# Numerical Methods Lecture 5 - Curve Fitting Techniques

Topics

- motivation
- interpolation
- linear regression
- higher order polynomial form
- exponential form

## **Curve fitting -** motivation

For root finding, we used a given function to identify where it crossed zero

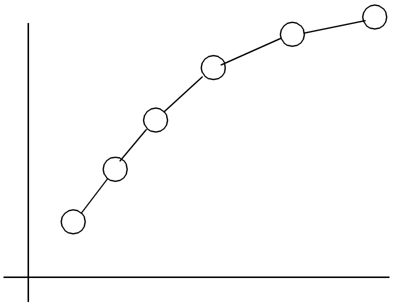
where does $f(x) = 0$ ??

Q: Where does this given function $f(x)$ come from in the first place?

- Analytical models of phenomena (e.g. equations from physics)
- Create an equation from observed data

**1) Interpolation** (connect the data-dots)

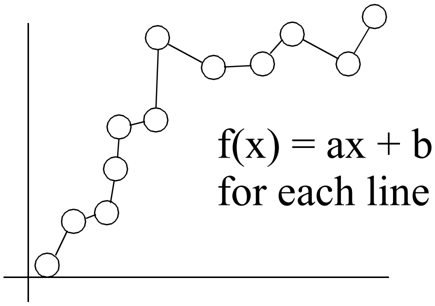
If data is reliable, we can plot it and connect the dots
This is piece-wise, linear interpolation



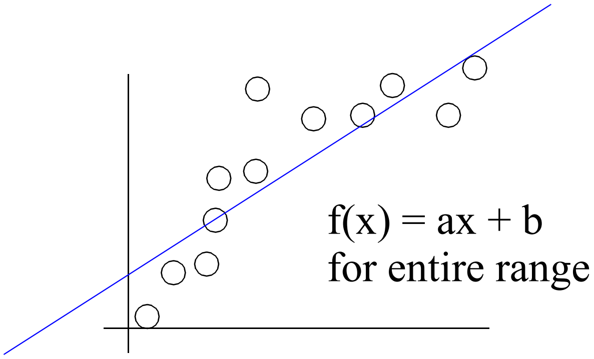
This has limited use as a general function $f(x)$

Since its really a group of small $f(x)$ s, connecting one point to the next it doesn't work very well for data that has built in random error (scatter)

**2) Curve fitting** - capturing the trend in the data by assigning a single function across the entire range. The example below uses a straight line function



