HUMBER ENGINEERING

MENG-3020
SYSTEMS MODELING & SIMULATION

LECTURE 11

HUMBER

WE ARE HUMBER

LECTURE 11
Model Validation Methods

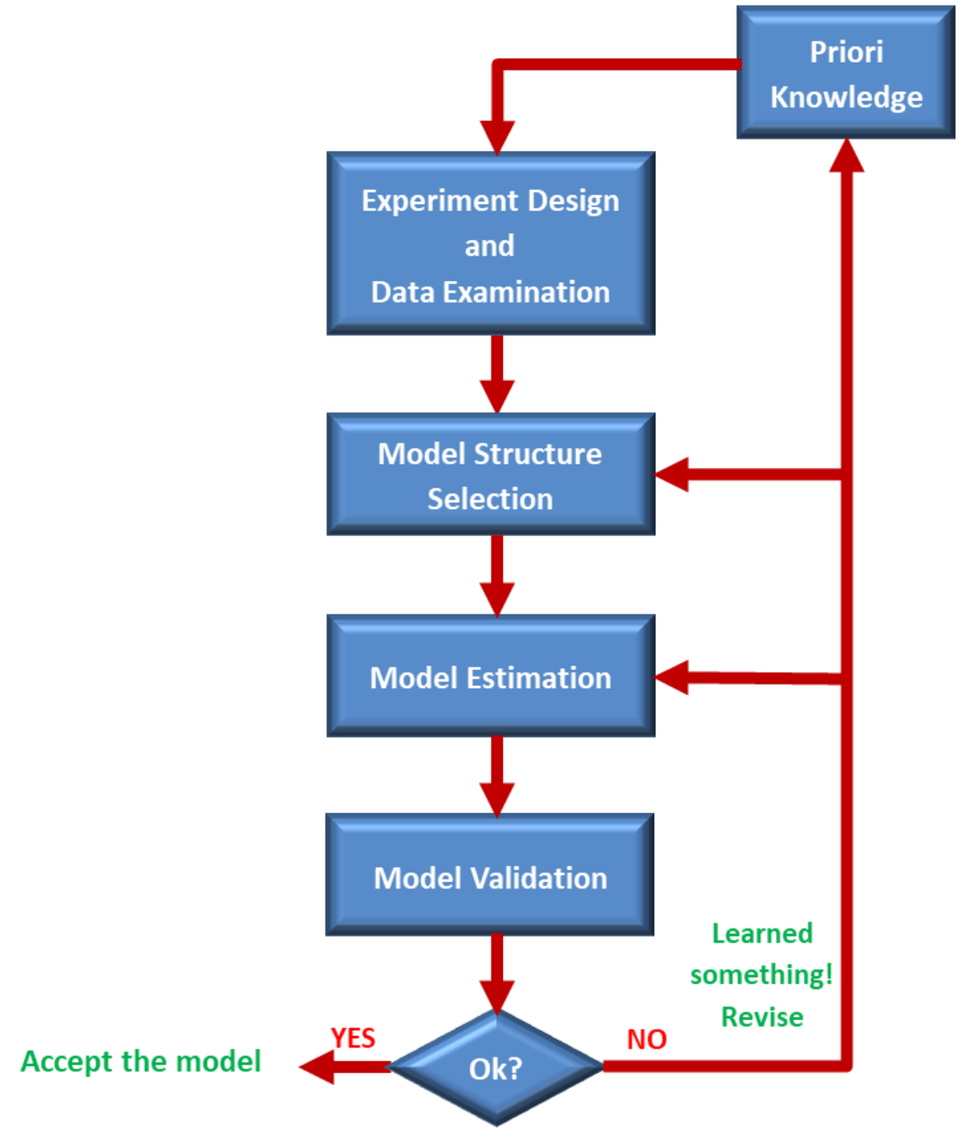
- Simulation and Cross-Validation
  - Prediction Error Method
- Model Validity Criterion
  - Akaike's Final Prediction Error (FPE)
  - Mean-Squares Error (MSE)
  - Parameters Confidence Intervals
- Pole-Zero Plots
- Residual Analysis

System Identification Procedure

❑ Model Validation Techniques

- The procedures that asses the quality of a model are generally called Model Validation techniques.

- **Simulation & Cross Validation**
  - Prediction Error Method
- **Model Validity Criteria**
  - Parameters Confidence Intervals
  - Final Prediction Error (FPE)
  - Mean-squared Error (MSE)
- **Pole-Zero Plots**
- **Residual Analysis**
  - Whiteness Test
  - Independency Test



Simulation & Cross-Validation

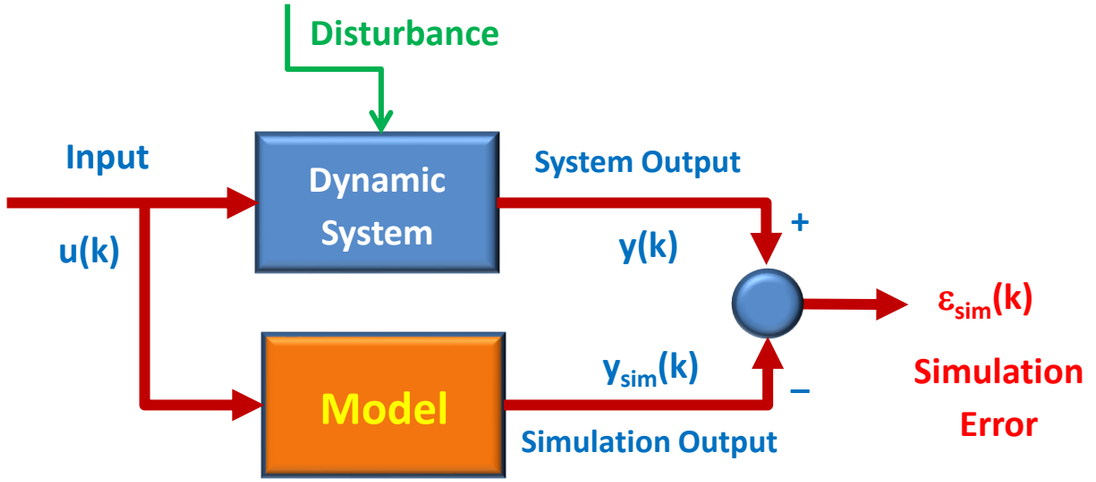
- Simulation and Cross-Validation are methods for testing whether a model can reproduce the observed output data when driven by the actual input.

❑ Simulation

- In simulation test we compare the desired model output (= system output), $y(k)$, with the simulated output by the model, $y_{sim}(k)$, and we only observe how much the plots of $y(k)$ and $y_{sim}(k)$ differ.

Simulation Error $\rightarrow$ $\boldsymbol{\varepsilon_{sim}(k) = y(k) - y_{sim}(k)}$

- In Simulation Test, there is a risk of **over-fitting** the data in noisy systems.
- Since over-fitting leads to a small simulation error, however, this does not mean that the model is accurate.
- To avoid the over-fitting issue the **Cross-validation** method is advised.



$$y_{sim}(k) = G(q)u(k) + H(q)e(k)$$

Simulation & Cross-Validation

❑ MATLAB Function for Simulation

- The Simulation approach can be applied by command **`sim`** in MATLAB

```
ysim = sim(data,sys)
```

Variable	Description
`ysim`	Simulated response
`sys`	Estimated or constructed dynamic system model
`data`	Identification data

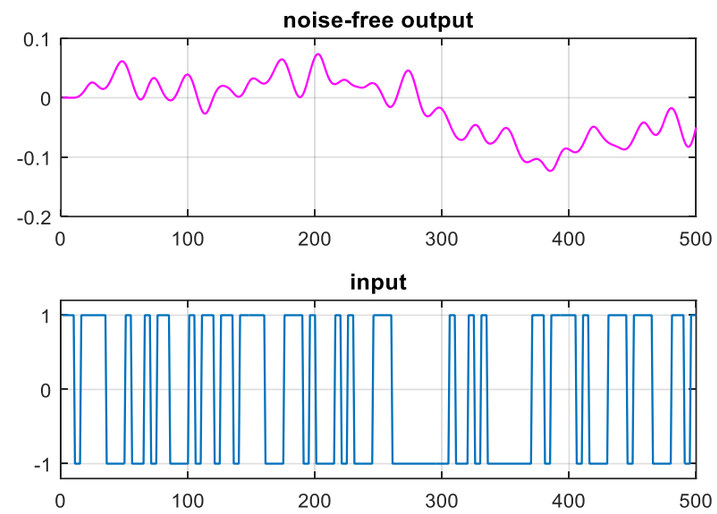
- **`sim`** returns the simulated response of an identified model **`sys`** based on the given identification data.

Simulation & Cross-Validation

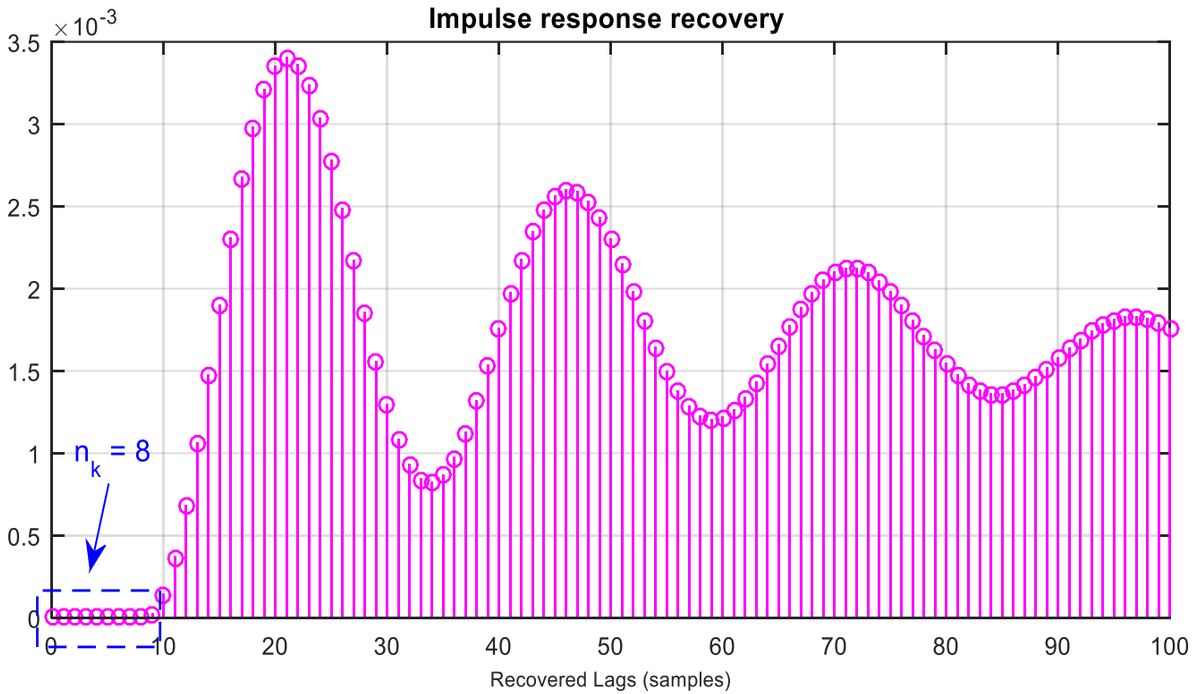
Example 1

Assume that the collected noise-free input-output data is available with $N = 500,\ T_s = 0.05\text{sec}.$

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



The impulse response coefficients are recovered as below



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

```
data = iddata(y,u,Ts);
na = 3;
nb = 3;
nk = 8;
M_ARX = arx(data,[na nb nk]);
```

Assume that the system is identified by an ARX model structure

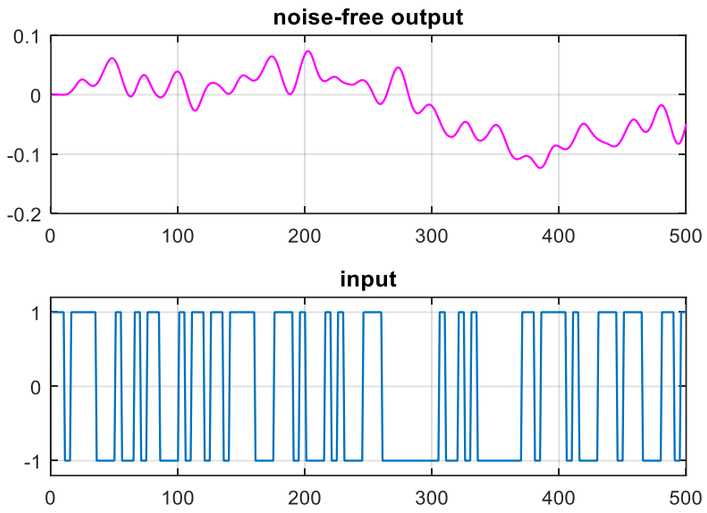
$$[n_a \quad n_b \quad n_k\ ] = [3 \quad 3 \quad 8]$$

Simulation & Cross-Validation

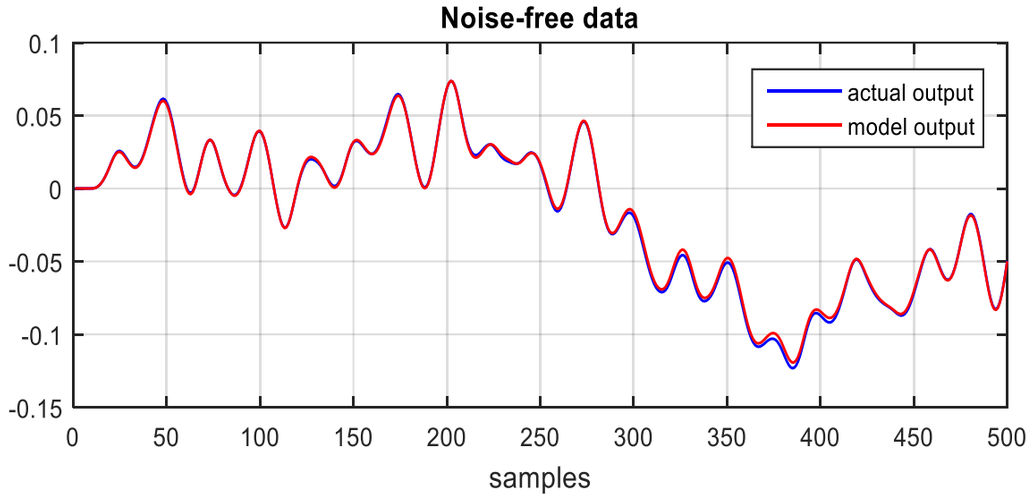
Example 1

Assume that the collected noise-free input-output data is available with $N = 500,\ T_s = 0.05\text{sec}.$

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



The identified model can be tested with the I/O data in a simulation using the **`sim`** function in MATLAB:




```
data = iddata(y,u,Ts);
na = 3;
nb = 3;
nk = 8;
M_ARX = arx(data,[na nb nk]);

ysim = sim(data,M_ARX);
plot([y ysim])
e_sim = norm(y-ysim);
```

- In noise-free case, we can easily compare the actual system output and the model output via simulation visually.

$$\varepsilon_{sim}(k) = y(k) - y_{sim}(k) \quad \rightarrow \quad \|\varepsilon_{sim}(k)\| = 0.0359$$

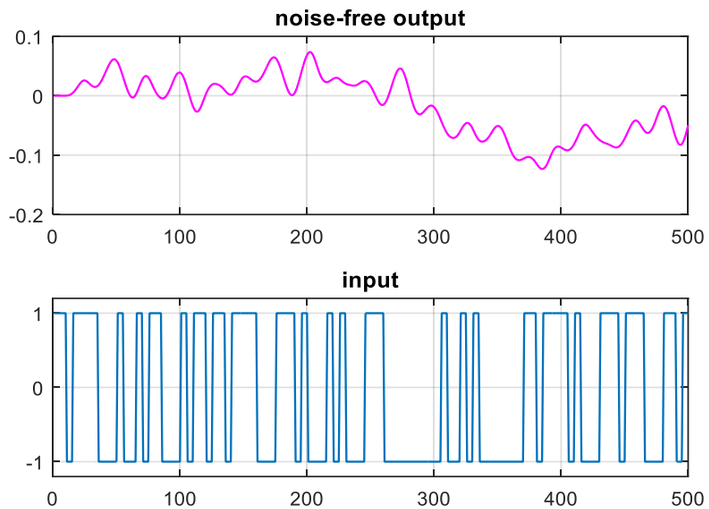
Since the simulation error is small the results are acceptable for noise-free data.

