

8. Linear least-squares

- definition
- examples and applications

Definition

Overdetermined linear equations

$$Ax = b \quad (A \text{ is } m \times n \text{ with } m > n)$$

for most b , cannot solve for x

Least-squares formulation

$$\text{minimize } \|Ax - b\| = \left(\sum_{i=1}^m \left(\sum_{j=1}^n a_{ij}x_j - b_i \right)^2 \right)^{1/2}$$

- $r = Ax - b$ is called the *residual* or *error*
- x with smallest residual norm $\|r\|$ is called the *least-squares solution*
- equivalent to minimizing $\|Ax - b\|^2$

Example

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Least-squares solution

$$\text{minimize } (2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2$$

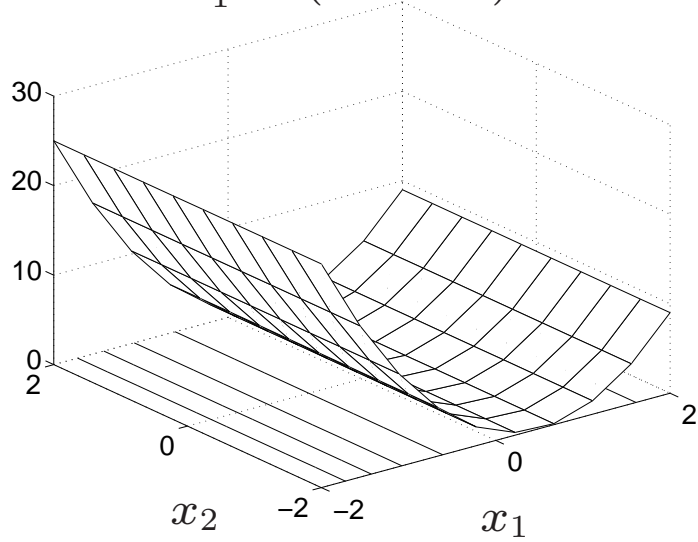
to find optimal x_1, x_2 , set derivatives w.r.t. x_1 and x_2 equal to zero:

$$10x_1 - 2x_2 - 4 = 0, \quad -2x_1 + 10x_2 + 4 = 0$$

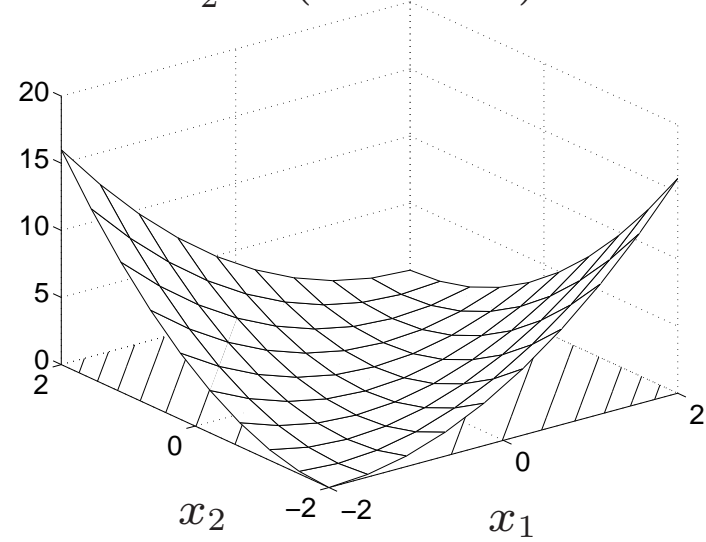
solution $x_1 = 1/3, x_2 = -1/3$

(much more on solving LS problems later)

