Line and curve fitting

Line fitting in MATLAB

- MATLAB provides a function for performing least-squares polynomial fitting
  - a line is a polynomial of degree 1

```
>> x = [-4; 3.7; 0; 2.5; 1.2; -2.8; -1.4];
>> y = [-37; 38; 0; 29; 21; -21; -8];
>> polyfit(x, y, 1)
ans =
    9.7436    4.2564
```

  - this says that the best fit line is $y = 9.7436x + 4.2564$

Line fitting in MATLAB

- MATLAB provides a function named **`polyval`** for evaluating the polynomial computed by **`polyfit`**

```
>> coeffs = polyfit(x, y, 1);
>> yfit = polyval(coeffs, x)
yfit =
  -34.7182
   40.3079
    4.2564
   28.6155
   15.9488
  -23.0258
   -9.3847
```

the best fit line $y = a + bx$ evaluated at each value in **`x`**

Line fitting in MATLAB

- computing the residual errors is easy using **`polyval`**

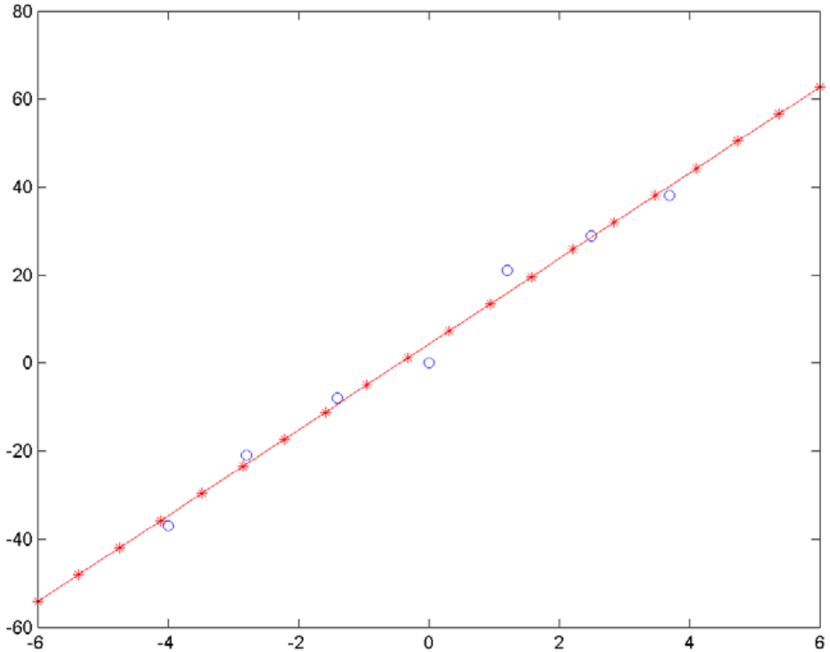
```
>> coeffs = polyfit(x, y, 1);
>> yfit = polyval(coeffs, x);
>> res = y - yfit
res =
   -2.2818
   -2.3079
   -4.2564
    0.3845
    5.0512
    2.0258
    1.3847
```

the residual errors $r_i = y_i - (a + bx_i)$

Line fitting in MATLAB

- you can use any vector of values **x** in **polyval**

```
>> xfit = linspace(-6, 6, 20);
>> yfit = polyval(coeffs, xfit);
>> plot(x, y, 'bo', xfit, yfit, 'r*-')
```