Simulation & Cross-Validation

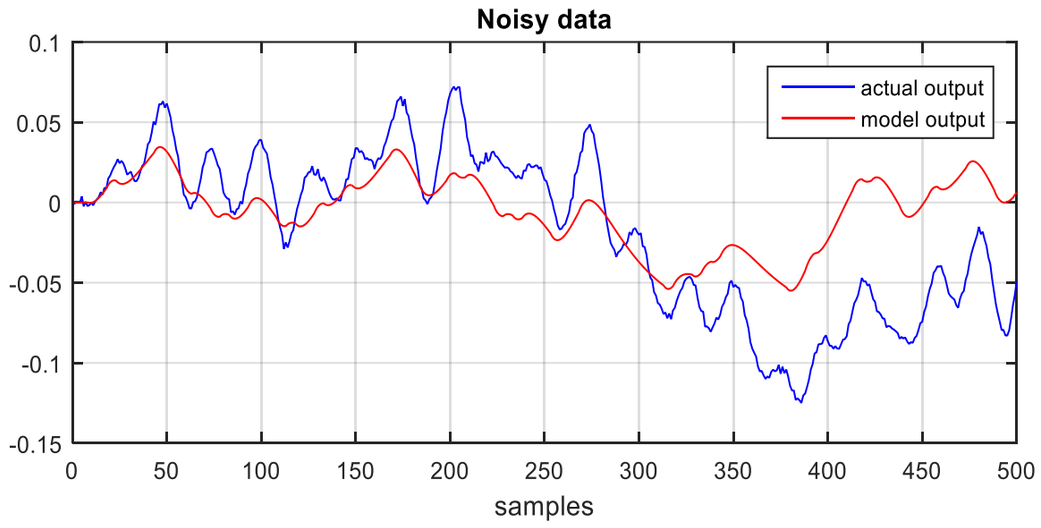
Example 1

Assume that the collected noise-free input-output data is available with $N = 500,\ T_s = 0.05\text{sec}.$

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



Assume that additive white noise is added to output signal with SNR=30dB. Then, the system is identified by noisy data set and tested with the I/O data in a simulation by using the **`sim`** function in MATLAB:



- In noisy case, the simulation error between the actual output and the model output is greater than the noise-free case.

$$\varepsilon_{sim}(k) = y(k) - y_{sim}(k) \ \rightarrow \ \|\varepsilon_{sim}(k)\| = 0.9196$$

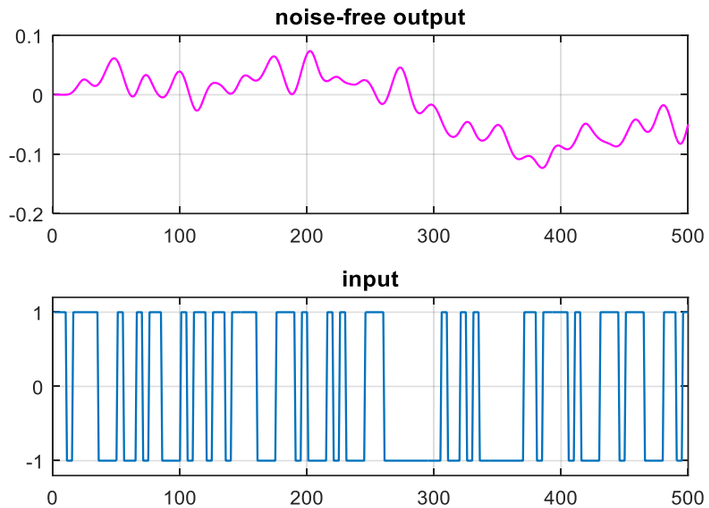
```
SNR = 30;
yn = awgn(y,SNR,'measured');
datan = iddata(yn,u,Ts);
na = 3;
nb = 3;
nk = 8;
M_ARXwn = arx(datan,[na nb nk]);

ynsim = sim(datan,M_ARXwn);
plot([yn ynsim])
e_sim = norm(yn-ynsim);
```

Simulation & Cross-Validation

Example 1

Assume that the collected noise-free input-output data is available with $N = 500,\ T_s = 0.05\text{sec}.$

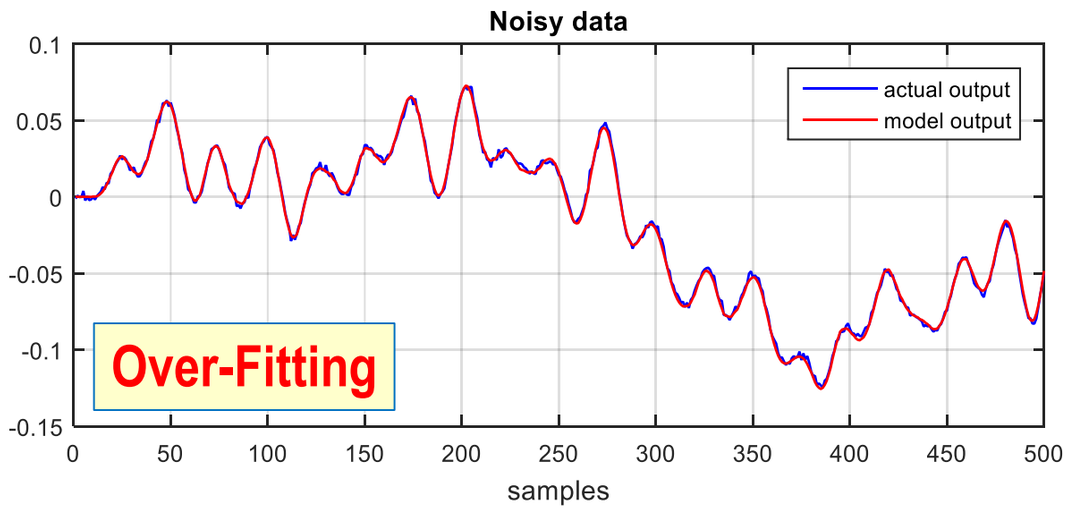
$$\{u(k)\ , y(k) |\ k = 1, \cdots, N\}$$



In noisy cases, usually increasing the model order decreases the simulation error. However, it leads to **over-fitting** the data.

- For example, by considering the following ARX structure the simulation error between model output and actual output decreases.

$$[n_a \quad n_b \quad n_k\ ] = [15 \quad 15 \quad 8]$$



$$\varepsilon_{sim}(k) = y(k) - y_{sim}(k) \ \rightarrow \ \|\varepsilon_{sim}(k)\| = 0.0431$$

```
SNR = 30;
yn = awgn(y,SNR,'measured');
datan = iddata(yn,u,Ts);
na = 15;
nb = 15;
nk = 8;
M_ARXwn = arx(datan,[na nb nk]);

ynsim = sim(datan,M_ARXwn);
plot([yn ynsim])
e_sim = norm(yn-ynsim);
```

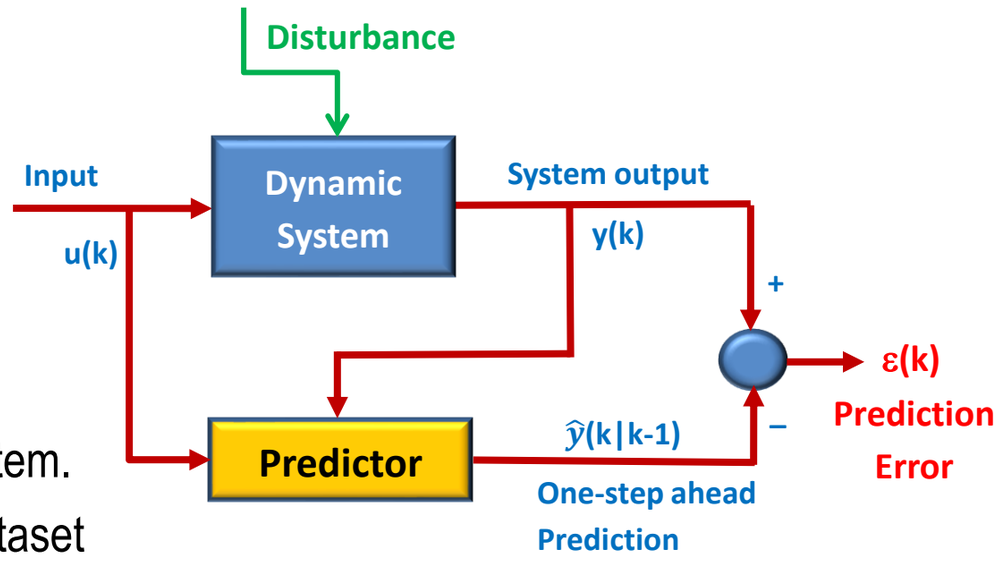
Simulation & Cross-Validation

❑ Cross-Validation

- Cross-validation is the procedure that compares the quality of identified models by validating them on a dataset where neither of them was estimated.
- In cross-validation method we split the I/O dataset Z in two subset:
  - Identification data (Z_{id}) → To compute the optimal parameters θ
  - Validation data (Z_{val}) → To check how the estimated model behaves on fresh data
- The validation criteria is defined based on minimizing the error between the desired model output (= system output) $y(k)$ and the one-step ahead predicted output $\hat{y}(k|k-1)$ for the estimated parameters θ by using the validation dataset.

One-step ahead Prediction Error → $\boldsymbol{\varepsilon(k) = y(k) - \hat{y}(k|k-1)}$

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



- No need to apply any statistical arguments and assumption about the system.
- Needs to save a fresh dataset for validation, and we cannot use all the dataset to identification.

Simulation & Cross-Validation

❑ MATLAB Function for Cross-Validation

- Cross-validation approach can be applied by command **`compare`** in MATLAB

```
compare(data,sys,kstep)
```

Variable	Description
`sys`	Estimated or constructed dynamic system model
`data`	Validation data
`kstep`	prediction horizon (default = 0)

- **`compare`** function plots the simulated (or predicted) response of a dynamic system model **`sys`** and superimposed over the validation data, **`data`**, for comparison.
- The plot also displays the Normalized Root Mean Square (NRMSE) measure of the goodness of the fit.

Simulation & Cross-Validation

Example 2

Assume the collected noisy input-output data from Example 1 with $N = 500,\ T_s = 0.05\text{sec}.$

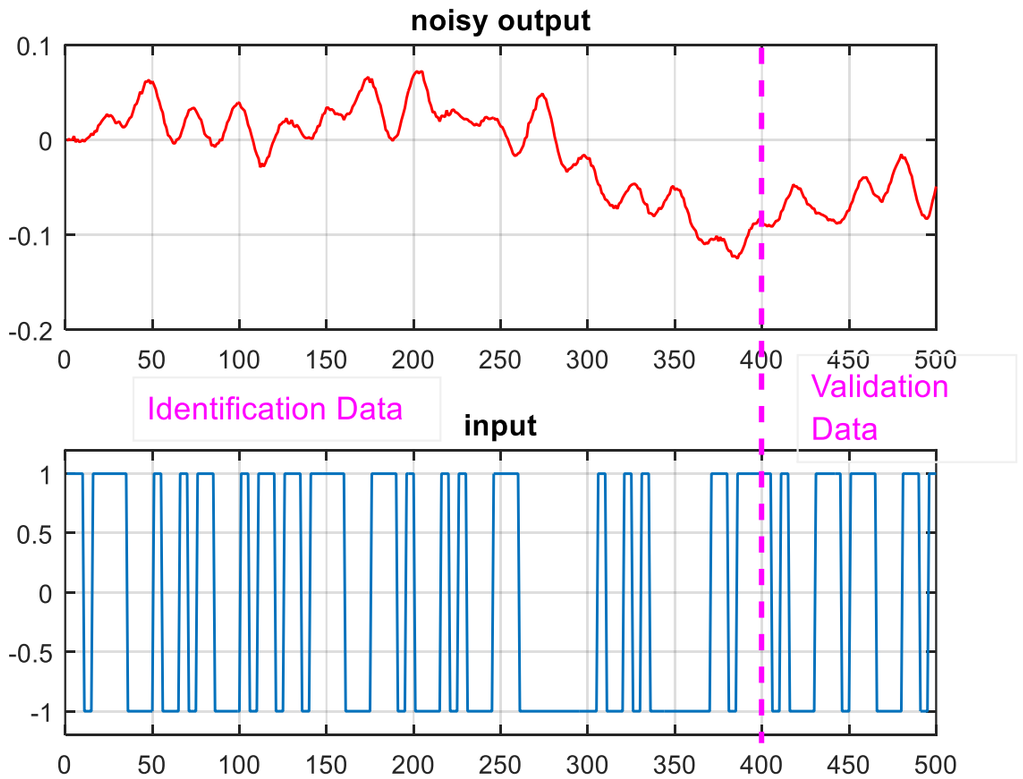
$$\{u(k)\ , y(k)|\ k = 1, \cdots, N\}$$

Here, we have $N = 500$ samples of I/O data. The first 400 samples are used for identification (estimation the model parameters), the rest of the data for validation.

$$N_{id} = 400, \qquad N_{val} = 100$$

```
datawn = iddata(yn,u,Ts);
datawni = datawn(1:400);
datawnv = datawn(401:N);
```

First, we identify the system by using the identification data and then validate the model by using the validation data and minimizing the one-step ahead prediction error.

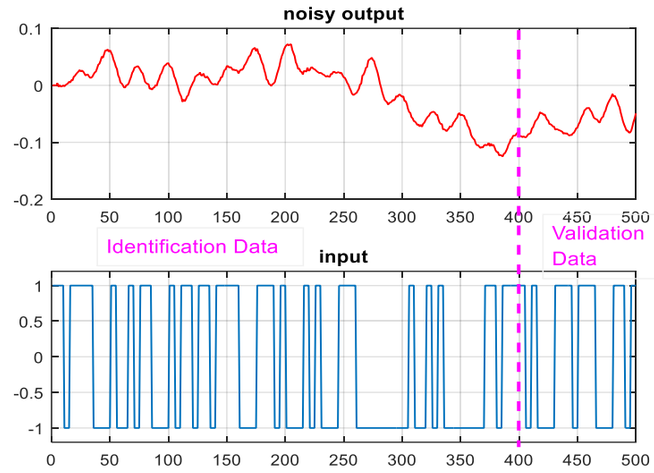


Simulation & Cross-Validation

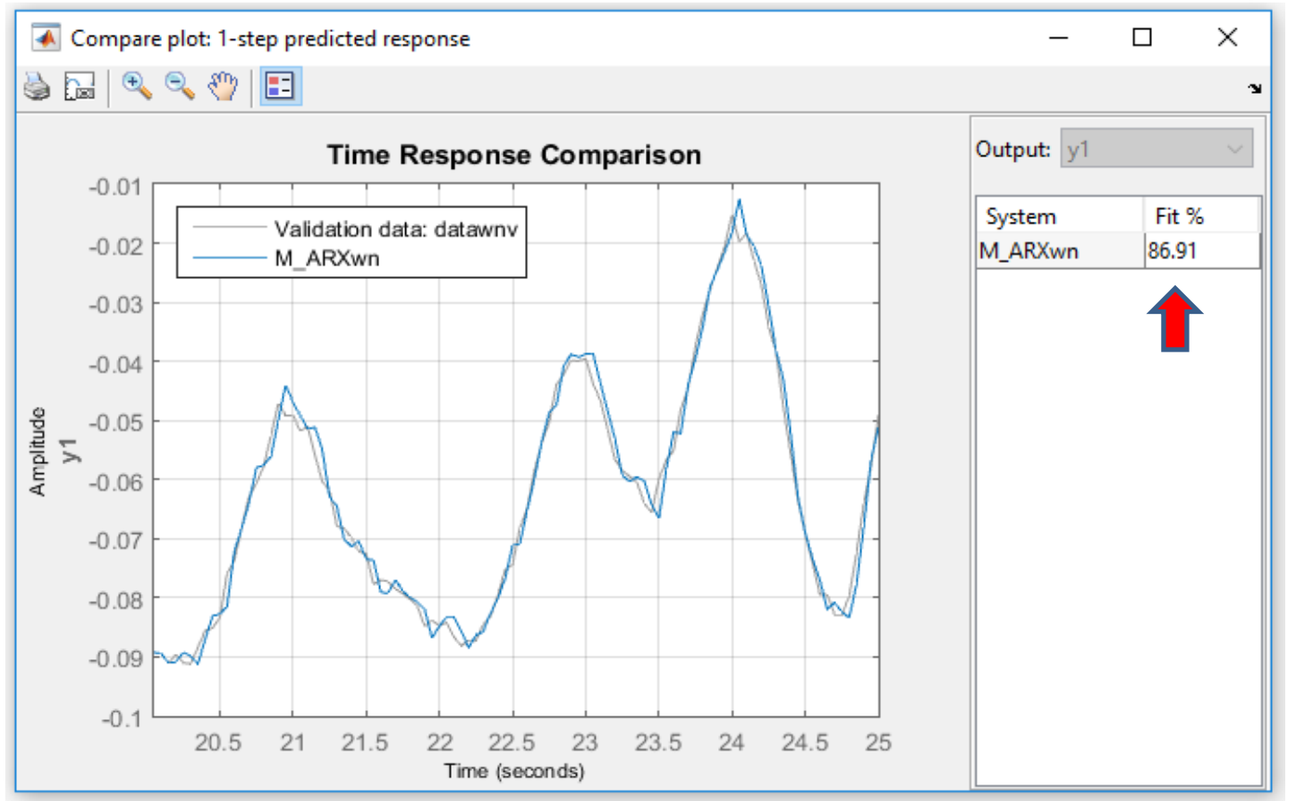
Example 2

Assume the collected noisy input-output data from Example 1 with $N = 500,\ T_s = 0.05\text{sec}.$

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



Case 1: ARX(3,3,8) Model Structure




```
datawn = iddata(yn,u,Ts);
datawni = datawn(1:400);
datawnv = datawn(401:N);
```

```
na = 3;
nb = 3;
nk = 8;
M_ARXwn = arx(datawni,[na nb nk]);
compare(datawnv,M_ARXwn,1)
```

The results show that using the Cross-validation method to validate the ARX(3,3,8) model gives better result (86.91% fit) without increasing the model order and no over-fitting issue.

