HUMBER ENGINEERING

MENG-3020
SYSTEMS MODELING & SIMULATION

LECTURE 10

HUMBER

WE ARE HUMBER

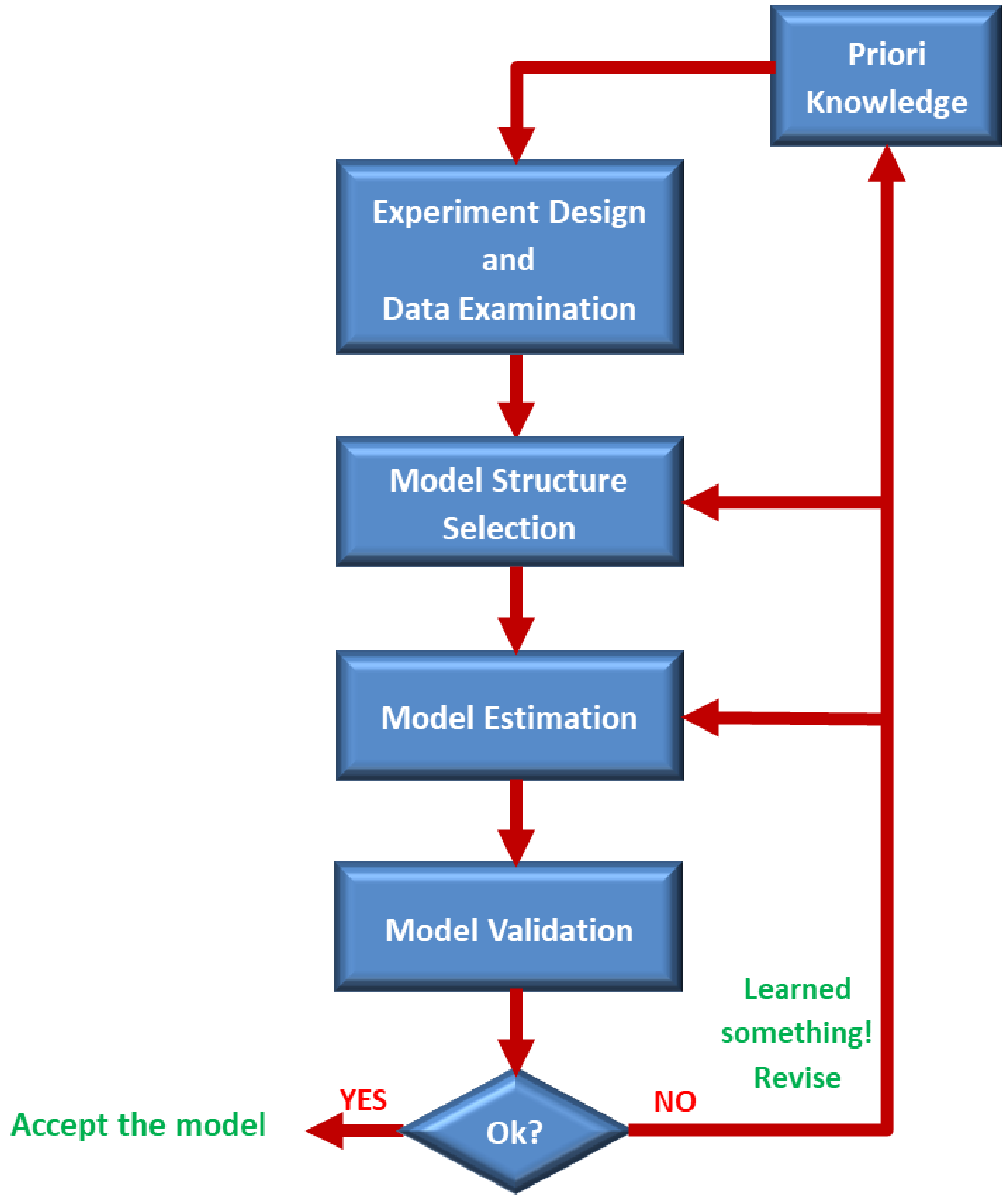
LECTURE 10
Model Structures & Estimation Methods

- System Identification Procedure
- Linear Regression Models & Least-Squares Estimation
    - Static Systems
    - Dynamic Systems
    - Discrete-time Transfer Function Models
- Determining Model Order and Delay
- Simulation Examples

System Identification Procedure

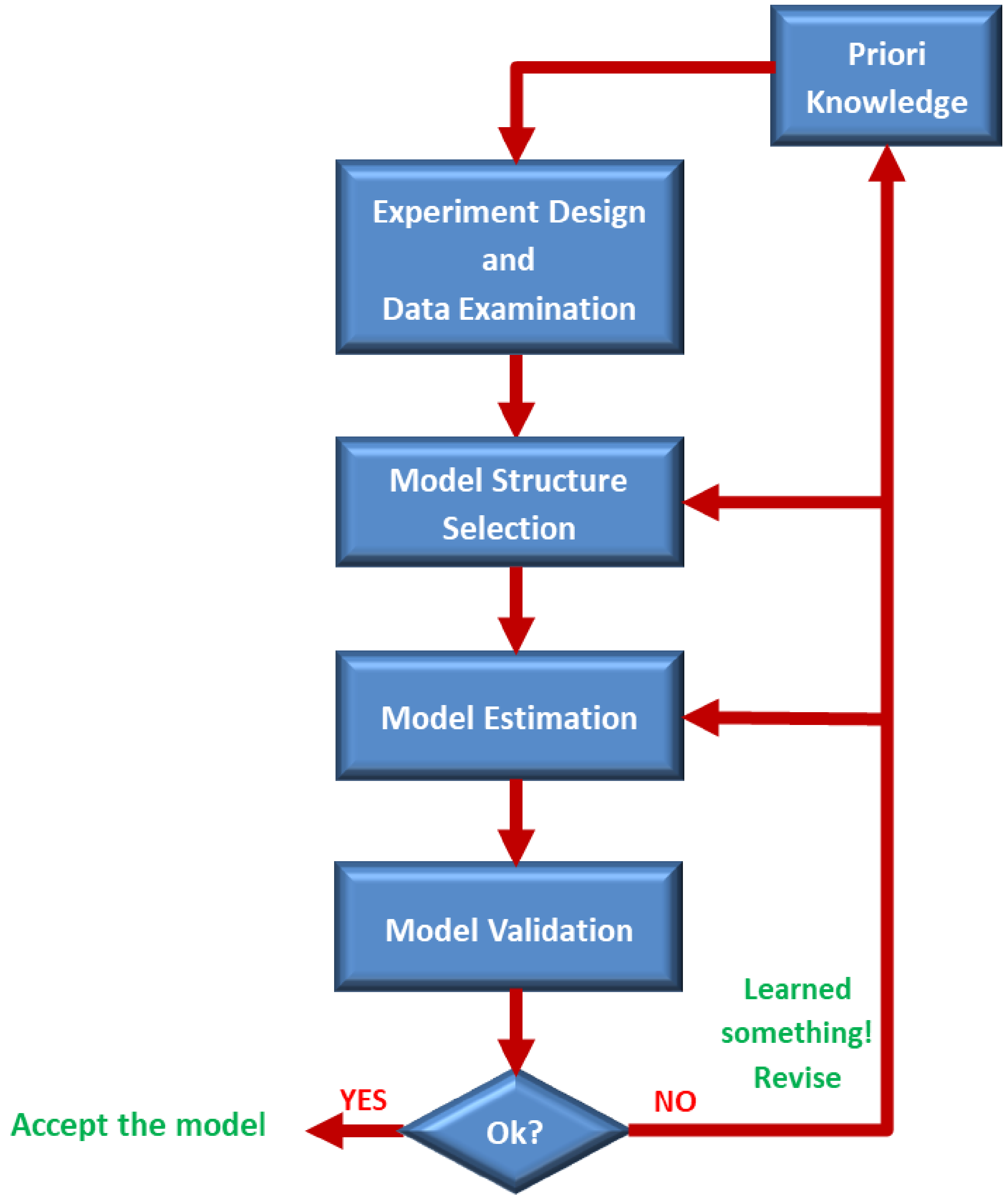
- **Prior Knowledge**
  - Purpose of Modeling
    - Control System Design
  - Grey-box Identification
    - Some part of the system is known
      - Model Order, Dominant Pole Locations, An Integrator, ….
  - Black-box Identification
    - No prior knowledge about the system

- **Experiment Design & Data Examination**
  - Choice of Input Signal and I/O Data Collection
  - I/O Data Examination
    - Aliasing, Outliers and Trends, Noise Filtering
  - Preliminary Diagnostic Experiments
    - Frequency Response Analysis
      - Bode Diagrams
    - Time Response Analysis
      - Impulse Response
      - Step Response



System Identification Procedure

- **Parametric Model Structures**
  - Continuous-Time Models
    - Transfer Function Model
    - State-space Model
  - Discrete-Time Models
    - ARX, OE, BJ
- **Model Estimation**
  - Parametric Models from the General I/O Data
    - Least Squares Method
      - Static Systems
      - Dynamic Systems
    - Order and Delay Estimation
- **Model Validation Techniques**
  - Simulation & Cross-Validation
  - Model Validity Criterion
  - Pole-Zero Plots
  - Residual Analysis



Linear Regression Modeling of Static Systems

- Assume that collected input-output data points of a static system are available.

$$\{u(k), y(k) |\ k = 1, 2, \cdots, N\}$$

- Linear regression model is the simplest type of a parametric model, which the model is linear with respect to its parameters. It can be used for curve fitting in the data points, by modeling the relation between the output variable and the regressors.

$$y(k) = \theta_0 + \theta_1 u(k) + \theta_2 u(k)^2 + \cdots + \theta_n u(k)^n + e(k)$$

where,

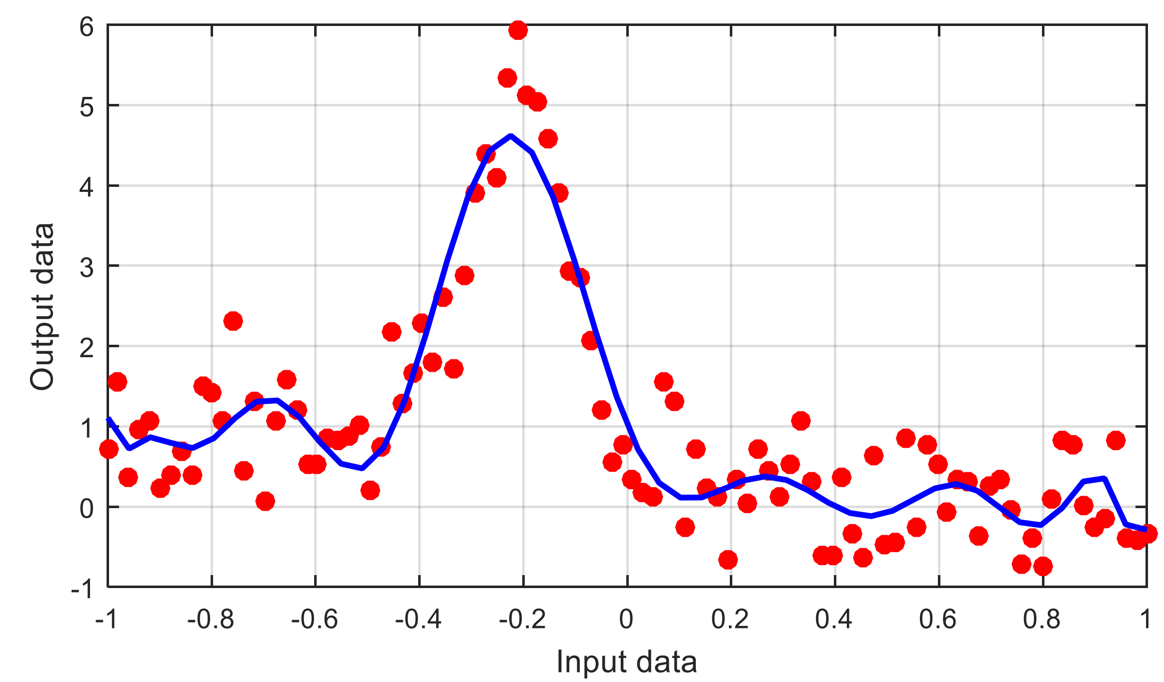
$y(k)$: the observed output data,

θ_i: unknown parameters,

$u(k)$: known quantities or regressors

$e(k)$: white noise (zero mean, variance σ_e^2, uncorrelated with regressors)

- In curve fitting procedure the goal is to find a Linear Regression Model with an appropriate order that provides the best fit to the data points by using Least-squares method.



Linear Regression Modeling of Static Systems

❑ Linear Regression Model Formulation

- The general form of the linear regression model for static systems is obtained as below

$$\boldsymbol{y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)}$$

- Based on the available dataset ($k = 1, 2, \cdots, N$) we have the following set of linear equations

$$
\begin{aligned}
y(1) &= \theta_n u(1)^n + \theta_{n-1} u(1)^{n-1} + \cdots + \theta_2 u(1)^2 + \theta_1 u(1) + \theta_0 + e(1) \\
y(2) &= \theta_n u(2)^n + \theta_{n-1} u(2)^{n-1} + \cdots + \theta_2 u(2)^2 + \theta_1 u(2) + \theta_0 + e(2) \\
&\vdots \\
y(N) &= \theta_n u(N)^n + \theta_{n-1} u(N)^{n-1} + \cdots + \theta_2 u(N)^2 + \theta_1 u(N) + \theta_0 + e(N)
\end{aligned}
$$

- We can represent these equations in matrix-vector form as below

$$
\underbrace{\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}}_{\text{Output observations } N \times 1} = \underbrace{\begin{bmatrix} u(1)^n & u(1)^{n-1} & \cdots & u(1) & 1 \\ u(2)^n & u(2)^{n-1} & \cdots & u(2) & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ u(N)^n & u(N)^{n-1} & \cdots & u(N) & 1 \end{bmatrix}}_{\text{Regressors matrix } N \times n} \underbrace{\begin{bmatrix} \theta_n \\ \theta_{n-1} \\ \vdots \\ \theta_1 \\ \theta_0 \end{bmatrix}}_{\text{Unknown parameters } n \times 1} + \underbrace{\begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(N) \end{bmatrix}}_{\text{White noise } N \times 1} \quad \Rightarrow \quad \boxed{\mathbf{Y} = \Phi \boldsymbol{\theta} + \boldsymbol{\epsilon}}
$$

Linear Regression Modeling of Static Systems

Least-Squares Estimation

- Assume the general matrix-vector form of the Linear Regression Model

$$\boxed{\mathbf{Y} = \Phi\boldsymbol{\theta} + \epsilon}$$

- The Normal Equation and Least-Squares Estimation are determined as

$$\textbf{Normal Equation} \rightarrow \quad \Phi^T\mathbf{Y} = \Phi^T\Phi\boldsymbol{\theta}$$

$$\textbf{Least} - \textbf{Squares Estimate} \rightarrow \quad \hat{\boldsymbol{\theta}} = (\Phi^T\Phi)^{-1}\,\Phi^T\mathbf{Y} = \Phi^{\#}\mathbf{Y}$$

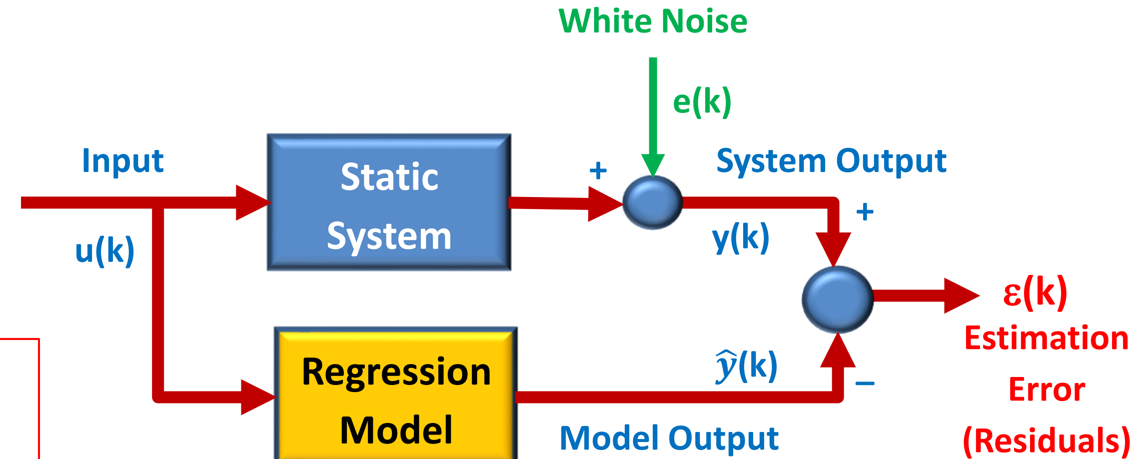
- The Least-squares estimation error (residuals) is defined as the difference between the observed output data points and the estimated output by the regression model.

$$\textbf{Least} - \textbf{Squares Estimation Error} \rightarrow \quad \boldsymbol{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}} = \mathbf{Y} - \Phi\hat{\boldsymbol{\theta}}$$

- The goal is to estimate the unknown parameters, $\boldsymbol{\theta}$, that minimizes the Mean Square of the Estimation Error.

- Therefore, the MSE loss function is defined as:

$$\boxed{\textbf{MSE Loss function} \quad \rightarrow \quad J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k;\boldsymbol{\theta}) = \frac{1}{N}\|\boldsymbol{\varepsilon}\|_2^2}$$



Linear Regression Modeling of Static Systems

❑ Statistical Properties of Least-Squares Estimation

- **THEOREM**: Consider the Least-Squares estimate of parameters as

$$\widehat{\boldsymbol{\theta}} = (\Phi^T\Phi)^{-1}\,\Phi^T\mathbf{Y}$$

- Assume that the data is generated from the following linear regression model, where θ^0 is the vector of true value of the parameters, $e(k)$ is a white noise with zero mean and variance σ_e^2.

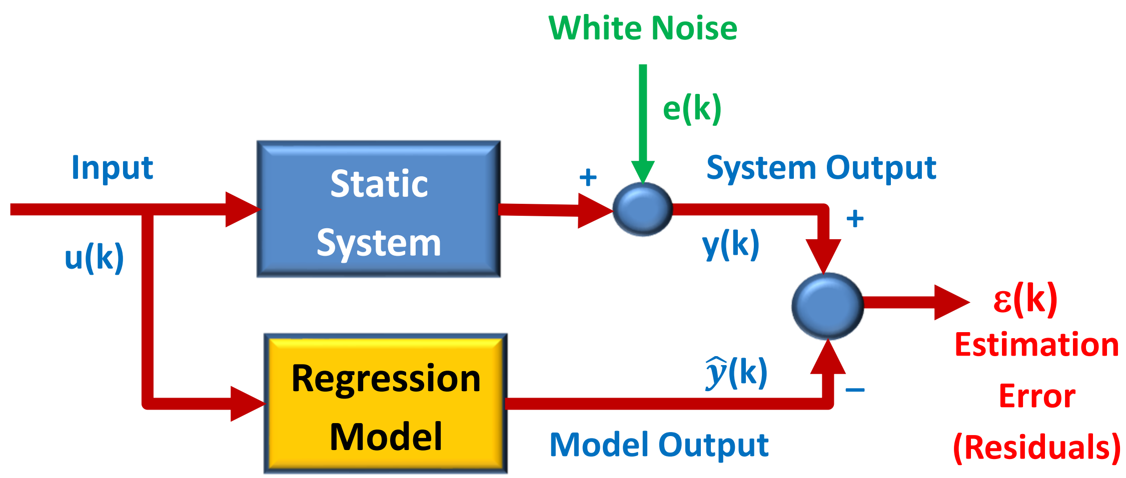
$$\boldsymbol{y}(\boldsymbol{k}) = \varphi^T(k)\theta^0 + \boldsymbol{e}(\boldsymbol{k})$$

- If $\Phi^T\Phi$ is nonsingular, then the estimates are **unbiased** and **consistent.**

- This means that the expected values of estimated parameters converges to the true optimal parameter values as the number of observations N increases toward infinity.

$$\boldsymbol{E}\left[\widehat{\boldsymbol{\theta}}(\boldsymbol{t})\right] = \boldsymbol{\theta}^{\mathbf{0}}$$

- The expected value is the **mean of the possible values a random variable can take.**



Linear Regression Modeling of Static Systems

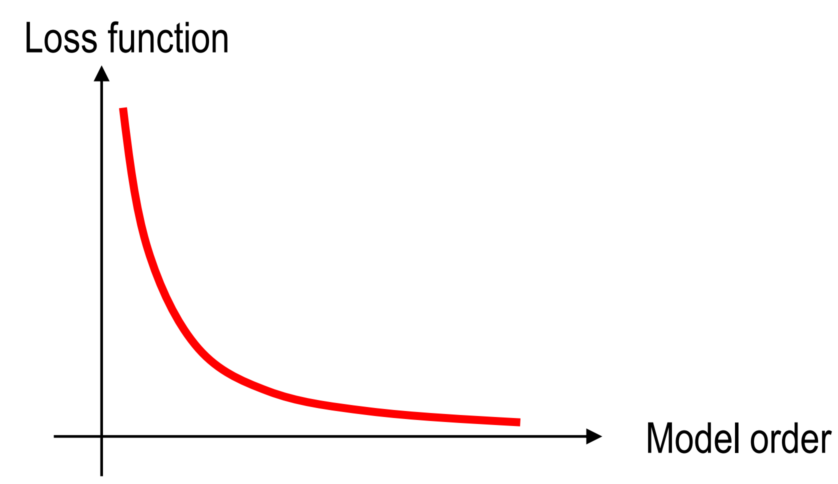
Model Order Selection

- In the Linear Regression Modeling the important part is appropriate selecting the model order (n) or the number of unknown parameters $\boldsymbol{\theta}$.
- Assume that the estimated models based on the collected N samples of data for different model orders are available:

$$
\begin{aligned}
n = 1 &\quad \rightarrow \quad \hat{y}(k) = \theta_1 u(k) + \theta_0 \\
n = 2 &\quad \rightarrow \quad \hat{y}(k) = \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 \\
n = 3 &\quad \rightarrow \quad \hat{y}(k) = \theta_3 u(k)^3 + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 \\
&\vdots
\end{aligned}
$$

- The best order can be determined by evaluating the loss function over a range of orders, for example $n = 1, 2, \cdots, 10$ and selecting the best order which minimizes the loss function.