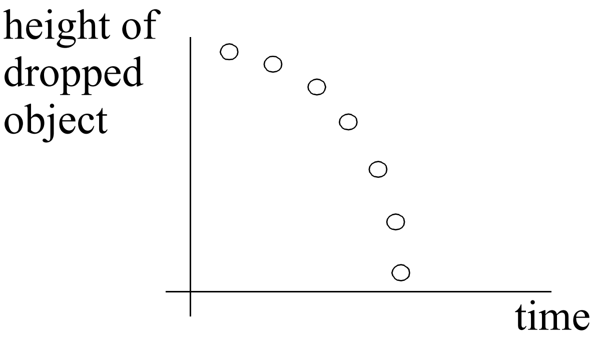
Interpolation

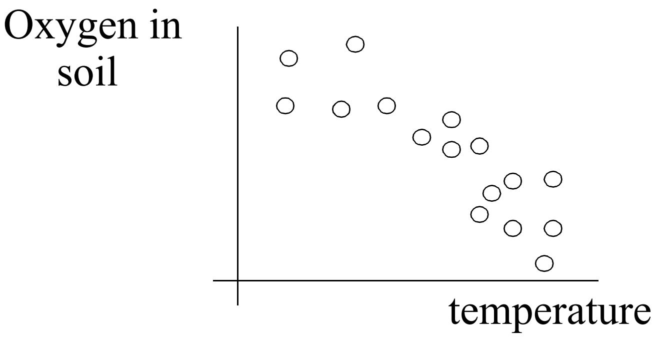


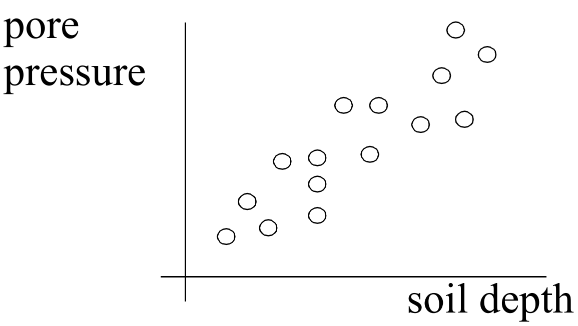
Curve Fitting

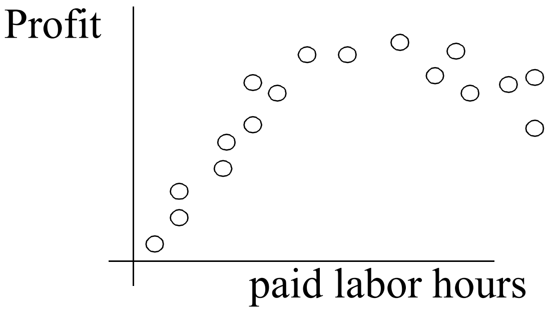
A straight line is described generically by f(x) = ax + b

**The goal** is to identify the coefficients 'a' and 'b' such that f(x) 'fits' the data well other examples of data sets that we can fit a function to.









Is a straight line suitable for each of these cases ?
No. But we're not stuck with just straight line fits. We'll start with straight lines, then expand the concept.

## Linear curve fitting (linear regression)

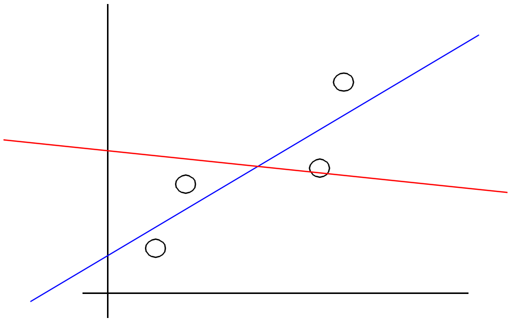
Given the general form of a straight line

$$f(x) = ax + b$$

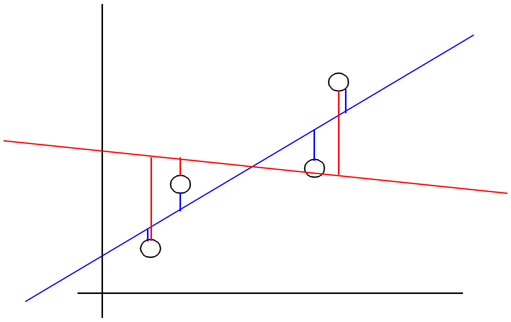
How can we pick the coefficients that best fits the line to the data?

First question: What makes a particular straight line a 'good' fit?

Why does the blue line appear to us to fit the trend better?



- Consider the distance between the data and points on the line
- Add up the length of all the red and blue verticle lines
- This is an expression of the 'error' between data and fitted line
- The one line that provides a <u>minimum</u> error is then the 'best' straight line



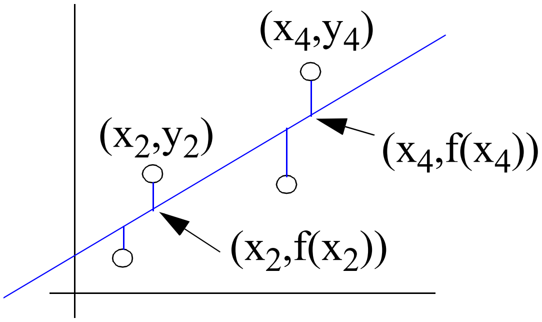
## *Quantifying error in a curve fit*

assumptions:

1) positive or negative error have the same value
(data point is above or below the line)
2) Weight greater errors more heavily

we can do both of these things by squaring the distance

denote data values as (x, y) ==============>>
denote points on the fitted line as (x, f(x))
sum the error at the four data points



$$err = \sum (d_i)^2 = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2$$
$$+ (y_3 - f(x_3))^2 + (y_4 - f(x_4))^2$$

Our fit is a straight line, so now substitute $f(x) = ax + b$

$$err = \sum_{i=1}^{\#\text{ data points}} (y_i - f(x_i))^2 = \sum_{i=1}^{\#\text{ data points}} (y_i - (ax_i + b))^2$$

The 'best' line has **minimum error** between line and data points

This is called the **least squares approach**, since we minimize the square of the error.