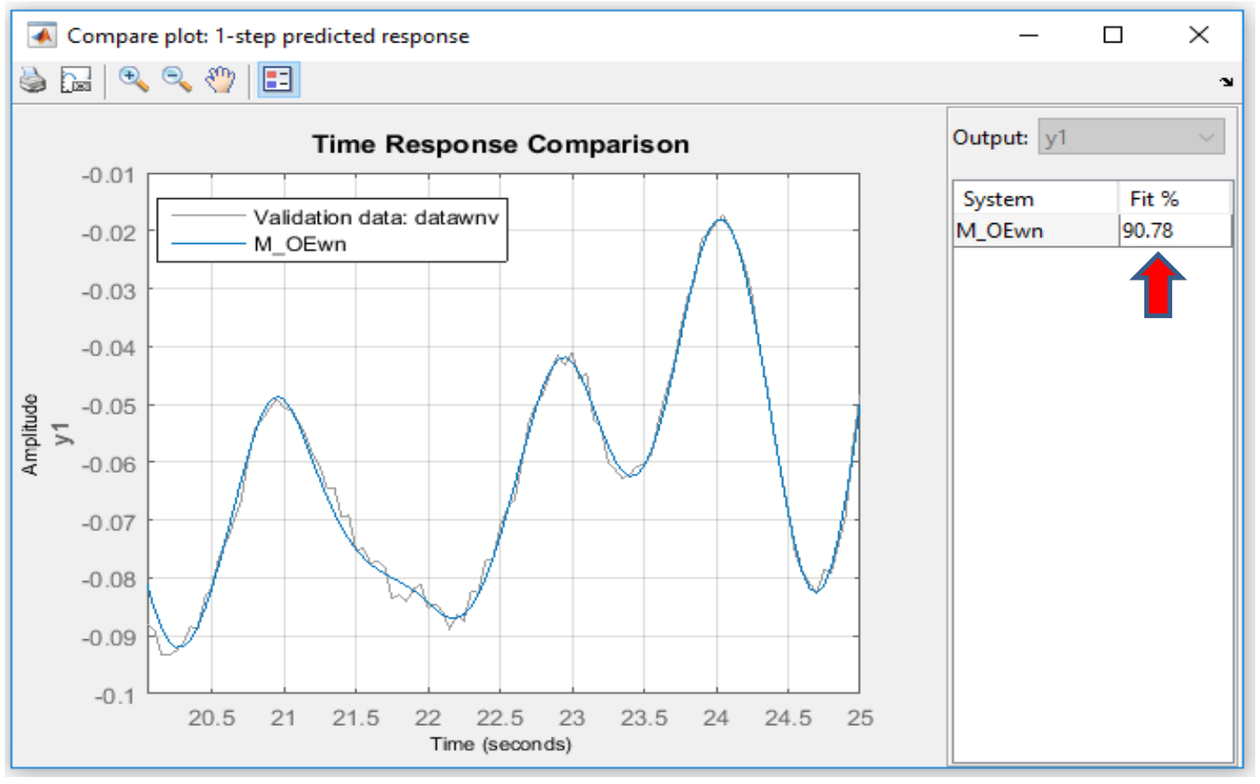
Simulation & Cross-Validation

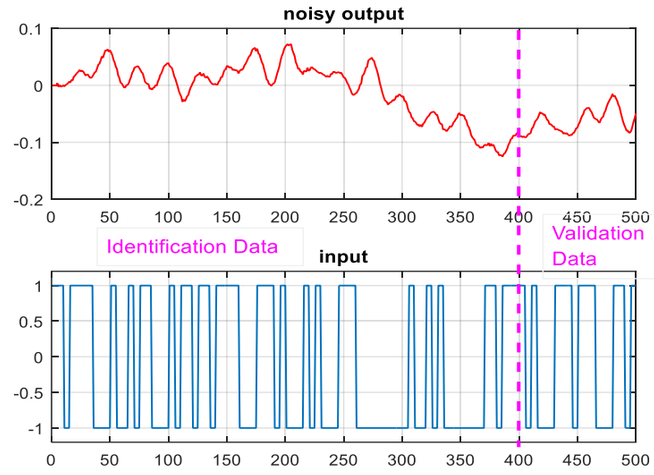
Example 2

Assume the collected noisy input-output data from Example 1 with $N = 500,\ T_s = 0.05\text{sec}.$

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

Case 2: OE(3,3,8) Model Structure






```
datawn = iddata(yn,u,Ts);
datawni = datawn(1:400);
datawnv = datawn(401:N);
```

```
nb = 3;
nf = 3;
nk = 8;
M_OEwn = oe(datawni,[nb nf nk]);
compare(datawnv,M_OEwn,1)
```

The results show that the OE(3,3,8) model gives better result (90.78% fit) than the ARX(3,3,8) model (86.91% fit). This is because the noise is the Additive White Noise not process noise.

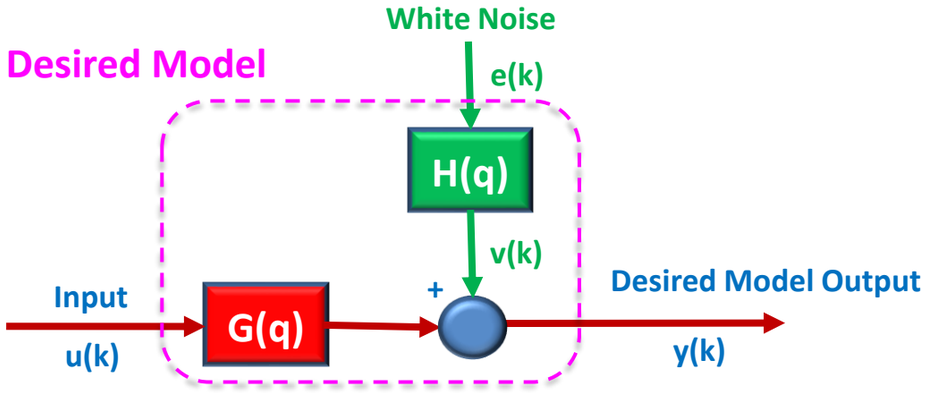
Prediction Error Method

- In Prediction Error Methods, we use a predictor to measure the performance of the corresponding model.
- The **Prediction Error** (or **Residuals**) is defined as the difference between the system output (= desired model output) $y(k)$ and the one-step ahead predicted output $\hat{y}(k|k-1)$ for the estimated parameters θ.

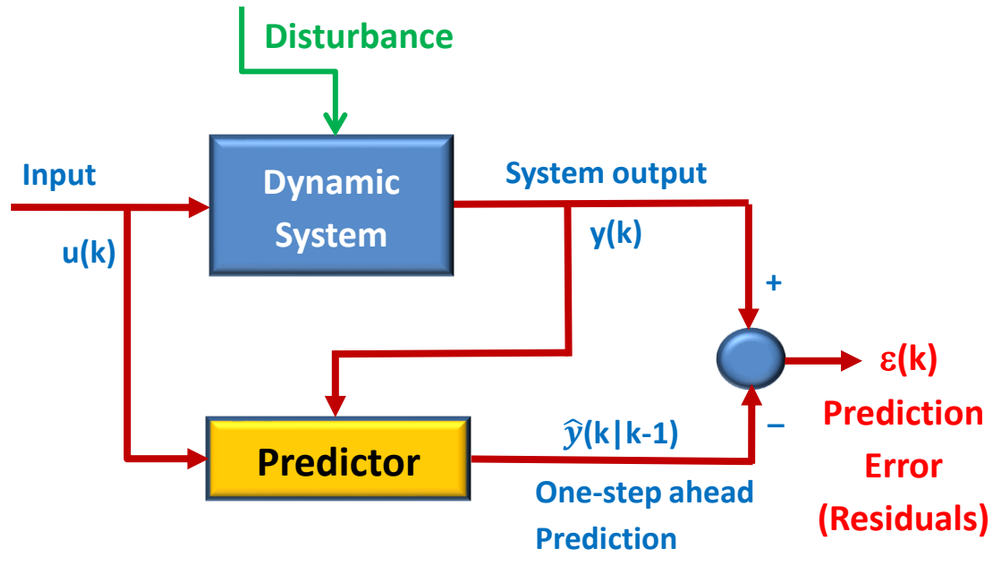
$$\textbf{Prediction Error} \rightarrow \varepsilon(k) = y(k) - \hat{y}(k|k-1)$$

Desired Model Output — One-step ahead Prediction

- The goal is to determine an optimal predictor such that the prediction error is minimized.



$$y(k) = G(q)u(k) + H(q)e(k)$$



Prediction Error Method

❑ Optimal Predictor

- The predictor can be defined as the linear combination of the past input-output data:

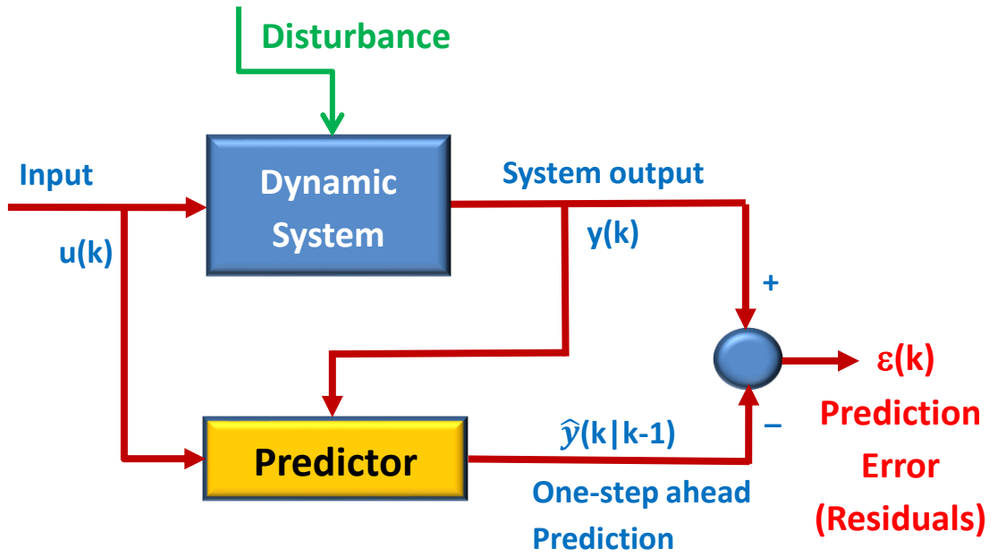
$$\hat{y}(k|k-1) = s_0 u(k) + s_1 u(k-1) + \cdots + s_{n_s} u(k-n_s) + t_1 y(k-1) + t_2 y(k-2) + \cdots + t_{n_t} y(k-n_t)$$

$$\hat{y}(k|k-1) = (s_0 + s_1 q^{-1} + s_2 q^{-2} + \cdots + s_{n_s} q^{-n_s}) u(k) + (t_1 q^{-1} + t_2 q^{-2} + \cdots + t_{n_t} q^{-n_t}) y(k)$$

Predictor $\hat{y}(k|k-1) = S(q)u(k) + T(q)y(k)$

- Since only the output, $y(k)$, is disturbed by noise, the term $S(q)u(k)$ contains information about the deterministic part of the predictor and term $T(q)y(k)$ introduces a stochastic component into the predictor.

What are the optimal filters for $S(q)$ and $T(q)$ to achieve the minimum prediction error?



Prediction Error Method

❑ Optimal Predictor

- The predictor can be defined as the linear combination of the past input-output data:

Predictor $\boldsymbol{\hat{y}(k|k-1) = S(q)u(k) + T(q)y(k)}$

- The optimal predictor is achievable if the difference between the output $y(k)$ and the predicted output $\hat{y}(k|k-1)$ is equal to the White Noise $e(k)$

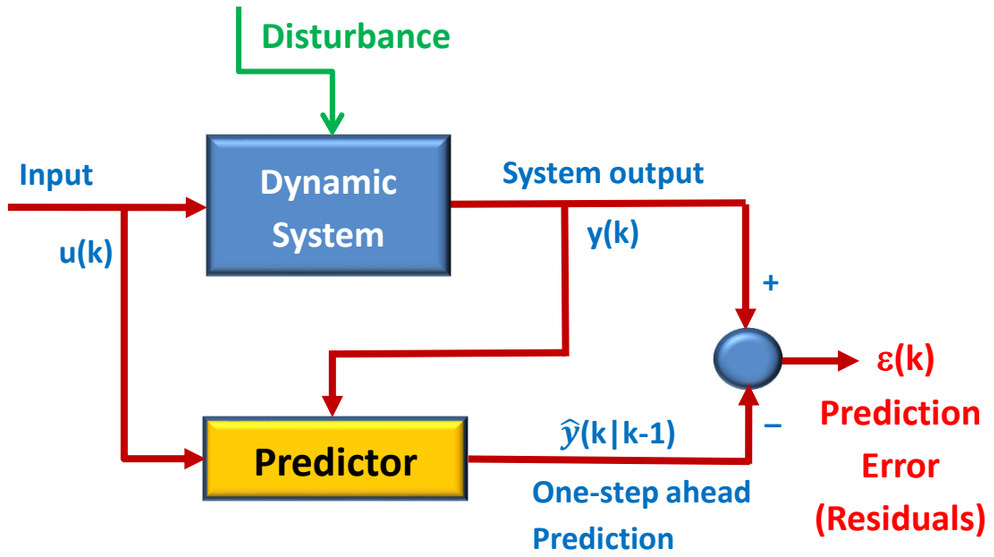
Optimal Predictor Condition $\boldsymbol{\varepsilon(k) = y(k) - \hat{y}(k|k-1) = e(k) = \text{White Noise}}$

- The optimal predictor can be determined as below:

$$y(k) = G(q)u(k) + H(q)e(k)$$

$$y(k) = G(q)u(k) + H(q)[\,y(k) - \hat{y}(k|k-1)]$$

Optimal Predictor $\boldsymbol{\hat{y}(k|k-1) = \underbrace{\frac{G(q)}{H(q)}}_{S(q)}u(k) + \underbrace{\left[1 - \frac{1}{H(q)}\right]}_{T(q)}y(k)}$



Prediction Error Method

- We can determine the optimal predictor for different model structures.

Optimal Predictor $\hat{y}(k|k-1) = \frac{G(q)}{H(q)}u(k) + \left[1 - \frac{1}{H(q)}\right]y(k)$

❑ Optimal Predictor of ARX Model

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

Optimal ARX Predictor →

$$\hat{y}(k|k-1) = B(q)u(k) + [1 - A(q)]y(k)$$

❑ Optimal Predictor of OE Model

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

Optimal OE Predictor →

$$\hat{y}(k|k-1) = \frac{B(q)}{C(q)}u(k)$$

❑ Optimal Predictor of BJ Model

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

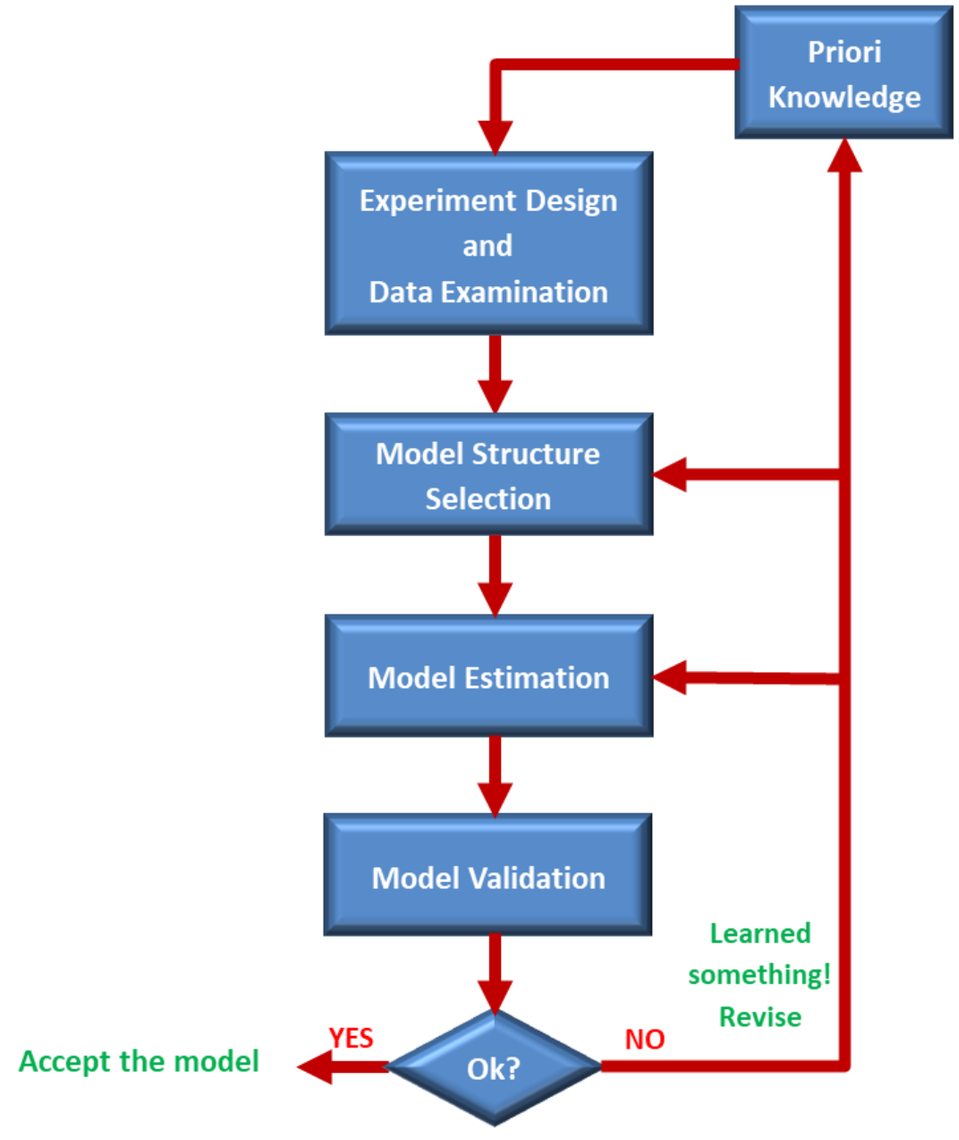
Optimal BJ Predictor →

$$\hat{y}(k|k-1) = \frac{B(q)D(q)}{F(q)C(q)}u(k) + \left[1 - \frac{D(q)}{C(q)}\right]y(k)$$

System Identification Procedure

❑ Model Validation Techniques

- The procedures that asses the quality of a model are generally called Model Validation techniques.
  - **Simulation & Cross Validation**
    - Prediction Error Method
  - **Model Validity Criteria**
    - Parameters Confidence Intervals
    - Final Prediction Error (FPE)
    - Mean-squared Error (MSE)
  - **Pole-Zero Plots**
  - **Residual Analysis**
    - Whiteness Test
    - Independency Test



Model Validity Criteria

- It is possible to determine the suitable model order by studying some criteria, which depend on the model order.
- Some of such criteria are:
  - Mean-squared Error (MSE)
  - Akaike's Final Prediction Error (FPE)
  - Percentage of the Fit to Estimated Data
  - Parameters Confidence Intervals
- The model validity criteria are given by command **`present`** in MATLAB.

```
present(sys)
```

Variable	Description
`sys`	Estimated or constructed dynamic system model

- These criteria are indices that represent goodness of the fit presented by the model.
- The smaller the indices, the better the result.