Loss Function $\rightarrow$ $$\boldsymbol{J(\theta) = \frac{1}{N}\sum_{k=1}^{N} \varepsilon^2(k;\theta) = \frac{1}{N}\sum_{k=1}^{N}\big(y(k) - \hat{y}\,(k;\theta)\big)^2}$$

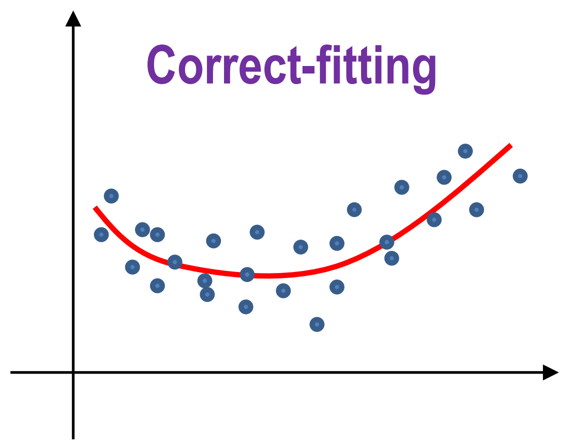


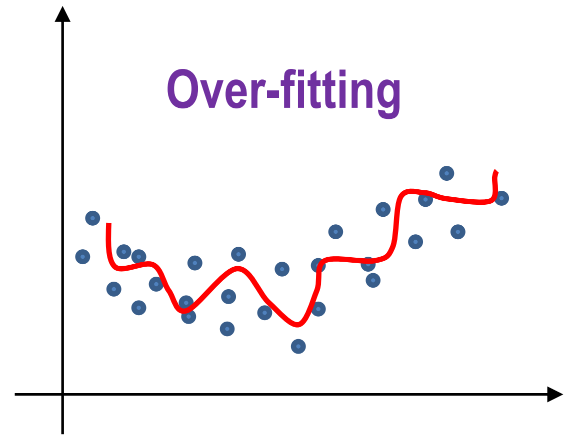
- In general, by increasing the order of model (number of parameters) the Least-Squares Estimation error and the loss function value will decrease monotonically.
- However, it does not mean to have a good estimation, because it makes the model more complex than we need and causes the Over-fitting to the data.

Linear Regression Modeling of Static Systems

❑ Model Order Selection

- Typically, when the model order be selected too complex, we actually models the noise dynamic not the true system. This issue is called **Over-fitting**.
- In the case of over-fitting if we validate the estimated model by the same identification data the result looks good, but if we use new fresh dataset for validation the results will be **poor**.
- In order to avoid the over-fitting the dataset (selecting high order model) we have to split the I/O dataset Z (*N* samples) in three datasets:
  - Identification dataset $\rightarrow$ Z_{id}
  - Validation dataset $\rightarrow$ Z_{val}
  - Order-selection dataset $\rightarrow$ Z_{os}





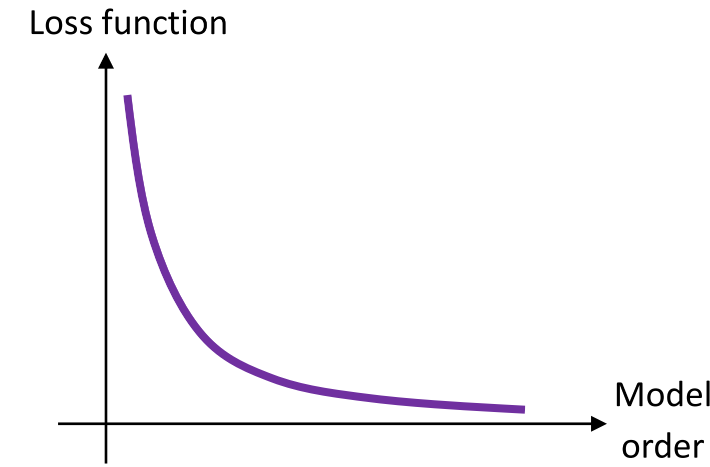
- Then by applying this technique we can determine the optimum order of the given data and compute the quality of fit of the estimated model by validating the model on a dataset where neither it was estimated.
- Note that if the number of available samples of data is not enough, we can consider only two sets of data (identification and validation) and determine the order based on the validation dataset.

Linear Regression Modeling of Static Systems

❑ Model Order Selection

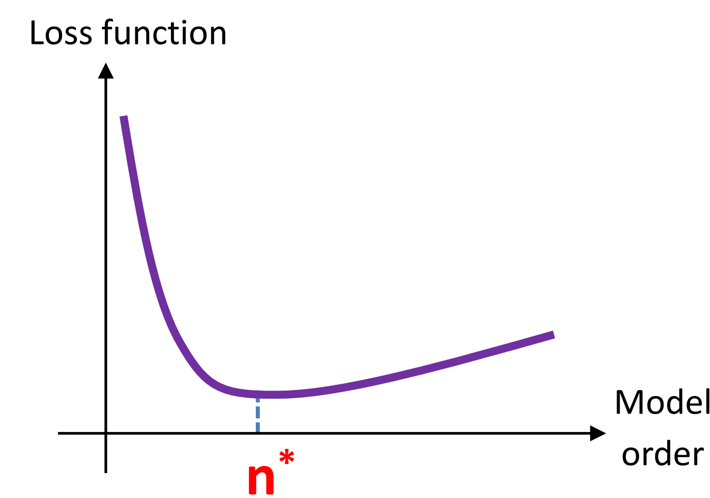
- If we evaluate the loss function using the identification data, by increasing the order of model the Least-squares estimation error and Loss function value decreases.

Loss function using ID data $\rightarrow \quad J(\theta, Z_{id}) = \dfrac{1}{N_{id}} \displaystyle\sum_{i=1}^{N_{id}} \left(\boldsymbol{y}(\boldsymbol{k}) - \hat{\boldsymbol{y}}\,(\boldsymbol{k}; \boldsymbol{\theta})\right)^{2}$



- On the order hand, the loss function has a different behavior when we apply it on the new fresh data. It starts to increase when the model complexity (order) becomes excessive (over-fitting data).

Loss function using new data $\rightarrow \quad J(\theta, Z_{val}) = \dfrac{1}{N_{val}} \displaystyle\sum_{i=1}^{N_{val}} \left(\boldsymbol{y}(\boldsymbol{k}) - \hat{\boldsymbol{y}}\,(\boldsymbol{k}; \boldsymbol{\theta})\right)^{2}$



- The optimum order (n^*) can be selected by minimizing the loss function using the new fresh dataset (a dataset other than the identification data) over a range of different orders (e.g. $n = 1, 2, \cdots, 10$) and selecting the best result for minimization.

Linear Regression Modeling of Static Systems

Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

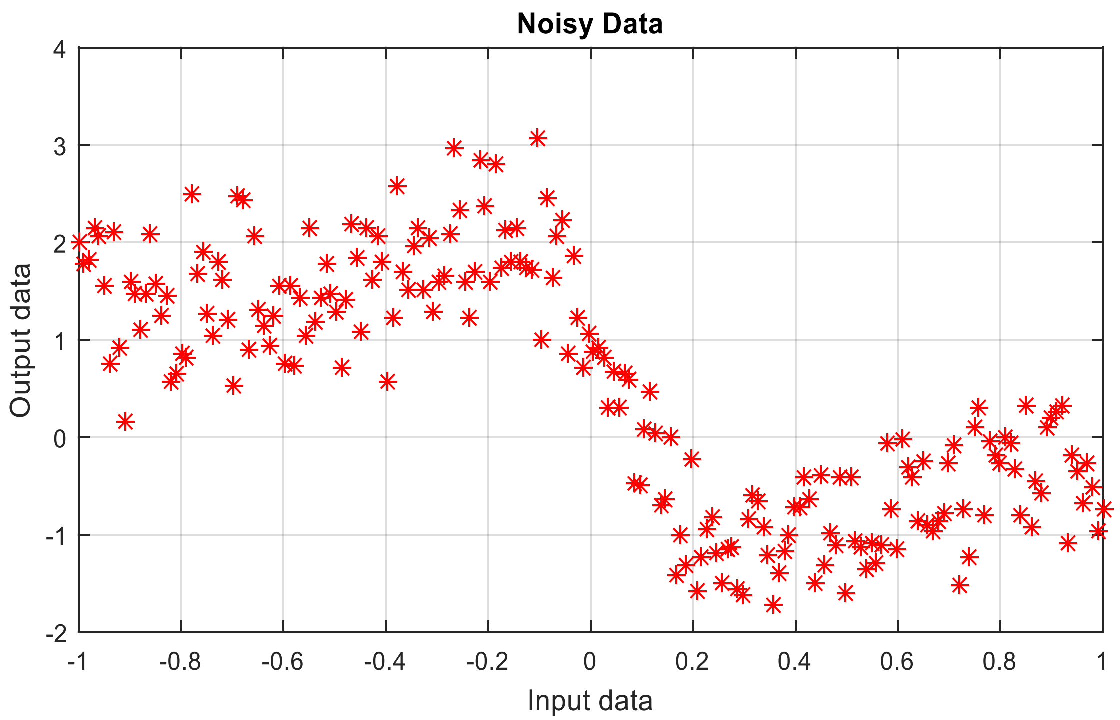
$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

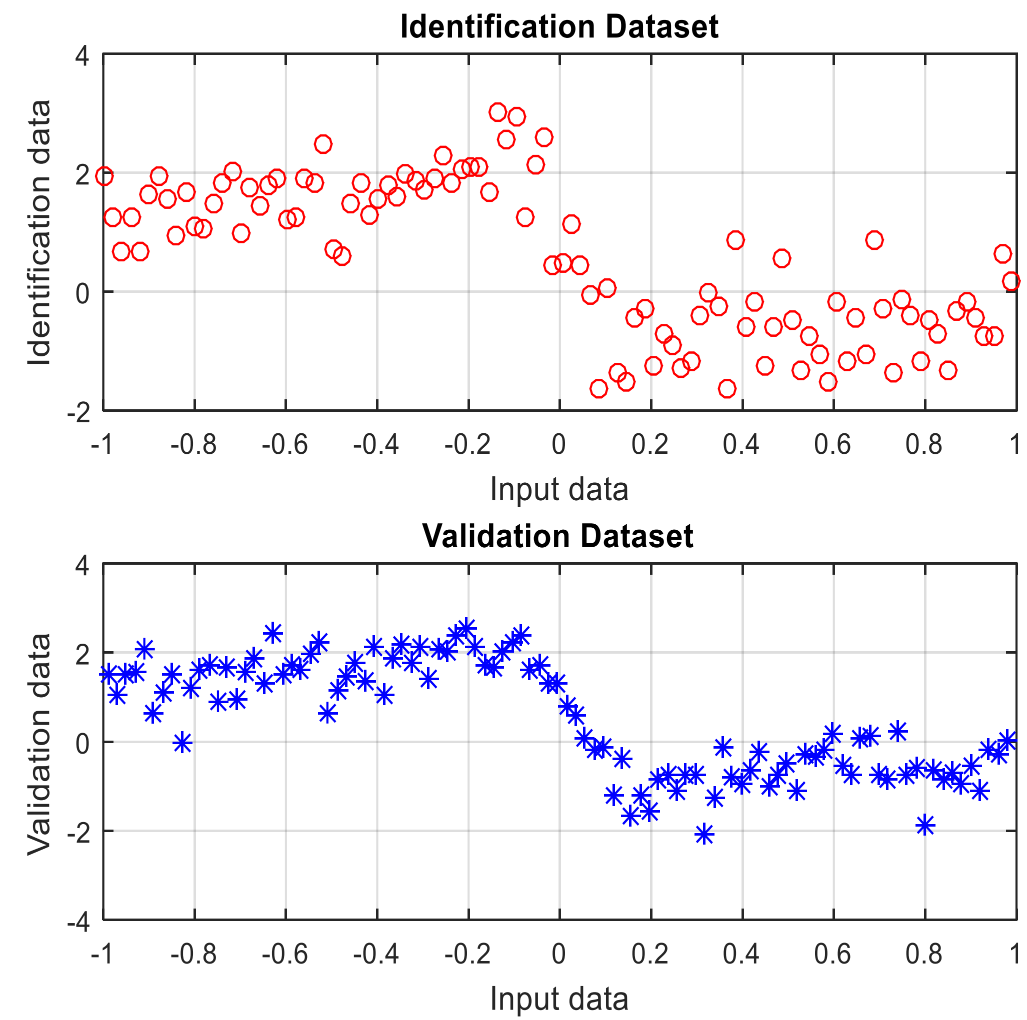
```
plot(u,y,'* r')    ← Plot I/O noisy data (u and y are column vectors)
```

First, split the data to identification and validation datasets.

```
plot(uid,yid,'o r')          ← Plot I/O identification data
plot(uval,yval,'* b')        ← Plot I/O validation data
```

$$N_{id} = 100\ , \qquad N_{val} = 100$$





Linear Regression Modeling of Static Systems

Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

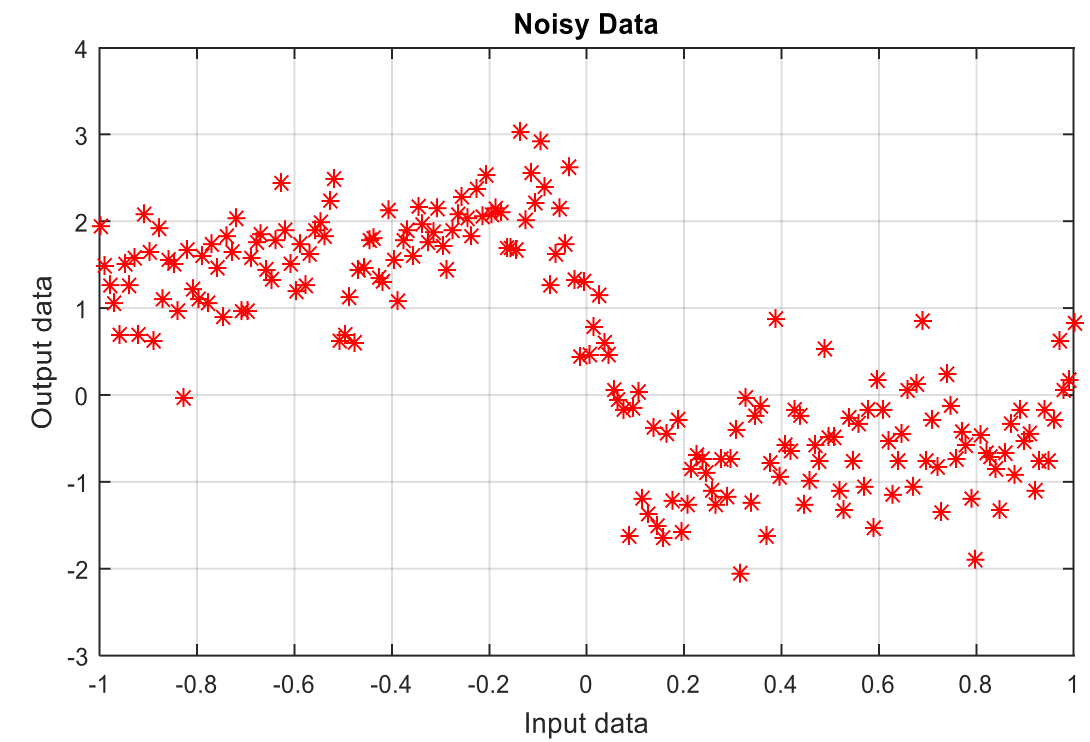
Next, we have to formulate the data in the Linear Regression form using the available samples of data:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N_{id}) \end{bmatrix} = \begin{bmatrix} u(1)^n & u(1)^{n-1} & \cdots & u(1) & 1 \\ u(2)^n & u(2)^{n-1} & \cdots & u(2) & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ u(N_{id})^n & u(N_{id})^{n-1} & \cdots & u(N_{id}) & 1 \end{bmatrix} \begin{bmatrix} \theta_n \\ \theta_{n-1} \\ \vdots \\ \theta_1 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(N_{id}) \end{bmatrix} \rightarrow \boxed{\mathbf{Y} = X\boldsymbol{\theta} + \boldsymbol{\epsilon}}$$

$$\boxed{\textbf{LS Estimate} \rightarrow \quad \hat{\boldsymbol{\theta}} = X^{\#}\mathbf{Y}}$$

We can use the following MATLAB code to formulate the linear regression model.

```
Yid = yid;                        ← Vector of output observations
Xid = zeros(Nid,n+1);             ← Initialize matrix of regressors
    for i=1:n+1
        Xid(:,i) = uid.^(n+1-i);  ← Generate matrix of regressors
    end
theta_hat = pinv(Xid)*Yid;        ← LS estimated parameters
```



Linear Regression Modeling of Static Systems

Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

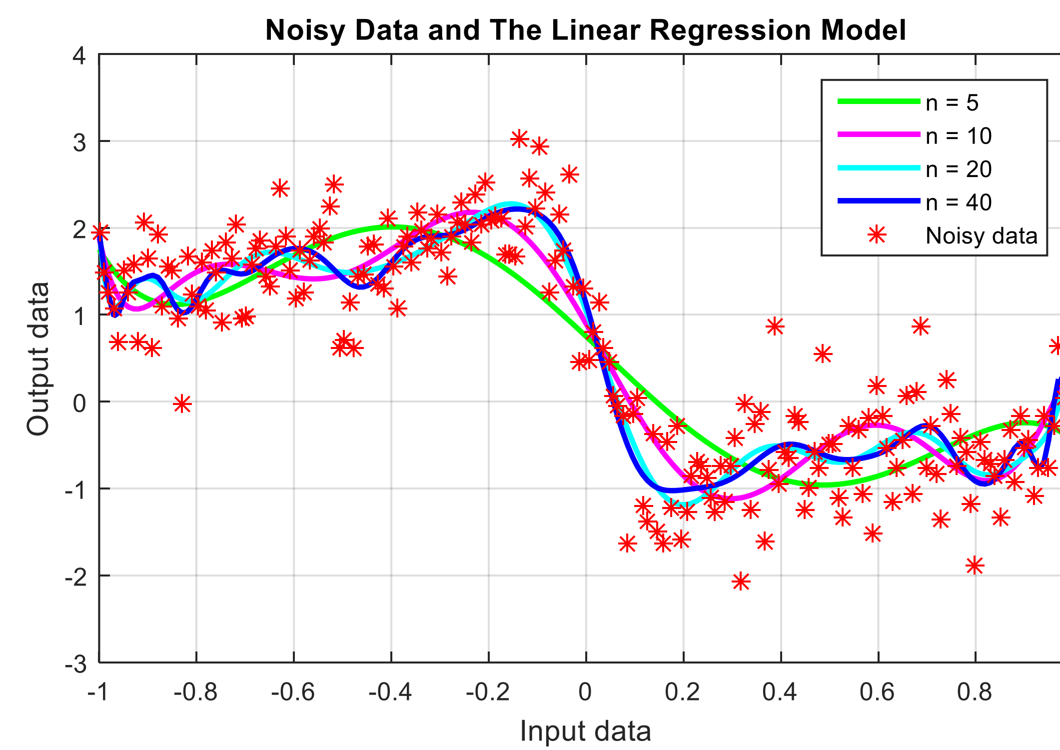
$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

