11. Subspace methods

- introduction
- geometric tools
- PO-MOESP algorithm
  1. input and output equation
  2. remove input and noise effects
  3. estimation of system matrices
- N4SID algorithm
- MATLAB examples

Introduction

consider a stochastic discrete-time linear system

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad y(t) = Cx(t) + Du(t) + v(t)$$

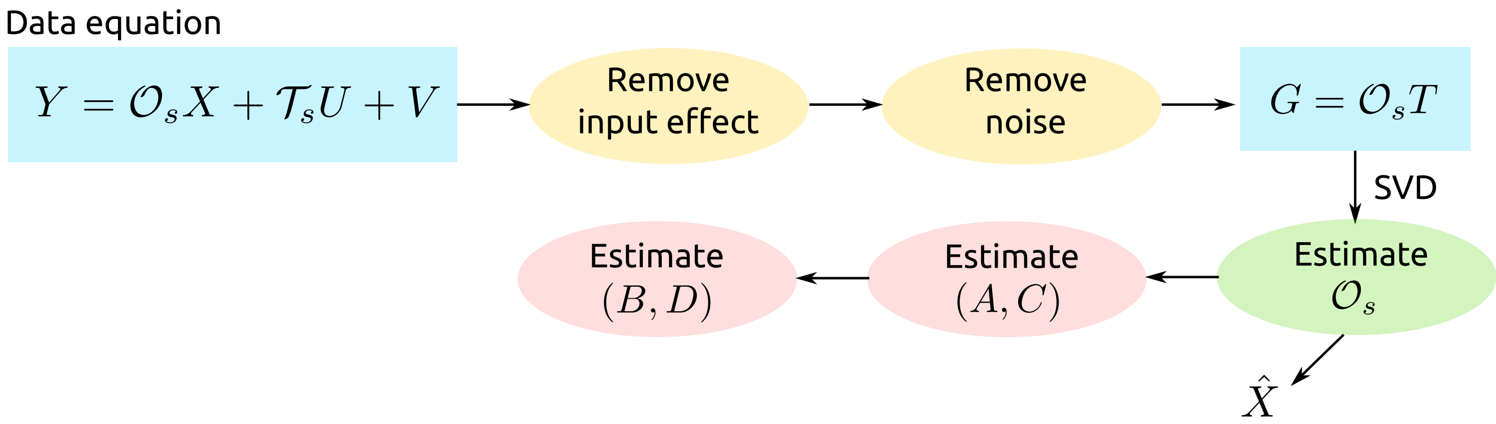
where $x \in \mathbf{R}^n, u \in \mathbf{R}^m, y \in \mathbf{R}^p$ and $\mathbf{E} \begin{bmatrix} w(t) \\ v(t) \end{bmatrix} \begin{bmatrix} w(t) \\ v(t) \end{bmatrix}^T = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \delta(t,s)$

problem statement: given input/output data $\{u(t), y(t)\}$ for $t = 0, \ldots, N$

- find an appropriate order n
- estimate the system matrices (A, B, C, D)
- estimate the noice covariances: Q, R, S

Overall scheme

the overall scheme has essential elements

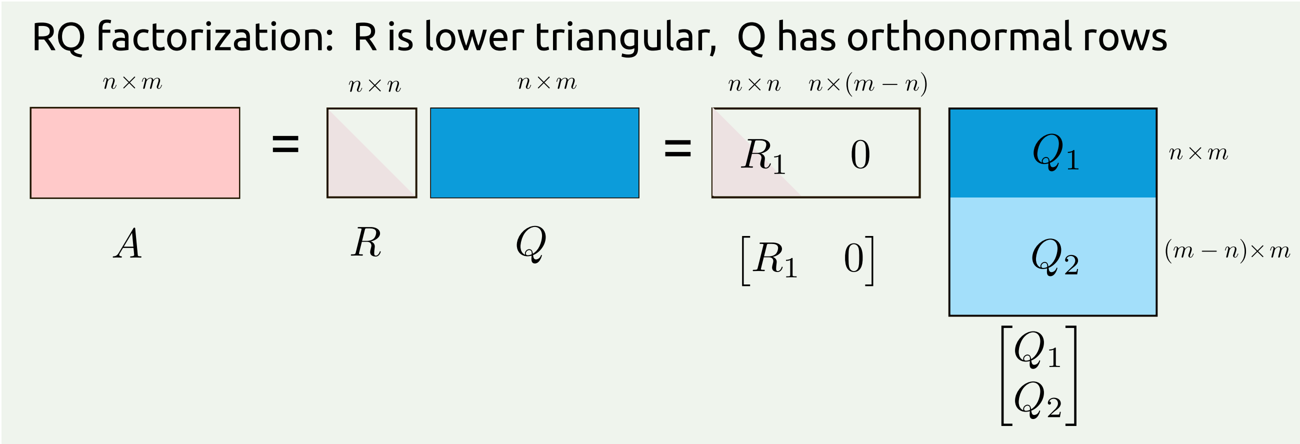


- extended observability matrix: $\mathcal{O}_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix} \in \mathbf{R}^{ps \times n}$