Model Validity Criteria

Example 3

Assume the collected noisy input-output data from Example 1 with $N = 500,\ T_s = 0.05\text{sec}.$

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

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

Here, we selected ARX(3,3,8) model structure. Now, we want to calculate the model validity criteria of the model using the **present** function in MATLAB.

```
na = 3;
nb = 3;
nk = 8;
M_ARXwn = arx(datawni,[na nb nk]);
present(M_ARXwn)                    ← Display model and validity criteria
```

```
M_ARXwn =
Discrete-time ARX model:  A(z)y(t) = B(z)u(t) + e(t)


A(z) = 1 - 1.24 (+/- 0.04436) z^-1 - 0.2424 (+/- 0.07659) z^-2 + 0.4907 (+/- 0.04394) z^-3

B(z) = -0.0003379 (+/- 0.000279) z^-8 + 0.0004769 (+/- 0.0003742) z^-9 + 0.0004133 (+/- 0.0002829) z^-10

Status:

Fit to estimation data: 93.44% (prediction focus)

FPE: 9.955e-06

MSE: 9.873e-06
```

Model Validity Criteria

❑ Parameters Confidence Intervals

- The standard deviation of the estimated parameters, which has to be a small value.
- When the parameter confidence intervals are large, it means that:
  - There is a large variance in the estimation
  - The model orders have been chosen too large
  - The number of data points is too small

```
M_ARXwn =

Discrete-time ARX model:  A(z)y(t) = B(z)u(t) + e(t)

A(z) = 1 - 1.24 (+/- 0.04436) z^-1 - 0.2424 (+/- 0.07659) z^-2 + 0.4907 (+/- 0.04394) z^-3

B(z) = -0.0003379 (+/- 0.000279) z^-8 + 0.0004769 (+/- 0.0003742) z^-9 + 0.0004133 (+/- 0.0002829) z^-10

Status:

Fit to estimation data: 93.44% (prediction focus)

FPE: 9.955e-06

MSE: 9.873e-06
```

Model Validity Criteria

❑ Percentage of the Fit to Estimated Data

$$Fit\ Percentage = \left(1 - \frac{\|y - y_m\|}{\|y - \bar{y}\|}\right) \times 100\%$$

- The Normalized Root Mean-Squared Error (NRMSE) expressed as a percentage, where:

  y: the measured output data,

  $\bar{y}$: the mean value of y,

  y_m: the simulated or predicted response of the model

- High percent of fit not necessarily represents the good model, it might be result of over-fitting or over-parametrizing.
  The other criteria FPE and MSE must also be considered.

```
M_ARXwn =
Discrete-time ARX model:  A(z)y(t) = B(z)u(t) + e(t)

A(z) = 1 - 1.24 (+/- 0.04436) z^-1 - 0.2424 (+/- 0.07659) z^-2 + 0.4907 (+/- 0.04394) z^-3
B(z) = -0.0003379 (+/- 0.000279) z^-8 + 0.0004769 (+/- 0.0003742) z^-9 + 0.0004133 (+/- 0.0002829) z^-10
Status:
Fit to estimation data: 93.44% (prediction focus)
FPE: 9.955e-06
MSE: 9.873e-06
```

Model Validity Criteria

❑ Mean-Squared Error (MSE)

- Mean-squared Error measure is defined as:
  - This criterion is an estimate of the prediction error variance, which shows the model accuracy.
  - MSE is the loss function to be minimized for SISO models.
  - Small MSE value indicates small prediction error for this estimated model.

$$MSE = \frac{1}{N}\sum_{k=1}^{N} \varepsilon^2(k)$$

```
M_ARXwn =
Discrete-time ARX model:  A(z)y(t) = B(z)u(t) + e(t)

A(z) = 1 - 1.24 (+/- 0.04436) z^-1 - 0.2424 (+/- 0.07659) z^-2 + 0.4907 (+/- 0.04394) z^-3
B(z) = -0.0003379 (+/- 0.000279) z^-8 + 0.0004769 (+/- 0.0003742) z^-9 + 0.0004133 (+/- 0.0002829) z^-10
Status:
Fit to estimation data: 93.44% (prediction focus)
FPE: 9.955e-06
MSE: 9.873e-06
```

Model Validity Criteria

❑ Final Prediction Error (FPE)

$$FPE = \left(\frac{1 + N_\theta/N}{1 - N_\theta/N}\right)\frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k)$$

Model Complexity | Model Accuracy

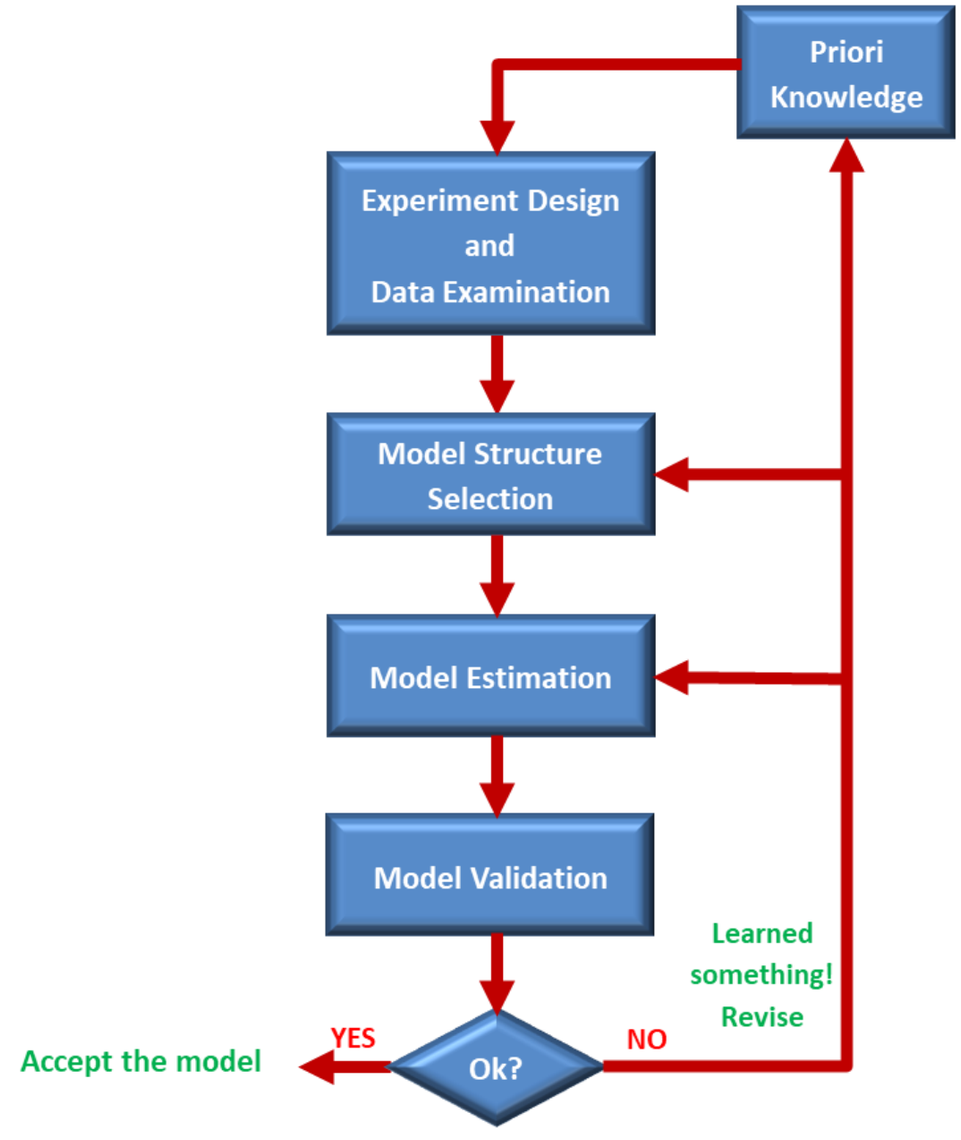
- The Final Prediction Error (FPE) criterion proposes to choose the model set that achieves a minimum value for the following expression:
- Here, N_θ is the number of parameters in the model set.
- This criterion has two terms:
  - To describe the model accuracy → Loss Function
  - To describe model complexity → Penalizing term to avoid over-fitting
- The Model Accuracy part is basically the loss function. In FPE the loss function is penalized by the model complexity term to avoid the over-fitting issues.
- Therefore, by comparing models using these criteria, we can pick a model that gives the best trade-off between the accuracy and the complexity.

```
Status:
Fit to estimation data: 93.44% (prediction focus)
FPE: 9.955e-06
MSE: 9.873e-06
```

System Identification Procedure

❑ Model Validation Techniques

- The procedures that asses the quality of a model are generally called Model Validation techniques.
  - **Simulation & Cross Validation**
    - Prediction Error Method
  - **Model Validity Criteria**
    - Parameters Confidence Intervals
    - Final Prediction Error (FPE)
    - Mean-squared Error (MSE)
  - **Pole-Zero Plots**
  - **Residual Analysis**
    - Whiteness Test
    - Independency Test



Pole-Zero Plots

- The Pole/Zero Plot helps to model reduction, if there is a sign of pole-zero cancellation in an over-parameterized model.
- If there are sign of pole-zero cancellation for model order higher than a certain value p , it means that a suitable model order n has to be selected as $n \leq p$.
- Over-parameterization makes the model unnecessarily complicated and be sensitive to parameter variations.
- Over-parameterization may result an over-fitted model, but these two terms are not same.
  - Over-fitting occurs when the model starts fitting the noise.
  - Over-parameterization occurs when the model order selected higher unnecessarily even in noise-free case.
- The Pole/Zero Plot is obtained by command **`pzmap`** in MATLAB.

```
pzmap(sys)
```

Variable	Description
sys	CT or DT dynamic system model

Pole-Zero Plots

Example 4

Assume the collected noise-free input-output data from Example 2 with $N = 500,\ T_s = 0.05\text{sec}$,

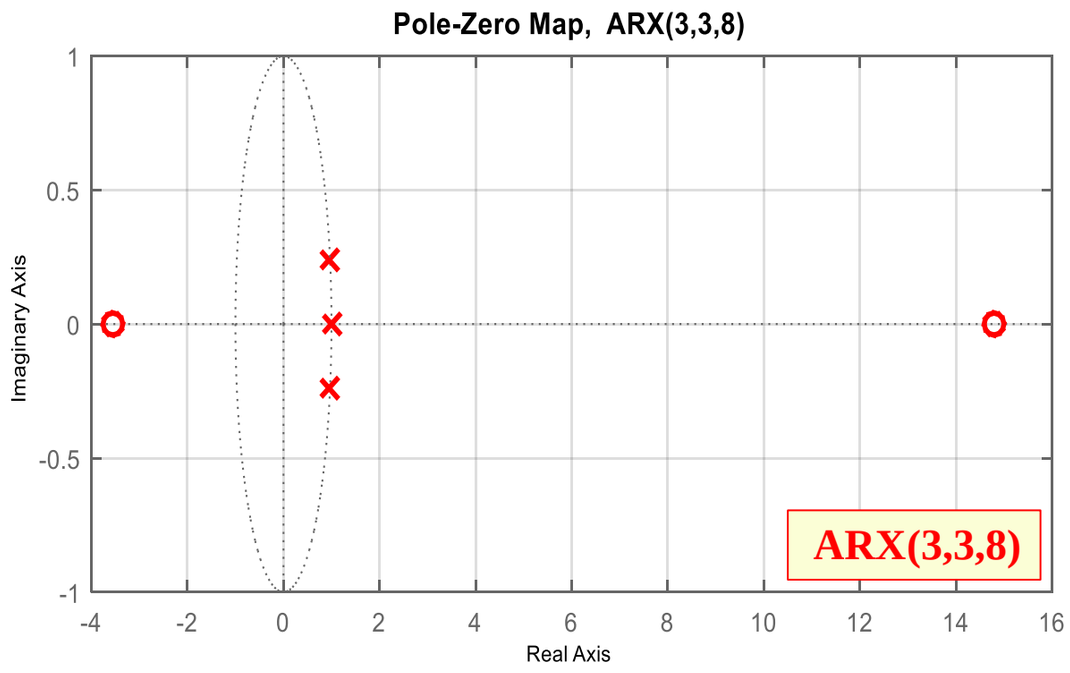
$$\{u(k)\ , y(k) |\ k = 1, \cdots, N\}$$

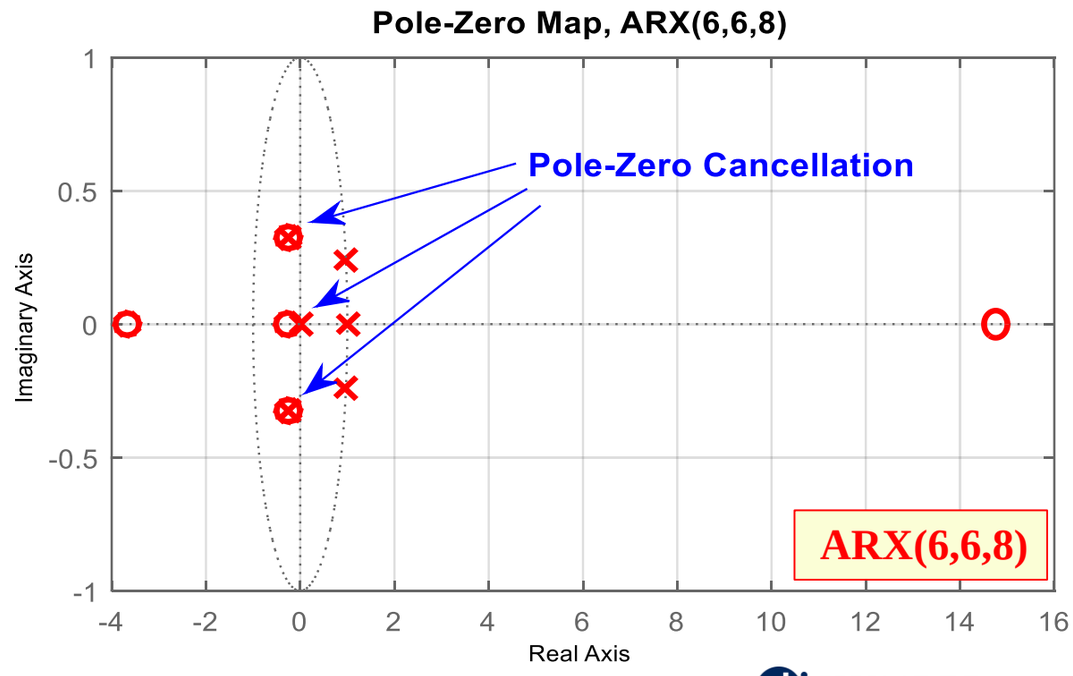
$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k)$$

Here, we selected ARX model with two different structures. We can determine the pole-zero locations using the **`pzmap`** function in MATLAB.

- The results show there are pole-zero cancellation in ARX(6,6,8) structure. It means the model is **over-parameterized** and it is **unnecessarily complicated**, which makes the model **sensitive to parameter variations**.