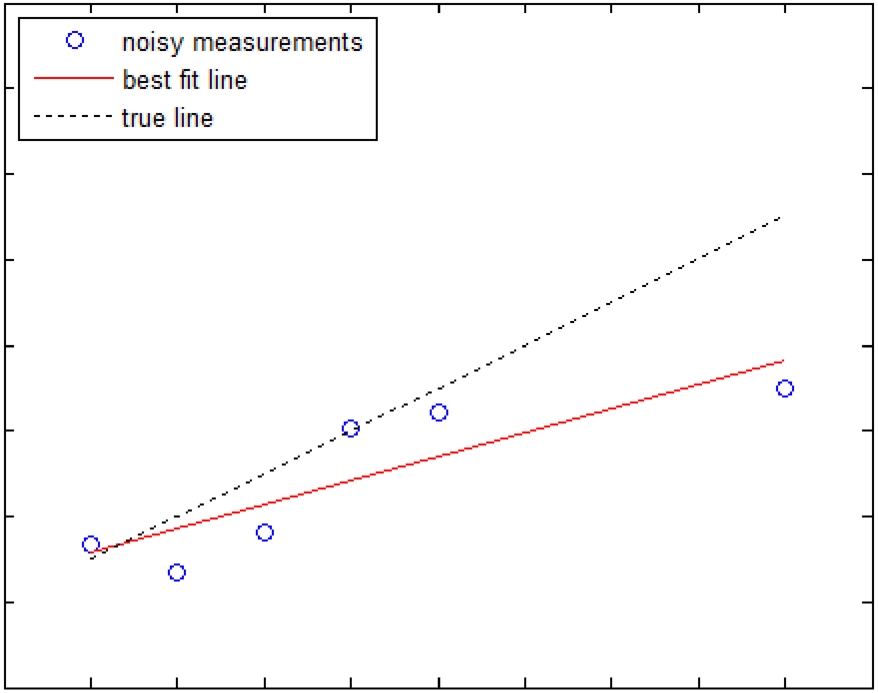


Leverage points in line fitting

- in line fitting, a high leverage point is a measurement made near the extremes of the range of independent variable
  - if this measurement is erroneous, or can only be made with low precision, then it will have a large effect on the fitted line