- input and output data equation

- geometrical tools (projection) to remove input and noise effects

RQ factorization



a fat matrix $A \in \mathbf{R}^{n \times m}$ has a RQ factorization: $A = RQ$

- $Q \in \mathbf{R}^{n \times m}$ has orthonormal rows ($Q_i Q_j^T = 0$ and $Q_i Q_i^T = I$)
- $R \in \mathbf{R}^{n \times n}$ is lower triangular with non-negative diagonals
- if $\mathbf{rank}(A) = n$ then the diagonals of R are all positive and

$$A = \begin{bmatrix} R_1 & 0 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = RQ \quad \text{(where } Q = Q_1 \text{ and } R = R_1\text{)}$$

Orthogonal projection

for $U \in \mathbf{R}^{r \times N}$, the orthogonal projection onto the row space of U $(\mathbf{row}(U))$ is

$$\Pi_u = U^T(UU^T)^{-1}U \qquad \text{(of size } N \times N\text{)}$$

- when Π_u is right-multiplied to a matrix M

$$M\Pi_u = MU^T(UU^T)^{-1}U$$

  is the matrix in $\mathbf{row}(U)$ that is closest to M (in Frobenius norm)

- the residual after projection is

$$M - MU^T(UU^T)^{-1}U = M(I - U^T(UU^T)^{-1}U)$$

- the matrix $\Pi_u^\perp = I - U^T(UU^T)^{-1}U$ is then called the orthogonal projection onto the orthogonal complement of $\mathbf{row}(U)$

- row space and column space are related by

$$\mathbf{row}(U)^{\perp} = \mathcal{R}(U^T)^{\perp} = \mathcal{R}(U)$$

- in other words, $\Pi_u^{\perp}$ is the orthogonal projection to the range space of U and

$$U\Pi_u^{\perp} = 0$$

(we will use this property to remove U from the data equation)

RQ factorization for least-squares

an RQ factorization relates to a least-squares problem of the form

$$\underset{\Theta}{\text{minimize}} \, \|Y - \Theta H\|_F^2 \quad \text{with solution} \quad \Theta_{\text{ls}} = YH^T(HH^T)^{-1}$$

this is to project Y to **row space** of H (since Θ is left-multiplied to H)

the LS solution can be obtained via RQ factorization

$$\begin{bmatrix} H \\ Y \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} R_{11}Q_1 \\ R_{21}Q_1 + R_{22}Q_2 \end{bmatrix}$$

define the orthogonal projection matrix (to row space of H)

$$\Pi_H = H^T(HH^T)^{-1}H \quad \text{and} \quad \Pi_H^{\perp} = I - H^T(HH^T)^{-1}H$$

we can verify that

$$
\begin{array}{rcll}
\Theta_{\mathrm{ls}} & = & YH^T(HH^T)^{-1} & = R_{21}R_{11}^{-1} \\
\Theta_{\mathrm{ls}}H & = & Y\Pi_H & = R_{21}Q_1 \\
Y - \Theta_{\mathrm{ls}}H & = & Y\Pi_H^{\perp} & = R_{22}Q_2
\end{array}
$$

- all quantities in LS problem can be computed from RQ factors
- $Y\Pi_H$ is the projection of Y onto the row space of H
- $Y\Pi_H^{\perp}$ is the residual after projection (which is the orthogonal complement of row space of H)
- the residual is the projection of Y onto the column space of H

Data equation

the state response to the system on page 11-2 is

$$x(t) = A^t x(0) + \sum_{\tau=0}^{t-1} A^{t-1-\tau}[Bu(\tau) + w(\tau)]$$

we can arrange y as a function of u and noise as

$$Y_{i,s,N} = \mathcal{O}_s X_{i,N} + \mathcal{T}_s U_{i,s,N} + V_{i,s,N} \tag{1}$$

where the matrices containing signals are

$$X_{i,N} = \begin{bmatrix} x(i) & x(i+1) & \cdots & x(i+N-1) \end{bmatrix} \tag{2}$$

$$Y_{i,s,N} = \begin{bmatrix} y(i) & y(i+1) & \cdots & y(i+N-1) \\ y(i+1) & y(i+2) & \cdots & y(i+N) \\ \vdots & \vdots & \ddots & \vdots \\ y(i+s-1) & y(i+s) & \cdots & y(i+N+s-2) \end{bmatrix} \tag{3}$$

($U_{i,s,N}$ has the same block Henkel structure, and $V_{i,s,N}$ contains w and v)

the matrices $\mathcal{O}_s$ and $\mathcal{T}_s$ contain system matrices

$$\mathcal{O}_s = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix}, \quad \mathcal{T}_s = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & & 0 \\ \vdots & & \ddots & \ddots & \\ CA^{s-2}B & CA^{r-3}B & \cdots & CB & D \end{bmatrix}$$