Over-Parameterized Model

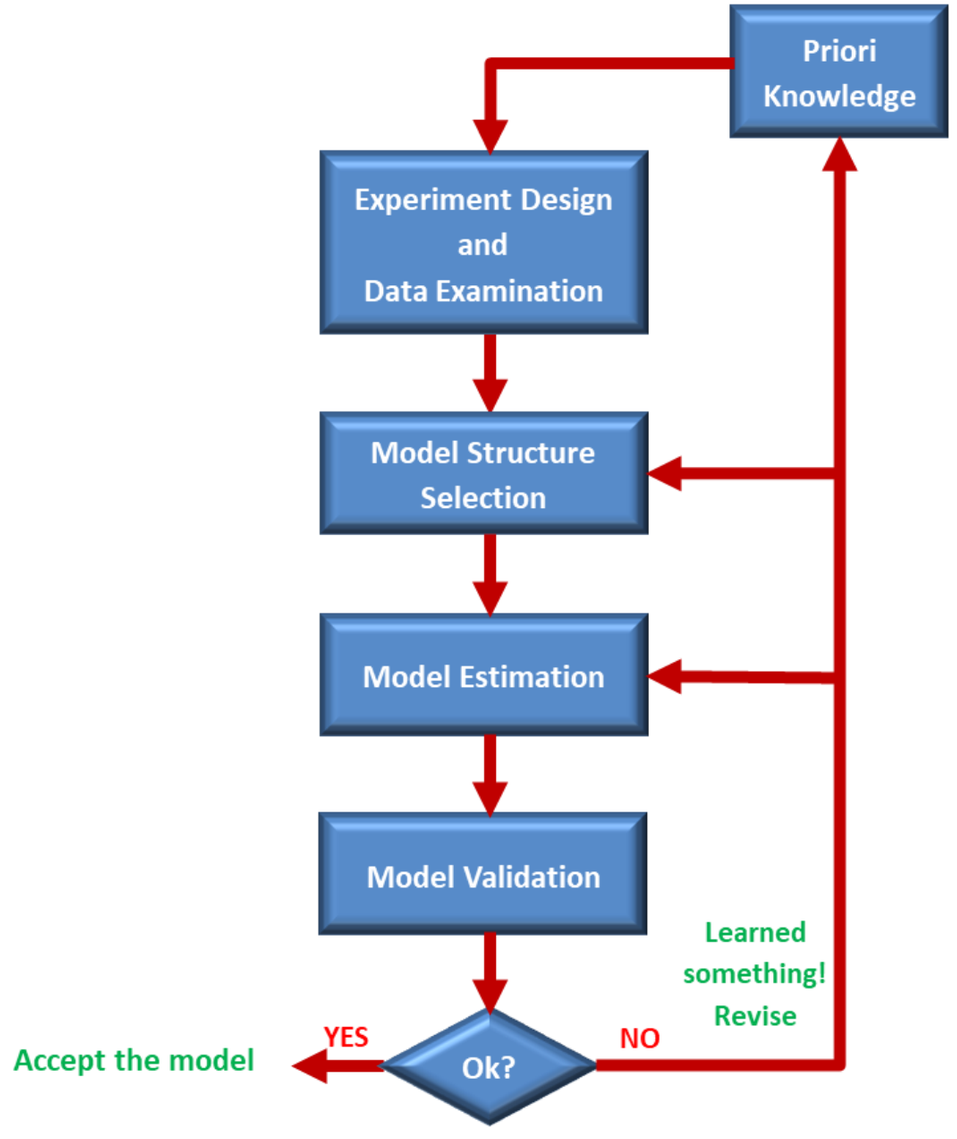




System Identification Procedure

❑ Model Validation Techniques

- The procedures that asses the quality of a model are generally called Model Validation techniques.
  - **Simulation & Cross Validation**
    - Prediction Error Method
  - **Model Validity Criteria**
    - Parameters Confidence Intervals
    - Akaike's Final Prediction Error (FPE)
    - Mean-squared Error (MSE) → Loss Function
  - **Pole-Zero Plots**
  - **Residual Analysis**
    - Whiteness Test
    - Independency Test



Residual Analysis

- Residual analysis is one of the powerful model validation methods, specifically when estimation is performed by Prediction Error Method.
- Recall the Prediction Error Method (PEM), the prediction error or residuals are determined as below

$$\textbf{Prediction Error} \quad \rightarrow \quad \boldsymbol{\varepsilon(k) = y(k) - \hat{y}(k|k-1)}$$

- Residuals are the leftovers from the modeling process, it means the part of the data that the model could not reproduce.
- Ideally and based on the definition of the Optimal Predictor, if the model is accurately describing the observed data, then the Prediction Error or Residuals, $\varepsilon(k)$, should be
  - **White noise** sequence with zero mean and variance of σ_e^2
  - **Uncorrelated** (independent) with the input data
- If this is not the case, then, for example, the model order, the model structure, the length of the data sequence, or the SNR level are inappropriate.
- The SNR level can be adjusted by changing the amplitude of the input signal.

Residual Analysis

- Based on this fact, there are two model validation techniques by Residual Analysis

$$\boldsymbol{\varepsilon(k) = y(k) - \hat{y}(k|k-1)}$$

❑ Whiteness Test of Residuals

- Under the assumption that the residuals $\varepsilon(k)$ is a white noise sequence with zero mean and variance of σ_e^2, then we can conduct the following test to check the Whiteness
  - **Normality Test → Visual Check**

❑ Independency between Residuals and Past Input Data

- Under the assumption that the residuals $\varepsilon(k)$ are uncorrelated with past inputs $u(k)$, then we can conduct the following test to check the Independency
  - **Normality Test → Visual Check**

- Fail in Whiteness Test means the residuals are not white noise. It means the noise model $H(q)$ is incorrect
- Fail in Independency Test means the residuals are correlated with the input data.
  It means the system model $G(q)$ model is incorrect.

Whiteness Test of Residuals

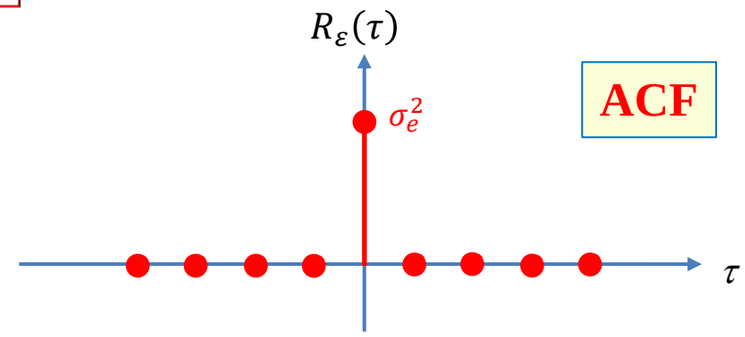
- To test the Whiteness of the residuals $\varepsilon(k)$, we compute the Auto-Correlation Function (ACF) of the residuals

$$\boldsymbol{ACF\ of\ Residuals} \quad \rightarrow \quad \boldsymbol{R_\varepsilon(\tau) = E[\varepsilon(k)\varepsilon(k-\tau)]}$$

- Theoretically, if $\varepsilon(k)$ is a White noise sequence with zero mean, then

$$\begin{cases} R_\varepsilon(\tau) = 0 & \text{for} \quad \tau \neq 0 \\ R_\varepsilon(\tau) = \sigma_e^2 & \text{for} \quad \tau = 0 \end{cases}$$

where σ_e^2 is the variance of the White noise.



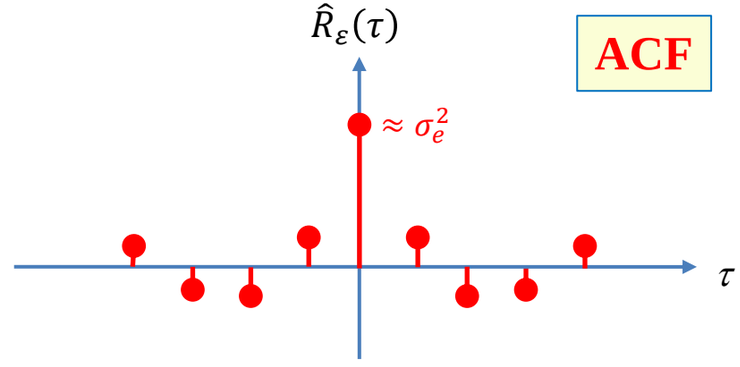
- Practically, for a finite number of data $\{k = 0, 1, \cdots, N-1\}$ the ACF is estimated as below

$$\hat{R}_\varepsilon(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \varepsilon(k)\varepsilon(k-\tau), \qquad |\tau| \leq N-1$$

- In this case, if $\varepsilon(k)$ is a White noise sequence, then the estimated ACF values are small values for all $\tau \neq 0$ but will never be all zero.

$$\begin{cases} \hat{R}_\varepsilon(\tau) \rightarrow 0 & \text{for} \quad \tau \neq 0 \\ \hat{R}_\varepsilon(\tau) \rightarrow \sigma_e^2 & \text{for} \quad \tau = 0 \end{cases} \quad \text{as} \quad N \rightarrow \infty$$

What does small mean in numerical context?



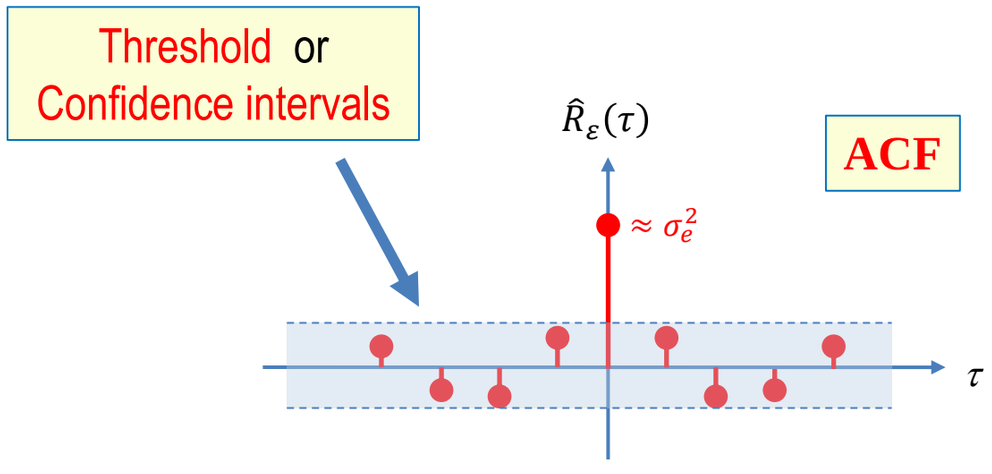
Whiteness Test of Residuals

- According to the Whiteness Test Criteria if the ACF values are smaller than the threshold value, it means inside the confidence intervals, we can assume that the residuals are small enough to be considered as white noise sequence.
- Here, we have to define a threshold value or confidence intervals, but how?
- If we know the statistical distribution of the $\hat{R}_\varepsilon(\tau)$, then we can compute a confidence interval with the given probability for $\hat{R}_\varepsilon(\tau)$.

ACF Estimates

$$\hat{R}_\varepsilon(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \varepsilon(k)\varepsilon(k-\tau), \qquad |\tau| \le N-1$$

- Based on the **Central Limit Theorem**, we can show that the Normalized ACF Estimates will be asymptotically Standard Normal distributed as $N \to \infty$



Whiteness Test of Residuals

❑ Central Limit Theorem

- The Central Limit Theorem states that a large sum of independent random variables with finite variance tends to behave like a Normal random variable.
- Suppose that $X_1, X_2, \cdots, X_N$ are a sequence of
  - Independent, identically distributed (i.i.d.) random variables
  - Mean value $\rightarrow$ μ
  - Variance $\rightarrow$ σ^2
- Then the distribution of random variable Y

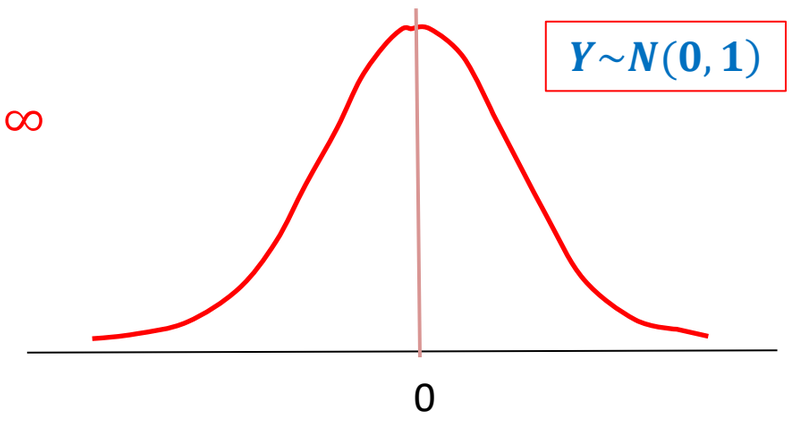
$$Y = \frac{X_1 + X_2 + \cdots + X_N - N\mu}{\sigma\sqrt{N}} = \frac{\sum_{i=1}^{N} X_i - N\mu}{\sigma\sqrt{N}}$$

tends to be Standard Normal Distribution (zero mean and unit variance) as $N \rightarrow \infty$

$$Y \rightarrow N(0,1)$$

Random variable — Mean value — Variance

Standard Normal Distribution



Whiteness Test of Residuals

- Assume that the residuals $\varepsilon(k)$ is a white noise sequence with zero mean and variance of σ_e^2
- Then from the Central Limit Theorem, the Normalized ACF Estimates will be asymptotically Standard Normal distributed as $N \to \infty$

❑ **Proof:**

Central Limit Theorem

$$Y = \frac{\sum_{i=1}^{N} X_i - N\mu}{\sigma\sqrt{N}}$$

$$Y \to N(0,1)$$

$$Y = \frac{\sum_{k=0}^{N-1}\varepsilon(k)\varepsilon(k-\tau) - N \times 0}{\sigma_e^2\sqrt{N}} = \frac{\sum_{k=0}^{N-1}\varepsilon(k)\varepsilon(k-\tau)}{\sigma_e^2\sqrt{N}} = \frac{\frac{1}{N}\sum_{k=0}^{N-1}\varepsilon(k)\varepsilon(k-\tau)}{\sigma_e^2\frac{\sqrt{N}}{N}}$$

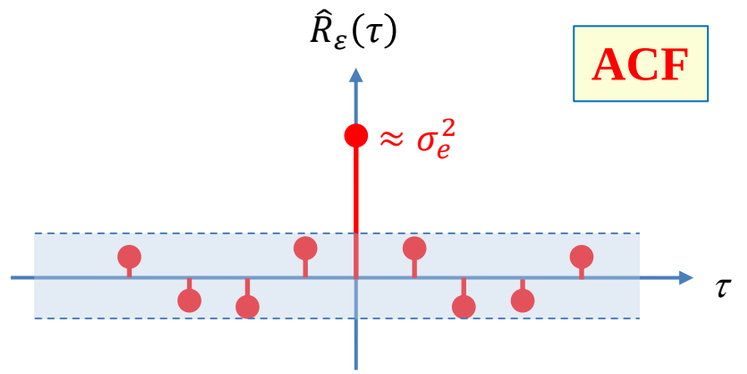
$$\hat{R}_\varepsilon(\tau) = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\varepsilon(k)\varepsilon(k-\tau)$$

If $N \to \infty$ the random variable Y can be simplified as below

$$Y = \sqrt{N}\frac{\hat{R}_\varepsilon(\tau)}{\hat{R}_\varepsilon(0)} \quad \to \quad N(0,1)$$

$$\mathrm{var}(XY) = (\sigma_X^2 + \mu_X^2)(\sigma_Y^2 + \mu_Y^2) + \mu_X^2\,\mu_Y^2$$

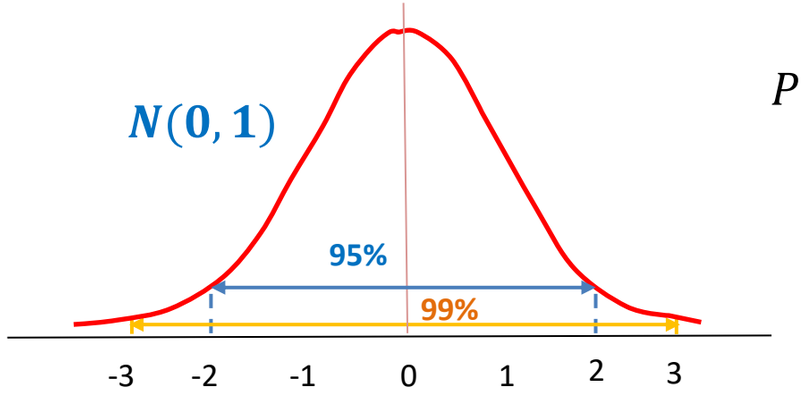
- Therefore, if the Normalized ACF Estimates satisfy the Normal Distribution characteristics the White noise assumption of the Residuals will be correct.



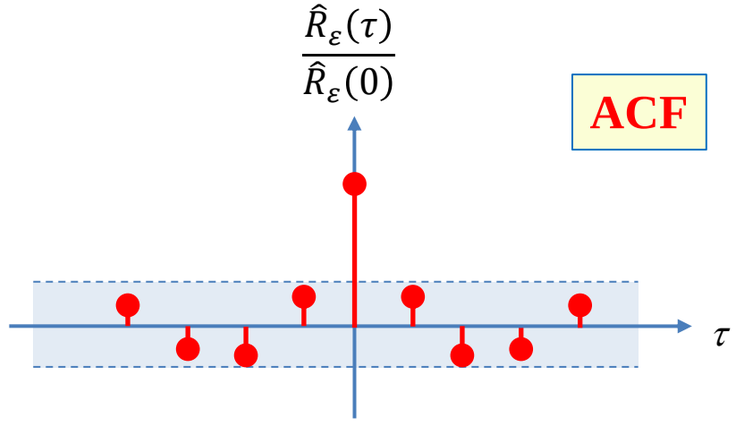
Whiteness Test of Residuals

❑ Normality Test

- **Hypothesis:** The residuals $\varepsilon(k)$ is a **white noise** sequence
- Plot the Normalized ACF Estimates of the residuals
- Calculate and draw the 99% or the 95% confidence intervals of the Normalized ACF Estimates
- The Normality test is passed with the confidence level of β (0.99 or 0.95) if the following condition is satisfied for each residual individually.



$$P\left(\sqrt{N}\left|\frac{\hat{R}_\varepsilon(\tau)}{\hat{R}_\varepsilon(0)}\right| \le N_\beta\right) = \beta \quad \Rightarrow \quad \left|\frac{\hat{R}_\varepsilon(\tau)}{\hat{R}_\varepsilon(0)}\right| \le \frac{N_\beta}{\sqrt{N}}$$



- Visually check if all Normalized ACF Estimates for $\tau \neq 0$ are located inside the selected confidence interval.
- If the ACF passes the Normality Test, then the hypothesis of White noise assumption of the Residuals will be correct.