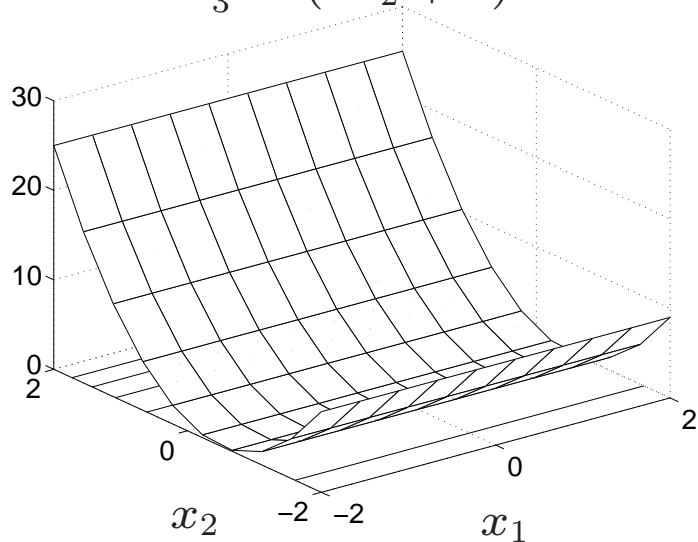
$$r_1^2 = (2x_1 - 1)^2$$



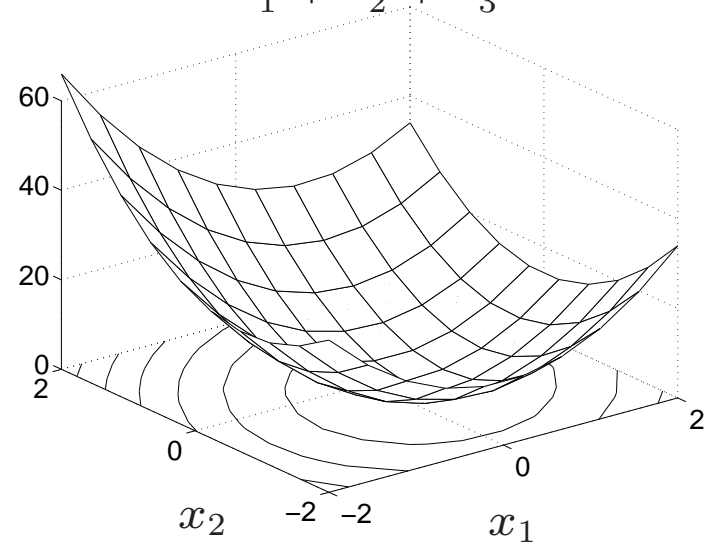
$$r_2^2 = (-x_1 + x_2)^2$$



$$r_3^2 = (2x_2 + 1)^2$$



$$r_1^2 + r_2^2 + r_3^2$$



Data fitting

fit a function

$$g(t) = x_1g_1(t) + x_2g_2(t) + \cdots + x_n g_n(t)$$

to data $(t_1, y_1), \dots, (t_m, y_m)$, *i.e.*, choose coefficients x_1, \dots, x_n so that

$$g(t_1) \approx y_1, \quad g(t_2) \approx y_2, \quad \dots, \quad g(t_m) \approx y_m$$

- $g_i(t) : \mathbf{R} \rightarrow \mathbf{R}$ are given functions (*basis functions*)
- problem variables: the coefficients x_1, x_2, \dots, x_n
- usually $m \gg n$, hence no exact solution with $g(t_i) = y_i$ for all i
- applications: developing simple, approximate model of observed data

Least-squares data fitting

compute x by minimizing

$$\sum_{i=1}^m (g(t_i) - y_i)^2 = \sum_{i=1}^m (x_1 g_1(t_i) + x_2 g_2(t_i) + \cdots + x_n g_n(t_i) - y_i)^2$$

in matrix notation: minimize $\|Ax - b\|^2$ where

$$A = \begin{bmatrix} g_1(t_1) & g_2(t_1) & g_3(t_1) & \cdots & g_n(t_1) \\ g_1(t_2) & g_2(t_2) & g_3(t_2) & \cdots & g_n(t_2) \\ \vdots & \vdots & \vdots & & \vdots \\ g_1(t_m) & g_2(t_m) & g_3(t_m) & \cdots & g_n(t_m) \end{bmatrix}, \quad b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Example: data fitting with polynomials