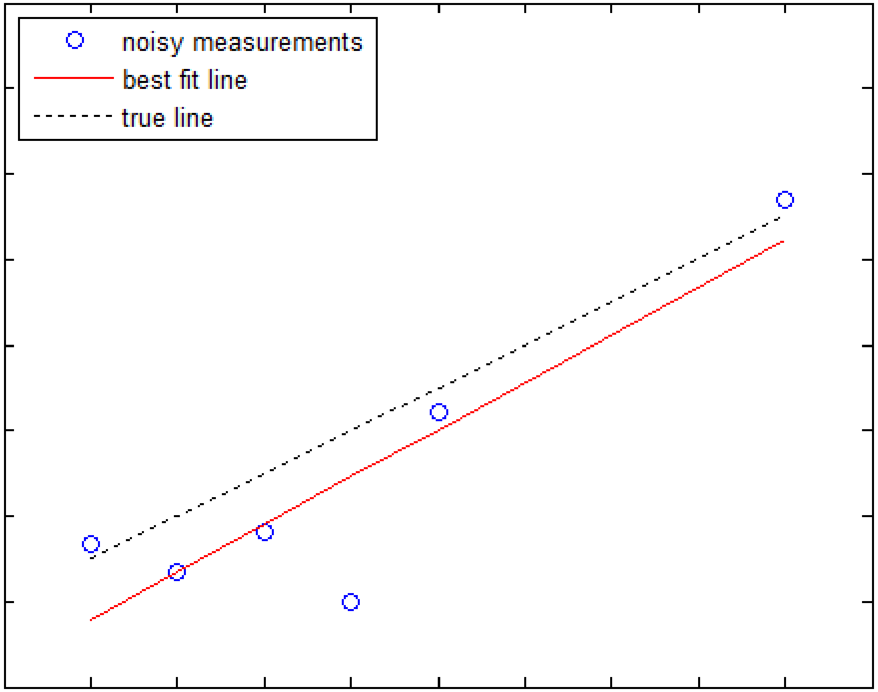
Leverage points in line fitting

- effect of a high leverage point on a line fit



Leverage points in line fitting

- effect of a low leverage point on a line fit



Transforming non-linear relationships

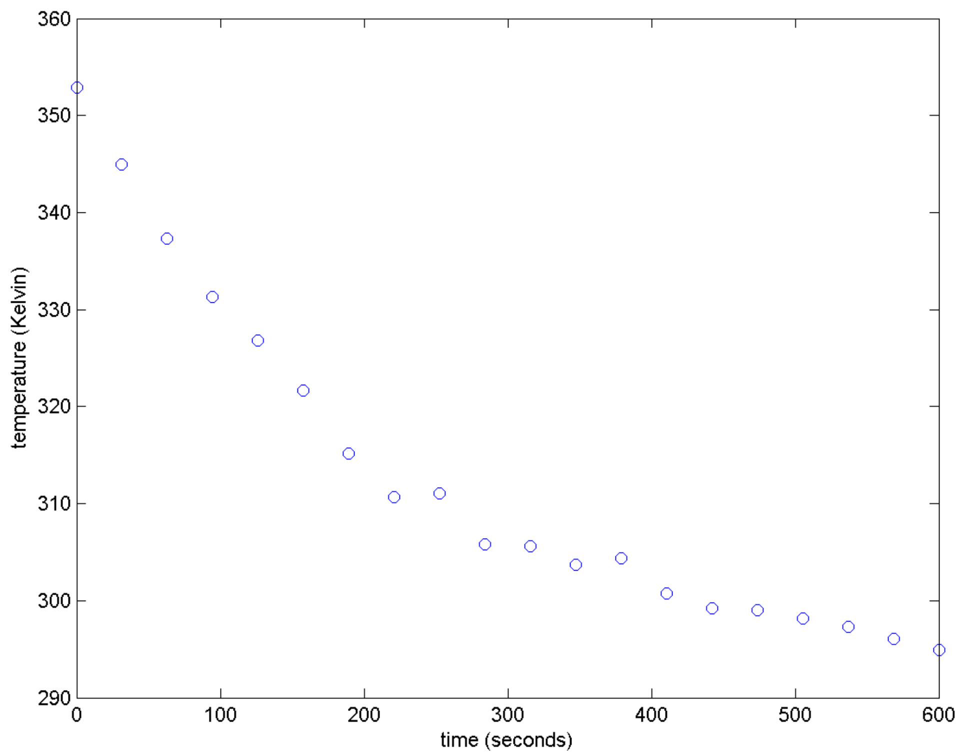
- line fitting can be applied to a non-linear problem if the problem can be transformed into a linear one; e.g.,
- Newton's law of cooling states that the rate of change of the temperature of an object is proportional to the difference between the temperature of the object and its surrounding environment
  - it can be shown that the temperature of the object as a function of time is:

$$T(t) = T_{env} + (T_0 - T_{env})e^{-rt}$$

where T_0 is the temperature at $t = 0$, T_{env} is the temperature of the surrounding environment, and r is a constant

Transforming non-linear relationships

- suppose that you have 20 measurements $T(t)$ and a measurement of T_{env}; what is the value of r?



$$T(t) = 293.15 + (353.15 - 293.15)e^{-0.005t} + \mathcal{N}(0, 1^2)$$

Transforming non-linear relationships

- to use line fitting, we need a linear relationship in t
  - in this case, a straightforward transformation exists:

