17. Subspace methods

- Main idea
- Algorithm outline
- Numerical example

Introduction

- A method of estimating state-space models using LS techniques
- Any linear system can always be represented in state-space form as

$$x(t+1) = Ax(t) + Bu(t) + w(t)$$
$$y(t) = Cx(t) + Du(t) + \nu(t)$$

- Assume that x, y, u can be measured, we can form a linear regression:

$$Y(t) = \Theta H(t) + \eta(t)$$

to estimate Θ where

$$Y(t) = \begin{bmatrix} x(t+1) \\ y(t) \end{bmatrix}, \quad \Theta = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$
$$H(t) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}, \quad \eta(t) = \begin{bmatrix} w(t) \\ \nu(t) \end{bmatrix}$$

Main idea

- How to obtain the state vector sequence x ?
- Let a system be given by the impulse response representation

$$y(t) = \sum_{j=0}^{\infty} [g(j)u(t-j) + h(j)e(t-j)]$$

- Define the k-step ahead predictors by

$$\hat{y}(t|t-k) = \sum_{j=k}^{\infty} [g(j)u(t-j) + h(j)e(t-j)]$$

and define

$$\hat{Y}_r(t) = \begin{bmatrix} \hat{y}(t|t-1) \\ \vdots \\ \hat{y}(t+r-1|t-1) \end{bmatrix}, \quad \hat{\mathbf{Y}} = \begin{bmatrix} \hat{Y}_r(1) & \dots & \hat{Y}_r(N) \end{bmatrix}$$

Then the following is true for $N \to \infty$

1. The linear system has an nth order minimal state space description *iff*

$$\mathbf{rank}(\hat{\mathbf{Y}}) = n, \quad \forall r \geq n$$

2. The state vector of any minimal realization can be chosen as linear combinations of $\hat{Y}_r(t)$, *i.e.*,

$$x(t) = L\hat{Y}_r(t)$$

   where L is such that $L\hat{\mathbf{Y}}$ spans $\hat{\mathbf{Y}}$

The facts above allow us to find a state vector from the data

The only remaining problem is to estimate the k-step ahead predictors

Estimating the k-step ahead predictor

- Use the fact that the innovation $e(j)$ can be written as a linear combination of past input-output data
- The predictor is approximated so that it only depends on s_1 past outputs and the s_2 past inputs

$$\hat{y}(t+k-1|t-1) = \theta_k^T H_s(t) + \gamma_k^T U_l(t) + \varepsilon(t+k-1)$$

where

$$H_s(t) = \begin{bmatrix} y^T(t-1) & \dots & y^T(t-s_1) & u^T(t-1) & \dots & u^T(t-s_2) \end{bmatrix}^T$$
$$U_l(t) = \begin{bmatrix} u^T(t) & \dots & u^T(t+l-1) \end{bmatrix}$$

- s_1, s_2 are up to the user and l typically equals to r

- The term U_l should be included in the least-squares estimation since $u(t+1), \ldots u(t+k)$ affects $y(t+k-1)$

- However, by the definition of the k-step ahead predictor, the influence of U_l should be ignore

- The k-step predictors are then given by

$$\hat{Y}_r(t) = \hat{\Theta} H_s(t)$$

where we throw out the term U_l and $\Theta = \begin{bmatrix} \theta_1 & \ldots & \theta_r \end{bmatrix}^T$

Basic subspace algorithm

1. Choose s_1, s_2, r, l and form $\hat{Y}_r(t)$ and $\mathbf{Y}$

2. Estimate the rank n of $\mathbf{Y}$ and determine L so that $x(t)$ corresponds to a well-conditioned basis for it

3. Estimate A, B, C, D and the noise covariance matrices by applying the LS method

- This is called *subspace projection* approach to estimating the state-space model
- The approach is suited well for multivariable systems
- There are several variants of this algorithm to do step 3