Since, the model order (n) is unknown , we consider polynomials with different orders to fit the data:

$$\begin{aligned}
n &= 1 \quad \rightarrow \quad \hat{y}(k) = \theta_1 u(k) + \theta_0 \\
n &= 2 \quad \rightarrow \quad \hat{y}(k) = \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 \\
n &= 3 \quad \rightarrow \quad \hat{y}(k) = \theta_3 u(k)^3 + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 \\
&\vdots
\end{aligned}$$

Model estimation for $n = 5, 10, 20$ and 40 are shown:

```
Yid = yid;                             ← Vector of output observations
for n=[5 10 20 40];                    ← Model order
   Xid = zeros(Nid,n+1);               ← Initialize matrix of regressors
     for i=1:n+1
         Xid(:,i) = uid.^(n+1-i);      ← Generate matrix of regressors
     end
   theta_hat = pinv(Xid)*Yid;          ← LS estimated parameters
   Y_hat = zeros(size(Yid));           ← Initialize estimated output
   Y_hat = Xid*theta_hat;              ← Estimated y
   plot(u,Y_hat), hold on
end
```



$$\mathbf{Y} = X\boldsymbol{\theta} \quad \rightarrow \quad \hat{\boldsymbol{\theta}} = X^{\#}\mathbf{Y} \quad \rightarrow \quad \hat{Y} = X\hat{\boldsymbol{\theta}}$$

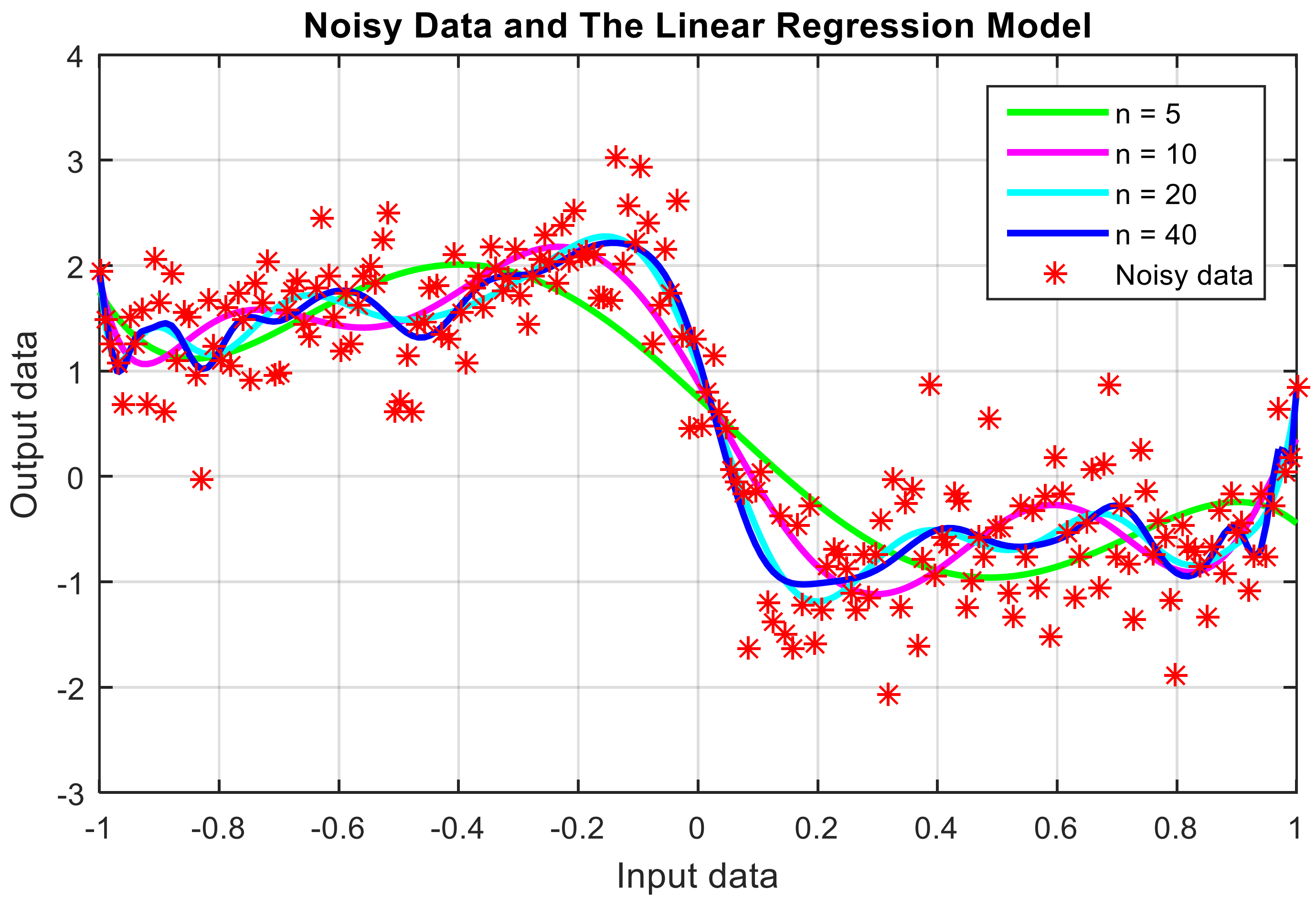
Linear Regression Modeling of Static Systems

Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

Model estimation for $n = 5, 10, 20$ and 40 are shown in the following figures.

- We can see that by increasing the order of polynomial the model starts **over-fitting** the data.
- It means that instead of modeling the true system, the model tries to model the noise.



HUMBER
Faculty of Applied Sciences & Technology

Linear Regression Modeling of Static Systems

Example 1

Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

The optimum order can be selected by minimizing the loss function using the validation dataset.

$$J(\theta, Z_{val}) = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \left(\boldsymbol{y}(\boldsymbol{k}) - \hat{\boldsymbol{y}}\,(\boldsymbol{k}; \boldsymbol{\theta})\right)^2$$

Loss function for VAL data

```
Yid = yid;  Yval = yval;            ← Vector of output observations
for n=1:20;                         ← Range of the model order
   Xid = zeros(Nid,n+1);            ← Initialize matrix of regressors for ID data
     for i=1:n+1
         Xid(:,i) = uid.^(n+1-i);   ← Generate matrix of regressors for ID data
     end
   theta_hat = pinv(Xid)*Yid;       ← LS estimated parameters using ID data

   Xval = zeros(Nval,n+1);          ← Initialize matrix of regressors for VAL data
     for i=1:n+1
         Xval(:,i) = uval.^(n+1-i); ← Generate matrix of regressors for VAL data
     end
   Y_hat = zeros(size(yval));       ← Initialize estimated output
   Y_hat = Xval*theta_hat;          ← Estimated y

   J(n) = immse(yval,Y_hat);        ← MSE Loss function
end
plot(J,'o- r')
```

$$\mathbf{Y}_{id} = X_{id}\boldsymbol{\theta} \;\rightarrow\; \hat{\boldsymbol{\theta}} = X_{id}^{\#}\mathbf{Y}_{id}$$

$$\hat{Y}_{val} = X_{val}\hat{\boldsymbol{\theta}}$$

Linear Regression Modeling of Static Systems

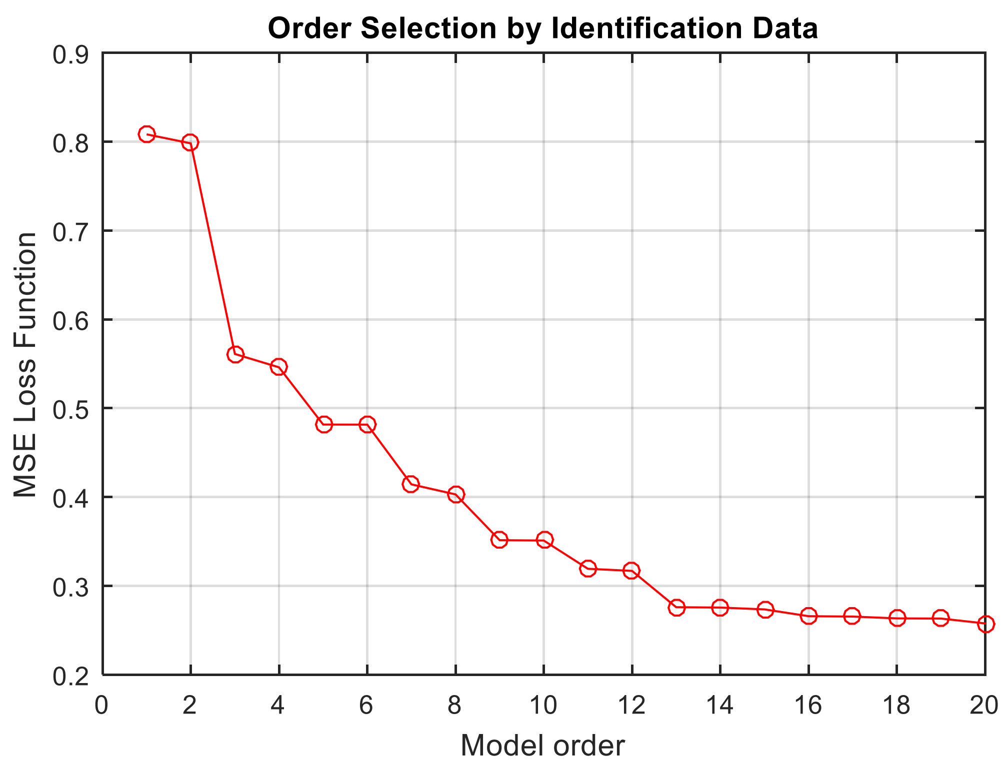
Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

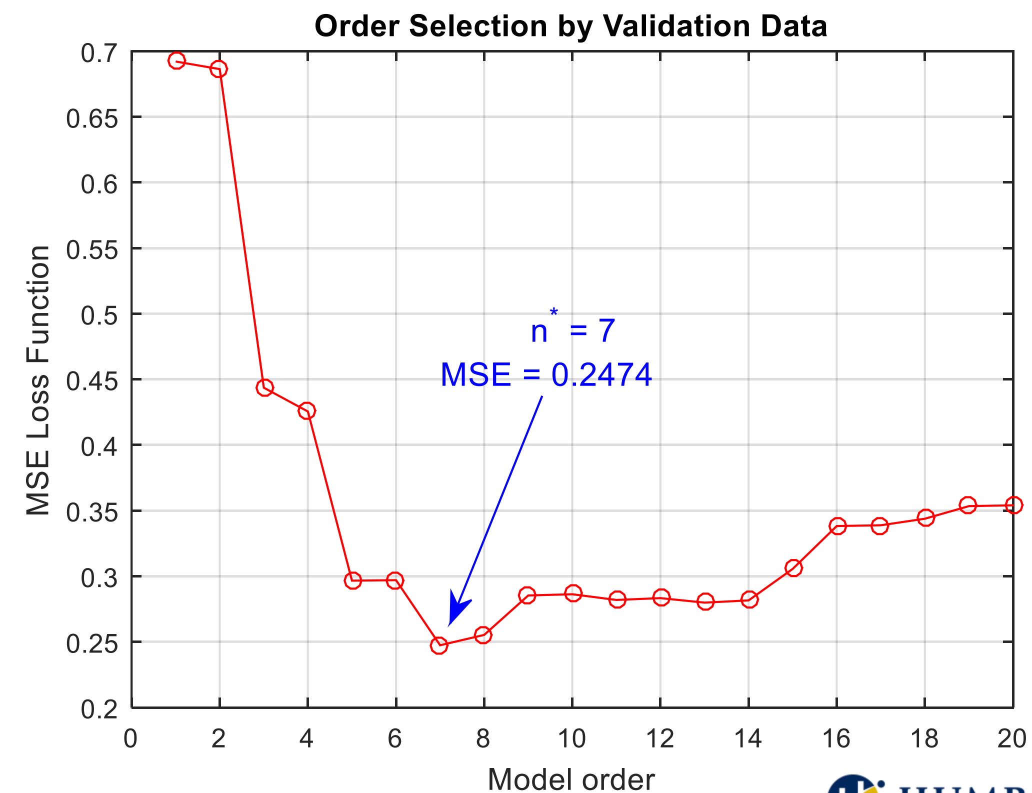
$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

Following figures compare the loss function evaluation for given range of model order $n = 1, 2, \cdots, 20$, using the validation dataset and the identification dataset.

$$\boxed{J(\theta, Z_{id}) = \frac{1}{N_{id}} \sum_{i=1}^{N_{id}} \left(\boldsymbol{y}(\boldsymbol{k}) - \hat{\boldsymbol{y}}\,(\boldsymbol{k}; \boldsymbol{\theta})\right)^2}$$

$$\boxed{J(\theta, Z_{val}) = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \left(\boldsymbol{y}(\boldsymbol{k}) - \hat{\boldsymbol{y}}\,(\boldsymbol{k}; \boldsymbol{\theta})\right)^2}$$





Linear Regression Modeling of Static Systems

Example 1 Assume that the following collected N = 200 samples of a noisy data is available. Estimate a polynomial model by Least-Squares Estimation method.

$$y(k) = \theta_n u(k)^n + \theta_{n-1} u(k)^{n-1} + \cdots + \theta_2 u(k)^2 + \theta_1 u(k) + \theta_0 + e(k)$$

Evaluation results show that the optimum order can be selected as

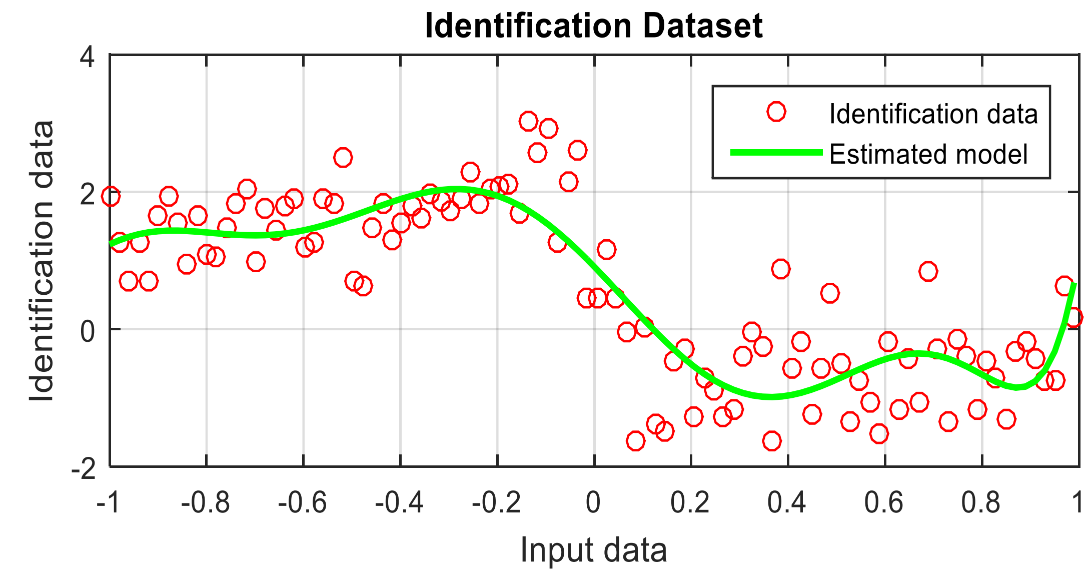
$$\boxed{\textbf{Optimum order} \rightarrow \; n^* = 7}$$

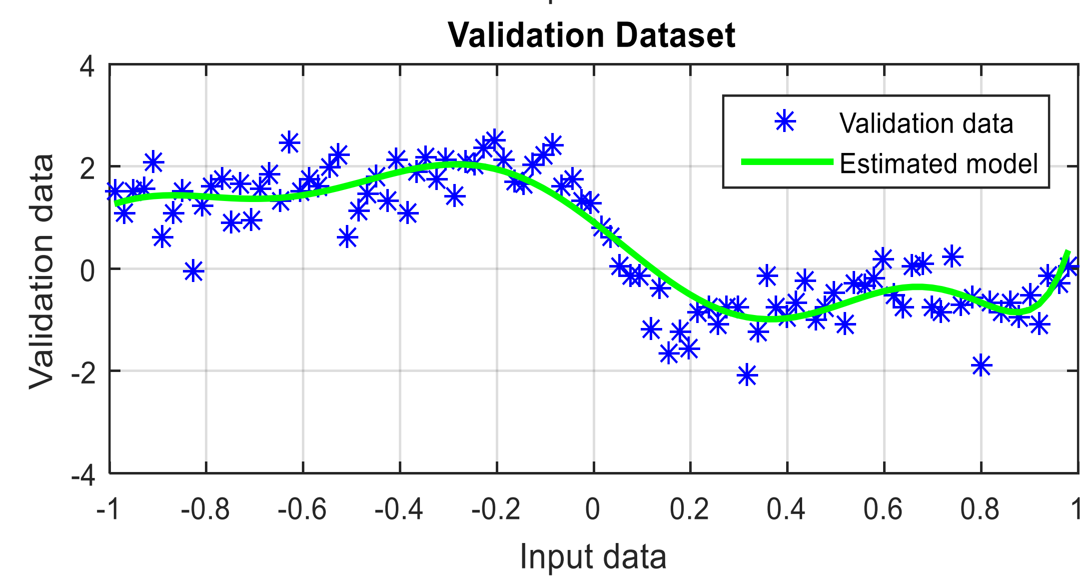
and the loss function evaluation is $MSE = 0.2474$

As a result, the best model structure to estimate the given data will be as

$$\boxed{y = \theta_7 u^7 + \theta_6 u^6 + \theta_5 u^5 + \theta_4 u^4 + \theta_3 u^3 + \theta_2 u^2 + \theta_1 u + \theta_0}$$

$$\textbf{Estimated Parameters} \rightarrow \begin{bmatrix} \theta_7 \\ \theta_6 \\ \theta_5 \\ \theta_4 \\ \theta_3 \\ \theta_2 \\ \theta_1 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 16.9586 \\ -29.9632 \\ -38.2327 \\ 22.8056 \\ 27.5786 \\ -6.1394 \\ -7.3441 \\ 0.8196 \end{bmatrix}$$





Linear Regression Modeling of Dynamic Systems

- Consider the linear regression model structure, which the model is linear with respect to its parameters.

Linear Regression Model $\boxed{y(k) = \boldsymbol{\varphi}^T(k)\boldsymbol{\theta} + e(k), \qquad k = 1, \cdots, N}$

- The linear regression models can be used to model the static systems and dynamic systems.
- In modeling of static systems, regressors depends only on the current input data.
  For example:

$$y(k) = \theta_0 + \theta_1 u(k) + \theta_2 u(k)^2 + \theta_3 u(k)^3$$

- In dynamic systems modeling, the regressors may depend on the past output data, and current and past input data.
  For example:

$$y(k) = \theta_0 y(k-2) + \theta_1 y(k-1) + \theta_2 u(k)^2 + \theta_3 u(k-3)$$

- The observations, $y(k)$, and regressors $\boldsymbol{\varphi}(k)$, values are obtained from the experimental I/O data.

- In dynamic systems modeling, the linear regression representation only applies to model structures that are linear in terms of the parameters.

Linear Regression Modeling of Dynamic Systems

Example 2

Determine the linear regression form of the following systems with input $u(k)$, output $y(k)$ and $e(k)$ is a white noise.

$$\boxed{y(k) = \boldsymbol{\varphi}^T(k)\boldsymbol{\theta} + e(k)}$$

a) $y(k) = b_0 + b_1 u(k) + b_2 u^2(k) + e(k)$

$$y(k) = \underbrace{\begin{bmatrix} 1 & u(k) & u^2(k) \end{bmatrix}}_{\boldsymbol{\varphi}^T(\boldsymbol{k})} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}}_{\boldsymbol{\theta}} + e(k)$$

b) $y(k) + a_1 y(k-1) = b_0 u(k) + b_1 \sin(u(k-1)) + e(k)$

$$y(k) = -a_1 y(k-1) + b_0 u(k) + b_1 \sin(u(k-1)) + e(k)$$

$$y(k) = \underbrace{\begin{bmatrix} -y(k-1) & u(k) & \sin(u(k-1)) \end{bmatrix}}_{\boldsymbol{\varphi}^T(\boldsymbol{k})} \underbrace{\begin{bmatrix} a_1 \\ b_0 \\ b_1 \end{bmatrix}}_{\boldsymbol{\theta}} + e(k)$$

c) $y(k) + a_1 y(k-1) + a_2 y(k-2) = b_1 u(k-1) + b_2 u(k-2) + e(k)$

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) + e(k)$$

$$y(k) = \underbrace{\begin{bmatrix} -y(k-1) & -y(k-2) & u(k-1) & u(k-2) \end{bmatrix}}_{\boldsymbol{\varphi}^T(\boldsymbol{k})} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix}}_{\boldsymbol{\theta}} + e(k)$$

The models are linear in terms of the parameters $\boldsymbol{\theta}$ not the regressors $\varphi(k)$.