- from data equation, only $Y_{i,s,N}$ and $U_{i,s,N}$ are available
- if we can approximate $\mathcal{O}_s$, we can estimate (A, C) first
- the effects of $U_{i,s,N}$ and $V_{i,s,N}$ on $Y_{i,s,N}$ should be removed – this can be done using an orthogonal projection

Remove input from data equation

from the concept of orthogonal projection, define

$$\Pi_u^{\perp} = I - U_{i,s,N}^T (U_{i,s,N} U_{i,s,N}^T)^{-1} U_{i,s,N}$$

and right-mutiply to the data equation (1)

$$\begin{aligned} Y_{i,s,N}\Pi_u^{\perp} &= \mathcal{O}_s X_{i,N}\Pi_u^{\perp} + \mathcal{T}_s U_{i,s,N}\Pi_u^{\perp} + V_{i,s,N}\Pi_u^{\perp} \\ &= \mathcal{O}_s X_{i,N}\Pi_u^{\perp} + V_{i,s,N}\Pi_u^{\perp} \end{aligned}$$

(use the fact that $U_{i,s,N}\Pi_u^{\perp} = 0$)

Remove noise from data equation

definition: a matrix $Z_N \in \mathbf{R}^{sz \times N}$ with $n < s \ll N$ is said to be an **instrument variable** if it has the following properties

$$\begin{aligned} \lim_{N\to\infty} \frac{1}{N} V_{i,s,N} \Pi_u^{\perp} Z_N^T &= 0 \\ \mathbf{rank}\left(\lim_{N\to\infty} \frac{1}{N} X_{i,N} \Pi_u^{\perp} Z_N^T \right) &= n \end{aligned}$$

- Z_N should be uncorrelated with noise $V_{i,s,N}$
- Z_N should be correlated with state variables since $X_{i,N}\Pi^{\perp} Z_N^T$ still has full rank
- an example of instrument variable is the past input/output sequences

$$Z_N = \begin{bmatrix} U_{0,s,N} \\ Y_{0,s,N} \end{bmatrix} \triangleq \begin{bmatrix} Z_s(0) & Z_s(1) & \cdots & Z_s(N-1) \end{bmatrix}$$

from the data equation (after the input was removed), set $i = s$

$$
\begin{aligned}
Y_{i,s,N}\Pi_u^{\perp} &= \mathcal{O}_s X_{i,N}\Pi_u^{\perp} + V_{i,s,N}\Pi_u^{\perp} \\
\lim_{N\to\infty} \frac{1}{N} Y_{s,s,N}\Pi_u^{\perp} Z_N^T &= \lim_{N\to\infty} \frac{1}{N}\mathcal{O}_s X_{s,N}\Pi_u^{\perp} Z_N^T + \underbrace{\lim_{N\to\infty} \frac{1}{N} V_{i,s,N}\Pi_u^{\perp} Z_N^T}_{=0} \\
&= \mathcal{O}_s \underbrace{\lim_{N\to\infty} \frac{1}{N} X_{s,N}\Pi_u^{\perp} Z_N^T}_{\mathbf{rank}=n \ \text{ when } Z_N \text{ is IV}}
\end{aligned}
$$

this can be expressed as

$$G = \mathcal{O}_s T$$

where G and T are generally fat matrices (if N is large)

- G contain measured input/output data
- T is generally unknown, as it contains state variables
- generally, $\mathcal{R}(G) \subseteq \mathcal{R}(\mathcal{O}_s)$

Performing SVD to estimate $\mathcal{O}_s$

main equation:

$$\lim_{N\to\infty} \frac{1}{N} Y_{s,s,N} \Pi_u^\perp Z_N^T = \mathcal{O}_s \lim_{N\to\infty} \frac{1}{N} X_{s,N} \Pi_u^\perp Z_N^T \quad \triangleq \quad G = \mathcal{O}_s T$$

- generally, $\mathbf{rank}(AB) \leq \min(\mathbf{rank}(A), \mathbf{rank}(B))$ – Sylvester rank inequality
- if $\mathbf{rank}(G) = \mathbf{rank}(\mathcal{O}_s) = n$ we conclude that $\mathcal{R}(G) = \mathcal{R}(\mathcal{O}_s)$ and performing SVD on G gives

$$U_n \Sigma_n V_n^T = \mathcal{O}_s T \quad \Rightarrow \quad U_n = \mathcal{O}_s T V_n \Sigma_n^{-1} = \mathcal{O}_s \tilde{T} \qquad \triangleq \quad \tilde{\mathcal{O}}_s$$