$$\text{minimize} \quad err = \sum_{i=1}^{\#\text{ data points} = n} (y_i - (ax_i + b))^2$$

time to pull out the **calculus**... finding the minimum of a function
1) derivative describes the slope
2) slope = zero is a minimum
==> take the derivative of the error with respect to $a$ and $b$, set each to zero

$$\frac{\partial err}{\partial a} = -2 \sum_{i=1}^{n} x_i(y_i - ax_i - b) = 0$$

$$\frac{\partial err}{\partial b} = -2 \sum_{i=1}^{n} (y_i - ax_i - b) = 0$$

Solve for the $a$ and $b$ so that the previous two equations both = 0
re-write these two equations

$$a\sum x_i^2 + b\sum x_i = \sum(x_i y_i)$$
$$a\sum x_i + b*n = \sum y_i$$

put these into **matrix form**

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}\begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum(x_i y_i) \end{bmatrix}$$

what’s unknown?
we have the data points $(x_i, y_i)$ for $i = 1, ..., n$, so we have all the summation terms in the matrix

so unknows are $a$ and $b$
Good news, we already know how to solve this problem
remember Gaussian elimination ??

$$A = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}, \quad X = \begin{bmatrix} b \\ a \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum(x_i y_i) \end{bmatrix}$$

so
$AX = B$

using built in Mathcad matrix inversion, the coefficients $a$ and $b$ are solved

$$>> X = A^{-1}*B$$

**Note:** $A$**,** $B$**, and** $X$ **are not the same as** $a$**,** $b$**, and** $x$

Let’s test this with an example:

| i | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $x$ | 0 | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 |
| $y$ | 0 | 1.5 | 3.0 | 4.5 | 6.0 | 7.5 |

First we find values for all the summation terms
$n = 6$

$$\sum x_i = 7.5, \quad \sum y_i = 22.5, \quad \sum x_i^2 = 13.75, \quad \sum x_i y_i = 41.25$$

Now plugging into the matrix form gives us:

$$\begin{bmatrix} 6 & 7.5 \\ 7.5 & 13.75 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 22.5 \\ 41.25 \end{bmatrix}$$

**Note**: we are using $\sum x_i^2$, NOT $(\sum x_i)^2$

$$\begin{bmatrix} b \\ a \end{bmatrix} = inv \begin{bmatrix} 6 & 7.5 \\ 7.5 & 13.75 \end{bmatrix} * \begin{bmatrix} 22.5 \\ 41.25 \end{bmatrix}$$ or use Gaussian elimination...

The solution is $\begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ ===> $f(x) = 3x + 0$

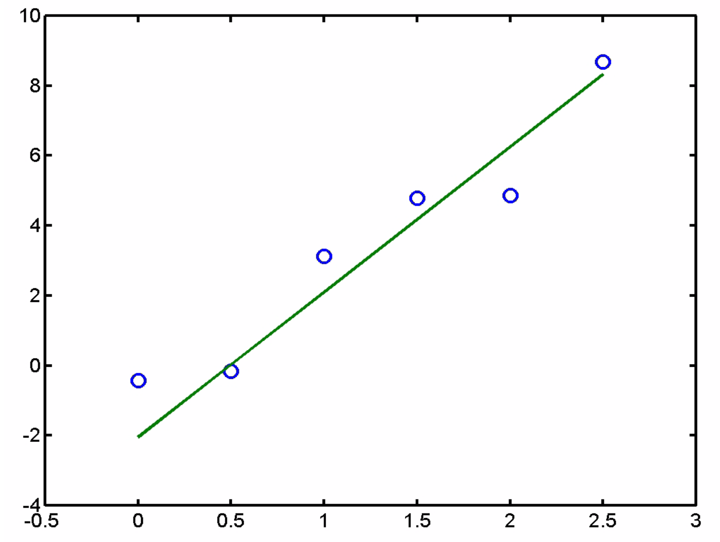
This fits the data exactly. That is, the error is zero. Usually this is not the outcome. Usually we have data that does not exactly fit a straight line.
Here's an example with some 'noisy' data

x = [0 .5 1 1.5 2 2.5], y = [-0.4326 -0.1656 3.1253 4.7877 4.8535 8.6909]

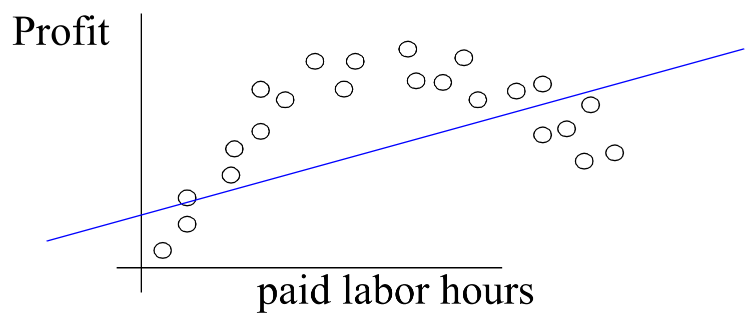
$$\begin{bmatrix} 6 & 7.5 \\ 7.5 & 13.75 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 20.8593 \\ 41.6584 \end{bmatrix}, \quad \begin{bmatrix} b \\ a \end{bmatrix} = inv \begin{bmatrix} 6 & 7.5 \\ 7.5 & 13.75 \end{bmatrix} * \begin{bmatrix} 20.8593 \\ 41.6584 \end{bmatrix}, \quad \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} -0.975 \\ 3.561 \end{bmatrix}$$

