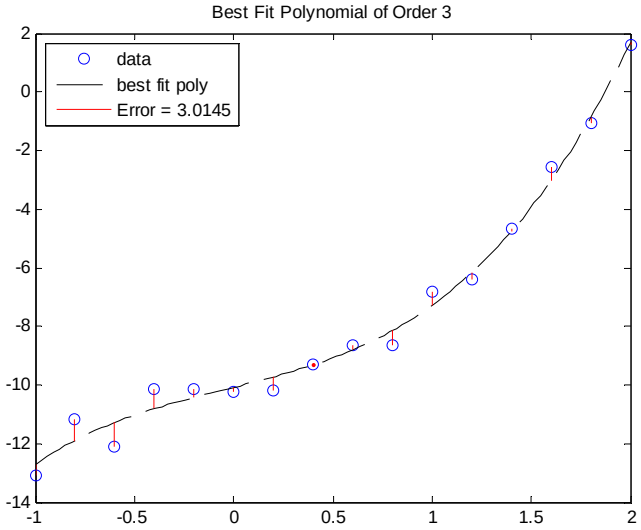
A. Linear Fit:



B. 2nd Degree Polynomial Fit:



C. Third Degree Polynomial Fit:



D. 15^{th} Degree Polynomial Fit (note: just and interpolating polynomial with 0 error, but would be a bit silly to present as a model).

IV. MATLAB CODE:

A. Creating the Noisy Data and Calling pfit.m

```
function go3;
close all;
x=-1:.2:2;
y=x.^3-3.^2+2.*x-1;
y=y+(2.*rand(1,length(x))-1);
plot(x,y,'o');
n=input(['Input Order of Polynomial (1<=n<=',num2str(length(x)-
1),') ']);
f=pfit(x,y,n),
```

B. The polynomial fitting function:

```
function f=pfit(x,y,n)
%pfit fits an nth-degree polynomial y=Pn(x) to a set of data
points X and
%Y.  Note: n<length(X)-1.  The resulting polynomial will be
plotted.
clf,close all;
plot(x,y,'o');%plot data;
hold
%Create x matrix.
X=[];
m=length(x);
if n>=m %need n+1 pts for nth order poly
    disp('Error--Not enough points to define polynomial');
    return;
end
for i=0:n; %build power matrix
    X=[x.^i',X];
end

%Determine coefficients
if n==m-1;
    A=inv(X)*y';
else
    A=(inv(X'*X)*X')*y';
end

%Plot result
xmin=min(x);
xmax=max(x);
xpts=linspace(xmin,xmax,100);
X=[]; %build new X matrix with 100 pts.
for i=0:n;
    X=[xpts.^i',X];
end
ypts=X*A;
plot(xpts, ypts,'k--');
```

```
%build function;
m=length(A),
f=num2str(A(m));
for i=m-1:-1:1;
    f=[f,'+',num2str(A(i)),'.*x.^',num2str(m-i)];
end
f=inline(f);

%plot and calulate errors
E=0;
for i=1:length(x);
    plot([x(i),x(i)],[y(i),f(x(i))],'r');
    E=E+(f(x(i))-y(i))^2;
end
title(['Best Fit Polynomial of Order ', num2str(n)]);
legend('data', 'best fit poly', ['Error =
',num2str(E)],'Location', 'NorthWest');
```