Discrete-time Transfer Function Models

- Consider a general form of a linear regression model with additive white noise,

$$y(k) + a_1 y(k-1) + a_2 y(k-2) + \cdots + a_{n_a} y(k-n_a) = b_1 u(k-1) + b_2 u(k-2) + \cdots + b_{n_b} u(k-n_b) + e(k)$$

- The backward-shift operator is defined as

$$\boxed{\boldsymbol{q^{-1} x(k) = x(k-1)}}$$

- We can rewrite the linear regression model in polynomial form:

$$y(k) + a_1 q^{-1} y(k) + a_2 q^{-2} y(k) + \cdots + a_{n_a} q^{-n_a} y(k) = b_1 q^{-1} u(k) + b_2 q^{-2} u(k) + \cdots + b_{n_b} q^{-n_b} u(k) + e(k)$$

$$\underbrace{\left(1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}\right)}_{\boldsymbol{A(q)}} y(k) = \underbrace{\left(b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}\right)}_{\boldsymbol{B(q)}} u(k) + e(k)$$

$$\boxed{\boldsymbol{A(q) y(k) = B(q) u(k) + e(k)}}$$

$$\boxed{\boldsymbol{y(k) = \frac{B(q)}{A(q)} u(k) + \frac{1}{A(q)} e(k)}}$$

ARX Model Structure

$$\boxed{\boldsymbol{G(q) = \frac{B(q)}{A(q)} = \frac{b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}}{1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}}}}$$

System model

$$\boxed{\boldsymbol{H(q) = \frac{1}{A(q)} = \frac{1}{1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}}}}$$

Noise filter

Discrete-time Transfer Function Models

❑ ARX Model (Auto-Regressive with Exogenous Input)

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k)$$

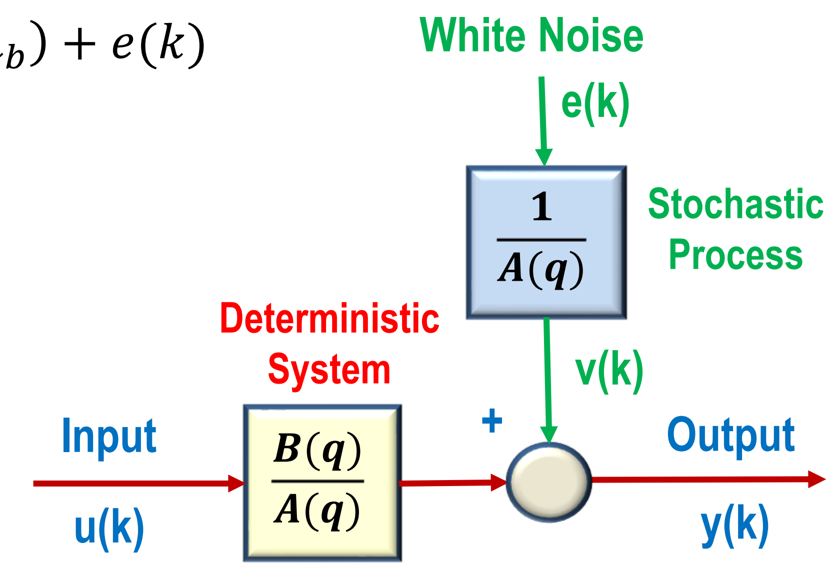
ARX Model Structure

- Auto-Regressive ($y(k)$ is a function of previous y values)
- Exogenous Input ($y(k)$ depends on extra input u)

- The ARX model is the most widely applied linear dynamic model, due to easy-to-compute parameters by linear least-squares technique, which makes it attractive as an initial solution to model identification in practice.
- The ARX model assumes that the input signal and noise signal are filtered by same dynamic.
- This is possible if source of disturbance enters early in the process (process noise), together with the input signal.
- The ARX model can predict the next output value given previous observations of input-output data.

$$y(k) = -a_1 y(k-1) - \cdots - a_{n_a} y(k-n_a) + b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k)$$

- The ARX model is quite general, it can describe arbitrary linear relationships between inputs and outputs.
- However, the noise enters the model in a restricted way.
- If the process noise **does not meet** the noise assumption made by ARX model, the parameters are estimated biased and non-consistent.
- In the absence of noise, the model reduces to a standard discrete-time transfer function.



Discrete-time Transfer Function Models

Linear Regression Model of ARX Model Structure

$$\boxed{\boldsymbol{y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k)}}$$

ARX Model Structure

- Recall the ARX model structure

$$A(q) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}$$

$$B(q) = b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}$$

- The parameter to be estimated is: $\boldsymbol{\theta} = [a_1 \quad a_2 \quad \cdots \quad a_{n_a} \quad b_1 \quad b_2 \quad \cdots \quad b_{n_b}]^T$
- The difference equation is obtained as below

$$\boxed{\boldsymbol{A(q)y(k) = B(q)u(k) + e(k)}} \longrightarrow \left(1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}\right) y(k) = \left(b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}\right) u(k) + e(k)$$

$$y(k) + a_1 y(k-1) + \cdots + a_{n_a} y(k-n_a) = b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k)$$

$$y(k) = -a_1 y(k-1) - \cdots - a_{n_a} y(k-n_a) + b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k)$$

$$y(k) = \underbrace{[-y(k-1) \quad \cdots \quad -y(k-n_a) \quad u(k-1) \quad \cdots \quad u(k-n_b)]}_{\substack{\boldsymbol{\varphi}^T(k) \\ \text{Past input-output data}}} \underbrace{\begin{bmatrix} a_1 \\ \vdots \\ a_{n_a} \\ b_1 \\ \vdots \\ b_{n_b} \end{bmatrix}}_{\substack{\boldsymbol{\theta} \\ \text{Unknown parameters}}} + \underbrace{e(k)}_{\text{White Noise}} \Rightarrow \boxed{y(k) = \boldsymbol{\varphi}^T(k)\boldsymbol{\theta} + e(k)}$$

Discrete-time Transfer Function Models

Example 3 Consider a dynamic system with sampled input $u(k)$, output $y(k)$ and $e(k)$ is white noise. According to the identification experiments we assume the following model for this system.

The a and b are unknown parameters.

$$y(k) = \frac{bq^{-1}}{1 + aq^{-1}} u(k) + \frac{1}{1 + aq^{-1}} e(k)$$

a) Determine structure of the model.

Here, the model has an ARX structure with the following polynomials:

$$y(k) = \frac{B(q)}{A(q)} u(k) + \frac{1}{A(q)} e(k) \quad \Rightarrow \quad \begin{aligned} A(q) &= (1 + aq^{-1}) \\ B(q) &= bq^{-1} \end{aligned}$$

b) Determine the linear regression model of the given structure.

$$(1 + aq^{-1}) y(k) = bq^{-1} u(k) + e(k) \longrightarrow y(k) = -ay(k-1) + bu(k-1) + e(k)$$

$$y(k) = \underbrace{[-y(k-1) \quad u(k-1)]}_{\varphi^T(k)} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta} + e(k)$$

Linear Regression Model

Discrete-time Transfer Function Models

Example 3

Consider a dynamic system with sampled input $u(k)$, output $y(k)$ and $e(k)$ is white noise. According to the identification experiments we assume the following model for this system.

The a and b are unknown parameters.

$$y(k) = \frac{bq^{-1}}{1+aq^{-1}}u(k) + \frac{1}{1+aq^{-1}}e(k)$$

c) Assume that the following I/O data samples are available. Determine the Least-Squares Estimation of the unknown parameters.

k	1	2	3	4
u	2	3	2	3
y	-23	48	-93	36

The least squares estimation of unknown parameters a and b are obtained as below

$$\boxed{\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y} \longrightarrow \begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

First, generate the regressors matrix Φ from the given data samples:

$$y(k) = -ay(k-1) + bu(k-1) + e(t)$$

$$y(2) = -ay(1) + bu(1) + e(2)$$

$$y(3) = -ay(2) + bu(2) + e(3)$$

$$y(4) = -ay(3) + bu(3) + e(4)$$

Vector-matrix form

$$\underbrace{\begin{bmatrix}y(2)\\ y(3)\\ y(4)\end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix}-y(1) & u(1)\\ -y(2) & u(2)\\ -y(3) & u(3)\end{bmatrix}}_{\Phi}\begin{bmatrix}a\\ b\end{bmatrix} + \begin{bmatrix}e(2)\\ e(3)\\ e(4)\end{bmatrix}$$

Discrete-time Transfer Function Models

Example 3

Consider a dynamic system with sampled input $u(k)$, output $y(k)$ and $e(k)$ is white noise. According to the identification experiments we assume the following model for this system.

The a and b are unknown parameters.

$$y(k) = \frac{bq^{-1}}{1+aq^{-1}}u(k) + \frac{1}{1+aq^{-1}}e(k)$$

c) Assume that the following I/O data samples are available. Determine the Least-Squares Estimation of the unknown parameters.

The least squares estimation of unknown parameters a and b are obtained as below

k	1	2	3	4
u	2	3	2	3
y	-23	48	-93	36

$$\boxed{\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y} \longrightarrow \begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

$$\begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix} = \left(\begin{bmatrix}-y(1) & -y(2) & -y(3)\\ u(1) & u(2) & u(3)\end{bmatrix}\begin{bmatrix}-y(1) & u(1)\\ -y(2) & u(2)\\ -y(3) & u(3)\end{bmatrix}\right)^{-1}\begin{bmatrix}-y(1) & -y(2) & -y(3)\\ u(1) & u(2) & u(3)\end{bmatrix}\begin{bmatrix}y(2)\\ y(3)\\ y(4)\end{bmatrix}$$

$$\begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix} = \left(\begin{bmatrix}23 & -48 & 93\\ 2 & 3 & 2\end{bmatrix}\begin{bmatrix}23 & 2\\ -48 & 3\\ 93 & 2\end{bmatrix}\right)^{-1}\begin{bmatrix}23 & -48 & 93\\ 2 & 3 & 2\end{bmatrix}\begin{bmatrix}48\\ -93\\ 36\end{bmatrix} \longrightarrow \begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix} = \left(\begin{bmatrix}11482 & 88\\ 88 & 17\end{bmatrix}\right)^{-1}\begin{bmatrix}8916\\ -111\end{bmatrix}$$

Least-Squares Estimation of the Parameters

$$\begin{bmatrix}\hat{\boldsymbol{a}}\\ \hat{\boldsymbol{b}}\end{bmatrix} = \begin{bmatrix}\mathbf{0.8607}\\ \mathbf{-10.9848}\end{bmatrix}$$

Discrete-time Transfer Function Models

Example 3

Consider a dynamic system with sampled input $u(k)$, output $y(k)$ and $e(k)$ is white noise. According to the identification experiments we assume the following model for this system.

The a and b are unknown parameters.

$$y(k) = \frac{bq^{-1}}{1 + aq^{-1}} u(k) + \frac{1}{1 + aq^{-1}} e(k)$$

d) Determine the estimated ARX model based on the estimated model parameters.

General form of the ARX model is

$$y(k) = \frac{bq^{-1}}{1 + aq^{-1}} u(k) + \frac{1}{1 + aq^{-1}} e(k)$$

From the estimated parameters $\hat{a} = 0.8607$ and $\hat{b} = -10.9848$ we have the following ARX model

$$\boxed{y(k) = \frac{-10.9848q^{-1}}{1 + 0.8607q^{-1}} u(k) + \frac{1}{1 + 0.8607q^{-1}} e(k)}$$

Estimated ARX Model from LS Method

Discrete-time Transfer Function Models

❑ Output Error (OE) Model Structure

$$\boldsymbol{y(k) = \frac{B(q)}{F(q)}u(k) + e(k)}$$

OE Model Structure

- The OE Model is a very simple model and more realistic process description.
- OE model assumes the disturbance is a White noise.

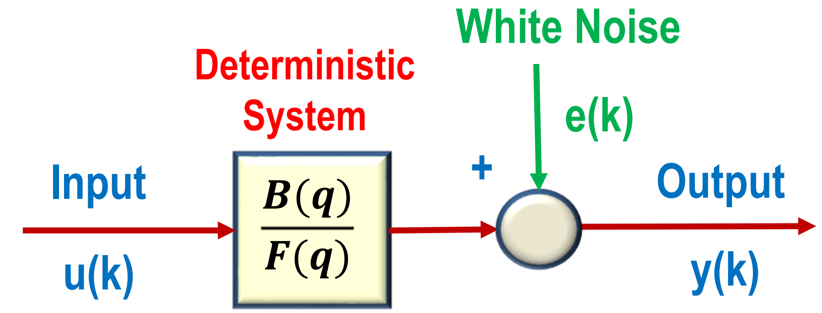
$$G(q) = \frac{B(q)}{F(q)} = \frac{b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}}{1 + f_1 q^{-1} + f_2 q^{-2} + \cdots + f_{n_f} q^{-n_f}}$$

- Since OE model has fewer parameters to estimate, it is often a good option of model structures in practice.
- OE model structure provides good results when system dominated additive measurement (sensor) white noise.
- OE model the parameters **cannot** be estimated by a simple LS method.

$$\boxed{F(q)y(k) = B(q)u(k) + F(q)e(k)}$$

$$y(k) + f_1 y(k-1) + \cdots + f_{n_f} y\left(k - n_f\right) = b_1 u(k-1) + \cdots + b_{n_b} u(k - n_b) + e(k) + f_1 e(k-1) + \cdots + f_{n_f} e\left(k - n_f\right)$$

- The regressors include the past samples of noise signal which are **unknown**.
- The parameters of OE model can be estimated by following methods exploiting the relationship to ARX model,
  - Nonlinear Optimization
  - Repeated Least-Squares and Filtering



Discrete-time Transfer Function Models

❑ Output Error (OE) Model Estimation

$$\boldsymbol{y(k) = \frac{B(q)}{F(q)}u(k) + e(k)}$$

OE Model Structure

- **Repeated LS and Filtering for OE Model Estimation**

1) Estimate an ARX model using LS method from the data $\{u(k), y(k)\}$

$$\boxed{\boldsymbol{y(k) = \frac{B(q)}{F(q)}u(k) + \frac{1}{F(q)}e(k)}} \quad \rightarrow \quad F(q)y(k) = B(q)u(k) + e(k)$$

$$y(k) + f_1 y(k-1) + \cdots + f_{n_f} y(k-n_f) = b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k)$$

$$\boxed{\hat{\boldsymbol{\theta}} = (\Phi^T \Phi)^{-1}\, \Phi^T \mathbf{Y}}$$

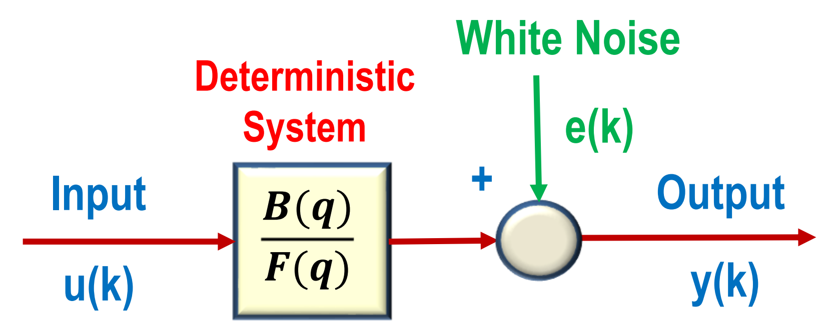
The parameters in $\hat{\boldsymbol{\theta}}$ includes f_i and b_i values

2) Filter the input-output data $\{u(k), y(k)\}$ through the estimated filter $\hat{F}(q)$.

$$u^F(k) = \frac{1}{\hat{F}(q)}u(k) \qquad y^F(k) = \frac{1}{\hat{F}(q)}y(k)$$

3) Estimate the OE model parameters f_i and b_i by an ARX model estimation with the filtered input and output.

4) Iterate Steps 2 to 3 until convergence is reached.



Discrete-time Transfer Function Models

Box-Jenkins (BJ) Model Structure

- Box-Jenkins (BJ) Model is a very general model.
- Includes all other models as special case.
- In BJ model, disturbance can have a completely different model from the process model.

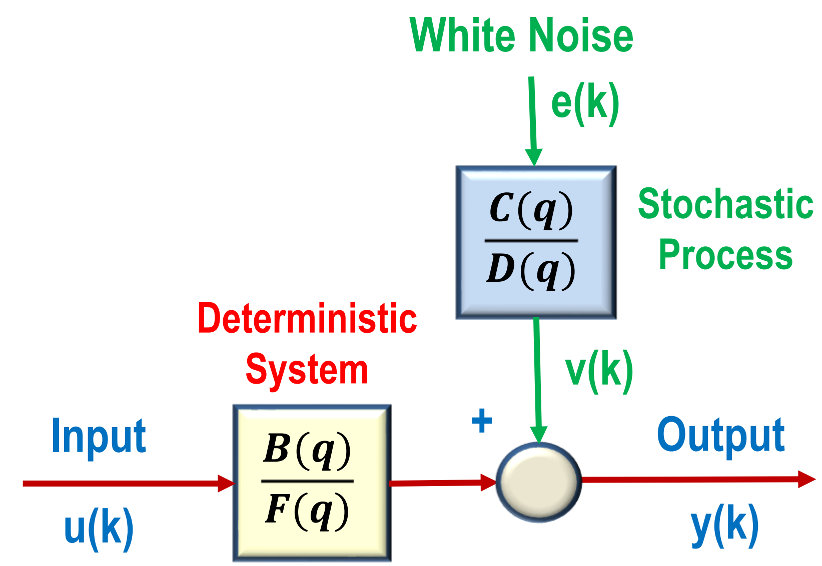
$$y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}e(k)$$

BJ Model Structure

$$G(q) = \frac{B(q)}{F(q)} = \frac{b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}}{1 + f_1 q^{-1} + f_2 q^{-2} + \cdots + f_{n_f} q^{-n_f}}$$

$$H(q) = \frac{C(q)}{D(q)} = \frac{1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_{n_c} q^{-n_c}}{1 + d_1 q^{-1} + d_2 q^{-2} + \cdots + d_{n_d} q^{-n_d}}$$

- Typically, a BJ model can be estimated by following approaches
  - Nonlinear Optimization
  - Independent Parameterization of $G(q)$ and $H(q)$



Discrete-time TF Model Estimation in MATLAB

- In System Identification Toolbox MATLAB, we have the following functions to obtain DT transfer function model structures
  - The **arx** function estimates parameters of ARX model

```
sys = arx(data,[na nb nk])
```

  - The **oe** function estimates parameters of OE model

```
sys = oe(data,[nb nf nk])
```

  - The **bj** function estimates parameters of BJ model

```
sys = bj(data,[nb nc nd nf nk])
```

Variable	Description
sys	Parametric model that fits the estimated data
data	Input-output data in **iddata** format
na	Order of the polynomial $A(q)$
nb	Order of the polynomial $B(q)$
nc	Order of the polynomial $C(q)$
nd	Order of the polynomial $D(q)$
nf	Order of the polynomial $F(q)$
nk	Number of I/O delays of discrete-time model

- We can create a data object by **iddata** command from the collected I/O data.

```
data = iddata(y,u,Ts)
```

Data object

Output data

Input data

Sample time

Note that the DT models must be converted to CT model.