Residual Analysis

- Based on this fact, there are two model validation techniques by Residual Analysis

$$\boldsymbol{\varepsilon(k) = y(k) - \hat{y}(k|k-1)}$$

❑ Whiteness Test of Residuals

- Under the assumption that the residuals $\varepsilon(k)$ is a white noise sequence with zero mean and variance of σ_e^2, then we can conduct the following test to check the Whiteness
  - **Normality Test → Visual Check**

❑ Independency between Residuals and Past Input Data

- Under the assumption that the residuals $\varepsilon(k)$ are uncorrelated with past inputs $u(k)$, then we can conduct the following test to check the Independency
  - **Normality Test → Visual Check**

- Fail in Whiteness Test means the residuals are not white noise. It means the noise model $H(q)$ is incorrect
- Fail in Independency Test means the residuals are correlated with the input data.
  It means the system model $G(q)$ model is incorrect.

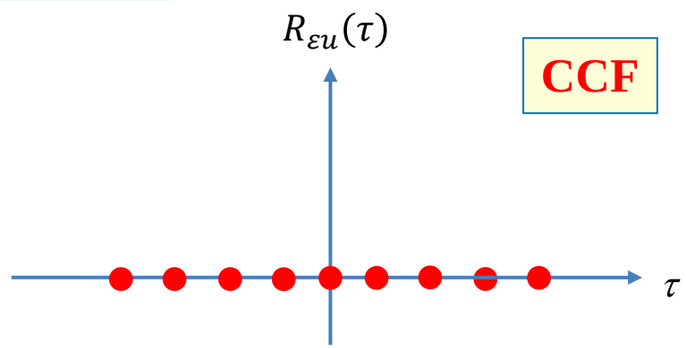
Independency Test between Residuals and Input Data

- To test the Correlation between residuals $\varepsilon(k)$ and the past input data $u(k)$, we compute the Cross-Correlation Function (CCF)

CCF between Residuals and Input Data $\rightarrow$ $\boldsymbol{R_{\varepsilon u}(\tau) = E[\varepsilon(k)u(k-\tau)]}$

- Theoretically, If $\varepsilon(k)$ and $u(k)$ are independent, then

$$R_{\varepsilon u}(\tau) = 0 \quad \text{for} \quad all\ \tau$$



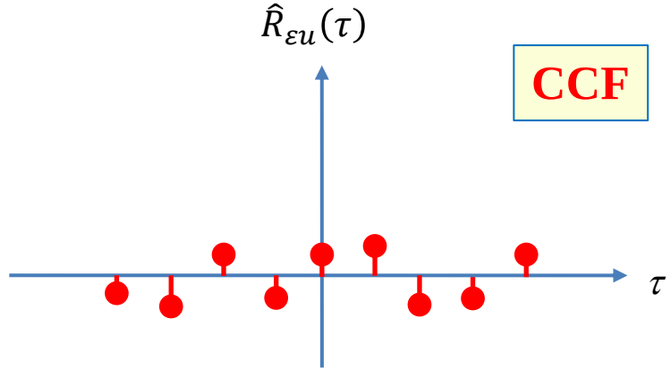
- Practically, for a finite number of data $\{k = 0, 1, \cdots, N-1\}$ the CCF is estimated as below

$$\hat{R}_{\varepsilon u}(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \varepsilon(k)u(k-\tau)\,, \qquad all\ \tau$$

- In this case, if $\varepsilon(k)$ and $u(k)$ are independent, then the estimated CCF values are small values for all τ but will never be all zero.

$$\hat{R}_{\varepsilon u}(\tau) \to 0 \quad \text{for} \quad all\ \tau \quad \text{as} \quad N \to \infty$$

What does small mean in numerical context?



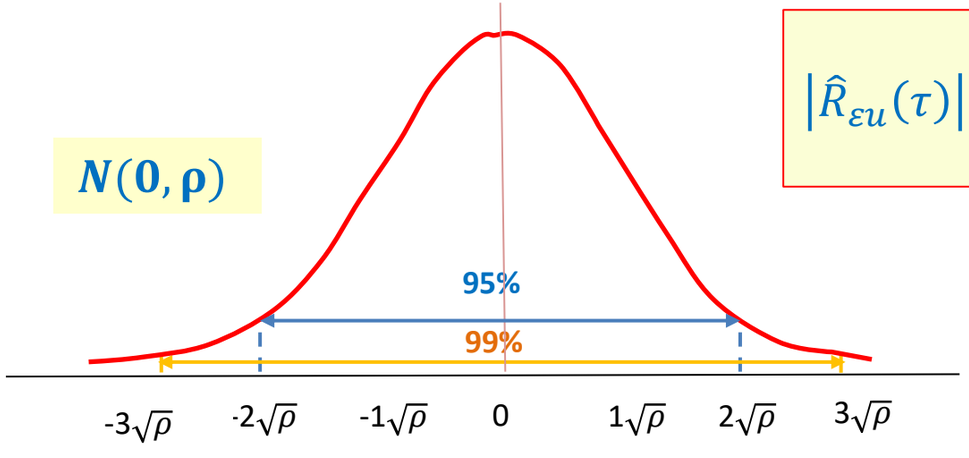
Independency Test between Residuals and Input Data

- According to the Independence Test Criteria if the CCF values are smaller than the threshold value, it means inside the confidence intervals, we can assume that the residuals are uncorrelated with past inputs.
- Assume that the residuals $\varepsilon(k)$ and the past input data $u(k)$ are uncorrelated.
- Based on the Central limit Theorem, the Normalized CCF Estimates are asymptotically Gaussian distributed as $N \to \infty$

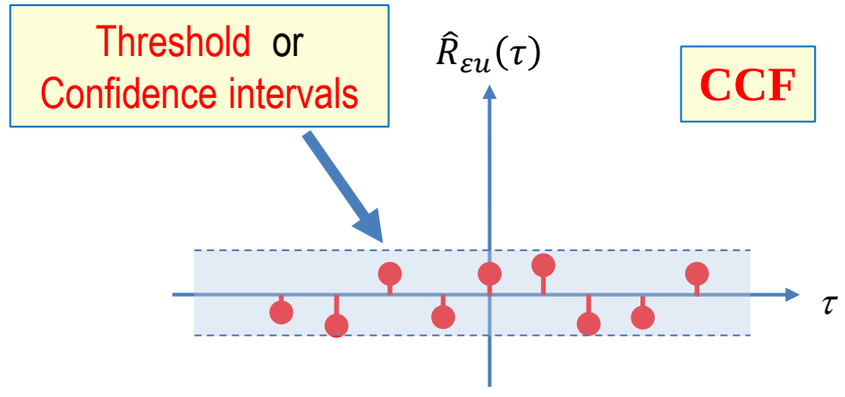
$$If \quad N \to \infty \quad \text{then} \quad \sqrt{N}\hat{R}_{\varepsilon u}(\tau) \to \quad N(0, \rho)$$

$$\rho = \sum_{k=-\infty}^{\infty} \hat{R}_{\varepsilon}(k)\hat{R}_{u}(k)$$

- Similar to the Whiteness test, here we have to check whether the following condition is satisfied for the selected Confidence level of β (0.99 or 0.95).



$$\left|\hat{R}_{\varepsilon u}(\tau)\right| \leq N_{\beta}\sqrt{\frac{\rho}{N}}$$



Independency Test between Residuals and Input Data

❑ Normality Test

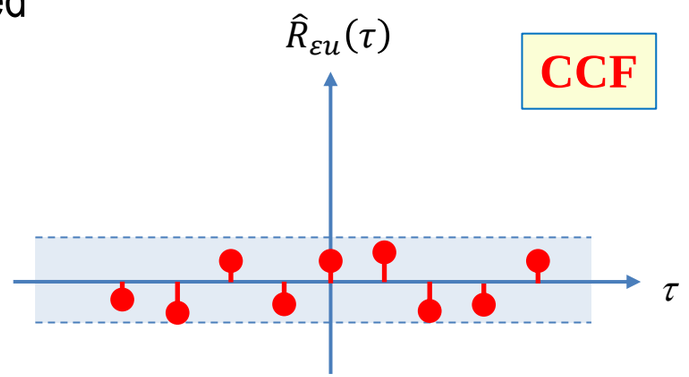
- **Hypothesis:** The residuals $\varepsilon(k)$ is and the past input data $u(k)$ are **uncorrelated**.

$$\hat{R}_{\varepsilon u}(\tau) \approx 0 \quad \text{for} \quad all\ \tau$$

- Plot the Normalized CCF Estimates of the residuals and the past input data
- Calculate and draw the 99% or the 95% confidence intervals of the Normalized CCF Estimates
- The Normality test is passed with the confidence level of β (0.99 or 0.95) if the following condition is satisfied

$$\left|\hat{R}_{\varepsilon u}(\tau)\right| \leq N_\beta \sqrt{\frac{\rho}{N}}$$

- Visually check if all Normalized CCF Estimates for all τ are located inside the selected confidence interval or not.
- If the CCF passes the Normality Test, then the hypothesis of Independency assumption of the Residuals and the Past Input Data will be correct.



Residual Analysis

❑ MATLAB Function for Residual Analysis

- In System Identification Toolbox MATLAB, we have the following function to compute and test the residuals for Normality

```
resid(data,sys)
```

Variable	Description
data	Input-output data in iddata format
sys	Identified model (ARX, ARMAX, OE, BJ)

- The **`resid`** function computes the one-step ahead prediction errors (residuals) for the identified model and plots the Normalized Auto-Correlation Function (ACF) of the residuals and the Normalized Cross-Correlation Function (CCF) of the residuals with the input signal for lags from -25 to 25.

- The 99% confidence interval has also been displayed as a shaded region around the x-axis.

Identification & Validation of Transfer Function Models

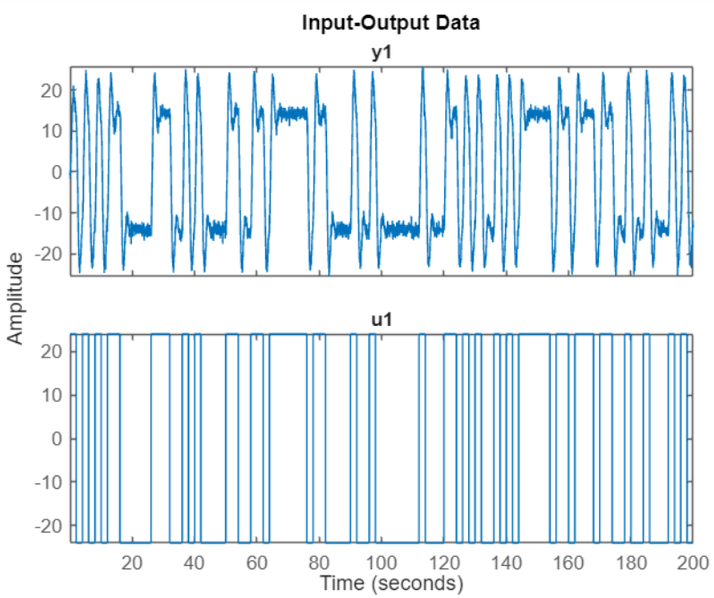
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 1: Collect and Examine the I/O Data

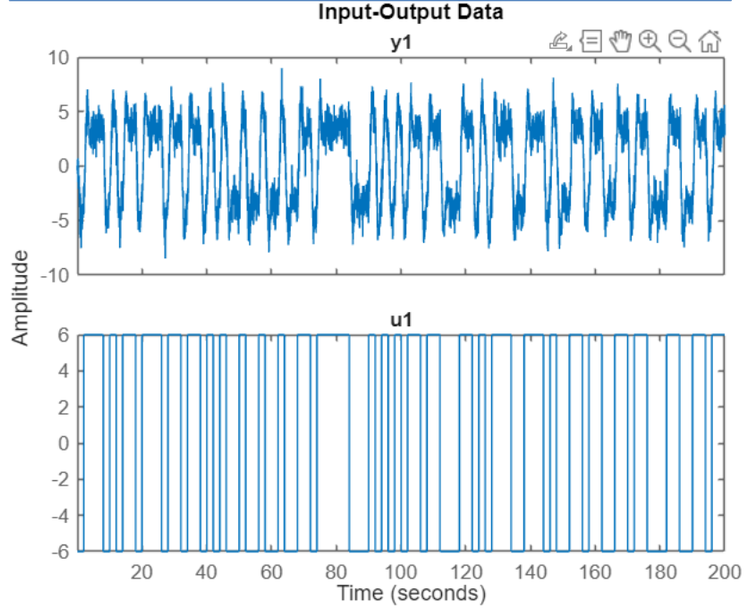
High SNR Dataset

```
Ts = 0.05;
dataH = iddata(yH,uH,Ts);
plot(dataH)
```



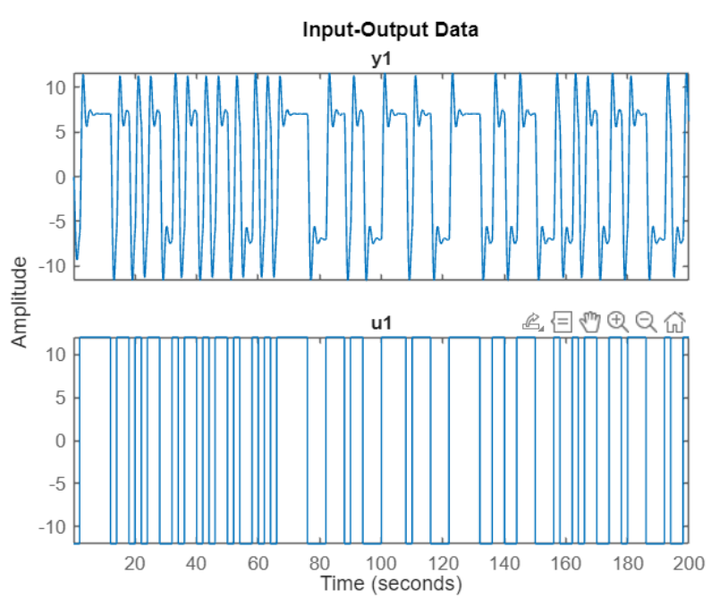
Low SNR Dataset

```
Ts = 0.05;
dataL = iddata(yL,uL,Ts);
plot(dataL)
```



Noise-free Dataset

```
Ts = 0.05;
data = iddata(y,u,Ts);
plot(data)
```



Since the sample time is 0.05sec, we collected data for 200sec, which means the total number of samples is 200/0.05=4000 samples.

Identification & Validation of Transfer Function Models

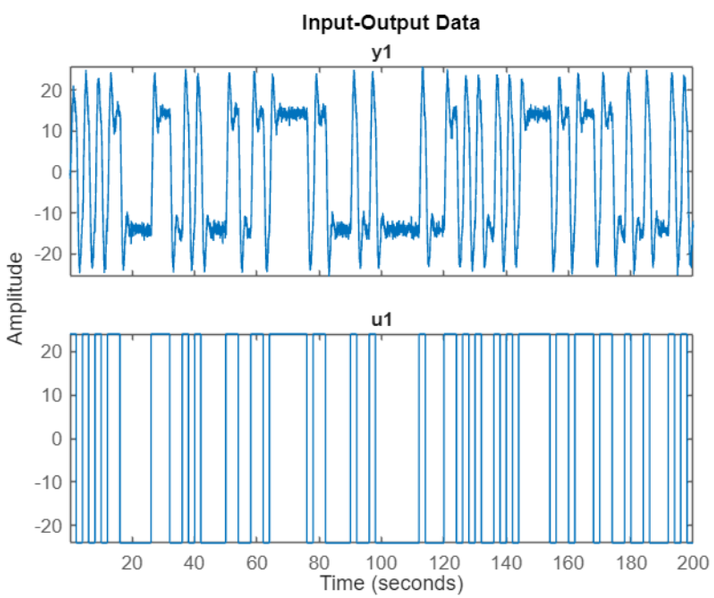
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

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

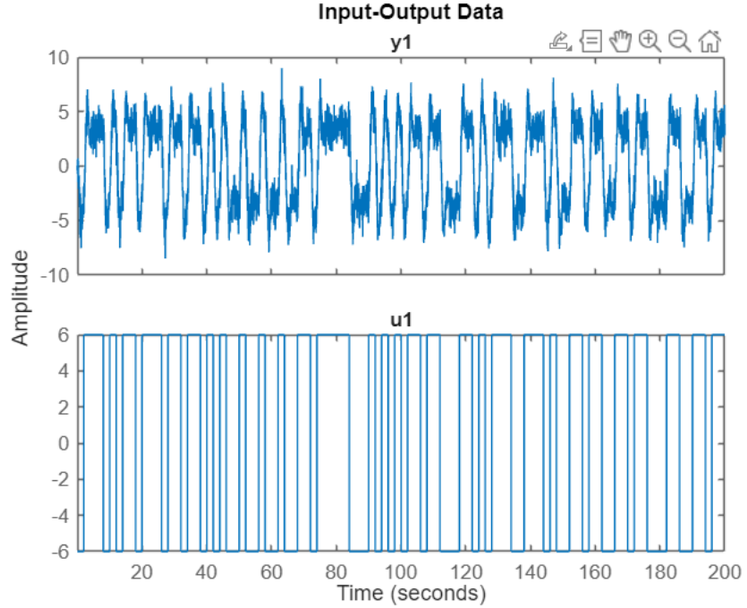
High SNR Dataset

```
dataHi = dataH(1:3000);
dataHv = dataH(3001:end);
```



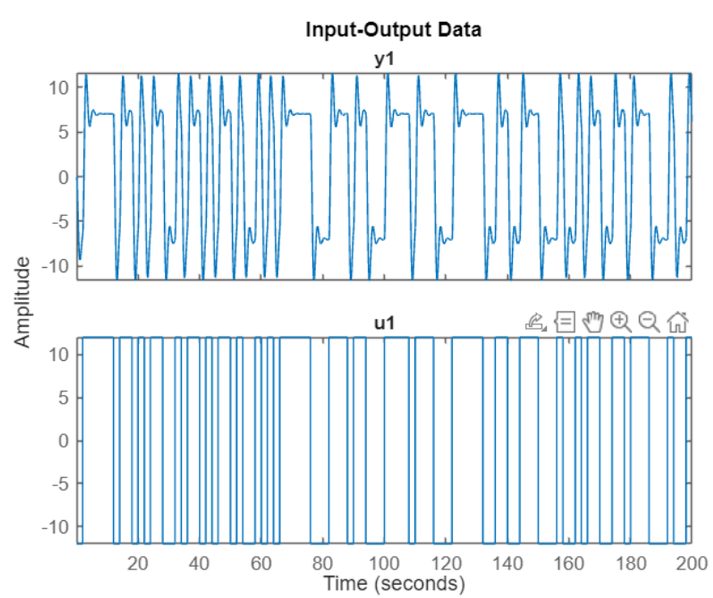
Low SNR Dataset

```
dataLi = dataL(1:3000);
dataLv = dataL(3001:end);
```



Noise-free Dataset

```
datai = data(1:3000);
datav = data(3001:end);
```



Here, we considered the first 3000 samples for identification and the last 1000 samples for validation.