$$g(t) = x_1 + x_2t + x_3t^2 + \cdots + x_nt^{n-1}$$

basis functions are $g_k(t) = t^{k-1}$, $k = 1, \dots, n$

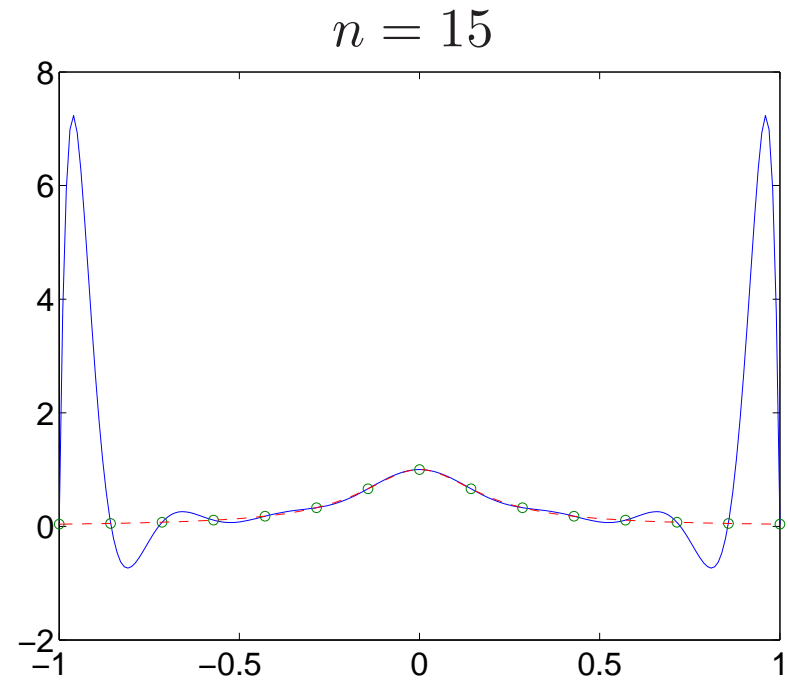
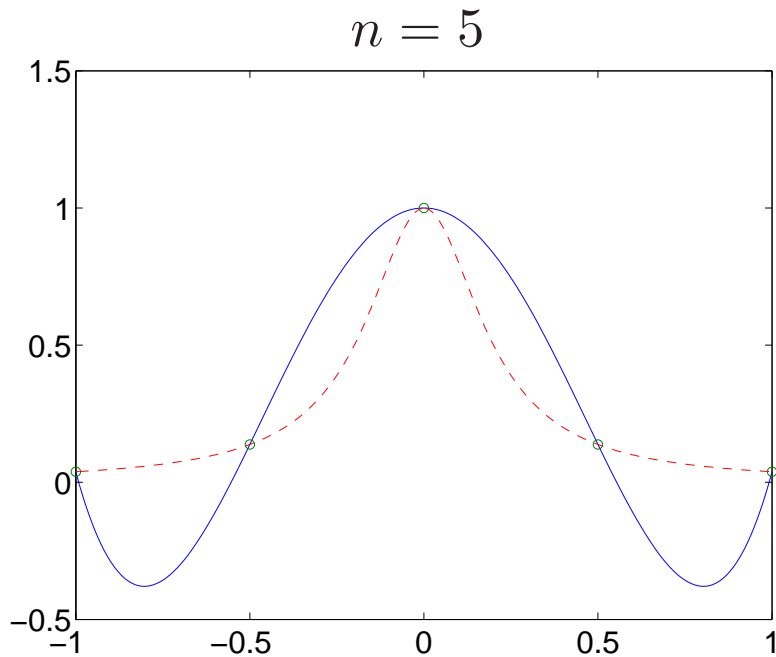
$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}, \quad b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Interpolation ($m = n$): can satisfy $g(t_i) = y_i$ exactly by solving $Ax = b$

Approximation ($m > n$): make error small by minimizing $\|Ax - b\|$

Example. fit a polynomial to $f(t) = 1/(1 + 25t^2)$ on $[-1, 1]$

- pick $m = n$ points t_i in $[-1, 1]$, and calculate $y_i = 1/(1 + 25t_i^2)$
- interpolate by solving $Ax = b$

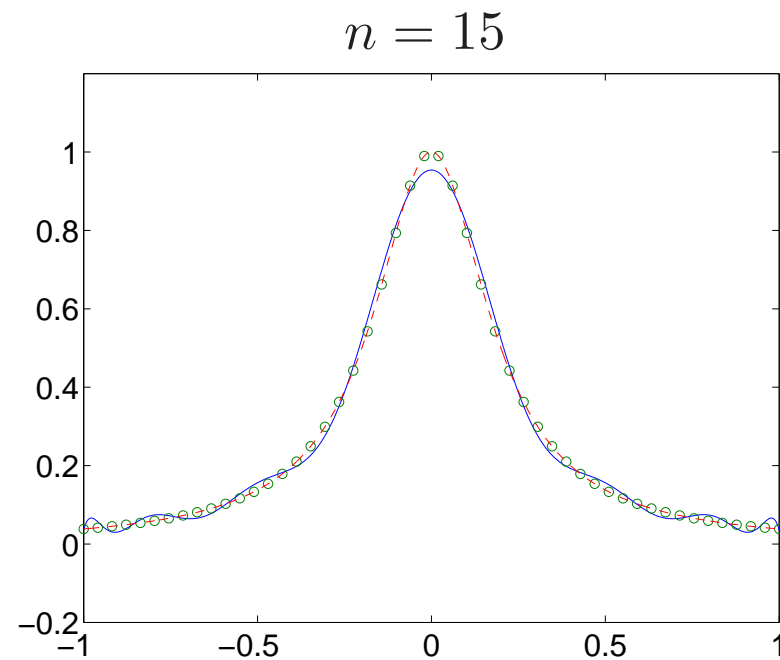
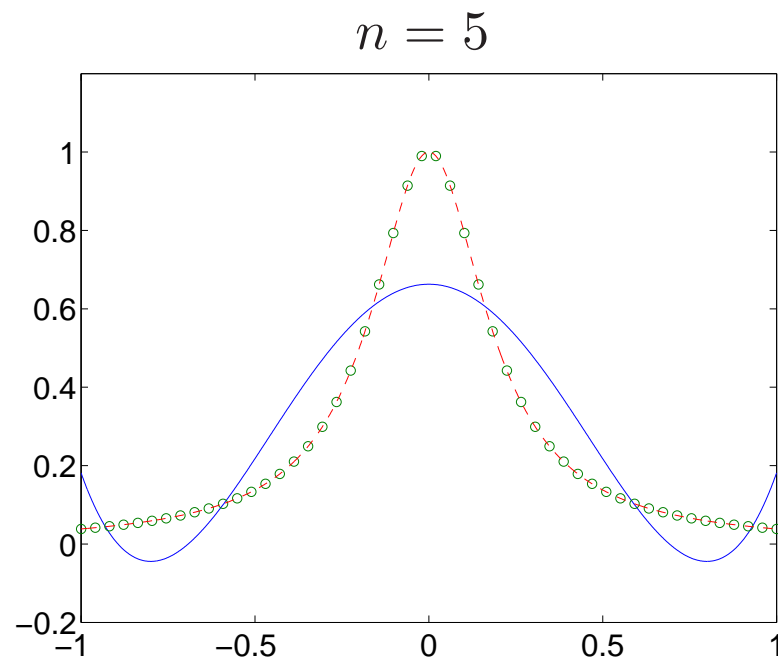