so our fit is $f(x) = 3.561\ x - 0.975$

Here's a plot of the data and the curve fit:



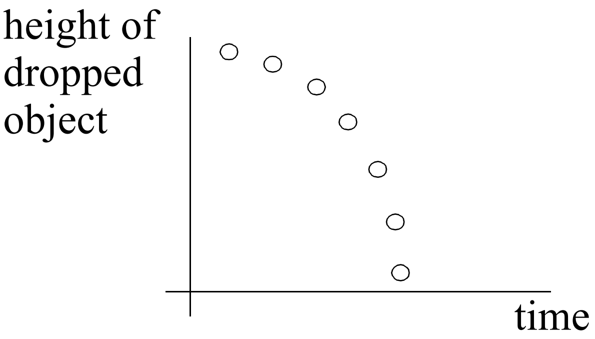
So...what do we do when a straight line is not suitable for the data set?

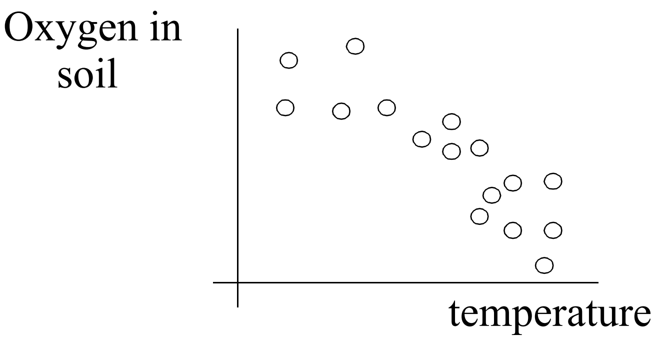


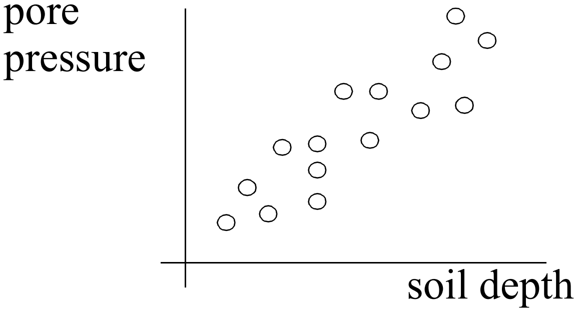
Straight line will not predict diminishing returns that data shows

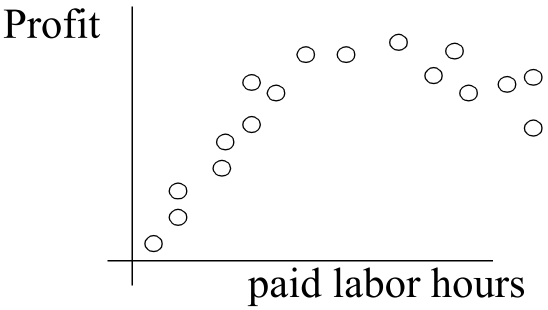
**Curve fitting -** higher order polynomials

We started the linear curve fit by choosing a generic form of the straight line f(x) = ax + b
This is just one kind of function. There are an infinite number of generic forms we could choose from for almost any shape we want. Let's start with a simple extension to the linear regression concept
recall the examples of sampled data









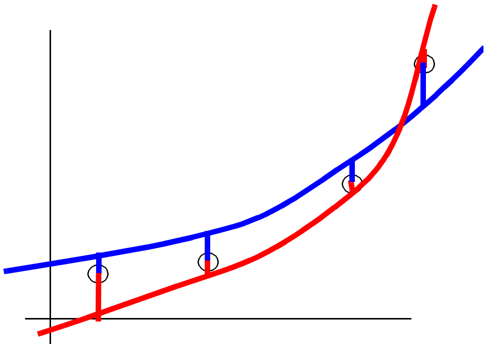
Is a straight line suitable for each of these cases ? Top left and bottom right don't look linear in trend, so why fit a straight line? No reason to, let's consider other options. There are lots of functions with lots of different shapes that depend on coefficients. We can choose a form based on experience and trial/error. Let's develop a few options for non-linear curve fitting. We'll start with a simple extension to linear regression...higher order polynomials