Identification & Validation of Transfer Function Models

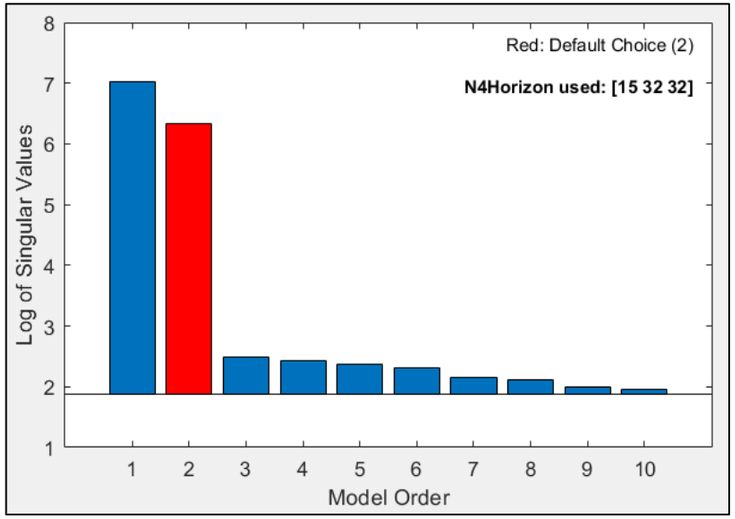
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 3: Order Selection & Delay Estimation

High SNR Dataset

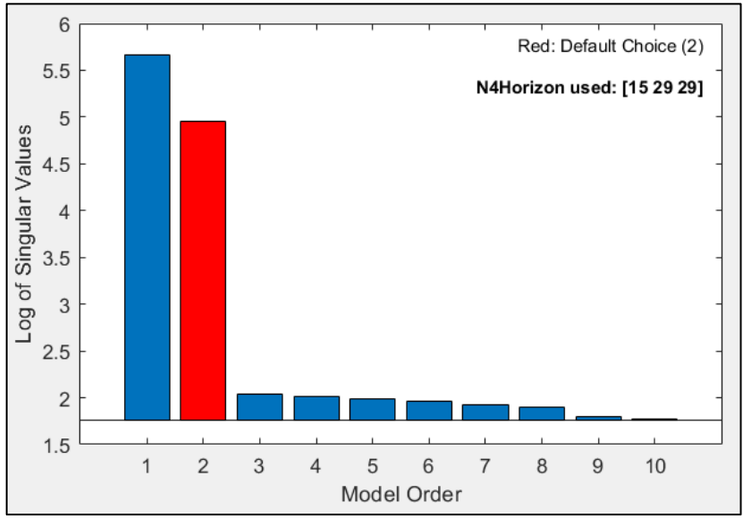
```
n4sid(dataH,1:10);
```



Selected order = 2

Low SNR Dataset

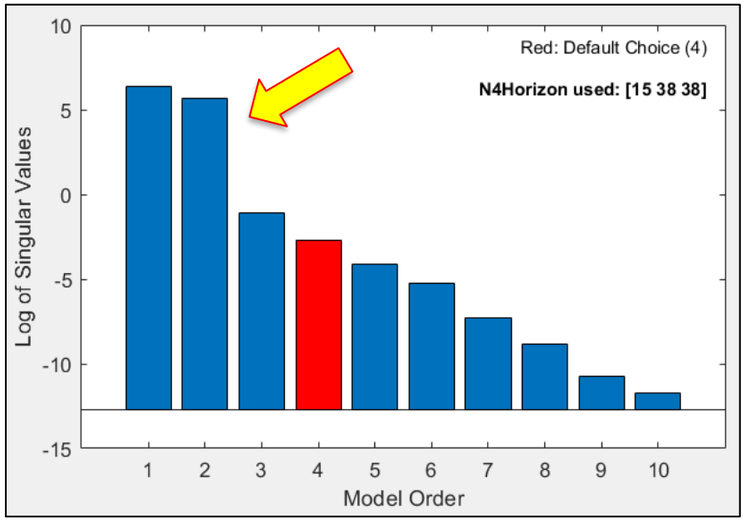
```
n4sid(dataL,1:10);
```



Selected order = 2

Noise-free Dataset

```
n4sid(data,1:10);
```



Recommended order = 4
(Possibility of the second-order model)

Identification & Validation of Transfer Function Models

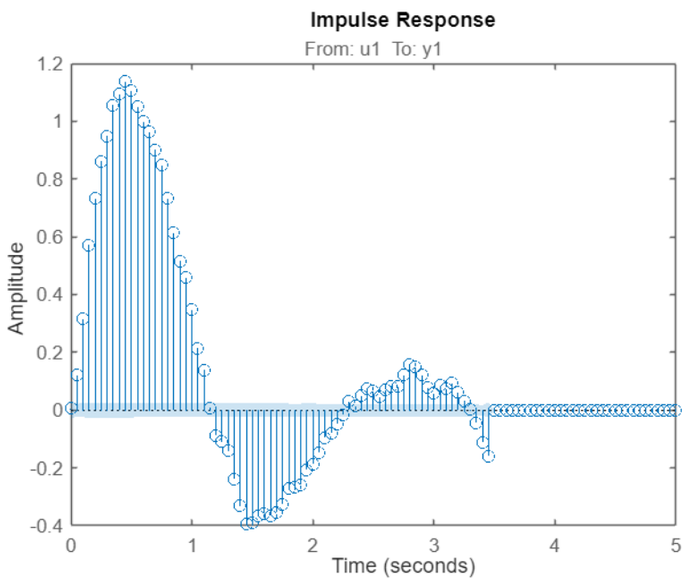
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 3: Order Selection & Delay Estimation

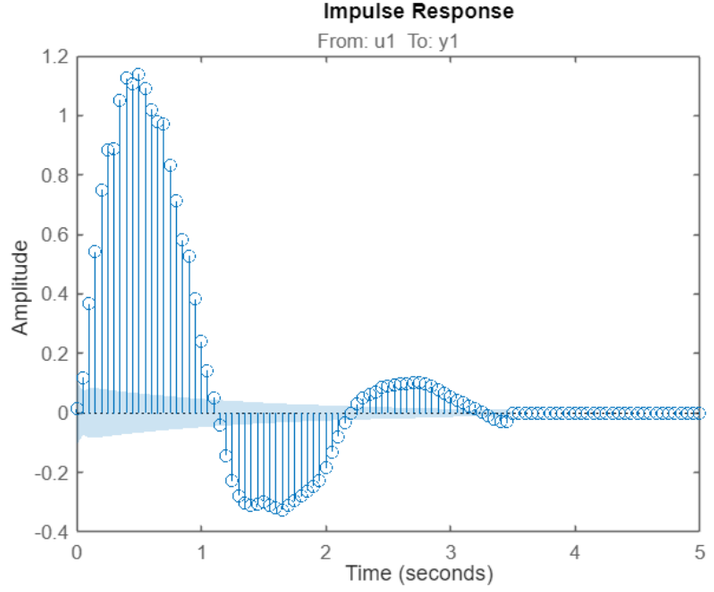
High SNR Dataset

```
ir = impulseest(dataH);
ir_plot = impulseplot(ir);
showConfidence(ir_plot,1)
```



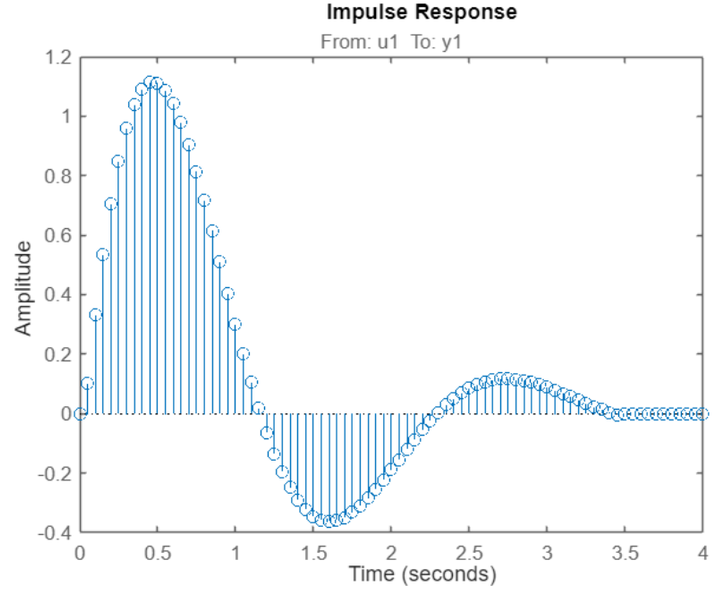
Low SNR Dataset

```
ir = impulseest(dataL);
ir_plot = impulseplot(ir);
showConfidence(ir_plot,1)
```



Noise-free Dataset

```
ir = impulseest(data);
ir_plot = impulseplot(ir);
showConfidence(ir_plot,1)
```



Determine the number of delay samples n_k from the estimated impulse response samples.
Note that there is always one delay sample due to the ZOH.
Therefore, there is no delay time in this system.

$$t_d = T_s(n_k - 1)$$

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 4: Select the Model Structure & Identify the Transfer Function Model

High SNR Dataset

```
np = 2;
nz = 0;
sys1 = tfest(dataHi,np,nz,0)
```

```
sys1 =
 From input "u1" to output "y1":
         5.004
  ---------------------
  s^2 + 1.957 s + 8.447

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters and
their uncertainties.

Status:
Estimated using TFEST on time domain data "dataHi".
Fit to estimation data: 93.46%
FPE: 0.9967, MSE: 0.9934
```

Second-order model with no zero.

$$G_1(s) = \frac{5.004}{s^2 + 1.957s + 8.447}$$

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation

High SNR Dataset

1- Model validation criteria

```
present(dataHi)
```

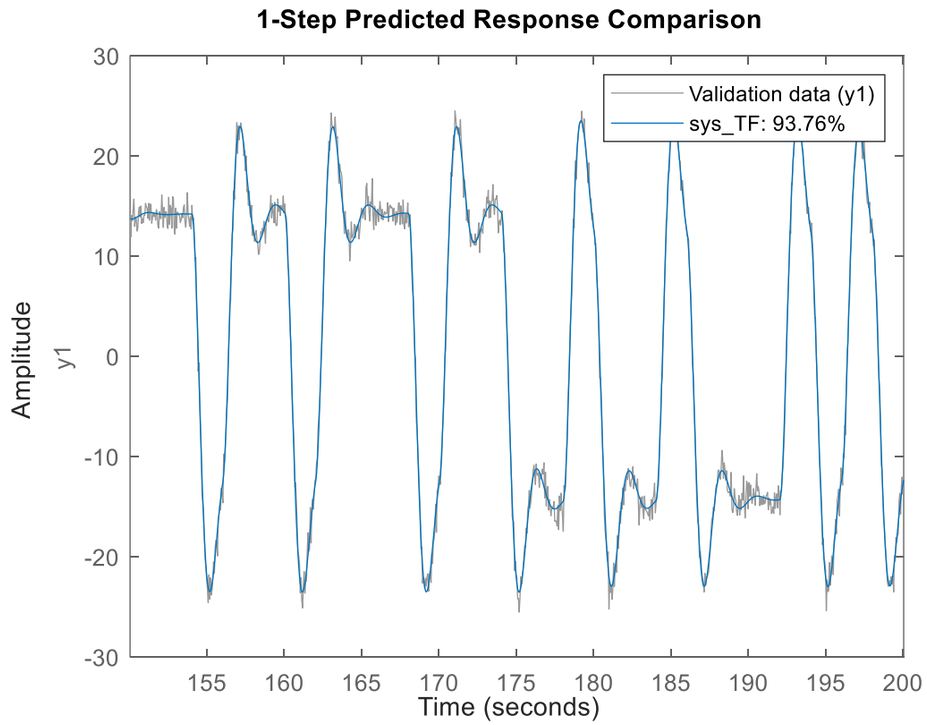
```
sys1 =

                5.004  (+/- 0.01944)
  ----------------------------------------------
  s^2 + 1.957  (+/- 0.009539)  s + 8.447  (+/- 0.02627)


Fit to estimation data: 93.46%
FPE: 0.9967, MSE: 0.9934
```

- Confidence intervals of the estimation → Must be small
- Fit to estimation data → Between 70% to 100% depends on the SNR
- Mean Square Error (MSE) → Must be small enough (less than 1)
- Final Prediction Error (FPE) → Must be small enough (less than 1)

2 - Cross-validation

```
compare(dataHv,sys1,1)
```

Validate the identify model based on the validation dataset.



Identification & Validation of Transfer Function Models

Example 5

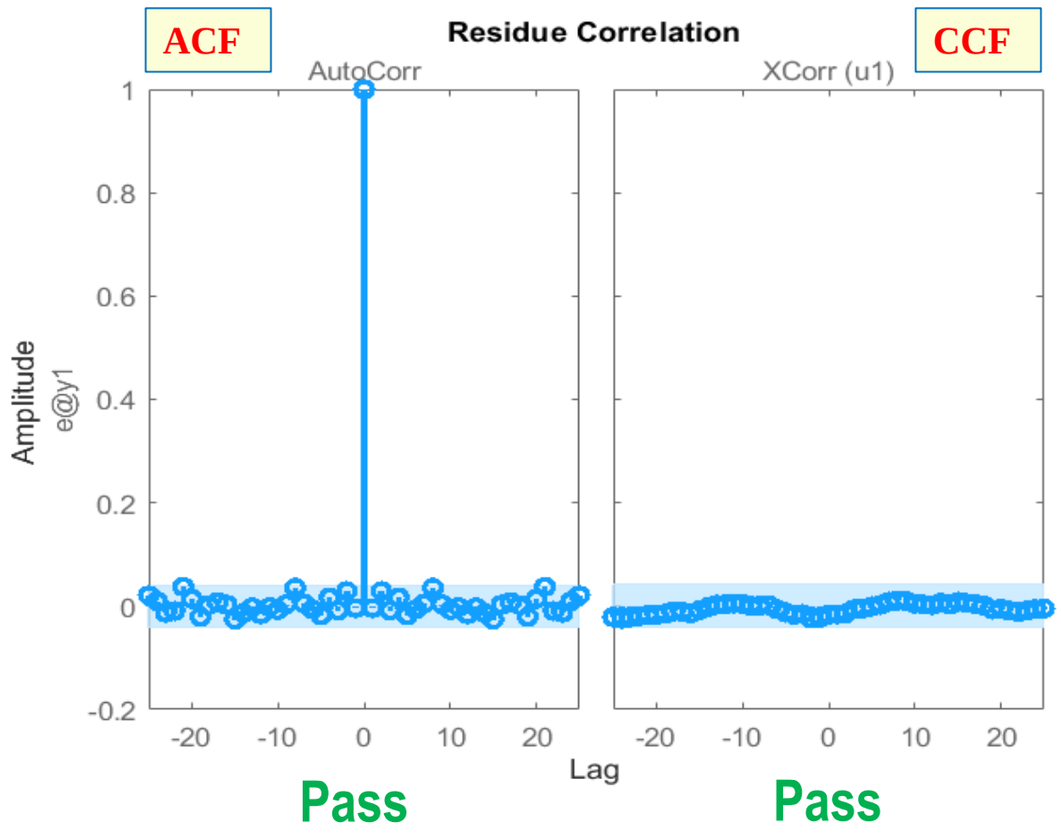
This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation

High SNR Dataset

3- Residual Analysis

```
resid(sys1,dataH)
```



- Visually check the Normality test for:
  - Whiteness of the residuals
  - Independency of the residuals from input data

- Both ACF and CCF must pass the Normality test.
- If it failed, check for
  - the model order,
  - the model structure,
  - the length of the data sequence, } Usually, CCF fails.
  - the SNR level (input amplitude), → Usually, ACF fails.