(dashed line: f ; solid line: polynomial g ; circles: the points (t_i, y_i))

increasing n does not improve the overall quality of the fit

Same example by approximation

- pick $m = 50$ points t_i in $[-1, 1]$
- fit polynomial by minimizing $\|Ax - b\|$



(dashed line: f ; solid line: polynomial g ; circles: the points (t_i, y_i))

much better fit overall

Least-squares estimation

$$y = Ax + w$$

- x is what we want to estimate or reconstruct
- y is our measurement(s)
- w is an unknown *noise* or *measurement error* (assumed small)
- i th row of A characterizes i th sensor or i th measurement

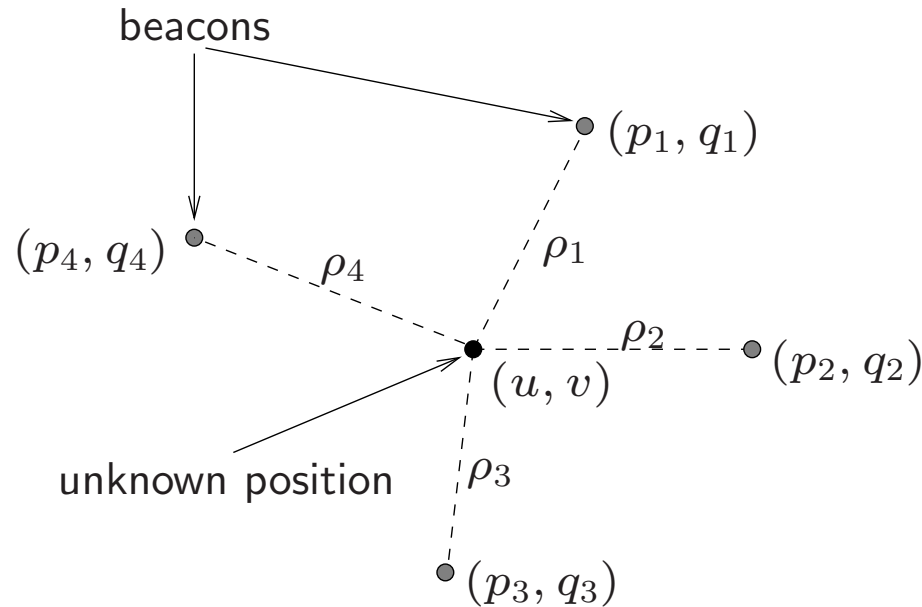
Least-squares estimation: choose as estimate the vector \hat{x} that minimizes

$$\|A\hat{x} - y\|$$

i.e., minimize the deviation between what we actually observed (y), and what we would observe if $x = \hat{x}$, and there were no noise ($w = 0$)

Navigation by range measurements

find position (u, v) in a plane from distances to beacons at positions (p_i, q_i)



four **nonlinear** equations in two variables u, v :

$$\sqrt{(u - p_i)^2 + (v - q_i)^2} = \rho_i \quad \text{for } i = 1, 2, 3, 4$$

ρ_i is the measured distance from unknown position (u, v) to beacon i

Linearized distance function: assume $u = u_0 + \Delta u$, $v = v_0 + \Delta v$ where