  U_n relates to the extended observability matrix in another coordinate

- once $\tilde{O}_s$ is estimated, the system matrices (A, C) can be estimated

Performing SVD on RQ factor

it is more numerically efficient to compute SVD of RQ factor of G

- when G is fat with M columns, SVD matrix has size of $M \times M$ (not cheap to compute)
- finding the inverse of $\Pi_u^\perp$ of size $N \times N$ is also computationally expensive
- we should perform RQ factor of G before performing SVD
- if $G = RQ$ and $\mathbf{rank}(G) = n$ then the diagonals of R are positive and $\mathbf{rank}(G) = \mathbf{rank}(R)$
- consider RQ factorization for G (without the limit when $N \to \infty$)

$$\begin{bmatrix} U_{s,s,N} \\ Z_N \\ Y_{s,s,N} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \quad \Rightarrow \quad Y_{s,s,N} \Pi_u^\perp Z_N^T = R_{32} R_{22}^T$$

where $R_{32} \in \mathbf{R}^{sp \times sp}$ and $R_{22} \in \mathbf{R}^{sz \times sz}$ when using Z_N as on page 11-12

Theorem: [Verhegan, Thm 9.4] if u is persistently exciting of a certain order then

$$\mathbf{rank}\left(\lim_{N\to\infty}\frac{1}{N}Y_{s,s,N}\Pi_u^{\perp}Z_N^T\right) = \mathbf{rank}\left(\lim_{N\to\infty}\frac{1}{\sqrt{N}}R_{32}\right) = n$$

and that R_{22} is invertible; hence, with this result, we can conclude that

$$\mathcal{R}(\lim_{N\to\infty}\frac{1}{\sqrt{N}}R_{32}) = \mathcal{R}(\mathcal{O}_s)$$

- for finite N, we may not see exactly n nonzero SVD of R_{32}
- in practice, user gets to choose the model order, so when performing SVD

$$R_{32} = \begin{bmatrix} U_1 & U_2 \end{bmatrix}\begin{bmatrix}\Sigma_1 & 0 \\ 0 & \Sigma_2\end{bmatrix}\begin{bmatrix}V_1^T \\ V_2^T\end{bmatrix} \quad \Rightarrow \quad \Sigma_1 \text{ has size } \tilde{n}\times\tilde{n}, \text{ neglected } \Sigma_2$$

- with $R_{32} = U_1\Sigma_1V_1^T$ we have $U_1 = \mathcal{O}_sTR_{22}^{-T}V_1^{-T}\Sigma_1^{-1} \triangleq \tilde{\mathcal{O}}_s \in \mathbf{R}^{sp\times\tilde{n}}$

Estimation of A and C

using definition of $\mathcal{O}_s$ on page 11-3 that contains s blocks, each of size $p \times \tilde{n}$

- $\hat{C}$ is obtained by extracting the first row block of $\tilde{\mathcal{O}}_s$
- to get $\hat{A}$, notice that

$$\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-2} \\ \hline CA^{s-1} \end{bmatrix} \qquad \begin{bmatrix} C \\ \hline CA \\ \vdots \\ CA^{s-2} \\ CA^{s-1} \end{bmatrix}$$

(but the estimated $\tilde{\mathcal{O}}_s$ may not have the above structure exactly)

- estimate A by matching the top block on LHS with the bottom block on RHS in least-squares sense

$$\tilde{\mathcal{O}}_s(1 : p(s-1), :)\hat{A} = \tilde{\mathcal{O}}_s(p+1 : ps, :)$$

Estimation of B and D

consider the predicted output of state-space equation

$$\hat{y}(t) = C(qI - A)^{-1}Bu(t) + Du(t) = CA^t x(0) + C\sum_{\tau=0}^{t-1} A^{t-\tau-1}Bu(\tau) + Du(t)$$

when (A, C) is known, $\hat{y}(t)$ is **linear** in (B, D), so we can use LS

to do so, we re-arrange the equation by vectorizing B and D

$$\hat{y}(t) = CA^t x(0) + \left(\sum_{\tau=0}^{t-1} u(\tau)^T \otimes CA^{t-\tau-1}\right)\mathbf{vec}(B) + \big(u(t)^T \otimes I_p\big)\,\mathbf{vec}(D)$$

$$\triangleq H(t)\theta \quad \Rightarrow \quad \text{solve } \theta \text{ in least-squares sense}$$

all previously described procedures constitute the **Past Outputs Multivariable Output-Error State-Space** or PO-MOSEP method

N4SID algorithm

- innovation form of state-space and data equation
- estimation of state variables from input/output data
- performing SVD
- estimation of noise covariances and Kalman gain

Overall scheme of N4SID

from state-space equation, if $x(t), y(t), u(t)$ are known, we can use an LS problem:

$$\underset{A,B,C,D}{\text{minimize}} \left\| \begin{bmatrix} \hat{x}(t+1) \\ y(t) \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ u(t) \end{bmatrix} \right\|_F^2$$

to estimate (A, B, C, D)

- recall notation of $Y_{i,s,N}$ in (3) on page 11-9 (using i as a starting index)
- $(Y_{0,s,N}, U_{0,s,N})$ are past data and $(Y_{s,s,N}, U_{s,s,N})$ are future data
- $y(t)$ is a function of state x (which includes effect of past u) and the present u

$$Y_{s,s,N} = \text{Gain} \cdot \text{input} + \text{Gain} \cdot X_{0,s,N} + \text{Gain} \cdot X_{s,s,N} + \text{noise}$$