## Polynomial Curve Fitting

Consider the general form for a polynomial of order $j$

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_j x^j = a_0 + \sum_{k=1}^{j} a_k x^k \tag{1}$$

Just as was the case for linear regression, we ask:



How can we pick the coefficients that best fits the curve to the data? We can use the same idea:

The curve that gives minimum error between data $y$ and the fit $f(x)$ is 'best'

Quantify the error for these two second order curves...

- Add up the length of all the red and blue verticle lines
- pick curve with minimum total error

## Error - Least squares approach

The general expression for any error using the least squares approach is

$$err = \sum (d_i)^2 = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + (y_3 - f(x_3))^2 + (y_4 - f(x_4))^2 \qquad (2)$$

where we want to minimize this error. Now substitute the form of our eq. (1)

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_j x^j = a_0 + \sum_{k=1}^{j} a_k x^k$$

into the general least squares error eq. (2)

$$err = \sum_{i=1}^{n} \left( y_i - \left( a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \ldots + a_j x_i^j \right) \right)^2 \qquad (3)$$

where: $n$ - # of data points given, $i$ - the current data point being summed, $j$ - the polynomial order
re-writing eq. (3)

$$err = \sum_{i=1}^{n} \left( y_i - \left( a_0 + \sum_{k=1}^{j} a_k x^k \right) \right)^2 \qquad (4)$$

find the best line = minimize the error (squared distance) between line and data points
Find the set of coefficients $a_k$, $a_0$ so we can minimize eq. (4)

**CALCULUS TIME**

To minimize eq. (4), take the derivative with respect to each coefficient $a_0, a_k \quad k = 1, \ldots, j$ set each to zero

$$\frac{\partial err}{\partial a_0} = -2 \sum_{i=1}^{n} \left( y_i - \left( a_0 + \sum_{k=1}^{j} a_k x^k \right) \right) = 0$$

$$\frac{\partial err}{\partial a_1} = -2 \sum_{i=1}^{n} \left( y_i - \left( a_0 + \sum_{k=1}^{j} a_k x^k \right) \right) x = 0$$

$$\frac{\partial err}{\partial a_2} = -2 \sum_{i=1}^{n} \left( y_i - \left( a_0 + \sum_{k=1}^{j} a_k x^k \right) \right) x^2 = 0$$

$$\vdots$$

$$\vdots$$

$$\frac{\partial err}{\partial a_j} = -2 \sum_{i=1}^{n} \left( y_i - \left( a_0 + \sum_{k=1}^{j} a_k x^k \right) \right) x^j = 0$$

re-write these $j + 1$ equations, and put into matrix form

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \dots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \dots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \dots & \sum x_i^{j+2} \\ : & : & : & & : \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \dots & \sum x_i^{j+j} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ : \\ a_j \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \\ \sum \left( x_i^2 y_i \right) \\ : \\ \sum \left( x_i^j y_i \right) \end{bmatrix}$$

where all summations above are over $i = 1, ..., n$

what's unknown?

we have the data points $(x_i, y_i)$ for $i = 1, ..., n$

we want $a_0, a_k \quad k = 1, ..., j$

We already know how to solve this problem. Remember Gaussian elimination ??

$$A = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \dots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \dots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \dots & \sum x_i^{j+2} \\ : & : & : & & : \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \dots & \sum x_i^{j+j} \end{bmatrix}, \quad X = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ : \\ a_j \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \\ \sum \left( x_i^2 y_i \right) \\ : \\ \sum \left( x_i^j y_i \right) \end{bmatrix}$$

where all summations above are over $i = 1, ..., n$ data points

**Note:** No matter what the order $j$, we always get equations **LINEAR** with respect to the coefficients. This means we can use the following solution method

$$AX = B$$

using built in Mathcad matrix inversion, the coefficients $a$ and $b$ are solved

```
>> X = A^-1*B
```

*Example #1:*

Fit a second order polynomial to the following data

| i | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $x$ | 0 | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 |
| $y$ | 0 | 0.25 | 1.0 | 2.25 | 4.0 | 6.25 |

Since the order is 2 ($j = 2$), the matrix form to solve is

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}$$

Now plug in the given data.
**Before we go on**...***what answers do you expect for the coefficients after looking at the data?***