$$T(t) = T_{env} + (T_0 - T_{env})e^{-rt}$$

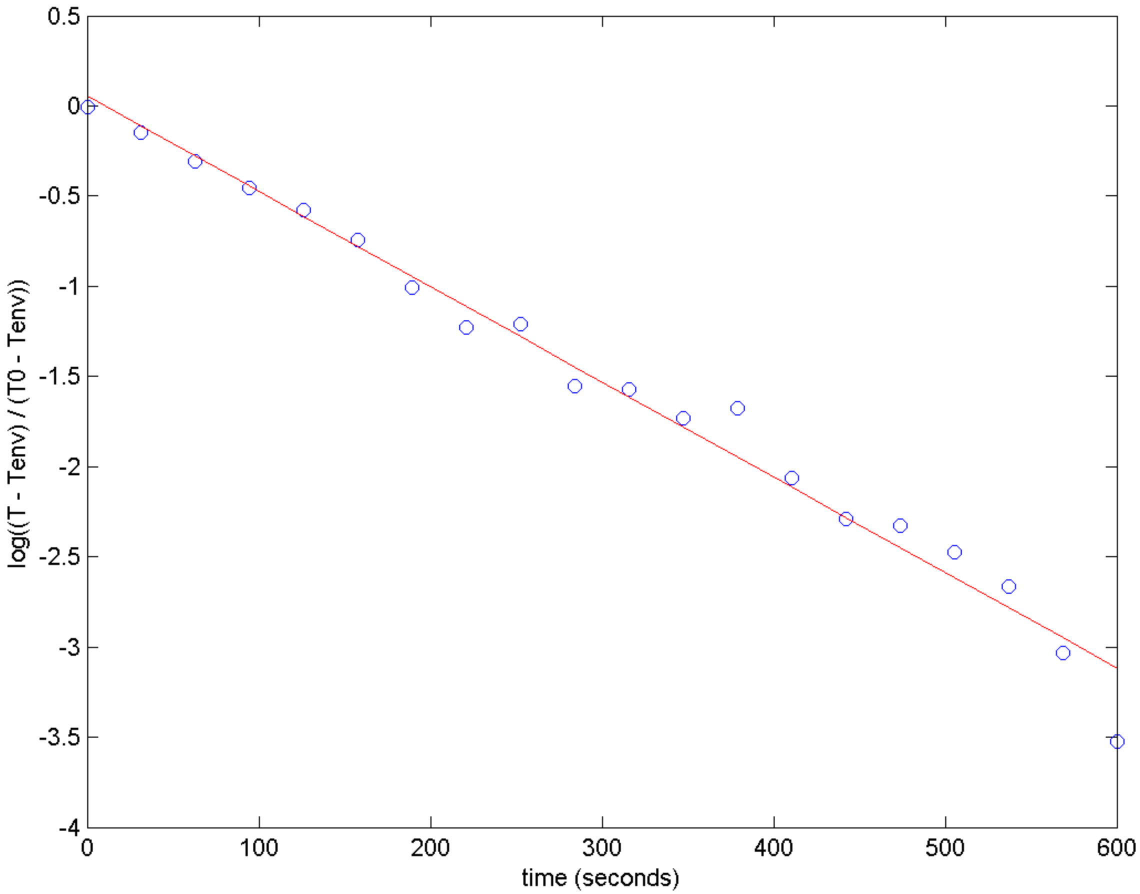
$$\frac{T - T_{env}}{T_0 - T_{env}} = e^{-rt}$$