- effect of $X_{0,s,N}$ on $Y_{s,s,N}$ dies out if s is large, so we focus on the relation between $Y_{s,s,N}$ and $X_{s,s,N}$ to estimate the states

Data equation

we use state-space in innovation form (notation of x, X here are **estimated states**)

$$x(t+1) = (A - KC)x(t) + (B - KD)u(t) + Ky(t), \quad y(t) = Cx(t) + Du(t) + e(t)$$

- e is called *innovation* which has white noise properties; K is the *Kalman* gain
- using $A_K = (A - KC)$ and $B_K = B - KD$ we can write block Hankel $X_{s,N}$ as

$$\begin{aligned} X_{s,N} &= A_K^s X_{0,N} \\ &\quad + \begin{bmatrix} A_K^{s-1}B_K & A_K^{s-2} & B_K & \cdots & B_K & A_K^{s-1}K & A_K^{s-2}K & \cdots & K \end{bmatrix} \begin{bmatrix} U_{0,s,N} \\ Y_{0,s,N} \end{bmatrix} \\ &\triangleq A_K^s X_{0,N} + F_s Z_N \qquad \text{(future states are function of past input/output)} \end{aligned}$$

$$Y_{s,s,N} = \mathcal{O}_s F_s Z_N + \mathcal{T}_s U_{s,s,N} + S_s E_{s,s,N} + \mathcal{O}_s A_K^s X_{0,N} \qquad \text{(proof as exercise)}$$

future output is described by past input/output (Z_N), future output $(U_{s,s,N})$, noise, and initial states

Estimation of states

we aim to estimate $X_{s,N}$ from past input/output data

- consider a regression of $Y_{s,s,N}$ on $U_{s,s,N}$ and Z_N

$$\underset{(L_u,L_z)}{\text{minimize}} \left\| Y_{s,s,N} - \begin{bmatrix} L_u & L_z \end{bmatrix} \begin{bmatrix} U_{s,s,N} \\ Z_N \end{bmatrix} \right\|_2^2$$

- it can be shown that part of $Y_{s,s,N}$ that is explained by Z_N has a connection with $X_{s,N}$ (Verhegan Theorem 9.5)

$$Y_{s,s,N} = L_u \cdot U_{s,s,N} + L_z \cdot Z_N \quad \Rightarrow \quad \lim_{N\to\infty} L_z Z_N \approx \mathcal{O}_s F_s Z_N \triangleq \mathcal{O}_s \hat{X}_{s,N}$$

(using properties of innovation e, A_K; see more details in J. Songsiri book)

- onces L_z is estimated, we form $L_z Z_N = \mathcal{O}_s \hat{X}_{s,N}$

Performing SVD to estimate states

main equation: $L_z Z_N = \mathcal{O}_s \hat{X}_{s,N}$

- use RQ to solve LS problem and to find $L_z Z_N$

$$\left[\begin{array}{c} U_{s,s,N} \\ Z_N \\ \hline Y_{s,s,N} \end{array}\right] = \left[\begin{array}{cc|c} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ \hline R_{31} & R_{32} & R_{33} \end{array}\right] \left[\begin{array}{c} Q_1 \\ Q_2 \\ \hline Q_3 \end{array}\right]$$

$$L_z Z_N = (Y_{s,s,N} \Pi_u^{\perp} Z_N^T)(Z_N \Pi_u^{\perp} Z_N^T)^{-1} Z_N = R_{32} R_{22}^{-1} (R_{21} Q_1 + R_{22} Q_2)$$

- the expression of $L_z Z_N$ is called the **oblique projection** of future y along the future input onto past data in Overshee book
- in N4SID algorithm, use RQ factor of $L_z Z_N$ to perform SVD

$$L_z Z_N = U_n \Sigma_n^{1/2} \Sigma_n^{1/2} V_n^T = \mathcal{O}_s \hat{X}_{s,N} \quad \Rightarrow \quad \tilde{O}_s = U_n \Sigma_n^{1/2}, \;\; \hat{X}_{s,N} = \Sigma_n^{1/2} V_n^T$$

(now estimated x are obtained and hence, (A, B, C, D) is estimated using LS)

Noise covariance estimation

after we obtained $\hat{X}_{s,N}$ and $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$, compute residuals

$$\begin{bmatrix} \hat{W}_{s,1,N-1} \\ \hat{V}_{s,1,N-1} \end{bmatrix} = \begin{bmatrix} \hat{X}_{s+1,N} \\ Y_{s,1,N-1} \end{bmatrix} - \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} \begin{bmatrix} \hat{X}_{s,N-1} \\ U_{s,1,N-1} \end{bmatrix}$$

and the sample covariance of noises is

$$\begin{bmatrix} \hat{Q} & \hat{S} \\ \hat{S}^T & \hat{R} \end{bmatrix} = \lim_{N \to \infty} \frac{1}{N} \begin{bmatrix} \hat{W}_{s,1,N-1} \\ \hat{V}_{s,1,N-1} \end{bmatrix} \begin{bmatrix} \hat{W}_{s,1,N-1} \\ \hat{V}_{s,1,N-1} \end{bmatrix}^T$$

the Kalman gain for the innovation form can be obtained by solving Riccati equation

$$\begin{aligned} P &= APA^T + Q - (S + APC^T)(CPC^T + R)^{-1}(S + APC^T)^T \\ K &= (S + APC^T)(R + CPC^T)^{-1} \end{aligned}$$