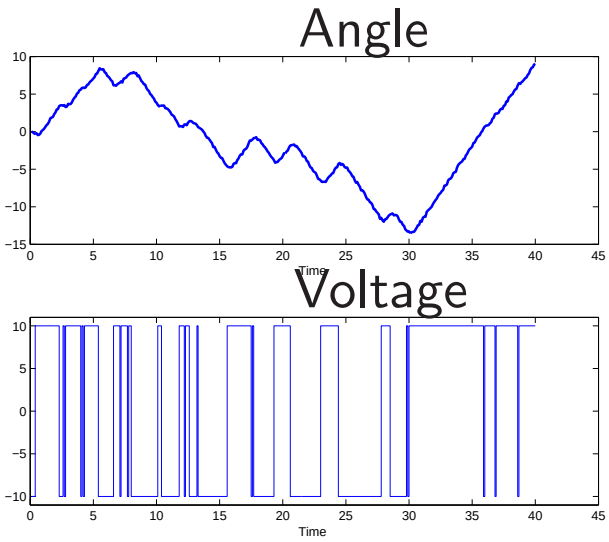
Example: DC motor

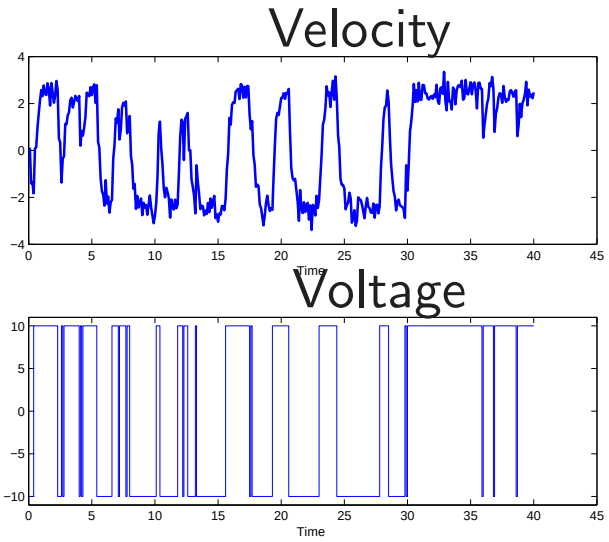
Time response of the second-order DC motor system

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 1/\tau \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \beta/\tau \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ \gamma/\tau \end{bmatrix} T_l(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

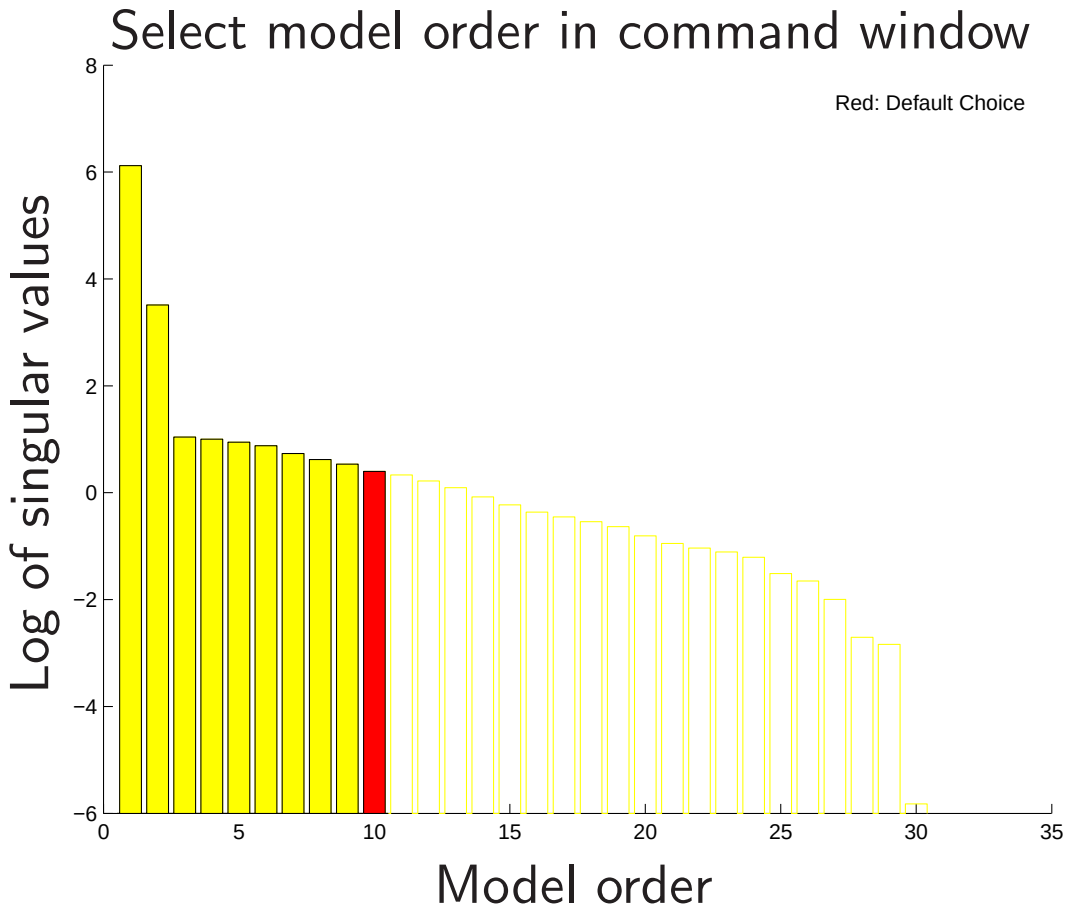
where τ, β, γ are parameters to be estimated





Use `n4sid` command in MATLAB

```
z = iddata(y,u,0.1);
m1 = n4sid(z,[1:10],'ssp','free','ts',0);
```



The software let the user choose the model order

Select $n = 2$ and the result from free parametrization is

$$A = \begin{bmatrix} 0.010476 & -0.056076 \\ 0.76664 & -4.0871 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0015657 \\ -0.040694 \end{bmatrix}$$
$$C = \begin{bmatrix} 116.37 & 4.6234 \\ 4.766 & -24.799 \end{bmatrix}, \quad D = 0$$