$$\ln\left(\frac{T - T_{env}}{T_0 - T_{env}}\right) = -rt$$

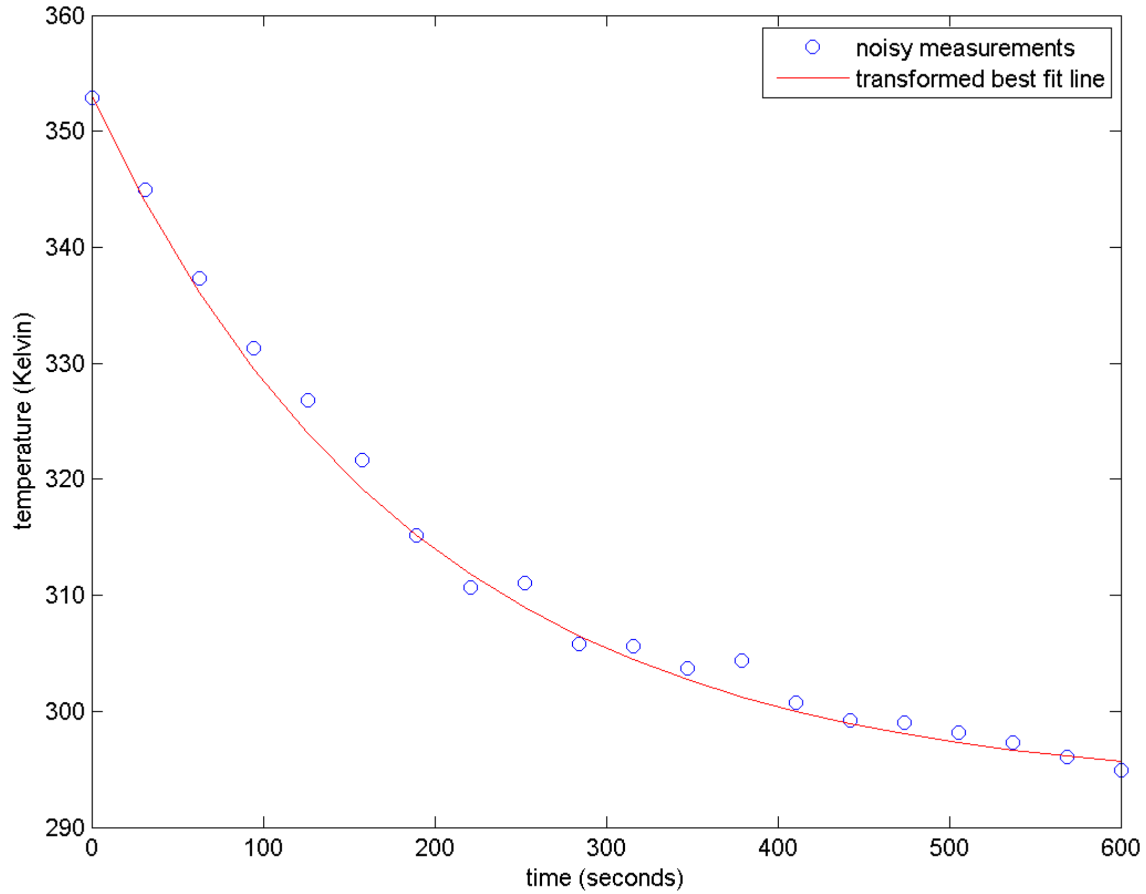
Transforming non-linear relationships

```
% t      time vector
% T      temperature measurements taken at t
% T0     353.15 K
% Tenv   293.15 K

U = log((T - Tenv) / (T0 - Tenv));
coeffs = polyfit(t, U, 1);
Ufit = polyval(coeffs, t);
plot(t, U, 'o', t, Ufit, 'r-');

slope = coeffs(1);     % what is coeffs(2)?
r = -slope             % should be 0.005
```