(use estimated system matrices in Riccati equation)

Numerical examples

- model of DC motor
- choosing model order with `n4sid`
- mass-spring model

State-space equation of DC motor

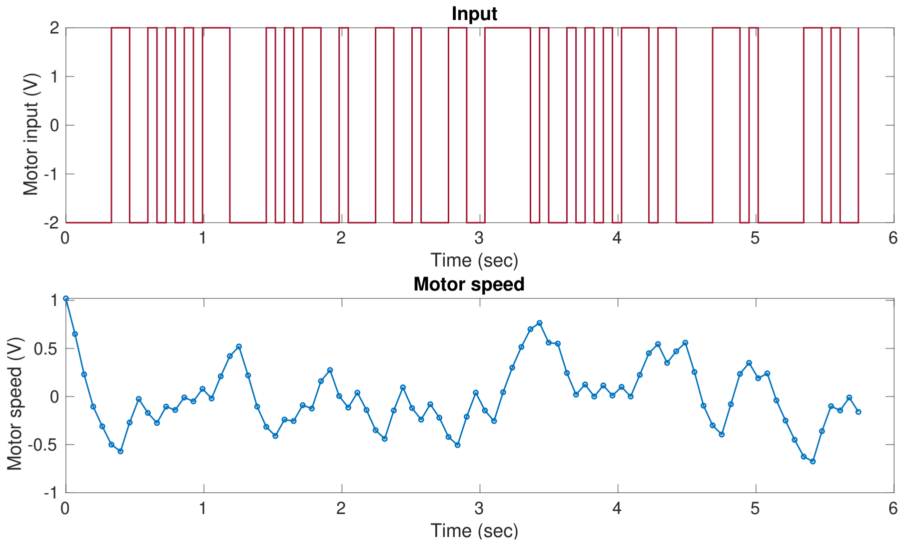
a discrete-time (ZOH) state-space representation of DC motor is

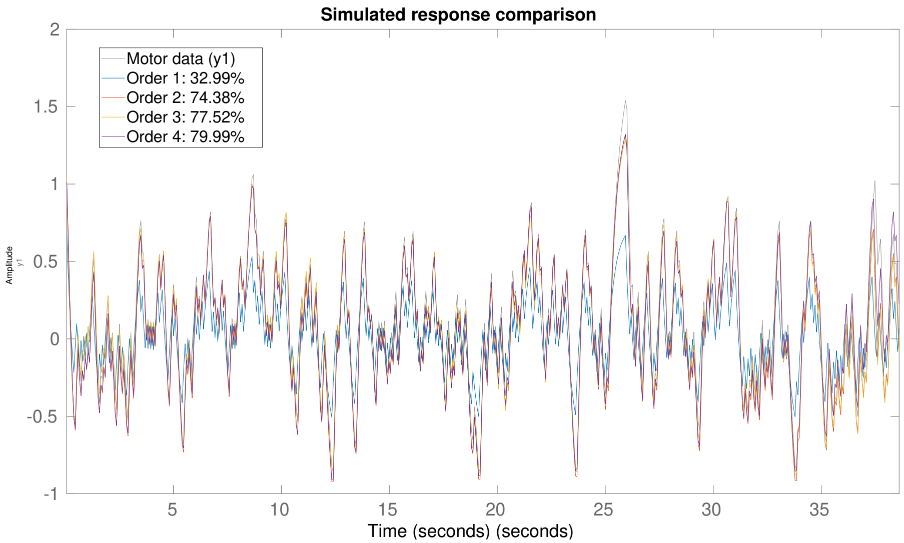
$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ b_1/\tau \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ b_2/\tau \end{bmatrix} \tau_l(t), \quad y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + v(t)$$

$$\begin{aligned} x(t+T) &= \begin{bmatrix} 1 & \tau(1 - e^{-T/\tau}) \\ 0 & e^{-T/\tau} \end{bmatrix} x(t) + \begin{bmatrix} b_1(\tau e^{-T/\tau} - \tau + T) \\ b_1/(1 - e^{-T/\tau}) \end{bmatrix} u(t) \\ &\quad + \begin{bmatrix} b_2(\tau e^{-T/\tau} - \tau + T) \\ b_2/(1 - e^{-T/\tau}) \end{bmatrix} \tau_l(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + v(t) \end{aligned}$$

- u is voltage input, y is motor angle
- $\tau_l(t)$ (or load torque) can be regarded as state noise and v is sensor noise
- the model is 2nd-order (by neglecting L in armature circuit)
- parameters τ, b_1, b_2 involves model parameters (J, R, K_a, K_v)