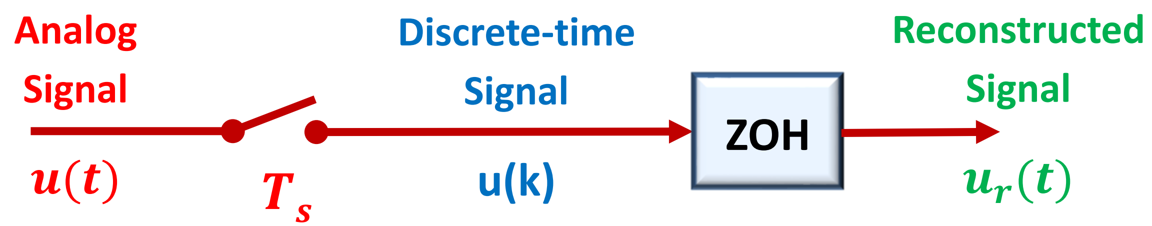
Discrete-time TF Model Estimation in MATLAB

❑ Sampling & Reconstruction

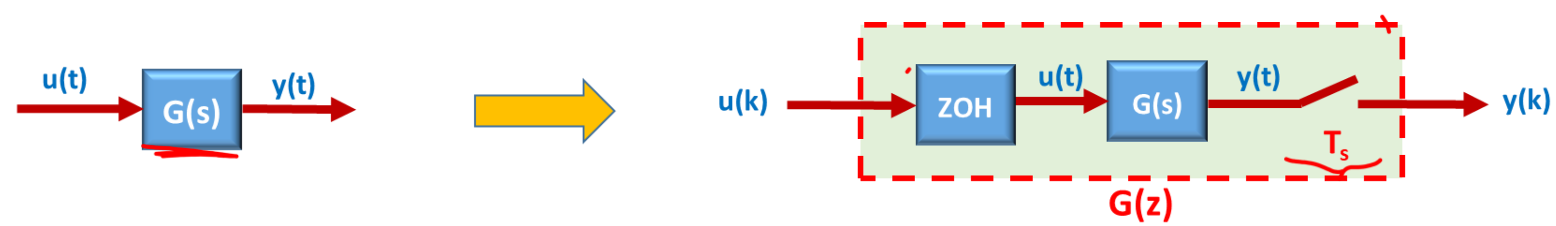
- Sampling is the process of converting the CT signal to a DT signal
- To avoid the aliasing effect the sampling rate must be selected based on the Sampling Theorem:

$$\boxed{f_s \geq 2f_{max}}$$

- There are several common techniques to reconstruct the sampled discrete-time signal. The simple way is using **Zero-Order-Hold (ZOH),** which assumes the input is piecewise constant over the sample time T_s.

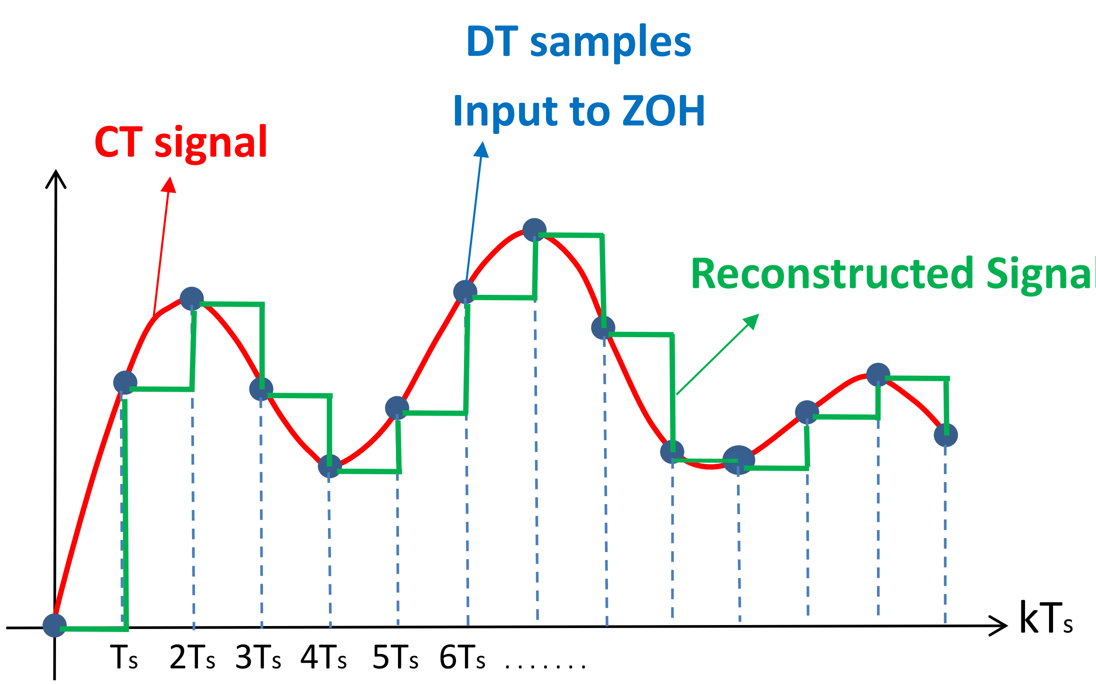


- ZOH adds **one sample delay** to the DT transfer function.



- MATLAB function to convert DT model to CT model.

```
sysCT = d2c(sysDT,'zoh')
```



Continuous-Time Model Estimation in MATLAB

- We can directly identify the CT Transfer Function Model and State-Space Model of a system using the time-domain input and output signals. *No need to do the DT to CT conversion*.

```
sys = tfest(data,np,nz,iodelay)
```

sys: Identified TF model
data: Data object
np: Number of poles
nz: Number of zeros
iodelay: Transport delays for I/O pair

- We can specify an unknown transport delay for the transfer function model by setting the `iodelay` to `nan.` Then MATLAB will estimate the appropriate delay-time based on the input-output data.

```
sys = ssest(data,nx)
```

sys: Identified SS model
data: Data object
nx: Model order

- We can estimate the optimal model order for state-space model by setting `nx` as a range. For example: `nx = 1:10;`

Determining Model Order and Delay

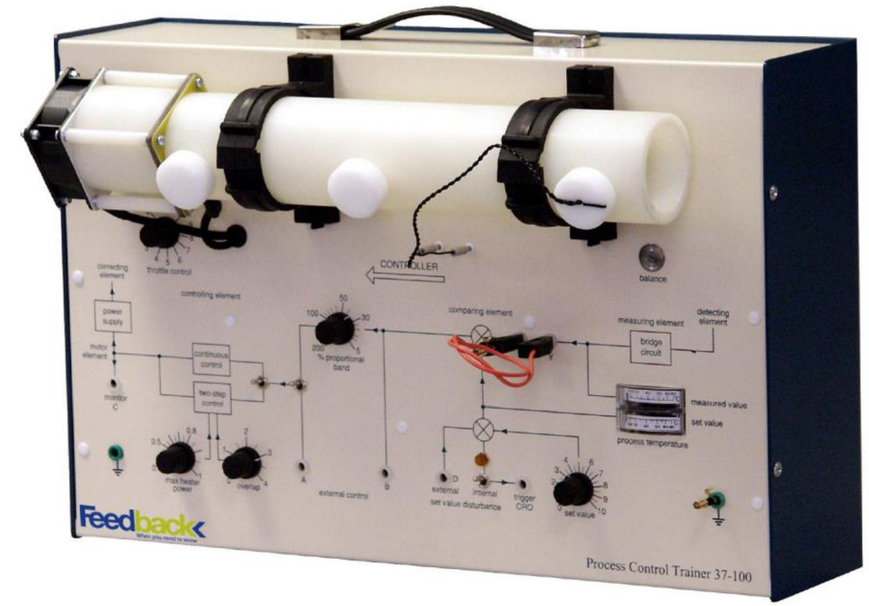
- Estimation of a model using I/O data requires selection of a model structure (such as DT or CT, state-space or transfer function), model order (e.g., number of poles and zeros) and the I/O delays in advance.
- The order and delays are not known in Black-box identification.

- We can determine the model order and delay in one of the following ways:
  - Guess their values by visually inspecting the data or based on the prior knowledge of the system.
  - Estimate delay as a part of transfer function model estimation in `tfest` function.
  - To estimate delays, you can also use one of the following tools:
    - Estimate delay using `delayest`.
    - Compute impulse response using `impulseest`.
  - To estimate model order, you can also use one of the following tools:
    - Select the model order in `n4sid` by specifying the model order range as a vector.
    - Choose the model order of an ARX model using `arxstruc` and `selstruc`. These command select the number of poles, zeros and delay.

Case Study: A Laboratory Scale Hairdryer

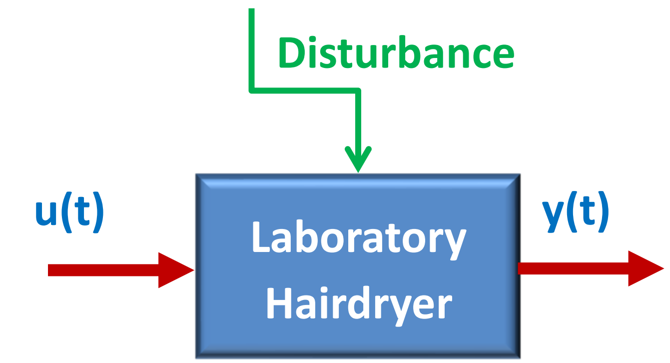
❑ Feedback's Process Control Trainer PT326

- This example shows how to develop a simple model from a real laboratory scale process data.
- The process function is like a hairdryer:
  - Air is fanned through a tube and heated at the inlet
  - The input $u(t)$ is the power of the heating device, which is a mesh of resistor wires.
  - The output $y(t)$ is the outlet air temperature.
  - The system is well behaved with simple dynamics and small disturbances.
- This example shows the basic steps of a system identification procedure:
  1) How to import the data and apply preprocessing steps
  2) How to estimate model order and delay
  3) How to estimate DT transfer function model
  4) How to validate the model to the actual output data from the experiment.
  5) Convert the DT model to CT transfer function model.

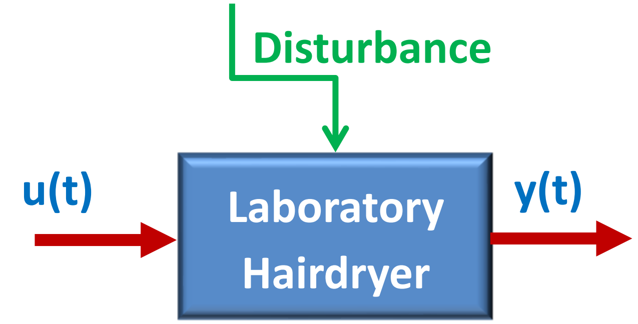


Process Control Trainer

http://www.feedback-instruments.com/pdf/brochures/37-100_datasheet_ProcessControlTrainer_04_2013.pdf



Case Study: A Laboratory Scale Hairdryer



$$\{u(k)\,,y(k)|\,k=1,\cdots,N\}$$

Step 1: Collect the I/O Data

Consider the following collected I/O data from a laboratory model of a hairdryer.

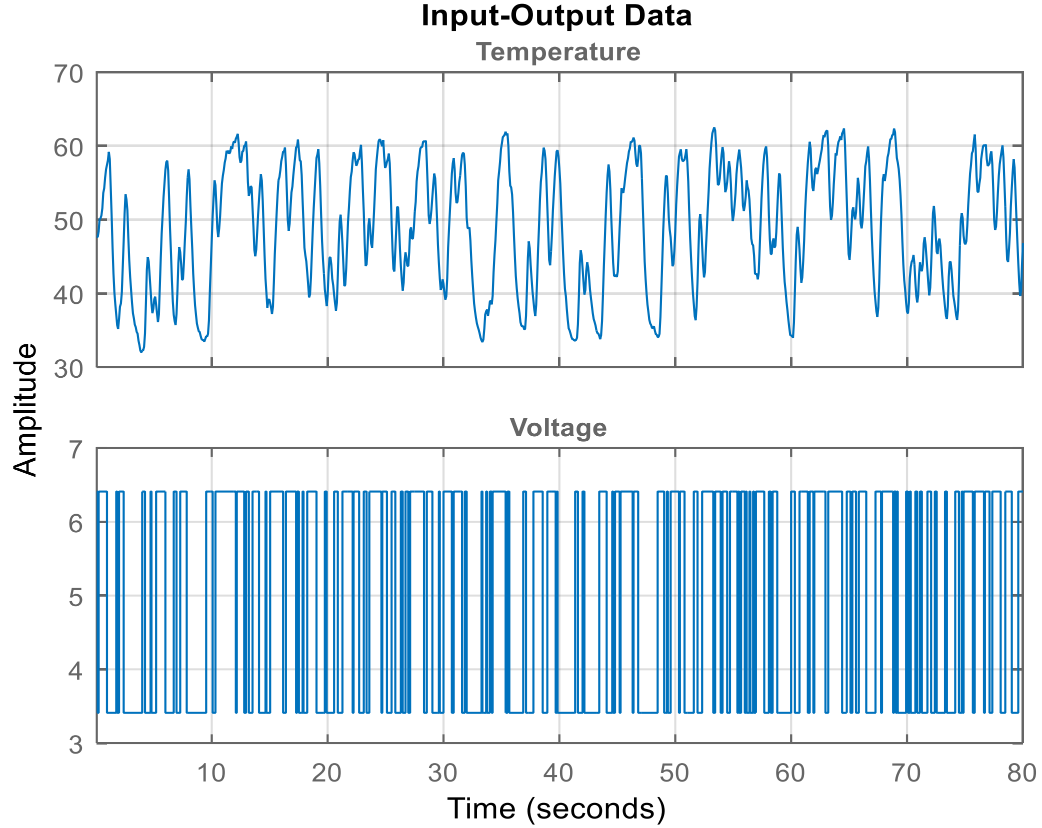
(This data is available in MATLAB System Identification Toolbox) $(N=1000, T_s=0.08\text{sec})$

```
load dry2              ← Load I/O data from MATLAB
data = dry2(1:N);      ← I/O data object
plot(data)             ← Plot raw I/O data
```

Input and output signals are defined as follows:

Input: The voltage over the heating device, which is a mesh of resistor wires.

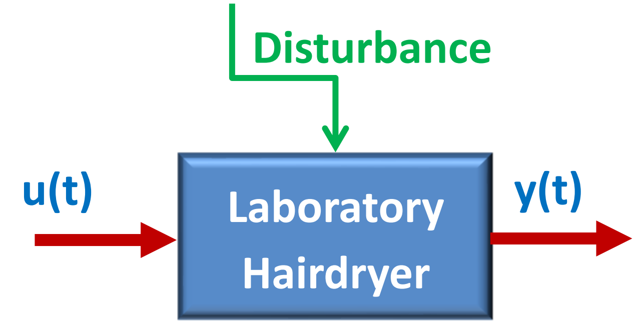
Output: The outlet air temperature represented by the measured thermocouple voltage.



- Here, the process has a simple dynamics with quite small disturbances
- The collected data has good SNR.

Case Study: A Laboratory Scale Hairdryer

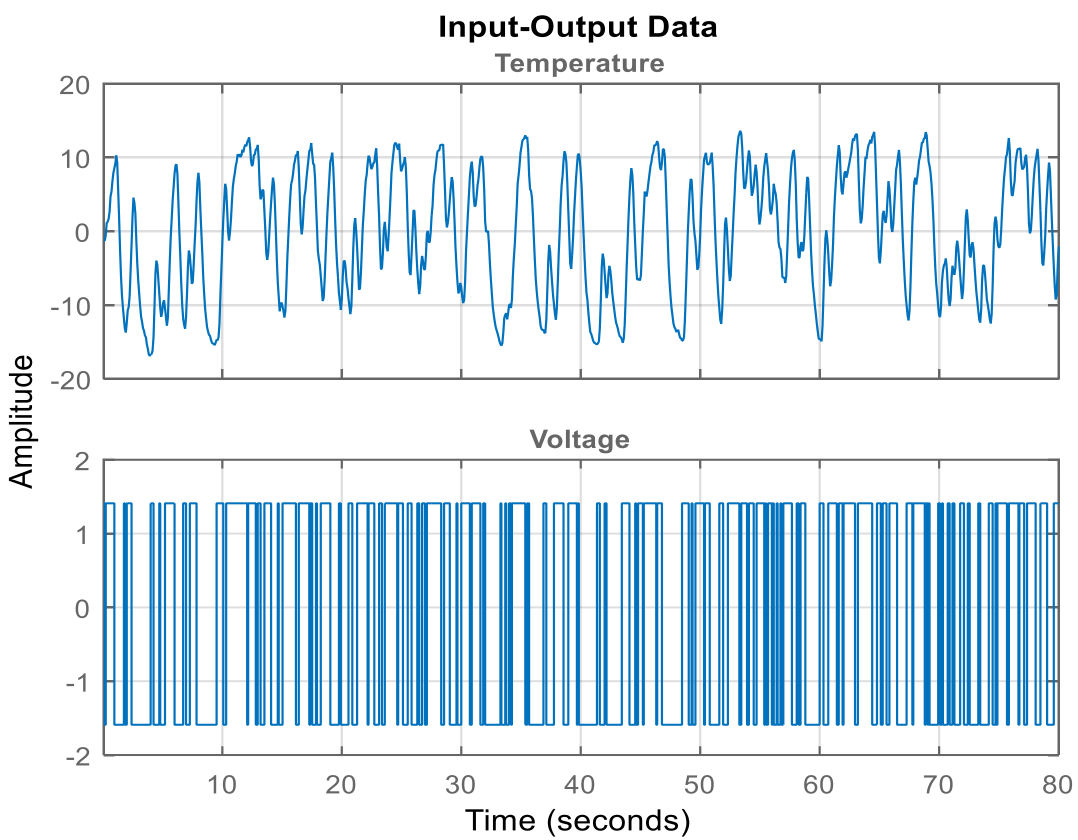
Step 2: Examine the I/O Data (Outliers, Trends and Aliasing effects)

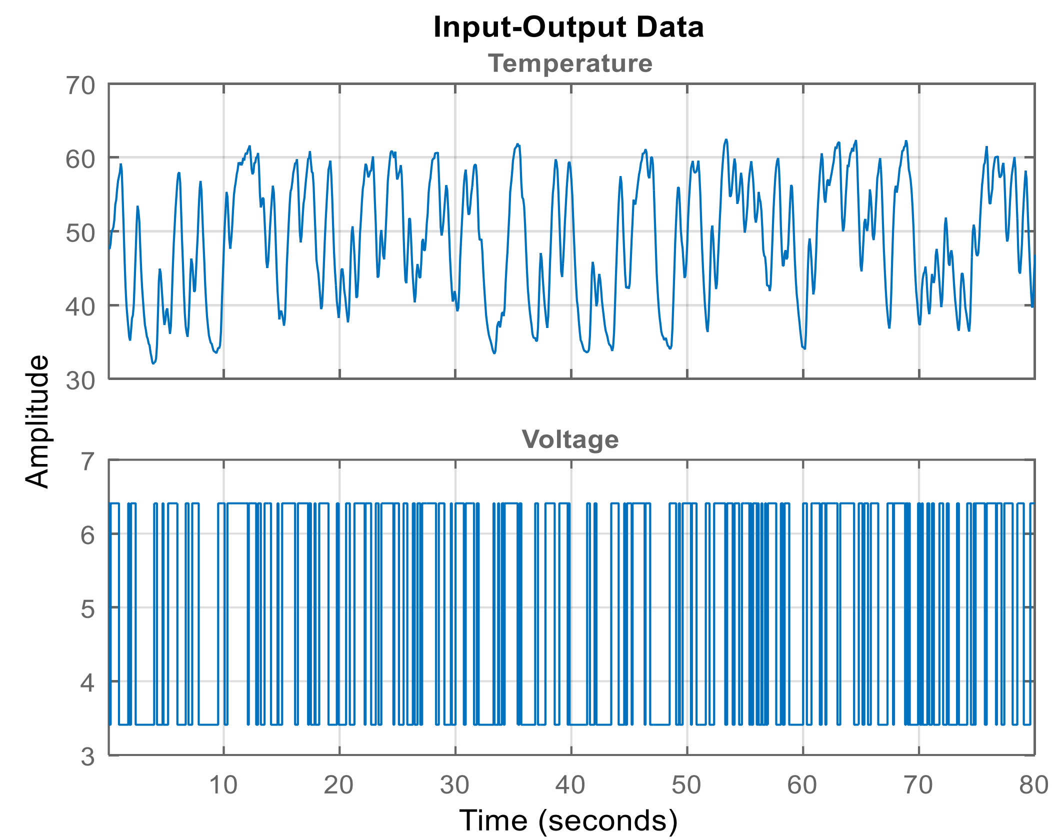


$$\{u(k)\,,y(k)|\;k=1,\cdots,N\}$$

From the data plot, it can be seen that the data has non-zero mean (DC-offset).
Therefore, first we have to remove the non-zero mean from the I/O data

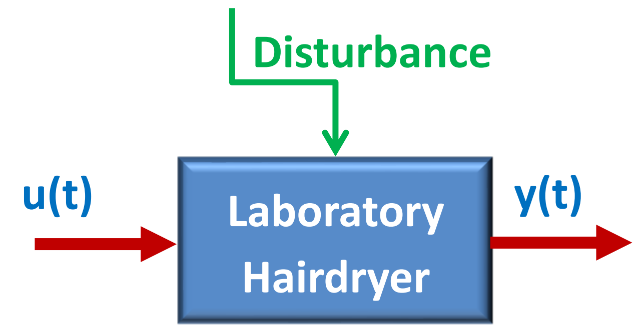
```
data = dtrend(data);   ← Remove non-zero mean value
```