Transforming non-linear relationships



Transforming non-linear relationships



Transforming non-linear relationships

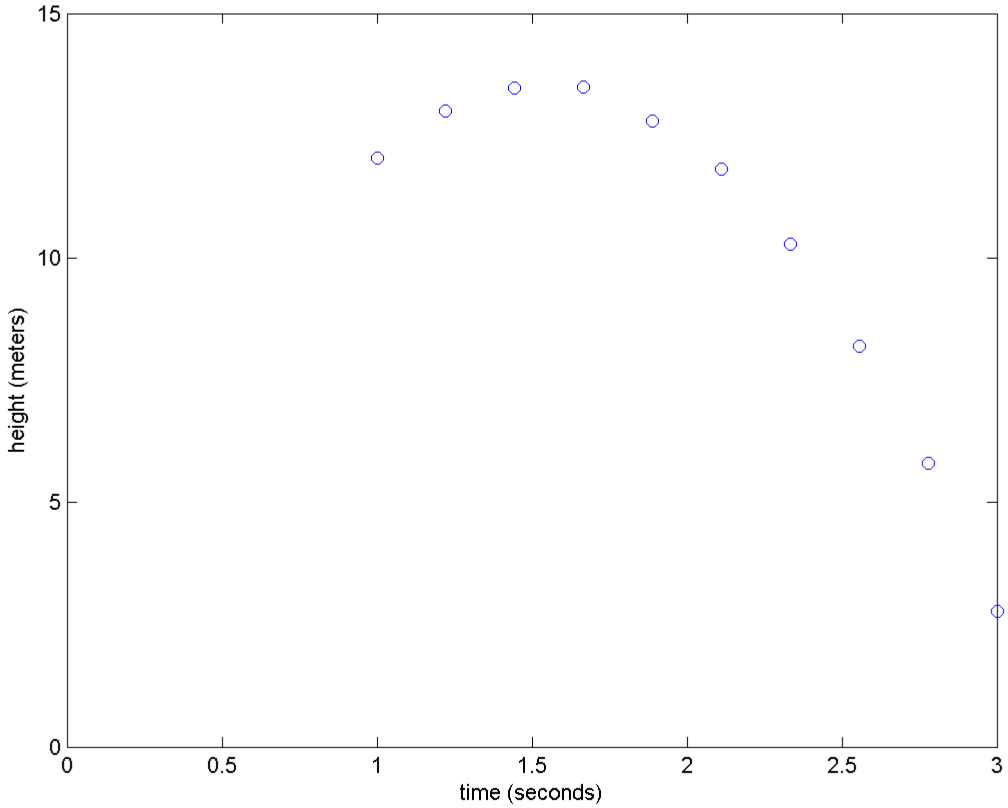
- exercise for the student:
  - you have to be very careful when using this approach; why?
  - hint
    - extend the measurements $T(t)$ to $t = 1200\ s$ and perform the same analysis
    - can you explain the appearance of the plot of **`U = log((T - Tenv) / (T0 - Tenv))`** when you extend the measurements to $t = 1200\ s$?

Polynomial fitting in MATLAB

- polyfit can be used to fit a polynomial of any degree

- suppose that at time $t = 0\ s$ you launch a ball straight up with an unknown initial velocity and unknown initial height
- starting at time $t = 1\ s$, you obtain measurements of the height $y(t)$ of the ball
- find the initial velocity and initial height of the ball

Polynomial fitting in MATLAB



$$v_0 = 15\,\frac{m}{s}, \quad y_0 = 2\,m \qquad y(t) = -\frac{1}{2}gt^2 + 15t + 2 + \mathcal{N}(0, 0.05^2)$$

Polynomial fitting in MATLAB

```
% t      time vector (from 1 to 3 s)
% y      height measurements taken at t

coeffs = polyfit(t, y, 2);

t2 = linspace(0, 3, 20);
yfit = polyval(coeffs, t2);
plot(t, y, 'o', t2, yfit, 'r-');

g = -2 * coeffs(1)     % should be 9.81
v0 = coeffs(2)         % should be 15
y0 = coeffs(3)         % should be 2
```