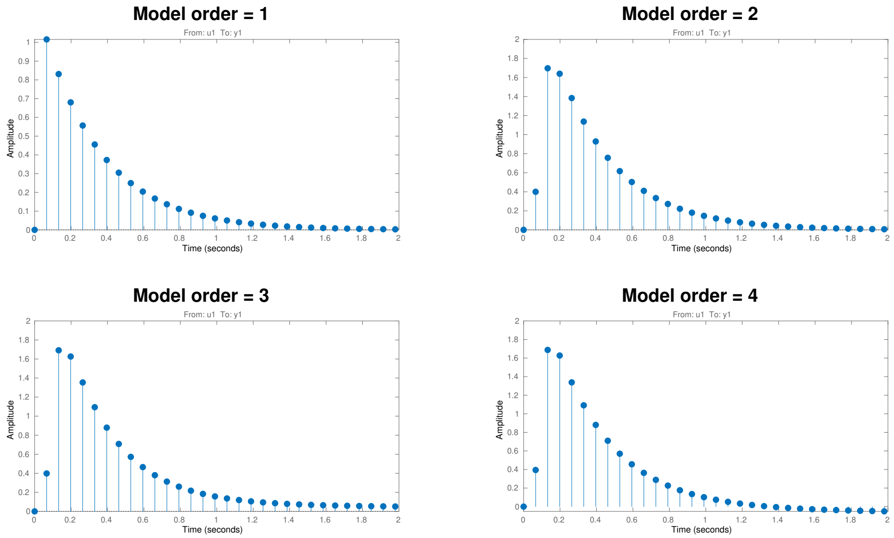
Results of fitting DC motor

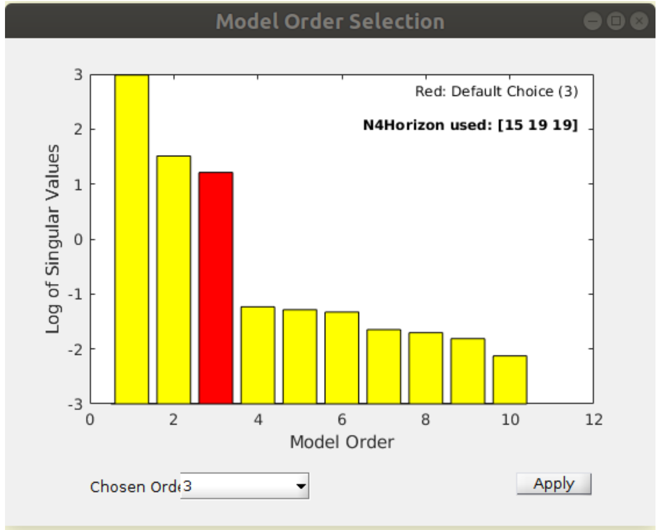




- u is a square pulse; the measured output y is the motor speed
- varying model order $n = 1, 2, \ldots, 10$
- Fit Percent values are not significantly different when $n \geq 2$

Model order selection in N4SID





- call `n4sid` to estimate the model of order $1, 2, \dots, 10$
- it suggests to pick the order that the log of SVD value (of the matrix $\mathcal{O}X$) is significant (indicating the rank of such matrix)
- the model equation suggests $n = 2$ (output is motor speed, including armature circuit in the model dynamics)
- impule response of first-order model is significantly different from the rest

example of MATLAB codes

```
load data-dcmotor-pulse

N = length(dat.y); Ts = dat.Ts; vect = (0:Ts: Ts*(N-1))' ;

ssmodel =cell(4,1); figure(1)
for k=1:4
    ssmodel{k} = n4sid(dat,k); ssmodel{k}.Name = ['Order ',num2str(k)];
end

figure(2); % compare fitting
compare(dat,ssmodel{1},ssmodel{2},ssmodel{3},ssmodel{4});

mss = n4sid(dat,[1:10],'InputDelay',0) ;
```