The structure of A, B, C, D matrices can be specified

```
As = [0 1; 0 NaN]; Bs = [0; NaN];
Cs = [1 0; 0 1]; Ds = [0; 0];
Ks = [0 0;0 0]; X0s = [0;0];
```

where `NaN` is free parameter and we assign this structure to `ms` model

```
A = [0 1; 0 -1]; B = [0; 0.28];
C = eye(2); D = zeros(2,1);
ms = idss(A,B,C,D); % nominal model (or initial guess)
setstruc(ms,As,Bs,Cs,Ds,Ks,X0s);
set(ms,'Ts',0);  % Continuous model
```

The structured parametrization can be used with `pem` command

```
m2 = pem(z,ms,'display','on');
```

The estimate now has a desired structure

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -4.0131 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1.0023 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = 0$$

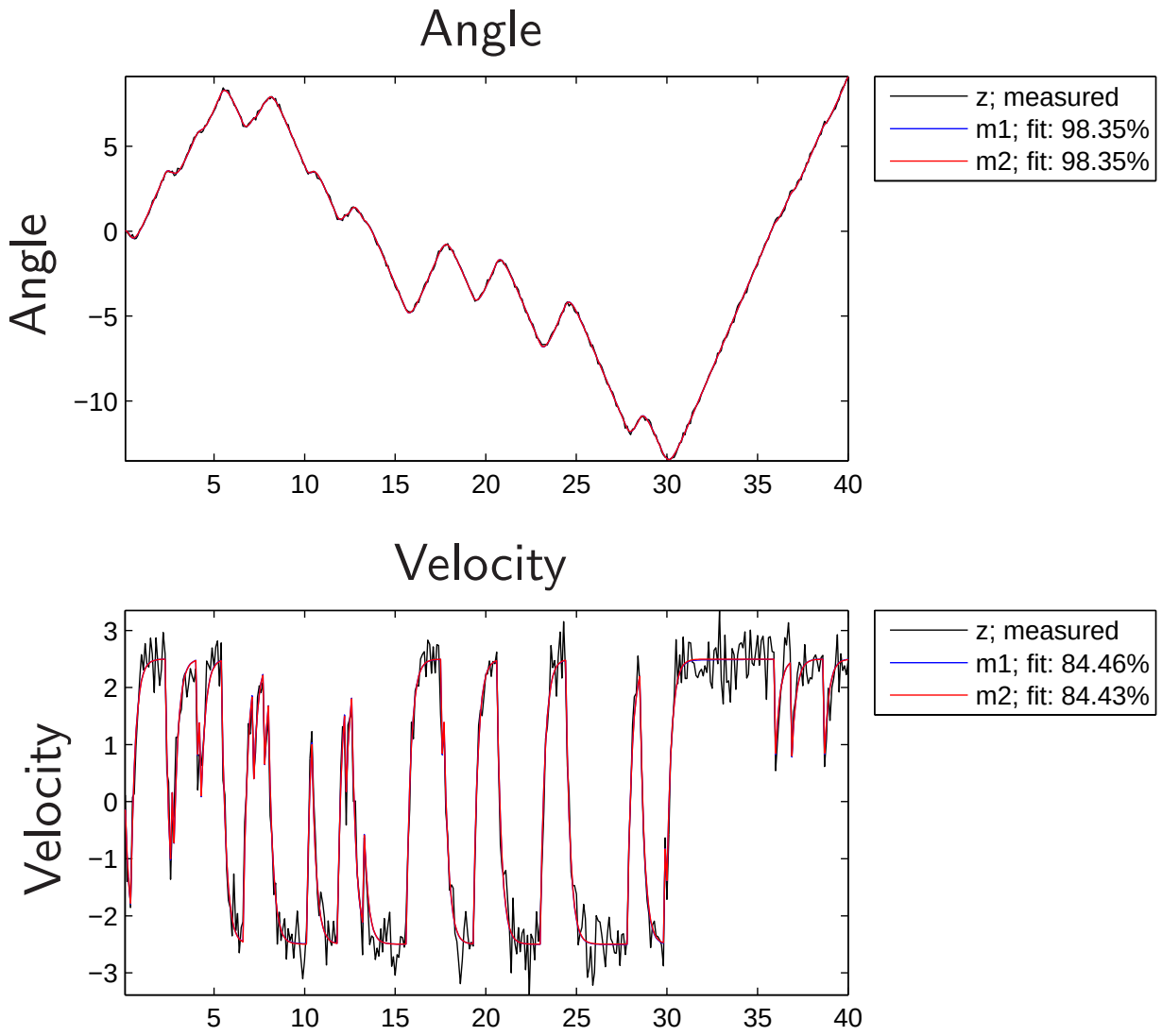
Choosing model order is included in `pem` command as well

```
m3 = pem(z,'nx',1:5,'ssp','free');
```

`pem` use the `n4sid` estimate as an initial guess

Compare the fitting from the two models

```
compare(z,m1,m2);
```



References

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