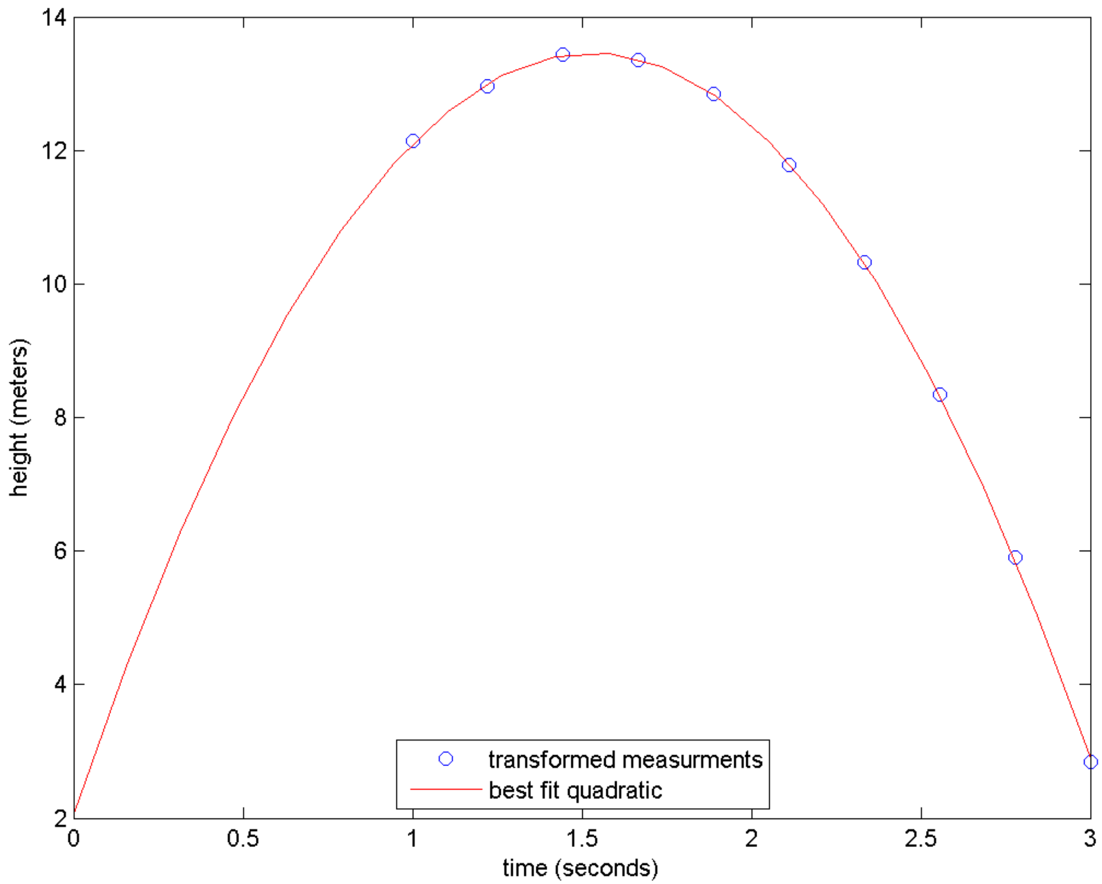
Polynomial fitting in MATLAB

values from fitting

$\hat{g} = 9.9842 \, \frac{m}{s^2}$

$\widehat{v_0} = 15.3073 \, \frac{m}{s}$,

$\widehat{y_0} = 1.7568 \, m$



Another view of line fitting

- we can view line fitting as solving a system of linear equations
  - given n measurements (x_i, y_i), find a and b

$$a + bx_1 = y_1$$

$$a + bx_2 = y_2$$

$$a + bx_3 = y_3$$

$$\vdots$$

$$a + bx_n = y_n$$

Another view of line fitting

- in matrix form:

$$
\left.\begin{aligned}
a + bx_1 &= y_1 \\
a + bx_2 &= y_2 \\
a + bx_3 &= y_3 \\
&\vdots \\
a + bx_n &= y_n
\end{aligned}\right\}
\quad
\underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ & \vdots \\ 1 & x_n \end{bmatrix}}_{\mathbf{A}}
\underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\mathbf{x}}
=
\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}}
$$

which can be solved in MATLAB in a least-squares sense as $\mathbf{x} = \mathbf{A} \setminus \mathbf{y}$

Another view of polynomial fitting

- polynomials are almost exactly the same:

$$\left.\begin{aligned} a + bx_1 + c{x_1}^2 &= y_1 \\ a + bx_2 + c{x_2}^2 &= y_2 \\ a + bx_3 + c{x_3}^2 &= y_3 \\ &\vdots \\ a + bx_n + cxn^2 &= y_n \end{aligned}\right\} \qquad \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ & \vdots & \\ 1 & x_n & x_n^2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}}$$

which can be solved in MATLAB in a least-squares sense as $\mathbf{x} = \mathbf{A} \setminus \mathbf{y}$

Another view of polynomial fitting

- a matrix **A** of the form

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix}$$

is called a Vandermonde matrix