$n = 6$

$\sum x_i = 7.5$, $\sum y_i = 13.75$

$\sum x_i^2 = 13.75$, $\sum x_i y_i = 28.125$

$\sum x_i^3 = 28.125$ $\sum x_i^2 y_i = 61.1875$

$\sum x_i^4 = 61.1875$

$$\begin{bmatrix} 6 & 7.5 & 13.75 \\ 7.5 & 13.75 & 28.125 \\ 13.75 & 28.125 & 61.1875 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 13.75 \\ 28.125 \\ 61.1875 \end{bmatrix}$$

**Note: we are using $\sum x_i^2$, NOT $(\sum x_i)^2$. There's a big difference**

using the inversion method $\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = inv \begin{bmatrix} 6 & 7.5 & 13.75 \\ 7.5 & 13.75 & 28.125 \\ 13.75 & 28.125 & 61.1875 \end{bmatrix} * \begin{bmatrix} 13.75 \\ 28.125 \\ 61.1857 \end{bmatrix}$

or use Gaussian elimination gives us the solution to the coefficients

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad ===> \quad f(x) = 0 + 0*x + 1*x^2$$

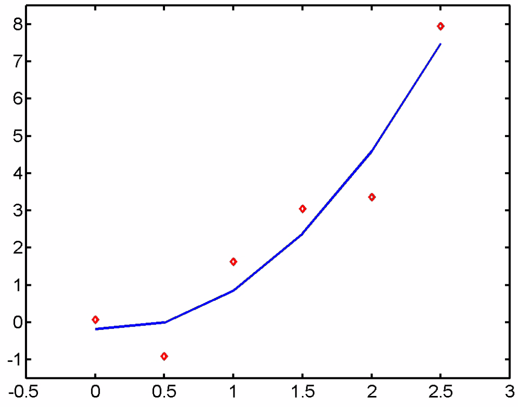
This fits the data exactly. That is, f(x) = y since y = x^2

## *Example #2: uncertain data*

Now we'll try some 'noisy' data

x = [0 .0 1 1.5 2 2.5]
y = [0.0674 -0.9156 1.6253 3.0377 3.3535 7.9409]
The resulting system to solve is:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = inv \begin{bmatrix} 6 & 7.5 & 13.75 \\ 7.5 & 13.75 & 28.125 \\ 13.75 & 28.125 & 61.1875 \end{bmatrix} * \begin{bmatrix} 15.1093 \\ 32.2834 \\ 71.276 \end{bmatrix}$$

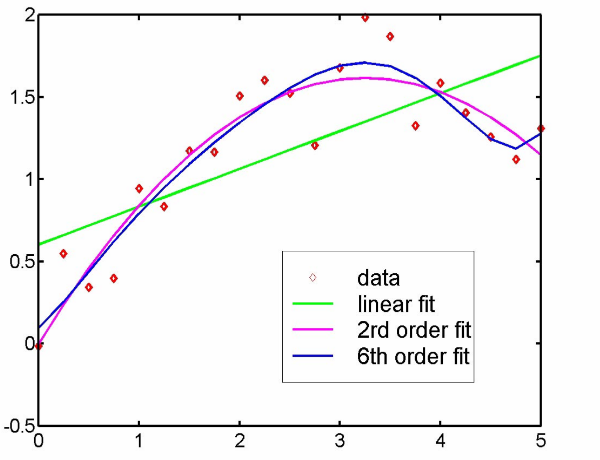


giving: $$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -0.1812 \\ -0.3221 \\ 1.3537 \end{bmatrix}$$

So our fitted second order function is:

$$f(x) = -0.1812 - 0.3221x* + 1.3537*x^2$$

## *Example #3 : data with three different fits*



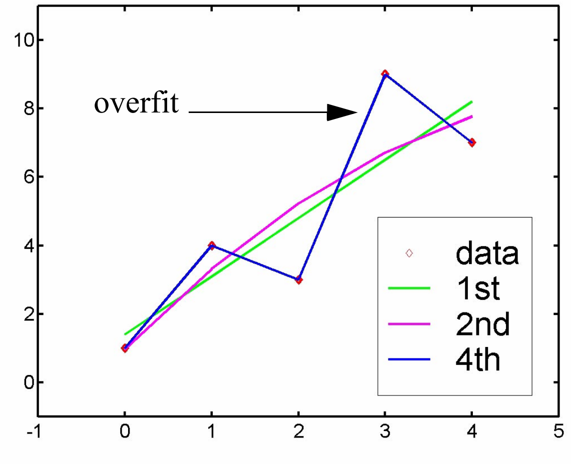
In this example, we're not sure which order will fit well, so we try three different polynomial orders
Note: Linear regression, or first order curve fitting is just the general polynomial form we just saw, where we use j=1,

- 2nd and 6th order look similar, but 6th has a 'squiggle to it. We may not want that...