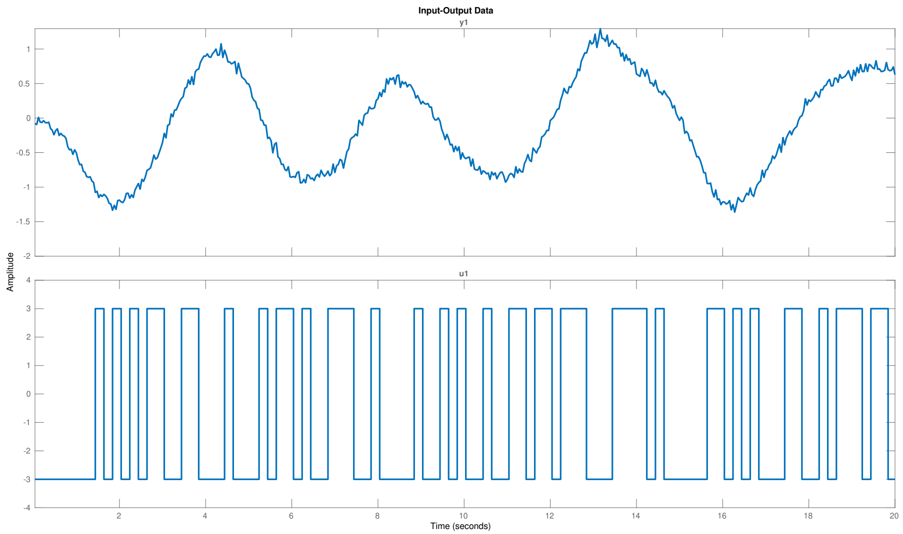
Mass-spring model

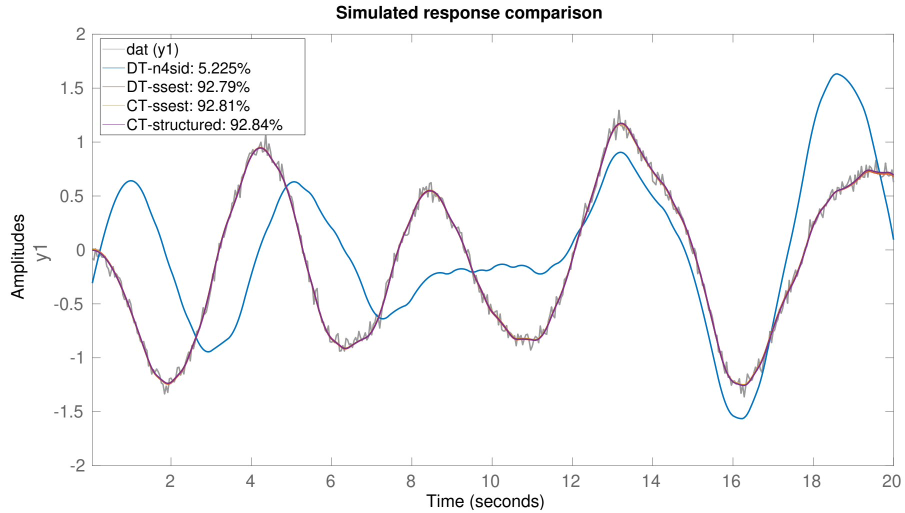
consider a mass-spring model with u as an applied force and y is the displacement

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

- CT model has a structure in A and B
- `ssest` initializes a model estimated by a subspace approach and then refines the parameter using PEM
- we can compare four models:
  1. DT-n4sid: DT model estimated by n4sid
  2. DT-ssest: DT model estimated by ssest (using DT-n4sid as the initial model)
  3. CT-ssest: CT model estimated by ssest directly
  4. CT-structured: CT model estimated ssest with the pre-defined structure

Results of fitting mass-spring system





- DT-n4sid has the lowest Fit Percent because the other models esimated by ssest (that refines the parameters using PEM for a better performance)
- the system matrices in CT-ssest are dense, while those of CT-structured model has the structure as desired
- pole locations of both CT-ssest and CT-structured are close – leading the models to have similar time responses

results:

```
Continuous-time transfer function.

trueTF =
          0.5
  ------------------
  s^2 + 0.25 s + 1.5

cTF =
   0.002356 s + 0.4979
  ----------------------
  s^2 + 0.2513 s + 1.505

fTF =
           0.5011
  ----------------------
  s^2 + 0.2519 s + 1.505

CT poles (True system, CT-ssest, CT-structured ) are

  -0.1250 + 1.2183i  -0.1256 + 1.2203i  -0.1259 + 1.2205i
  -0.1250 - 1.2183i  -0.1256 - 1.2203i  -0.1259 - 1.2205i
```

example of MATLAB codes:

```
% Estimation
load data-mass-spring dat

Ts = dat.Ts;
kk = 2; % model order
mn4 = n4sid(dat,kk); mn4.Name = 'DT-n4sid'; % DT model estimated by n4sid
mssest = ssest(dat,kk,'Ts',Ts); mssest.Name = 'DT-ssest'; % DT model estimated by ssest

csys = ssest(dat,kk); csys.Name = 'CT-ssest'; % CT model estimated by ssest

% Initialize a model structure
init_sys = idss([0 1;-1 -1],[0 1]',[1 0],0,[0 0]',[0 0]',0);
init_sys.Structure.A.Free = [false false; true true];
init_sys.Structure.B.Free = [false; true];
init_sys.Structure.C.Free = false;

% CT model where some structure is given
fsys = ssest(dat,init_sys); fsys.Name = 'CT-structured';
```

References

Chapter 9 in
M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least-square Approach*, Cambridge University Press, 2007.

Chapter 7 in
L. Ljung, *System Identification: Theory for the User*, 2nd edition, Prentice Hall, 1999

System Identification Toolbox demo
Building Structured and User-Defined Models Using System Identification Toolbox™

P. Van Overschee and B. De Moor, *Subspace Identification for Linear Systems*, KLUWER Academic Publishers, 1996

K. De Cock and B. De Moor, *Subspace identification methods*, 2003