- u_0, v_0 are known (*e.g.*, position a short time ago)
- $\Delta u, \Delta v$ are small (compared to ρ_i 's)

$$\begin{aligned} & \sqrt{(u_0 + \Delta u - p_i)^2 + (v_0 + \Delta v - q_i)^2} \\ & \approx \sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2} + \frac{(u_0 - p_i)\Delta u + (v_0 - q_i)\Delta v}{\sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2}} \end{aligned}$$

gives four **linear** equations in the variables $\Delta u, \Delta v$:

$$\frac{(u_0 - p_i)\Delta u + (v_0 - q_i)\Delta v}{\sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2}} \approx \rho_i - \sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2}$$

for $i = 1, 2, 3, 4$

Linearized equations

$$Ax \approx b$$

where $x = (\Delta u, \Delta v)$ and A is 4×2 with

$$\begin{aligned} b_i &= \rho_i - \sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2} \\ a_{i1} &= \frac{(u_0 - p_i)}{\sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2}} \\ a_{i2} &= \frac{(v_0 - q_i)}{\sqrt{(u_0 - p_i)^2 + (v_0 - q_i)^2}} \end{aligned}$$

- due to linearization and measurement error, we do not expect an exact solution ($Ax = b$)
- we can try to find Δu and Δv that 'almost' satisfy the equations

Numerical example

- beacons at positions $(10, 0)$, $(-10, 2)$, $(3, 9)$, $(10, 10)$
- measured distances $\rho = (8.22, 11.9, 7.08, 11.33)$
- (unknown) actual position is $(2, 2)$

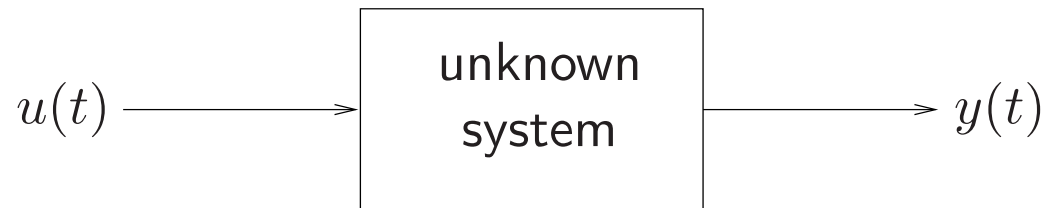
Linearized range equations (linearized around $(u_0, v_0) = (0, 0)$)

$$\begin{bmatrix} -1.00 & 0.00 \\ 0.98 & -0.20 \\ -0.32 & -0.95 \\ -0.71 & -0.71 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} \approx \begin{bmatrix} -1.77 \\ 1.72 \\ -2.41 \\ -2.81 \end{bmatrix}$$

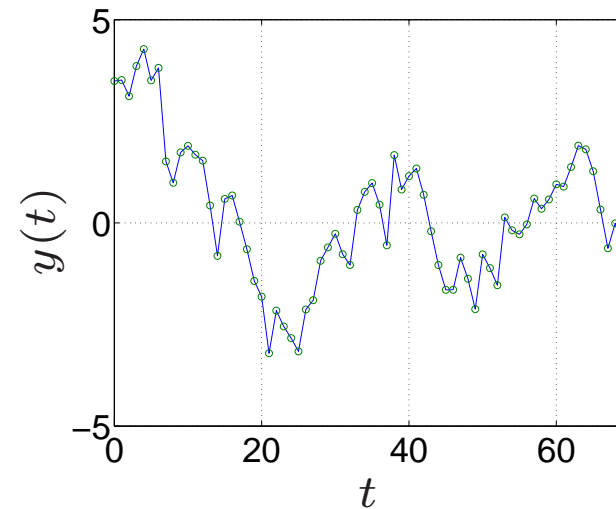
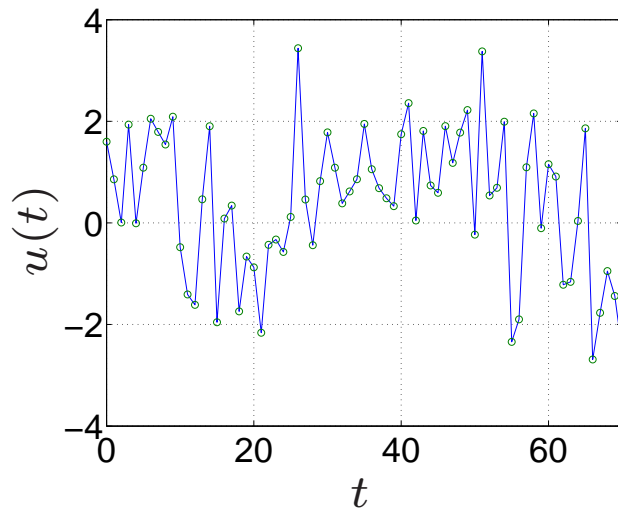
Least-squares solution: $(\Delta u, \Delta v) = (1.97, 1.90)$ (norm of error is 0.10)

Least-squares system identification

measure input $u(t)$ and output $y(t)$ for $t = 0, \dots, N$ of an unknown system



example ($N = 70$):



System identification problem: find reasonable model for system based on measured I/O data u, y