**Overfit / Underfit** - picking an inappropriate order

*Overfit -* over-doing the requirement for the fit to 'match' the data trend (order too high)

Polynomials become more 'squiggly' as their order increases. A 'squiggly' appearance comes from inflections in function



Consideration #1:

3rd order - 1 inflection point
4th order - 2 inflection points
nth order - n-2 inflection points

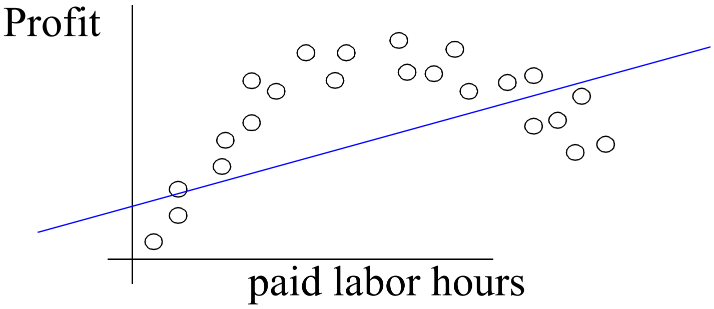
Consideration #2:

2 data points - linear touches each point
3 data points - second order touches each point
n data points - n-1 order polynomial will touch each point

SO: Picking an order too high will **overfit** data

**General rule:** pick a polynomial form at least several orders lower than the number of data points.
Start with linear and add order until trends are matched.

*Underfit -* If the order is too low to capture obvious trends in the data



Straight line will not predict
diminishing returns that data shows

**General rule:** View data first, then select an order that reflects inflections, etc.

For the example above:
1) Obviously nonlinear, so order > 1
2) No inflcetion points observed as obvious, so order < 3 is recommended
=====> I'd use 2nd order for this data

## **Curve fitting** - Other nonlinear fits (exponential)

Q: Will a polynomial of any order necessarily fit any set of data?
A: Nope, lots of phenomena don't follow a polynomial form. They may be, for example, exponential

## *Example :* Data (x,y) follows exponential form

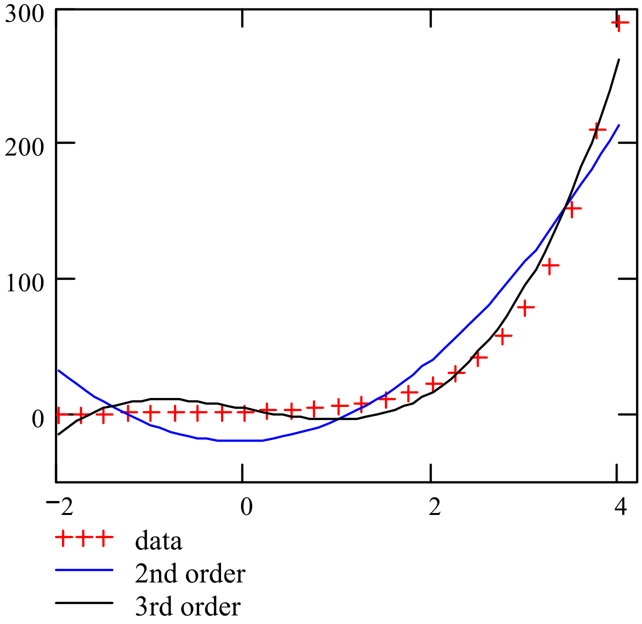
The next line references a separate worksheet with a function inside called Create_Vector. I can use the function here as long as I reference the worksheet first

```
Reference:C:\Mine\Mathcad\Tutorials\MyFunctions.mcd

X := Create_Vector(−2, 4, .25)        Y := 1.6·exp(1.3·X)

f2 := regress(X, Y, 2)                f3 := regress(X, Y, 3)

fit2(x) := interp(f2, X, Y, x)        fit3(x) := interp(f3, X, Y, x)     i := −2, −1.9.. 4
```



Note that neither 2nd nor 3rd order fit really describes the data well, but higher order will only get more 'squiggly'

We created this sample of data using an exponential function. Why not create a general form of the exponential function, and use the error minimization concept to identify its coefficients. That is, let's replace

the polynomial equation $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_jx^j = a_0 + \sum_{k=1}^{j} a_kx^k$

With a general exponential equation $f(x) = Ce^{Ax} = C\exp(Ax)$
where we will seek C and A such that this equation fits the data as best it can.