Case Study: A Laboratory Scale Hairdryer

Step 3: Split the I/O data to the Identification data and Validation data

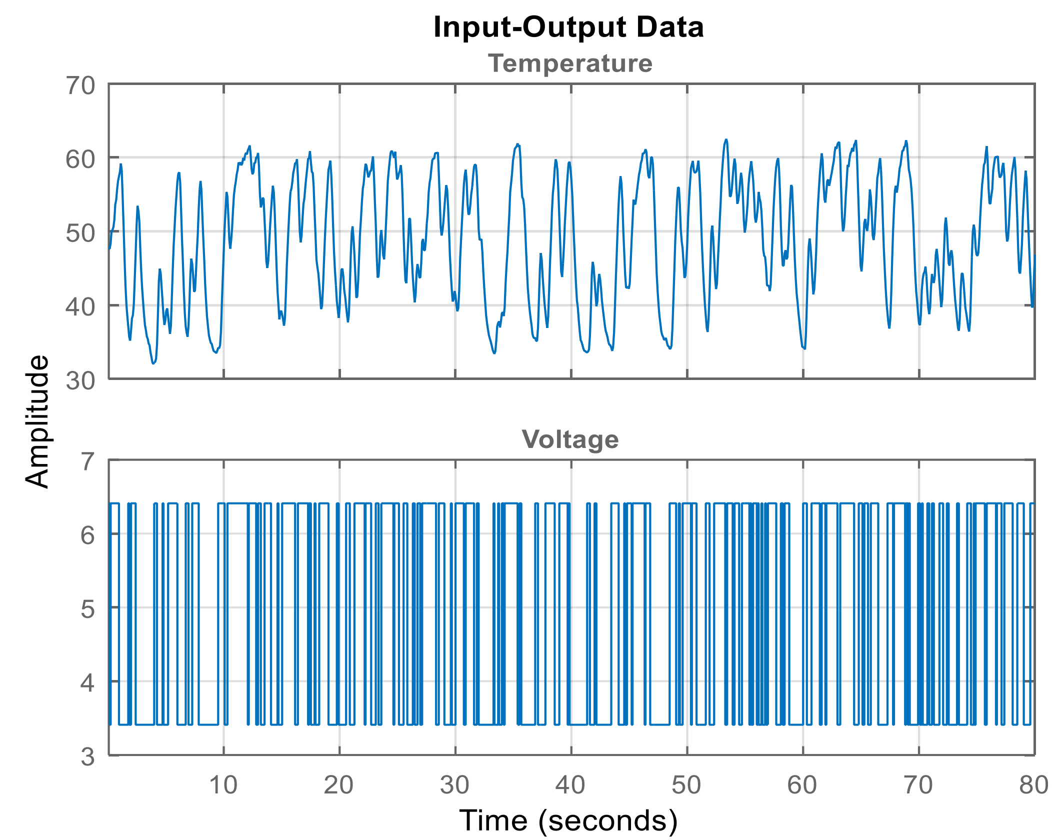


$$\{u(k), y(k) \mid k = 1, \cdots, N\}$$

Here, we have $N = 1000$ samples of I/O data. The first 700 samples are used for identification (estimation the model parameters), the rest for validation.

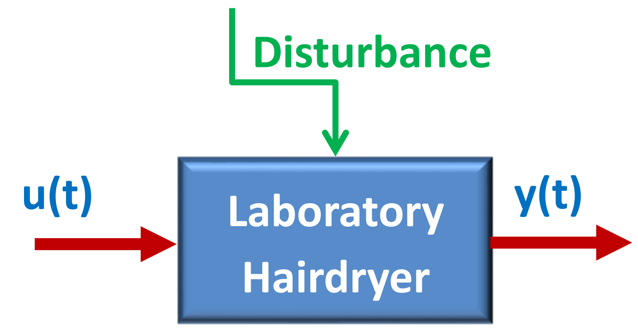
$$\boldsymbol{N_{id} = 700, \qquad N_{val} = 300}$$

```
datai = data(1:700);          ← Identification data
datav = data(701:N);          ← Validation data
```



Case Study: A Laboratory Scale Hairdryer

Step 4: Obtain some priori information about the system order and time-delay



$$\{u(k)\,, y(k) |\, k = 1, \cdots, N\}$$

❑ Delay Estimation:

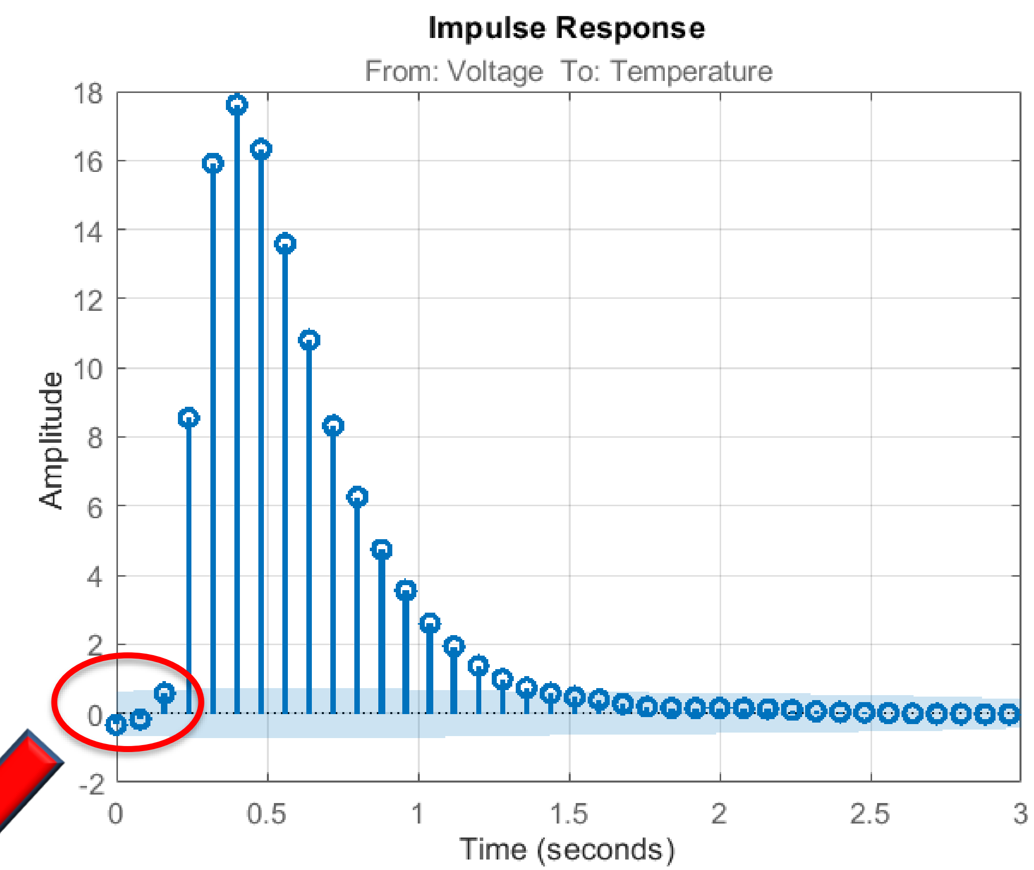
Here, we use the **`delayest`** and **`impulseest`** functions to estimate the delay.

```
delay = delayest(data)          ← Delay estimation
```

```
delay =

     3
```

```
ir = impulseest(data);         ← Impulse response estimation
ir_plot = impulseplot(ir);     ← Impulse response plot
showConfidence(ir_plot,5)      ← Show confidence intervals
```

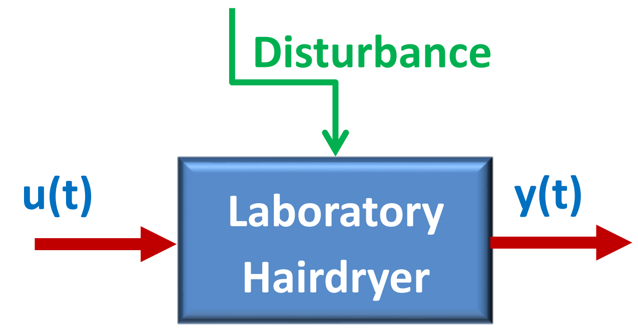
- We plot the impulse response with a confidence interval represented by 5 standard deviations.
- The results show that we can consider three sample delays, $n_k = 3$.
- Note that, the number of delay samples includes the one delay sample for ZOH.



$$\boxed{n_k = 3}$$

Case Study: A Laboratory Scale Hairdryer

Step 4: Obtain some priori information about the system order and time-delay



$$\{u(k)\,,y(k)|\;k=1,\cdots,N\}$$

❑ Order Estimation:

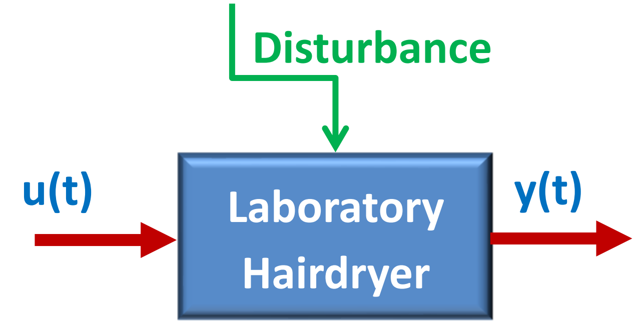
Here, we use the **n4sid** function to estimate the order from state-space structure.

```
n4sid(data,1:10,'InputDelay',2);    ← Order estimation
```

- We considered the range of orders from 1 to 10.
- The "**InputDelay**" was set to 2 because by default **n4sid** estimates a model accounting for one sample lag between input and output.
- The default order, indicated in the figure, is **two**.

Case Study: A Laboratory Scale Hairdryer

Step 5: Select the parametric model structure and estimate the model



$$\{u(k)\,,y(k)|\,k=1,\cdots,N\}$$

❑ OE Model Estimation:

```
nb = 2;                            ← Order of the B(q) polynomial
nf = 2;                            ← Order of the F(q) polynomial
nk = 3;                            ← Number of samples delay
M_OE = oe(datai,[nb nf nk]);       ← OE model from identification data
```

```
M_OE =
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.6795 z^-3 + 0.4541 z^-4
  F(z) = 1 - 1.274 z^-1 + 0.3946 z^-2

Sample time: 0.08 seconds
Parameterization:
   Polynomial orders:   nb=2   nf=2   nk=3
   Number of free coefficients: 4

Status:
Estimated using OE on time domain data "dry2".

Fit to estimation data: 88.76%
FPE: 0.9316, MSE: 0.9211
```

- The OE model estimation

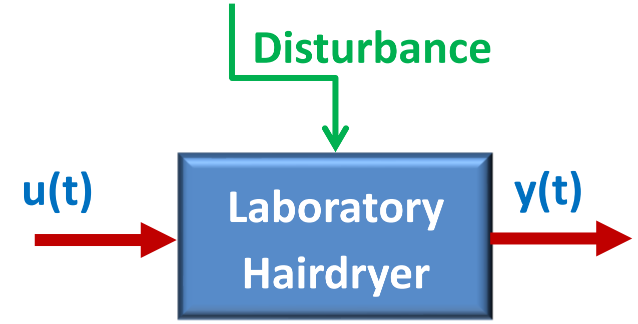
$$\boxed{y(k)=\frac{B(q)}{F(q)}u(k)+e(k)}$$

$$y(k)=\frac{0.6795q^{-3}+0.4541q^{-4}}{1-1.274q^{-1}+0.3946q^{-2}}u(k)+e(k)$$

$$\boxed{\boldsymbol{y(k)=\frac{0.6795+0.4541q^{-1}}{1-1.27q^{-1}+0.3946q^{-2}}u(k-3)+e(k)}}$$

Case Study: A Laboratory Scale Hairdryer

Step 5: Select the parametric model structure and estimate the model



$$\{u(k)\,, y(k) \mid k = 1, \cdots, N\}$$

❑ ARX Model Estimation:

```
na = 2;                         ← Order of the A(q) polynomial
nb = 2;                         ← Order of the B(q) polynomial
nk = 3;                         ← Number of samples delay
M_ARX = arx(datai,[na nb nk]);  ← ARX model from identification data
```

```
M_ARX =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.288 z^-1 + 0.4053 z^-2
  B(z) = 0.6552 z^-3 + 0.4324 z^-4

Sample time: 0.08 seconds
Parameterization:
   Polynomial orders:   na=2   nb=2   nk=3
   Number of free coefficients: 4

Status:
Estimated using ARX on time domain data "dry2".

Fit to estimation data: 95.32%
FPE: 0.1627, MSE: 0.16
```

- The ARX model estimation

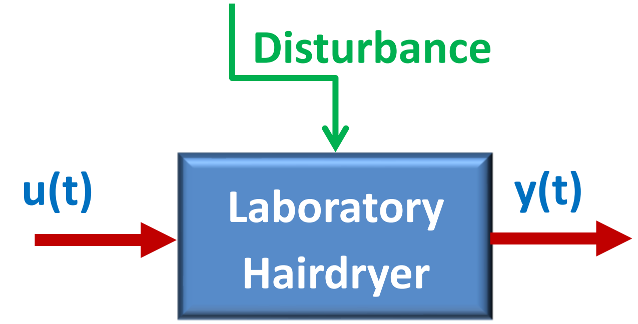
$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k)$$

- The ARX model has better fit to estimation data.

$$\boldsymbol{y(k) = \frac{0.6552 + 0.4324q^{-1}}{1 - 1.288q^{-1} + 0.4053q^{-2}}u(k-3) + \frac{1}{1 - 1.288q^{-1} + 0.4053q^{-2}}e(k)}$$

Case Study: A Laboratory Scale Hairdryer

Step 6: Model Validation



$$\{u(k)\,, y(k) \mid k = 1, \cdots, N\}$$

❑ OE Model Validation:

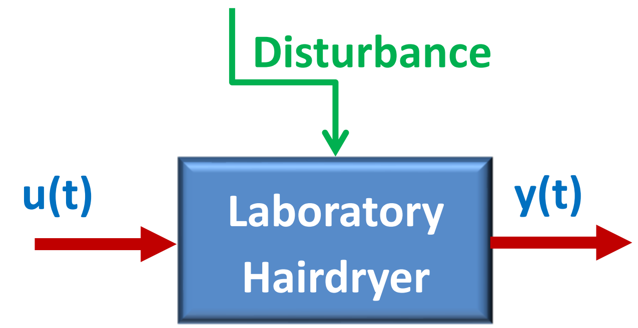
Here, we use the **present** function in MATLAB to validate the OE model parameters estimation.

```
present(M_OE)        ← Display model information
```

```
M_OE =
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.6795 (+/- 0.03021) z^-3 + 0.4541 (+/- 0.05239) z^-4
  F(z) = 1 - 1.274 (+/- 0.0253) z^-1 + 0.3946 (+/- 0.0228) z^-2

Sample time: 0.08 seconds
Parameterization:
   Polynomial orders:   nb=2   nf=2   nk=3
   Number of free coefficients: 4

Status:
Estimated using OE on time domain data "dry2".
Fit to estimation data: 88.76%
FPE: 0.9316, MSE: 0.9211
```

- Confidence intervals of the estimation
- Fit to estimation data
- Mean Square Error (MSE)
- Final Prediction Error (FPE)

Case Study: A Laboratory Scale Hairdryer



$$\{u(k), y(k) \mid k = 1, \cdots, N\}$$

Step 6: Model Validation

❑ ARX Model Validation:

Here, we use the `present` function in MATLAB to validate the ARX model parameters estimation.

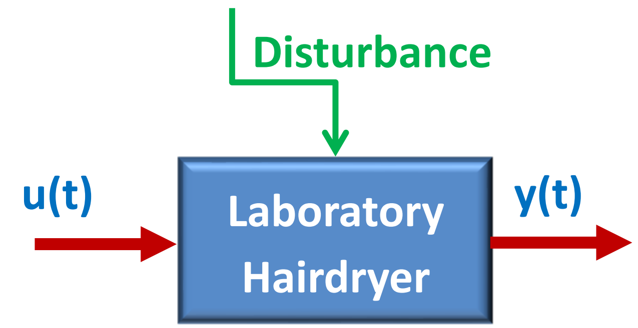
```
present(M_ARX)            ← Display model information
```

```
M_ARX =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.288 (+/- 0.01325) z^-1 + 0.4053 (+/- 0.01211) z^-2
  B(z) = 0.6552 (+/- 0.01326) z^-3 + 0.4324 (+/- 0.02026) z^-4

Sample time: 0.08 seconds
Parameterization:
   Polynomial orders:   na=2   nb=2   nk=3
   Number of free coefficients: 4

Status:
Estimated using ARX on time domain data "dry2".
Fit to estimation data: 95.32%
FPE: 0.1627, MSE: 0.16
```

- Confidence intervals of the estimation
- Fit to estimation data
- Mean Square Error (MSE)
- Final Prediction Error (FPE)

- We can see that for ARX model estimation the FPE and MSE criteria give a <u>better</u> results than the OE model.

Case Study: A Laboratory Scale Hairdryer



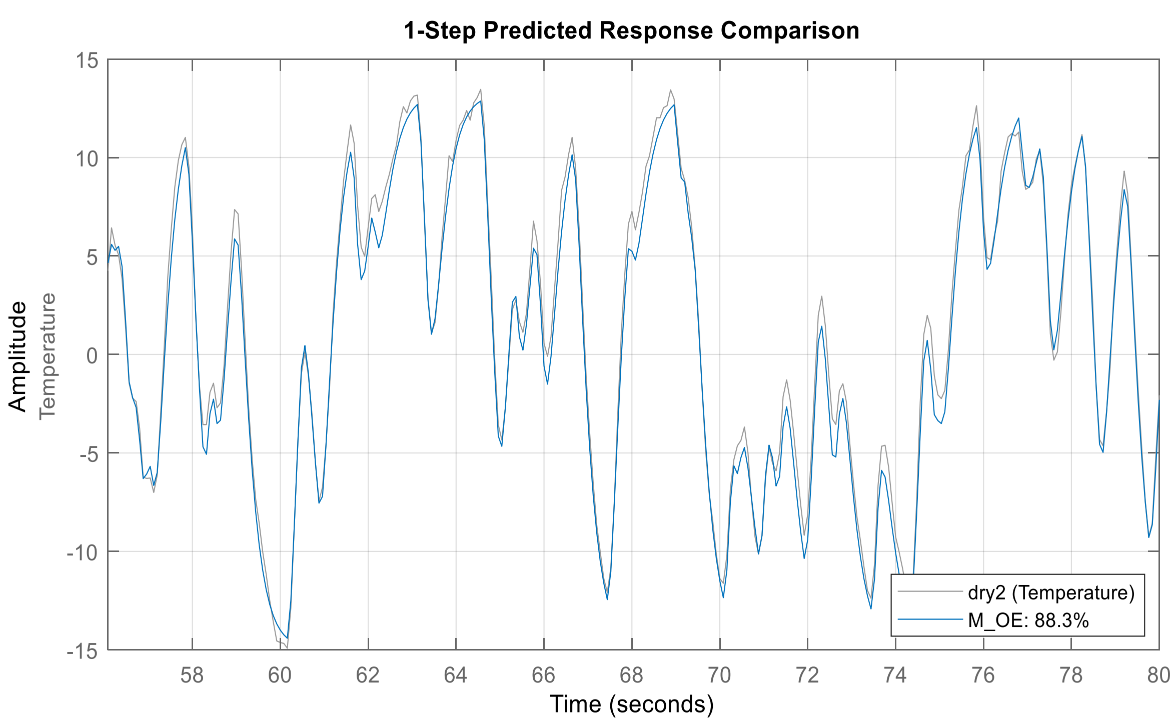
$$\{u(k), y(k) \mid k = 1, \cdots, N\}$$

Step 6: Model Validation

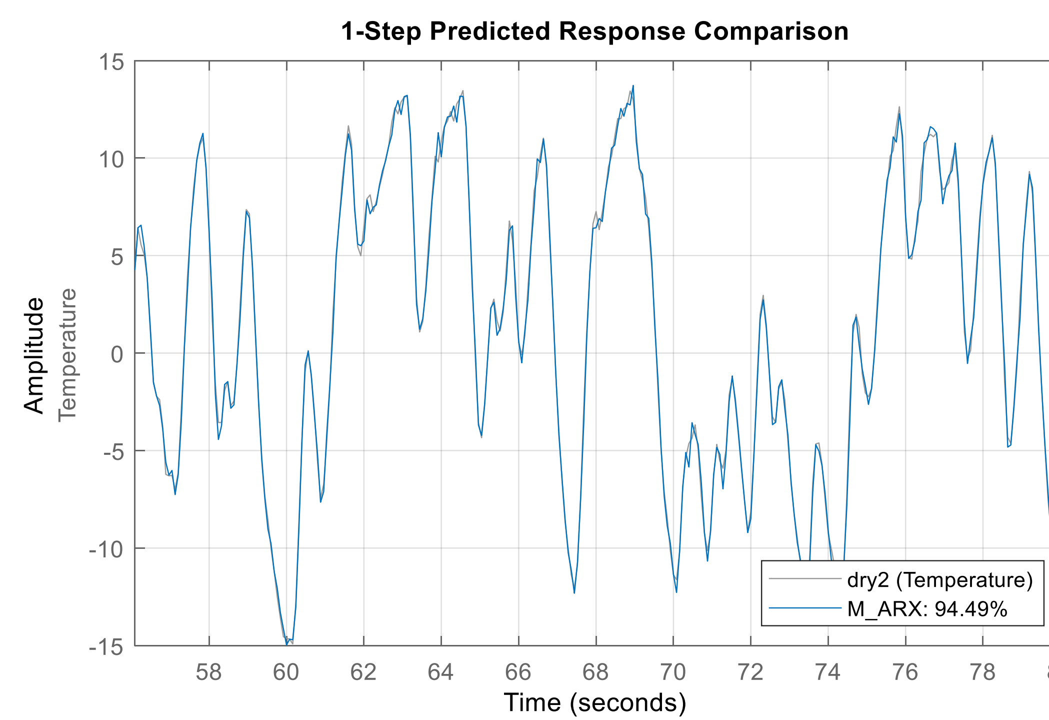
Here, we use the `compare` function in MATLAB to validate the model parameters estimation.

```
compare(datav,M_OE,1)      ← Compare OE model by validation data
compare(datav,M_ARX,1)     ← Compare ARX model by validation data
```

❑ **OE Model Validation:**

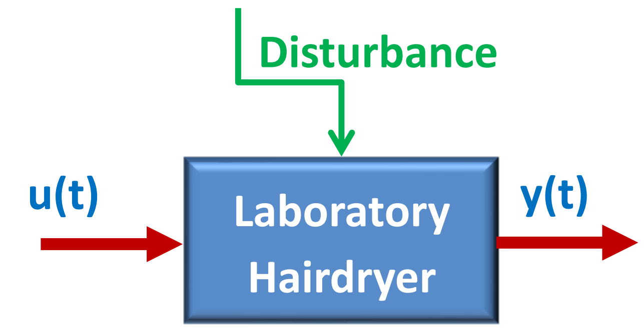


❑ **ARX Model Validation:**



- Since the ARX model shows better fit to the validation data, we can conclude that the data is more corrupted by process noise than the white noise.