Moving Average model

$$y_{\text{model}}(t) = h_0u(t) + h_1u(t - 1) + h_2u(t - 2) + \cdots + h_nu(t - n)$$

where $y_{\text{model}}(t)$ is the model output

- a simple and widely used model
- predicted output is a linear combination of current and n previous inputs
- h_0, \dots, h_n are *parameters* of the model
- called a *moving average* (MA) model with n delays

Least-squares identification: choose the model that minimizes the error

$$E = \left(\sum_{t=n}^N (y_{\text{model}}(t) - y(t))^2 \right)^{1/2}$$

formulation as a linear least-squares problem:

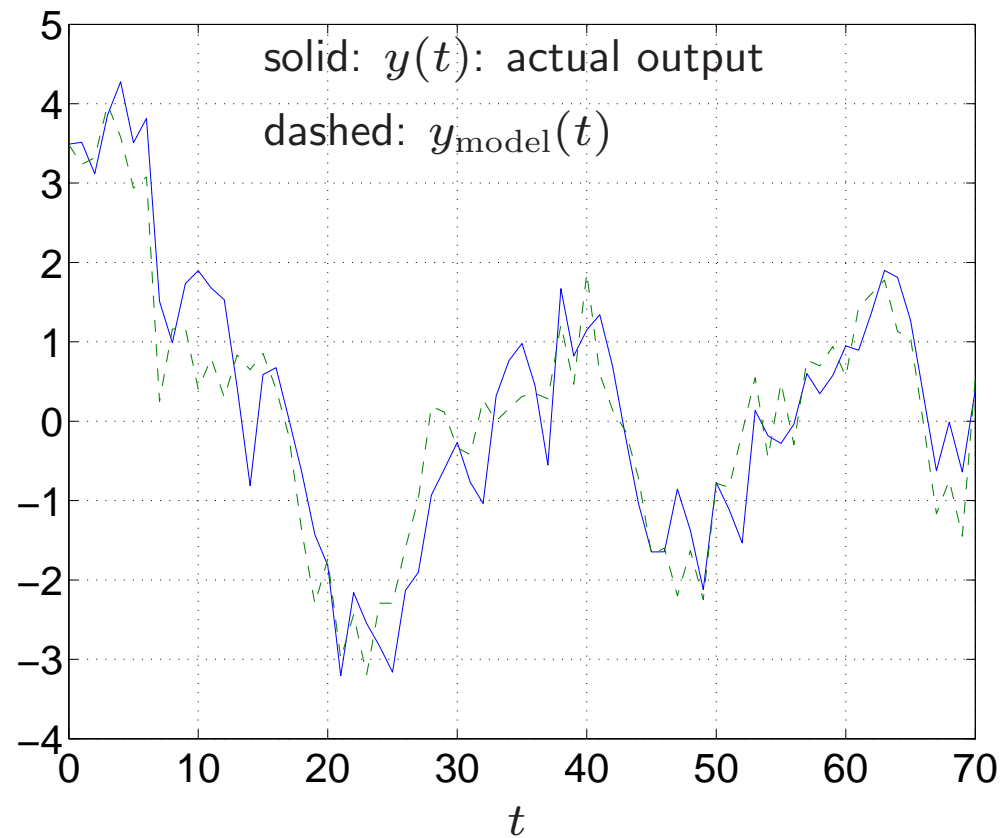
$$\begin{aligned}
 E &= \left(\sum_{t=n}^N (h_0 u(t) + h_1 u(t-1) + \cdots + h_n u(t-n) - y(t))^2 \right)^{1/2} \\
 &= \|Ax - b\|
 \end{aligned}$$

$$A = \begin{bmatrix}
 u(n) & u(n-1) & u(n-2) & \cdots & u(0) \\
 u(n+1) & u(n) & u(n-1) & \cdots & u(1) \\
 u(n+2) & u(n+1) & u(n) & \cdots & u(2) \\
 \vdots & \vdots & \vdots & & \vdots \\
 u(N) & u(N-1) & u(N-2) & \cdots & u(N-n)
 \end{bmatrix}$$

$$x = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix}, \quad b = \begin{bmatrix} y(n) \\ y(n+1) \\ y(n+2) \\ \vdots \\ y(N) \end{bmatrix}$$