Again with the error: solve for the coefficients $C, A$ such that the error is minimized:

minimize $$err = \sum_{i=1}^{n} (y_i - (C\exp(Ax)))^2$$

**Problem:** When we take partial derivatives with respect to $err$ and set to zero, we get two **NONLINEAR** equations with respect to $C, A$

So what? We can't use Gaussian Elimination or the inverse function anymore.
Those methods are for **LINEAR** equations only...

Now what?

## *Solution #1: Nonlinear equation solving methods*

Remember we used Newton Raphson to solve a single nonlinear equation? (root finding)
We can use Newton Raphson to solve a system of nonlinear equations.
Is there another way? For the exponential form, yes there is

## *Solution #2: Linearization:*

Let's see if we can do some algebra and change of variables to re-cast this as a linear problem...
Given: pair of data (x,y)

Find: a function to fit data of the general exponential form $y = Ce^{Ax}$

1) Take logarithm of both sides to get rid of the exponential $\ln(y) = \ln(Ce^{Ax}) = Ax + \ln(C)$

2) Introduce the following change of variables: $Y = \ln(y)$, $X = x$, $B = \ln(C)$

Now we have: $Y = AX + B$ which is a LINEAR equation

The original data points in the $x - y$ plane get mapped into the $X - Y$ plane.

This is called data linearization. The data is transformed as: $(x, y) \Rightarrow (X, Y) = (x, \ln(y))$

Now we use the method for solving a first order linear curve fit $\begin{bmatrix} n & \sum X \\ \sum X & \sum X^2 \end{bmatrix} \begin{bmatrix} B \\ A \end{bmatrix} = \begin{bmatrix} \sum Y \\ \sum XY \end{bmatrix}$

for $A$ and $B$, where above $Y = \ln(y)$, and $X = x$

Finally, we operate on $B = \ln(C)$ to solve $C = e^B$

And we now have the coefficients for $y = Ce^{Ax}$

*Example: repeat previous example, add exponential fit*

$X := \text{Create\_Vector}(-2, 4, .25)$ $\quad Y := 1.6 \cdot \exp(1.3 \cdot X)$

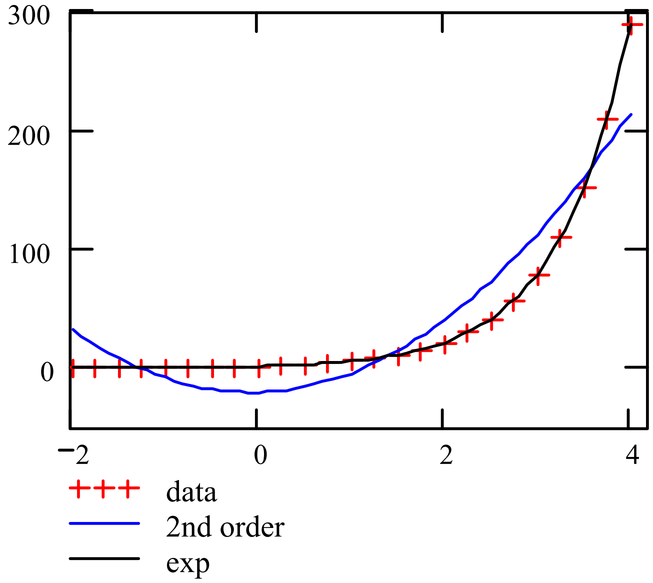
$f2 := \text{regress}(X, Y, 2)$ $\quad f3 := \text{regress}(X, Y, 3)$

$\text{fit2}(x) := \text{interp}(f2, X, Y, x)$ $\quad \text{fit3}(x) := \text{interp}(f3, X, Y, x)$

ADDING NEW STUFF FOR EXP FIT

$Y2 := \ln(Y)$ $\quad \text{fexp} := \text{regress}(X, Y2, 1)$ $\quad \text{coeff} := \text{submatrix}(\text{fexp}, 4, 5, 1, 1)$

$C := \exp\left(\text{coeff}_1\right)$ $\quad A := \text{coeff}_2$ $\quad \text{fitexp}(x) := C \cdot \exp(A \cdot x)$ $\quad i := -2, -1.9 .. 4$



$A = 1.3$

$C = 1.6$