Case Study: A Laboratory Scale Hairdryer



$$\{u(k)\,, y(k) |\; k = 1, \cdots, N\}$$

$$N = 1000, T_s = 0.08\text{sec}$$

Step 7: Determine the final DT and CT models

The ARX Model:

$$y(k) = \frac{0.6552 + 0.4324q^{-1}}{1 - 1.288q^{-1} + 0.4053q^{-2}} u(k-3) + \frac{1}{1 - 1.288q^{-1} + 0.4053q^{-2}} e(k)$$

- Convert DT model to CT model using **d2c** command in MATLAB:

```
mc_arx = d2c(M_ARX,'zoh')        ← convert DT ARX model to CT model G(s)
```

```
mc_arx =
Continuous-time ARMAX model: A(s)y(t) = B(s)u(t) + C(s)e(t)
  A(s) = s^2 + 11.29 s + 28.2
  B(s) = 0.5902 s + 261.9
  C(s) = s^2 + 26.56 s + 240.8

Input delays (listed by channel): 0.16
```

$$T_d = (n_k - 1)T_s = 2 \times 0.08 = 0.16sec$$

The CT Transfer Function Model

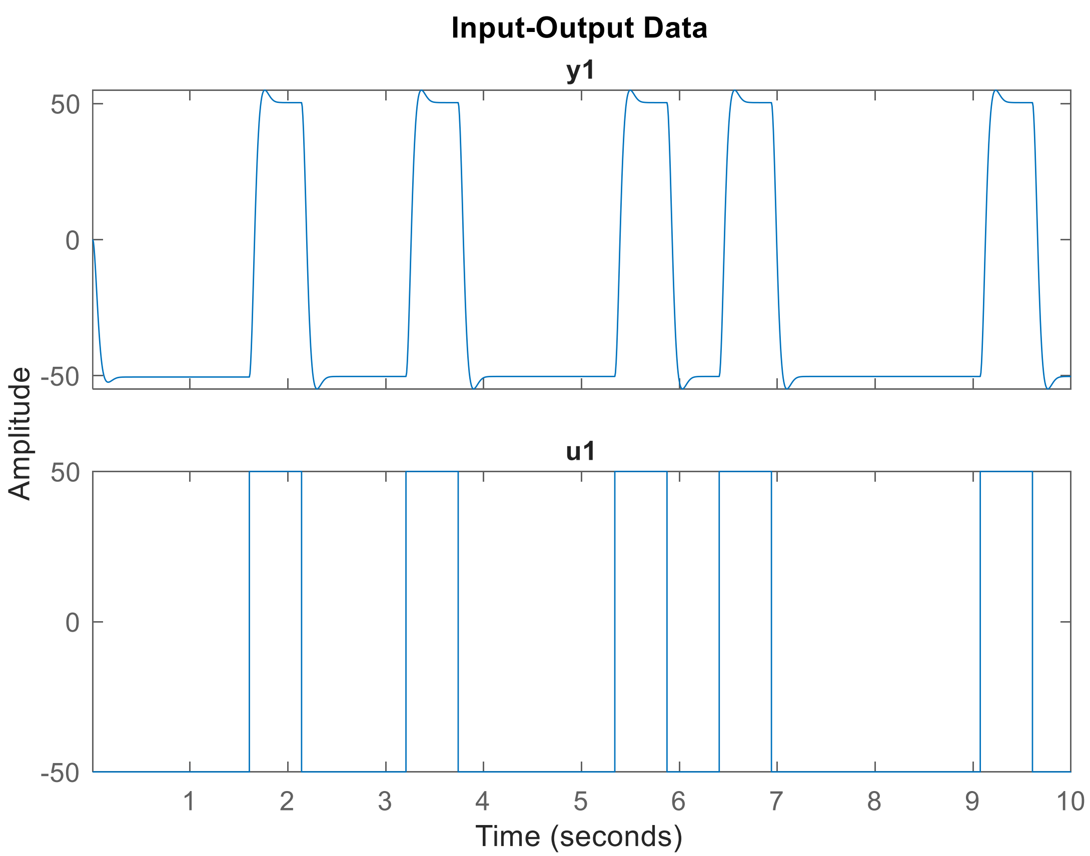
$$G(s) = e^{-0.16s} \frac{0.5902s + 261.9}{s^2 + 11.29s + 28.2}$$

Identification of CT Transfer Function Models

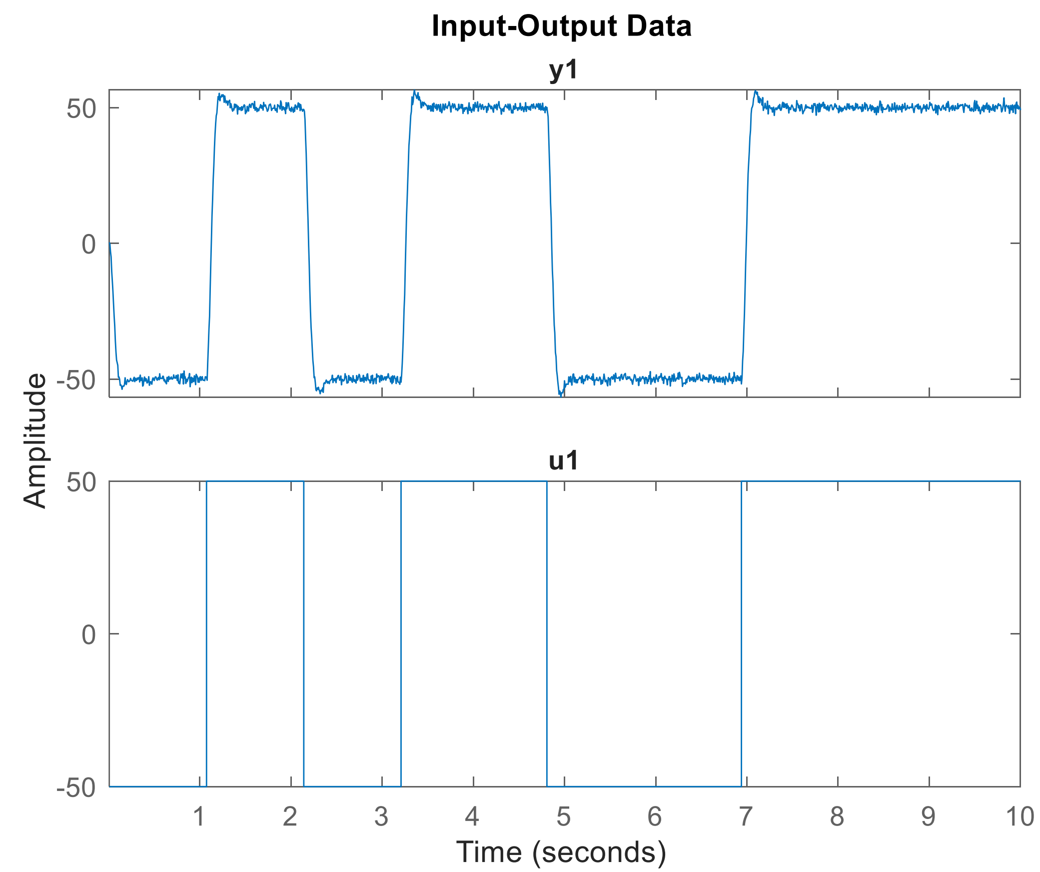
Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Assume that two sets of noisy and noise-free data has been collected from a closed-loop DC motor position control system. Input is the reference angular position in degrees, Output is the angular displacement of the DC motor.

Noise-free Dataset



Noisy Dataset



Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Step 1: Collect and Examine the I/O Data for Outliers and Trends

```
Ts = 0.0067;                  % Sampling time

data = iddata(y,u,Ts);        % Noise-free I/O data object
plot(data)                    % Plot noise-free I/O data

datan = iddata(yn,un,Ts);     % Noise-free I/O data object
plot(datan)                   % Plot raw I/O data
```

Step 2: Split the I/O Data to the Identification Data and Validation Data

- Here, we have $N = 1500$ samples of I/O data. The first 1000 samples are used for identification (estimation the model parameters), the rest for validation.

```
datai = data(1:1000);         % Noise-free identification data
datav = data(1001:end);       % Noise-free validation data

datani = datan(1:1000);       % Noisy identification data
datanv = datan(1001:end);     % Noisy validation data
```

Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Noise-free Data

Step 3: Select the parametric model structure, estimate the model.

Here, we can consider both second-order and third-order transfer function models to compare.

❑ Second-order TF Model Estimation:

```
np = 2;
nz = 1
sys_TF2 = tfest(datai,np,nz,nan);
```

```
sys_TF2 =
  From input "u1" to output "y1":
    -1.598 s + 798.4
  --------------------
  s^2 + 39.4 s + 792.2

Continuous-time identified transfer function.
Parameterization:
   Number of poles: 2   Number of zeros: 1
   Number of free coefficients: 4
```

$$G(s) = \frac{-1598s + 798.4}{s^2 + 39.4s + 792.2}$$

❑ Third-order TF Model Estimation:

```
np = 3;
nz = 1
sys_TF3 = tfest(datai,np,nz,nan);
```

```
sys_TF3 =
  From input "u1" to output "y1":
          538.1 s + 9.014e04
  --------------------------------
  s^3 + 148 s^2 + 5170 s + 8.94e04

Continuous-time identified transfer function.
Parameterization:
   Number of poles: 3   Number of zeros: 1
   Number of free coefficients: 5
```

$$G(s) = \frac{538.1s + 90140}{s^3 + 148s^2 + 5170s + 89400}$$

Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Noise-free Data

Step 4: Validate the model.

Here, we use the `present` function in MATLAB to validate the TF model parameters estimation.

❑ **Second-order TF Model Validation:**

```
present(sys_TF2)
```

```
Sys_TF2 =

  From input "u1" to output "y1":
   -1.598 (+/- 0.06281) s + 798.4 (+/- 2.801)
  ---------------------------------------------
  s^2 + 39.4 (+/- 0.1177) s + 792.2 (+/- 2.721)

Parameterization:
   Number of poles: 2   Number of zeros: 1
   Number of free coefficients: 4


Estimated using TFEST on time domain data "datai”
Fit to estimation data: 99.62%
FPE: 0.028, MSE: 0.02766
```

❑ **Third-order TF Model Validation:**

```
present(sys_TF3)
```

```
sys_TF3 =

  From input "u1" to output "y1":
                  538.1 (+/- 18) s + 9.014e04 (+/- 8751)
  --------------------------------------------------------------------
  s^3 + 148 (+/- 11.1) s^2 + 5170 (+/- 431.3) s + 8.94e04 (+/- 8710)

Parameterization:
   Number of poles: 3   Number of zeros: 1
   Number of free coefficients: 5


Estimated using TFEST on time domain data "datai".
Fit to estimation data: 99.71%
FPE: 0.01566, MSE: 0.01541
```

- The third-order model shows <u>better fit to estimation data</u>, but the parameter confidence intervals are large.

Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

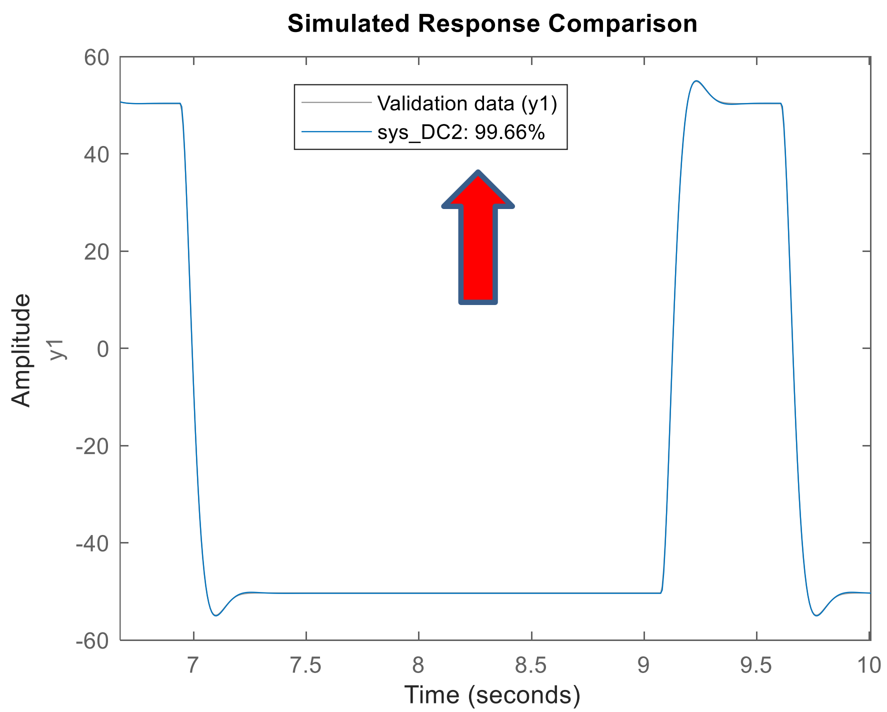
Noise-free Data

Step 4: Validate the model.

Here, we use the `compare` function in MATLAB to validate the estimated TF model using the validation dataset.

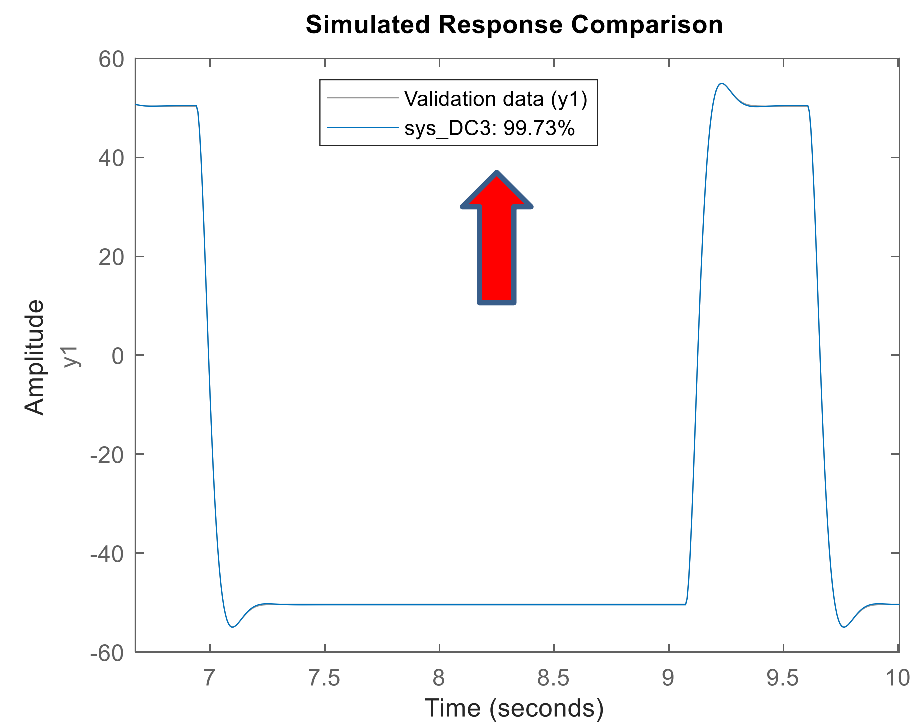
❑ **Second-order TF Model Validation:**

```
compare(sys_TF2,datav,1)
```



❑ **Third-order TF Model Validation:**

```
compare(sys_TF3,datav,1)
```



- The third-order model gives slightly better results. However, there is no significant difference between the models.