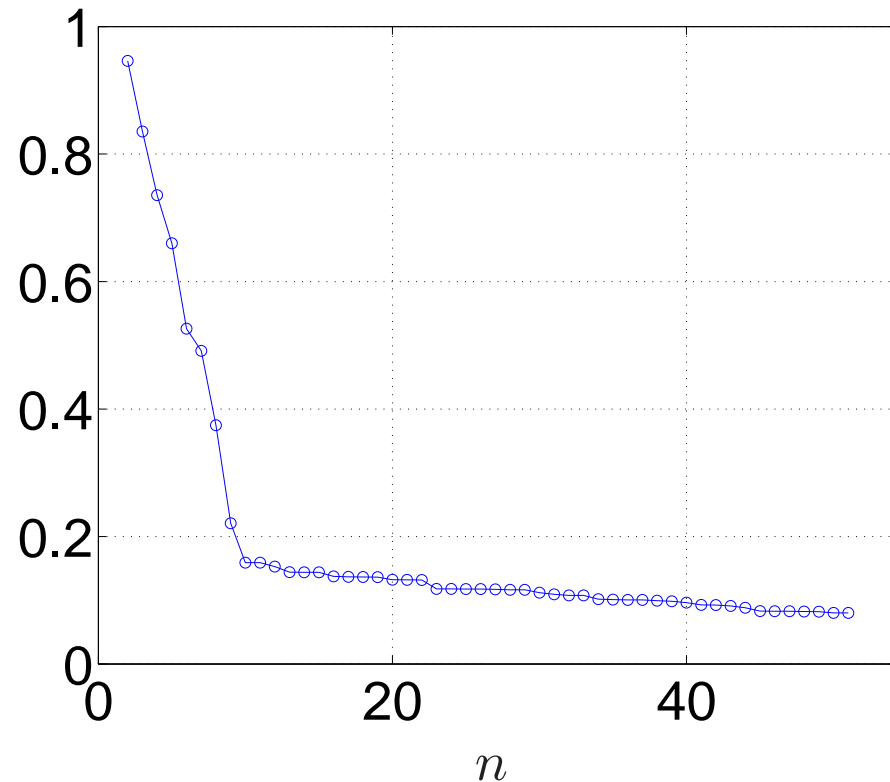
example (I/O data of page 8-15) with $n = 7$: least-squares solution is

$$\begin{array}{cccc} h_0 = 0.0240, & h_1 = 0.2819, & h_2 = 0.4176, & h_3 = 0.3536, \\ h_4 = 0.2425, & h_5 = 0.4873, & h_6 = 0.2084, & h_7 = 0.4412 \end{array}$$



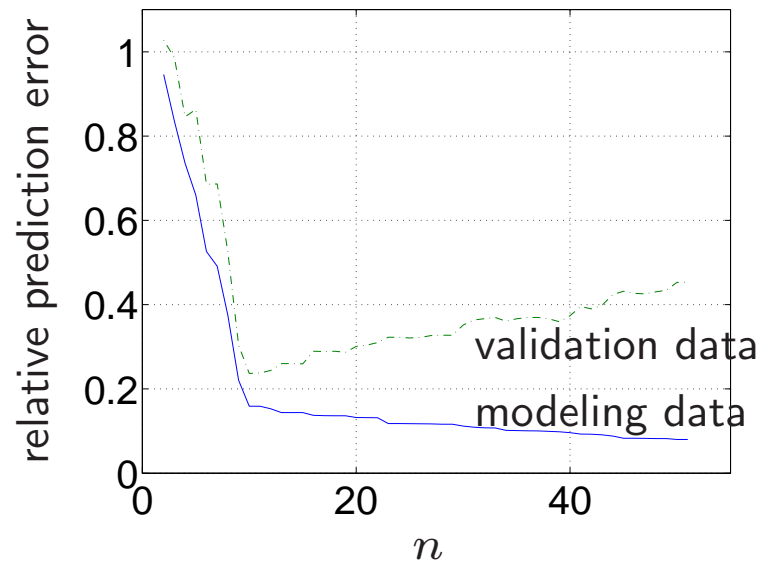
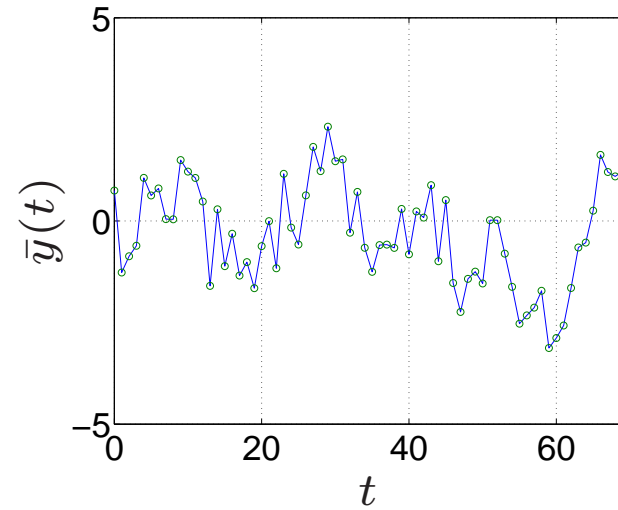
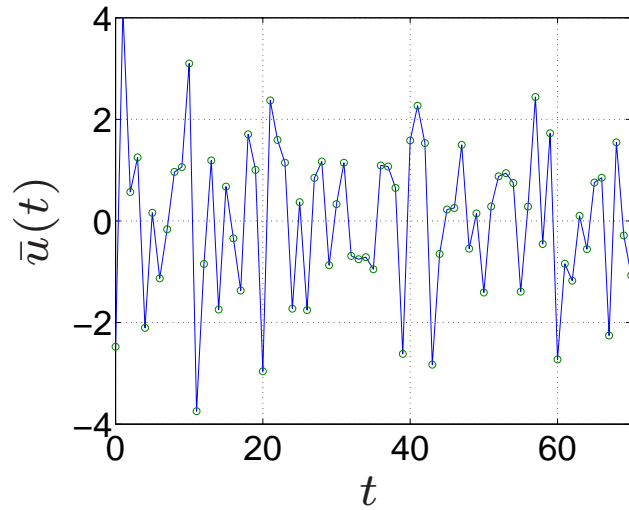
Model order selection: how large should n be?

relative error $E/\|y\|$



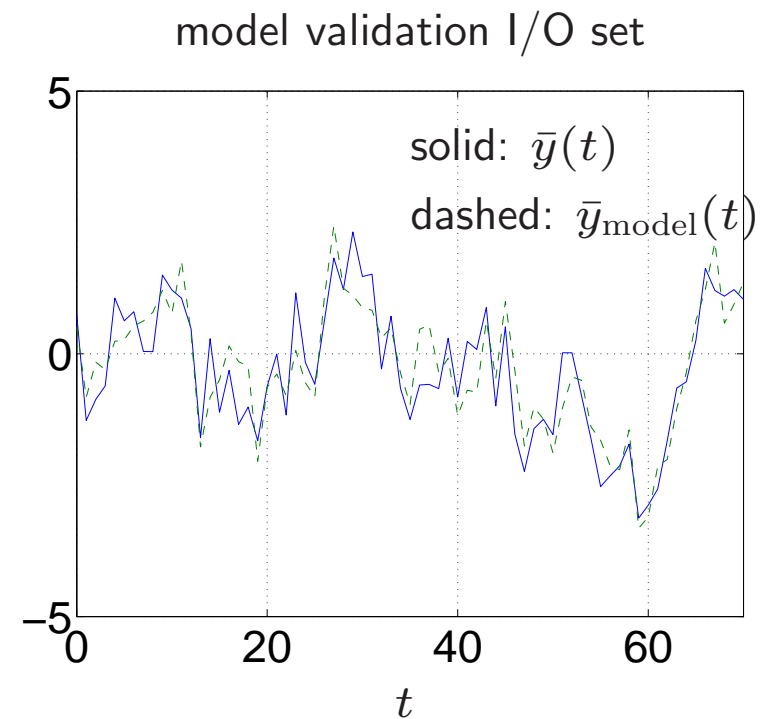
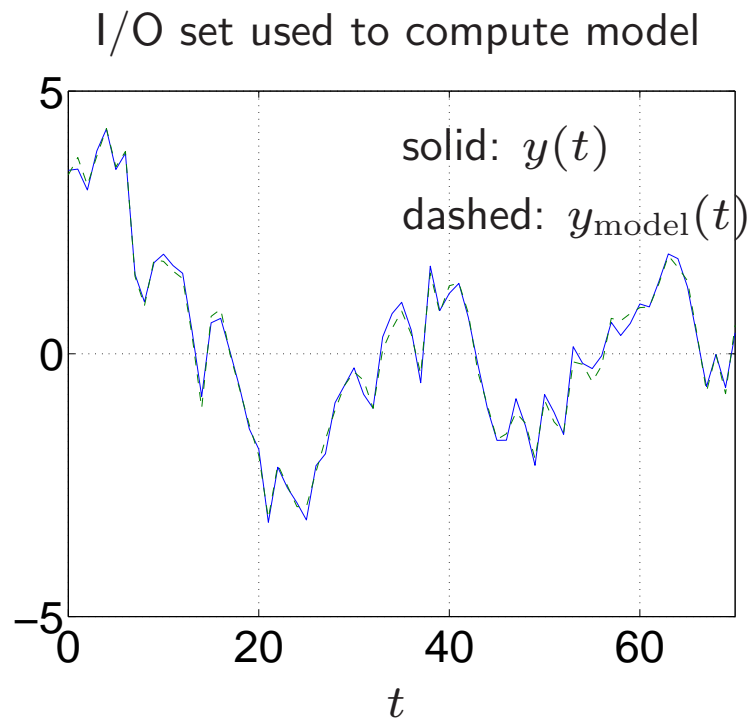
- suggests using largest possible n for smallest error
- a much more important question is: how good is the model at predicting *new data* (*i.e.*, not used to calculate the model)?

model **validation**: test model on a new data set (from the same system)



- for n too large the *predictive ability* of the model becomes worse!
- plot suggests $n = 10$

for $n = 50$ the actual and predicted outputs on system identification and model validation data are:



loss of predictive ability when n is too large is called *overfitting* or *overmodeling*