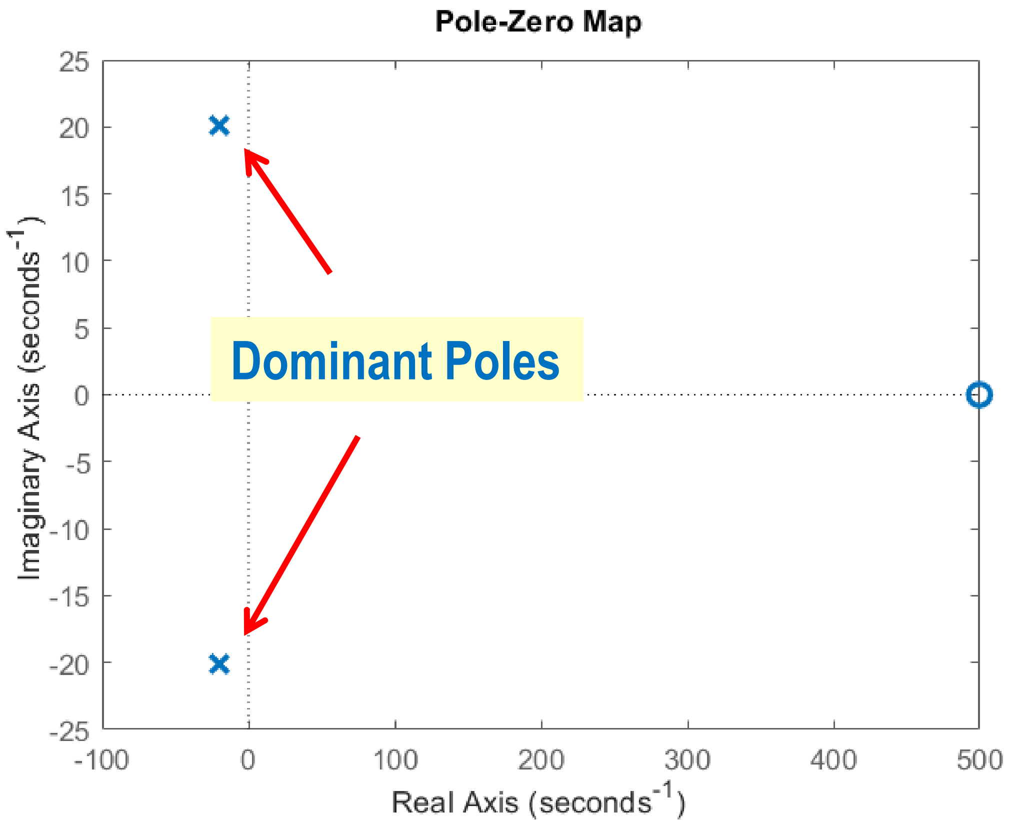
Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

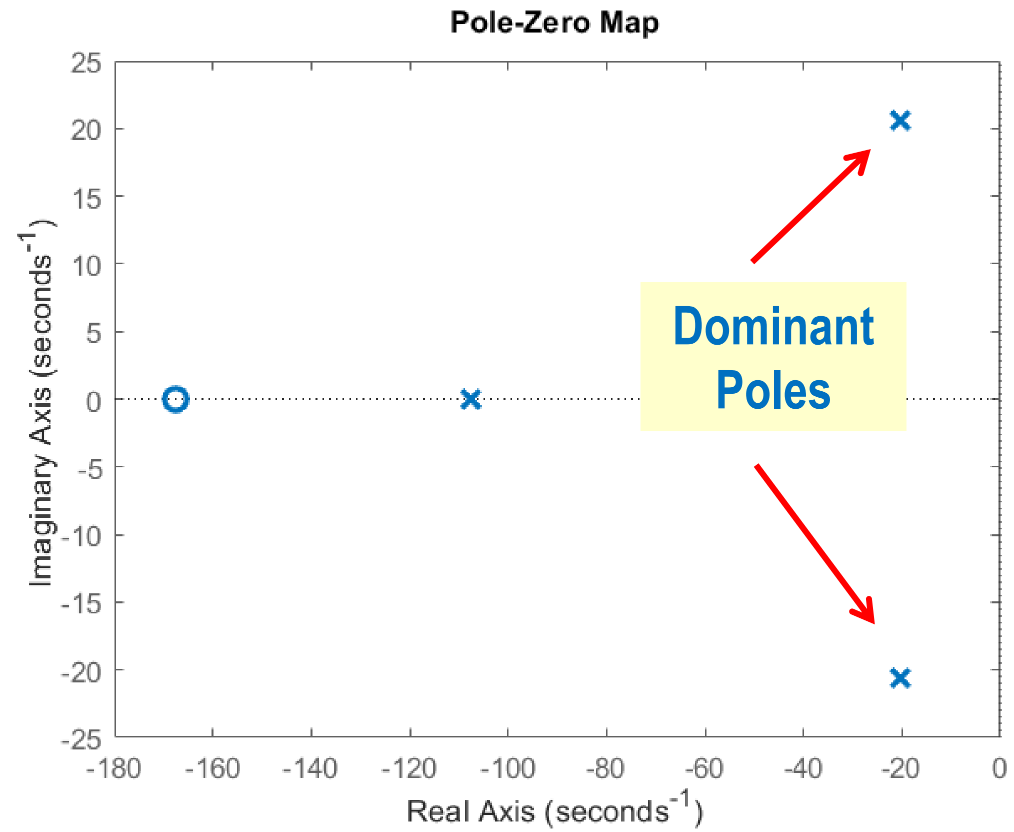
Noise-free Data

Step 5: Check the pole/zero locations for G(s)

❑ **Second-order TF Model:**



❑ **Third-order TF Model:**



- The pole/zero plots represent that system can be estimated as a second-order TF model with no zeroes.

Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Noise-free Data

Step 6: Modify the Model Order and Estimate and Validate the Model

❑ **Second-order TF Model with No Zero Estimation:**

```
np = 2;
nz = 0
sys_TF = tfest(datai,np,nz,nan);
```

```
sys_TF =
  From input "u1" to output "y1":
            740.4
  --------------------
  s^2 + 37.5 s + 734.6

Continuous-time identified transfer function.
Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
```

$$G(s) = \frac{740.4}{s^2 + 37.5s + 734.6}$$

```
sys_TF =

  From input "u1" to output "y1":
           740.4 (+/- 2.556)
  ---------------------------------------------
  s^2 + 37.5 (+/- 0.1368) s + 734.6 (+/- 2.437)

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Fit to estimation data: 99.24%
   FPE: 0.1086, MSE: 0.1075

Estimated using TFEST on time domain data "datai".
```

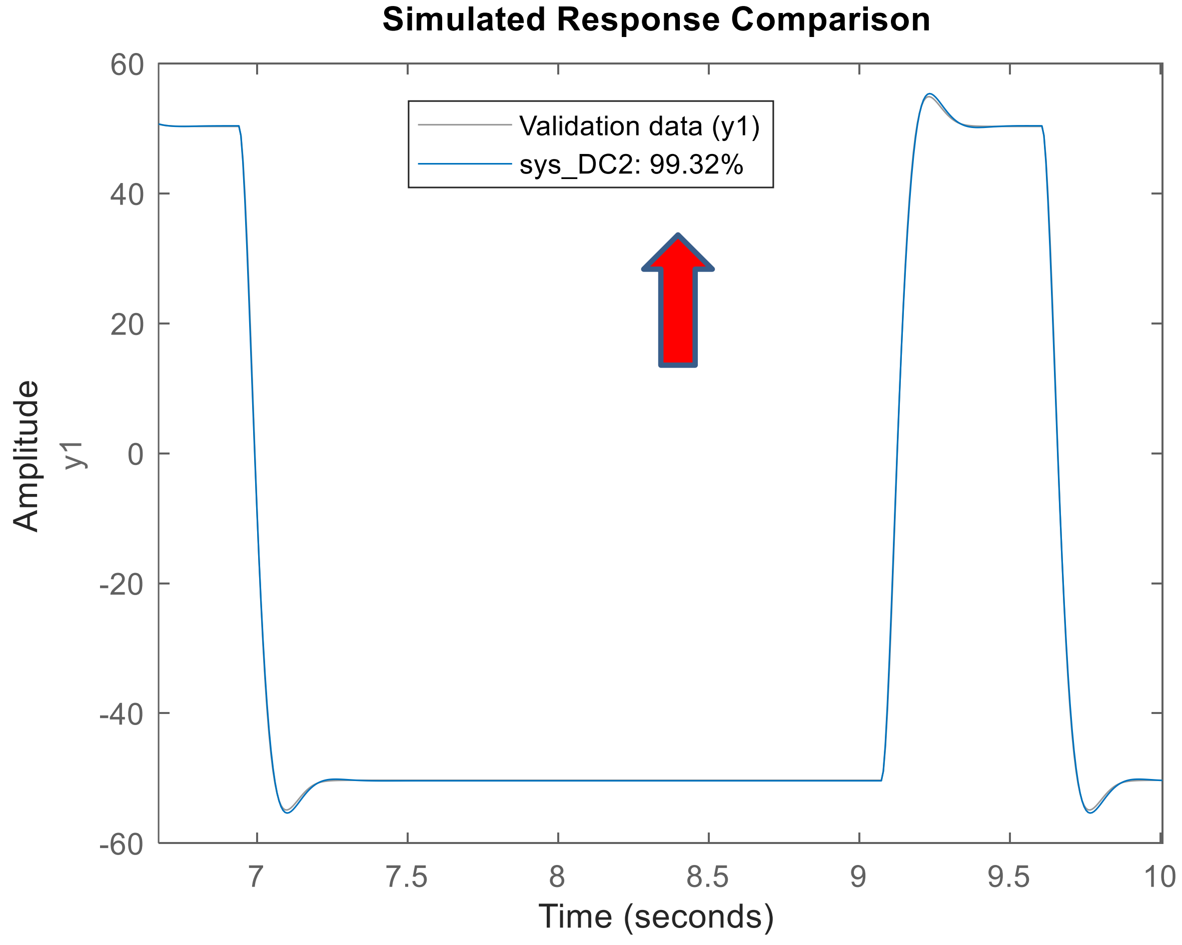
Identification of CT Transfer Function Models

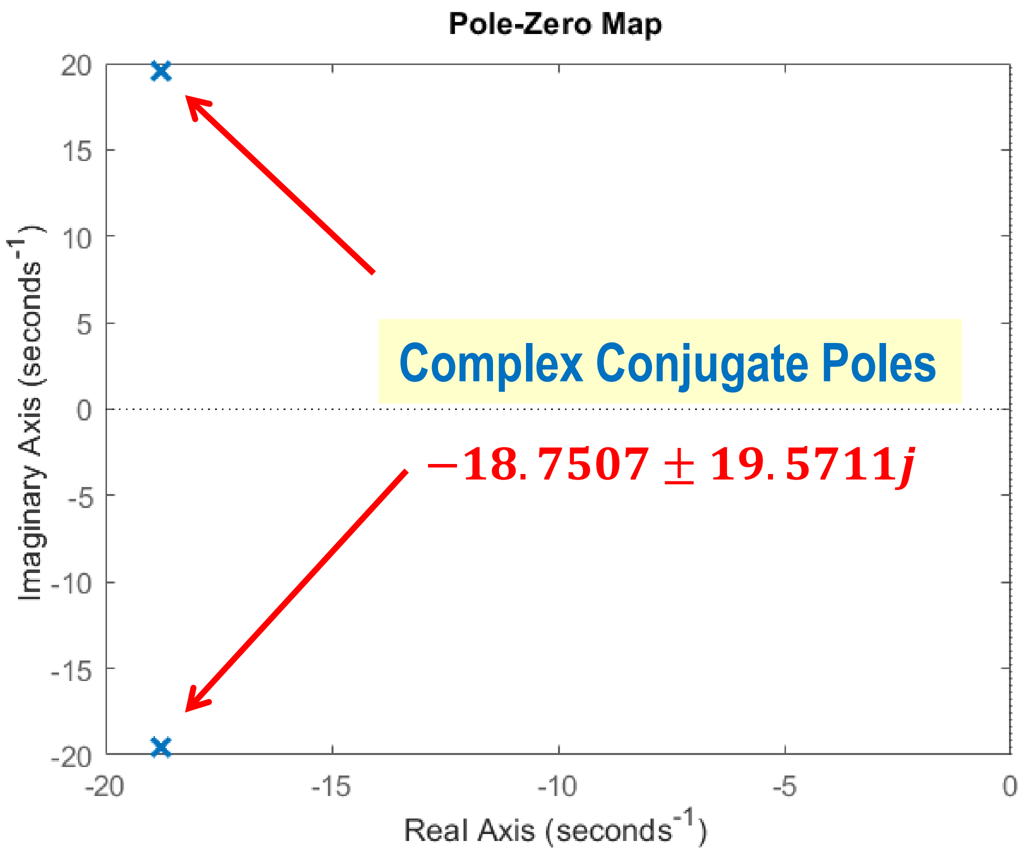
Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Noise-free Data

Step 6: Modify the Model Order and Estimate and Validate the Model

❑ **Second-order TF Model with No Zero Validation:**





Identification of CT Transfer Function Models

Example 5 This example shows how to develop a simple CT transfer function model from input-output data.

Noisy Data

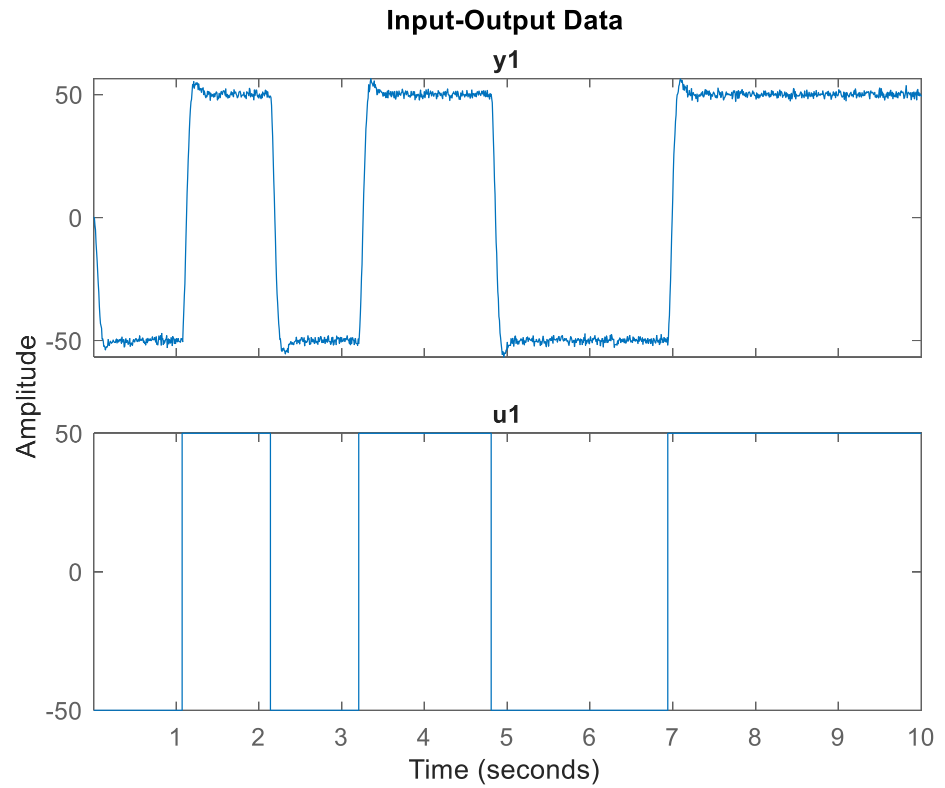
Step 3: Select the parametric model structure, estimate and validate the model.

Here, we can consider a second-order transfer function with no zero to compare with the noise-free case.

❑ Second-order TF Model Estimation:

```
np = 2;
nz = 0
sysn_TF = tfest(datani,np,nz);
```

```
sysn_TF =
  From input "u1" to output "y1":
         733.1
  ---------------------
  s^2 + 36.96 s + 732.9

Continuous-time identified transfer function.
Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3

Estimated using TFEST on time domain data "datani"
Fit to estimation data: 97.85%
FPE: 1.085, MSE: 1.074
```



$$G(s) = \frac{733.1}{s^2 + 36.96s + 732.9}$$

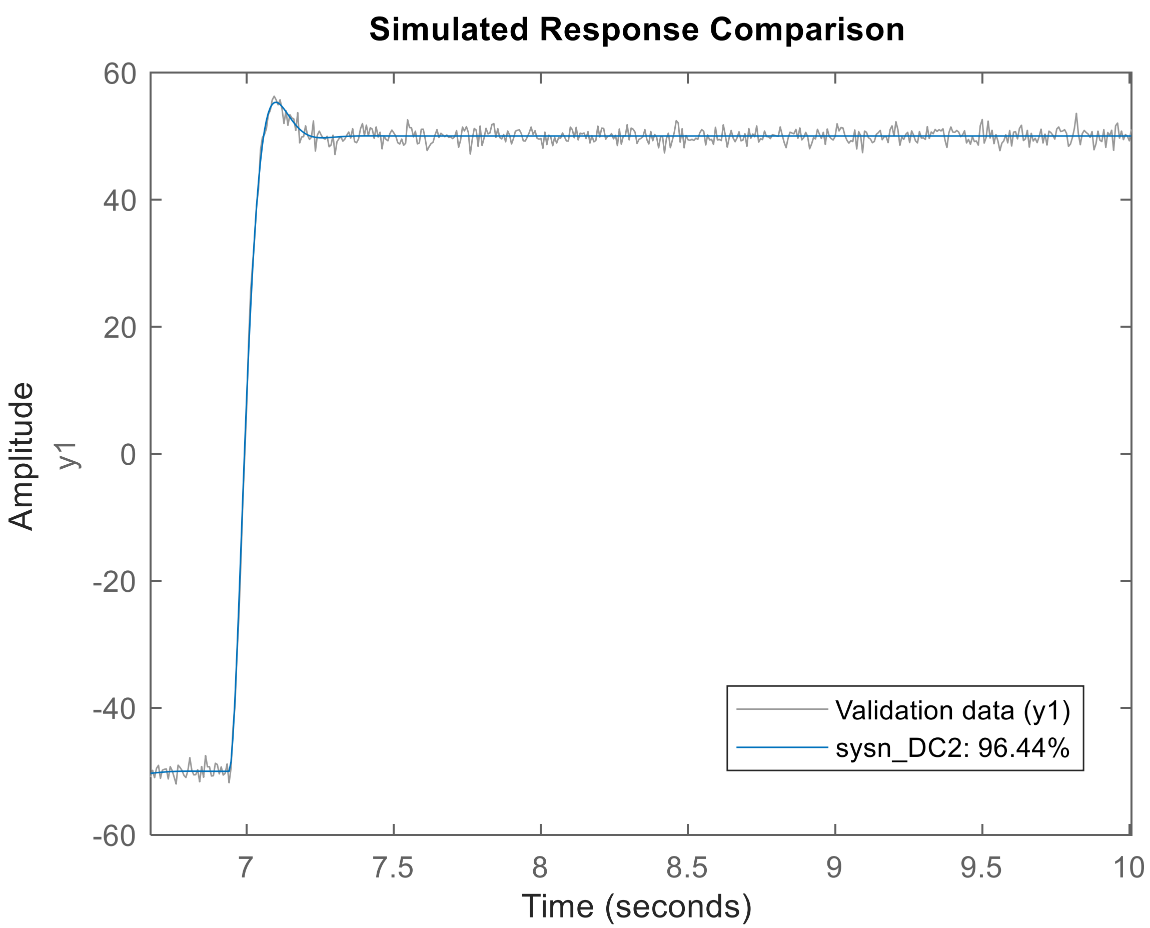
Identification of Transfer Function Models

Example 5 This example shows how to develop a simple continuous-time transfer function model from experimental input-output data.

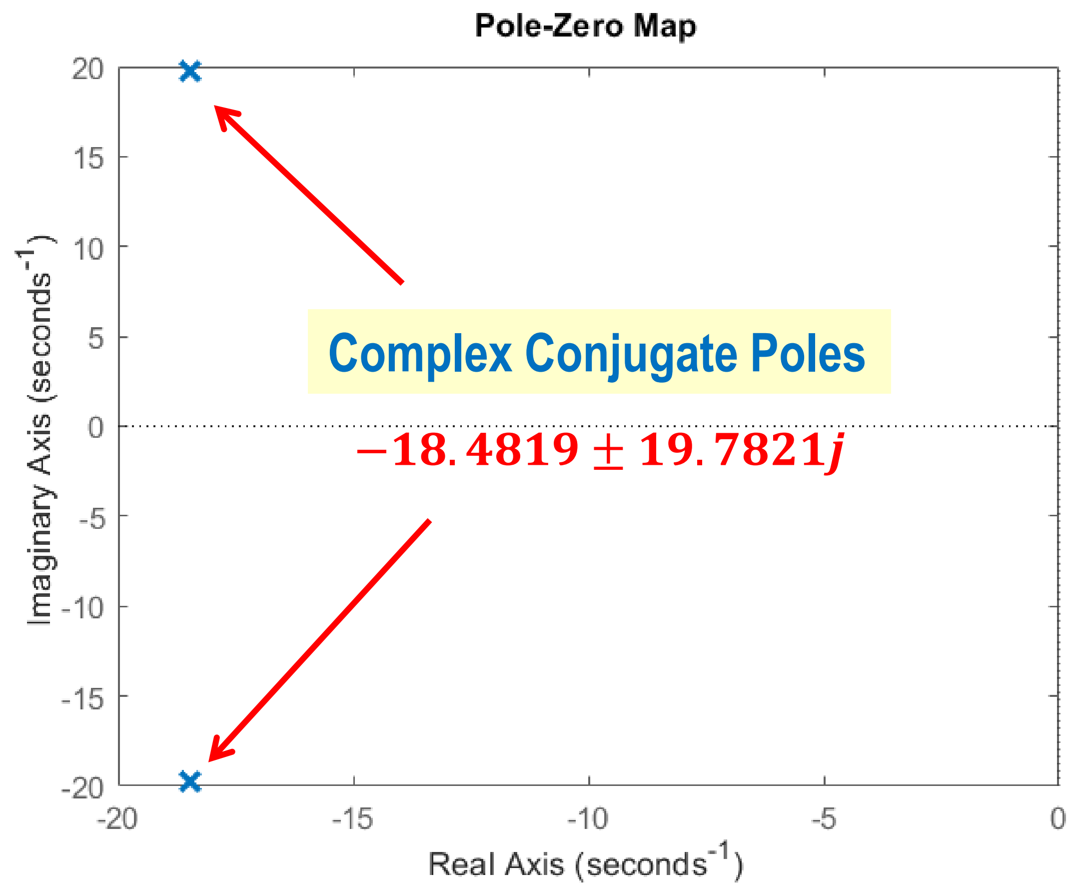
Step 3: Select the parametric model structure, estimate and validate the model.

Noisy Data

❑ **Second-order TF Model Validation:**



❑ **Second-order TF Model Pole/Zero Map:**



THANK YOU

WE ARE HUMBER