Note: Sometimes the ACF might fail the Whiteness test due to high SNR data. In that case decreasing the input amplitude and collecting more noisy data will be helpful.

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 6: Check the pole/zero locations & DC-gain for G(s)

High SNR Dataset

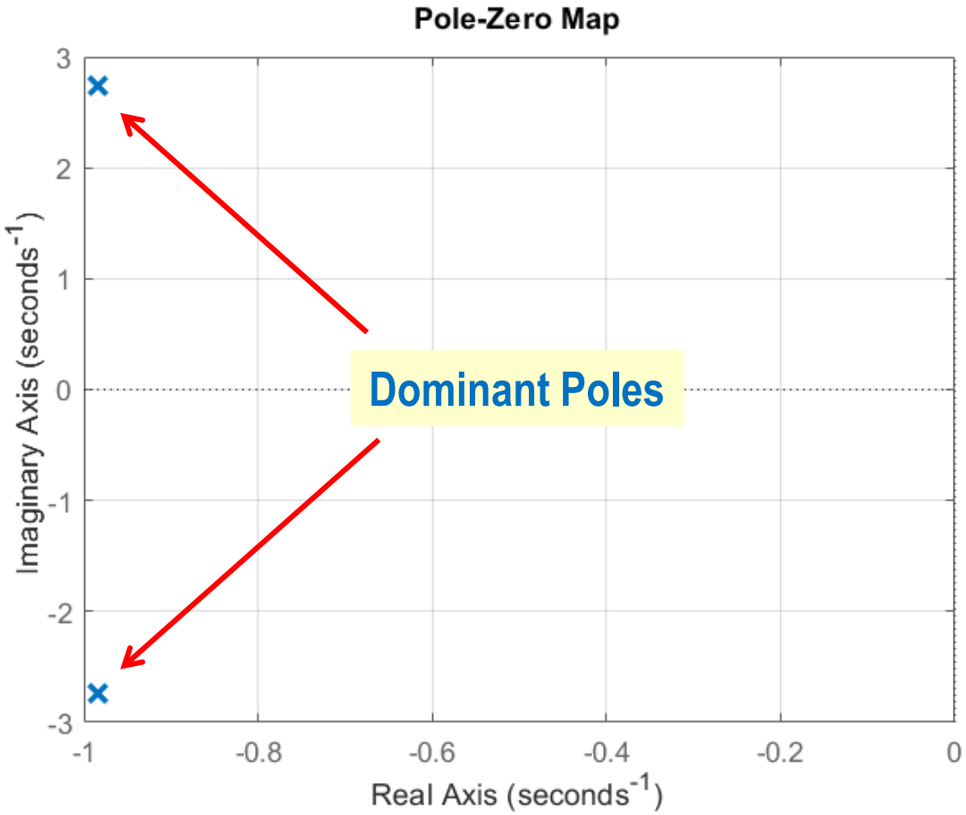
Second-order model with no zero.

$$G_1(s) = \frac{5.004}{s^2 + 1.957s + 8.447}$$

```
pzmap(sys1)
pole(sys1)
```

Poles: $-0.9785 \pm j2.7367$

DC-gain: $\lim_{s\to 0} G(s) = \frac{5.004}{8.447} = 0.592$



Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 7: Check the Step Response and Time-domain Specifications

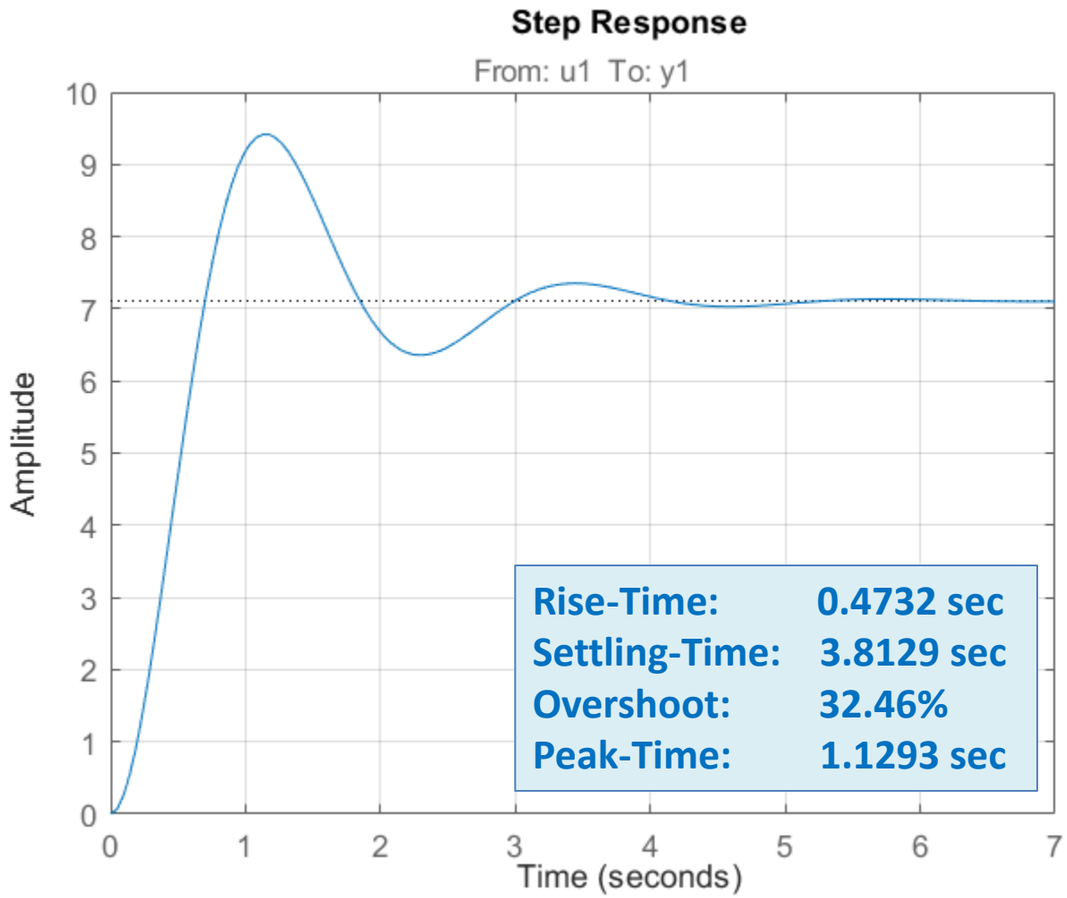
High SNR Dataset

Second-order model with no zero.

$$G_1(s) = \frac{5.004}{s^2 + 1.957s + 8.447}$$

```
opt = RespConfig;
opt.Amplitude = 12;
stepplot(sys1,opt)
stepinfo(sys1)
```

```
ans =
          RiseTime: 0.4732
     TransientTime: 3.8129
      SettlingTime: 3.8129
       SettlingMin: 0.5298
       SettlingMax: 0.7847
         Overshoot: 32.4649
        Undershoot: 0
              Peak: 0.7847
          PeakTime: 1.1293
```



Identification & Validation of Transfer Function Models

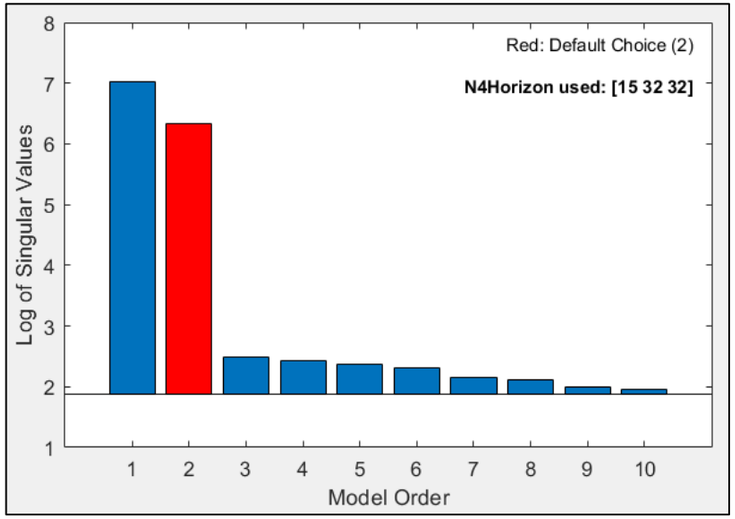
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 3: Order Selection & Delay Estimation

High SNR Dataset

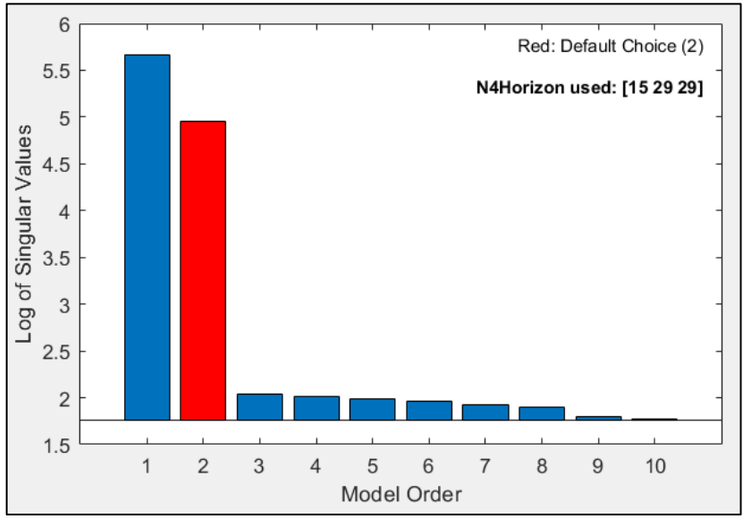
```
n4sid(dataH,1:10);
```



Selected order = 2

Low SNR Dataset

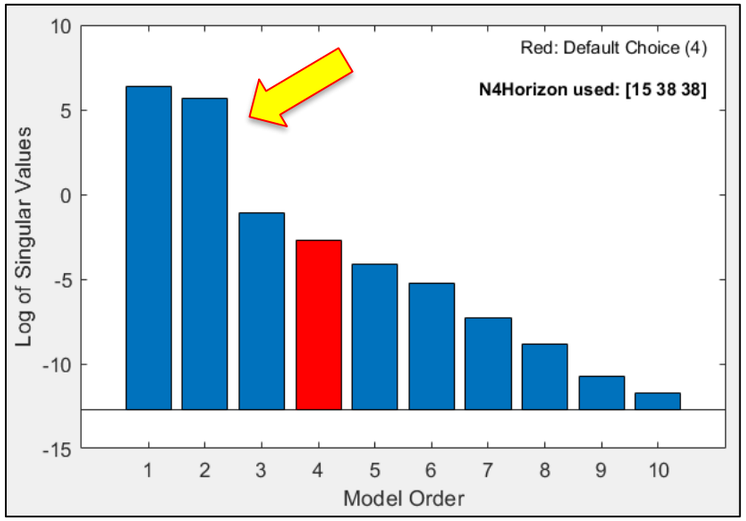
```
n4sid(dataL,1:10);
```



Selected order = 2

Noise-free Dataset

```
n4sid(data,1:10);
```



Recommended order = 4
(Possibility of the second-order model)

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 4: Select the Model Structure & Identify the Transfer Function Model

Low SNR Dataset

Second-order model with no zero.

```
np = 2;
nz = 0;
sys2 = tfest(dataLi,np,nz,0)
```

```
sys2 =
 From input "u1" to output "y1":
         5.069
  ---------------------
  s^2 + 1.951 s + 8.71

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters
and their uncertainties.

Status:
Estimated using TFEST on time domain data "dataLi".
Fit to estimation data: 74.38%
FPE: 1.005, MSE: 1.001
```

Third-order model with two zeros.

```
np = 3;
nz = 2;
sys32 = tfest(dataLi,np,nz,0)
```

```
sys32 =
 From input "u1" to output "y1":
  -0.03837 s^2 + 5.128 s + 0.08048
  ---------------------------------
  s^3 + 1.957 s^2 + 8.853 s + 0.1374

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 3   Number of zeros: 2
   Number of free coefficients: 6
   Use "tfdata", "getpvec", "getcov" for parameters
and their uncertainties.

Status:
Estimated using TFEST on time domain data "dataLi".
Fit to estimation data: 74.4%
FPE: 1, MSE: 1
```

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation

Low SNR Dataset

1- Model validation criteria

```
present(dataLi)
```

Second-order model with no zero.

```
sys2 =
                4.971  (+/- 0.07583)
  ------------------------------------------------
  s^2 + 1.978  (+/- 0.0404)  s + 8.559  (+/- 0.1082)


Fit to estimation data: 74.25%
FPE: 0.9653, MSE: 0.9621
```

Third-order model with two zeros.

```
sys32 =

   -0.03837  (+/- 0.03063)  s^2 + 5.128  (+/- 0.1014)  s + 0.0808  (+/- 0.1988)
  ------------------------------------------------------------------------------
  s^3 + 1.957  (+/- 0.05093)  s^2 + 8.853  (+/- 0.1605)  s + 0.1374  (+/- 0.3291)


Fit to estimation data: 74.4%
FPE: 1, MSE: 1
```

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

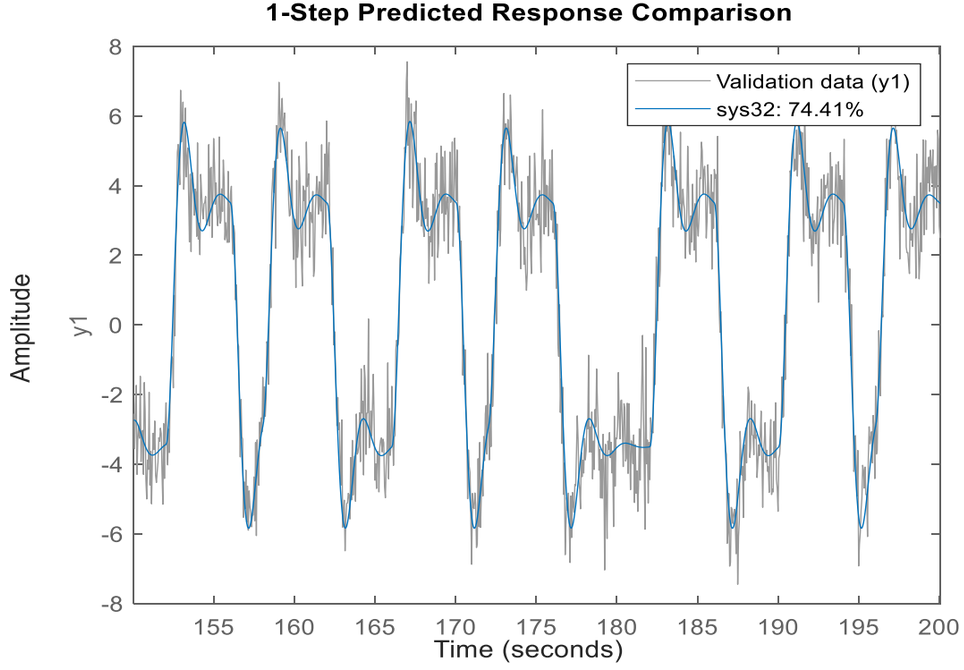
Step 5: Model Validation

Low SNR Dataset

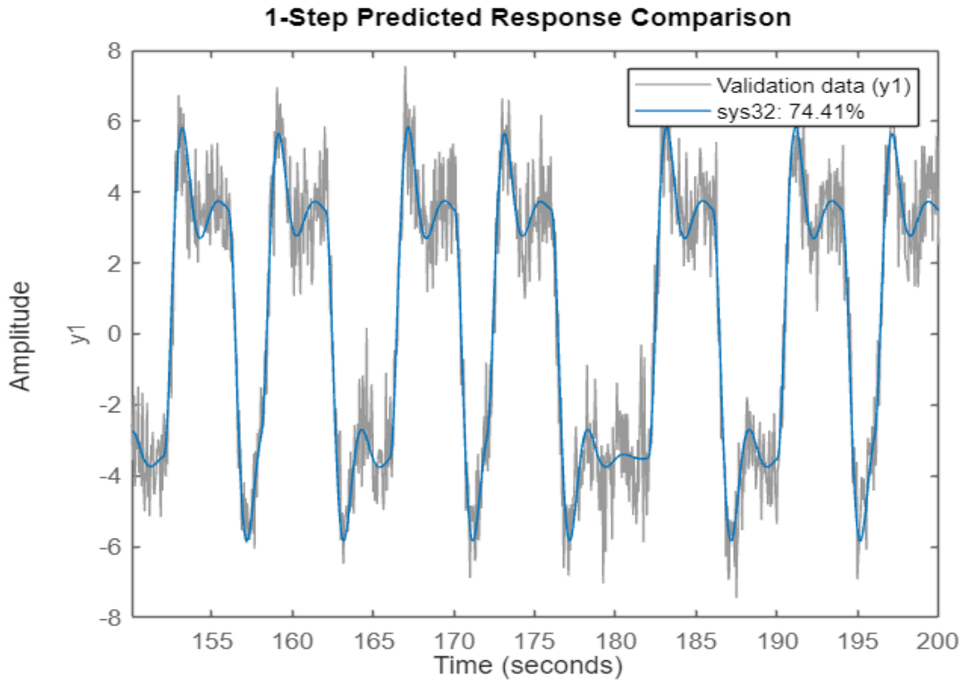
2- Cross-Validation

```
compare(dataLv,sys2,1)
```

Second-order model with no zero.



Third-order model with two zeros.



The cross-validation results are almost similar for both models.

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

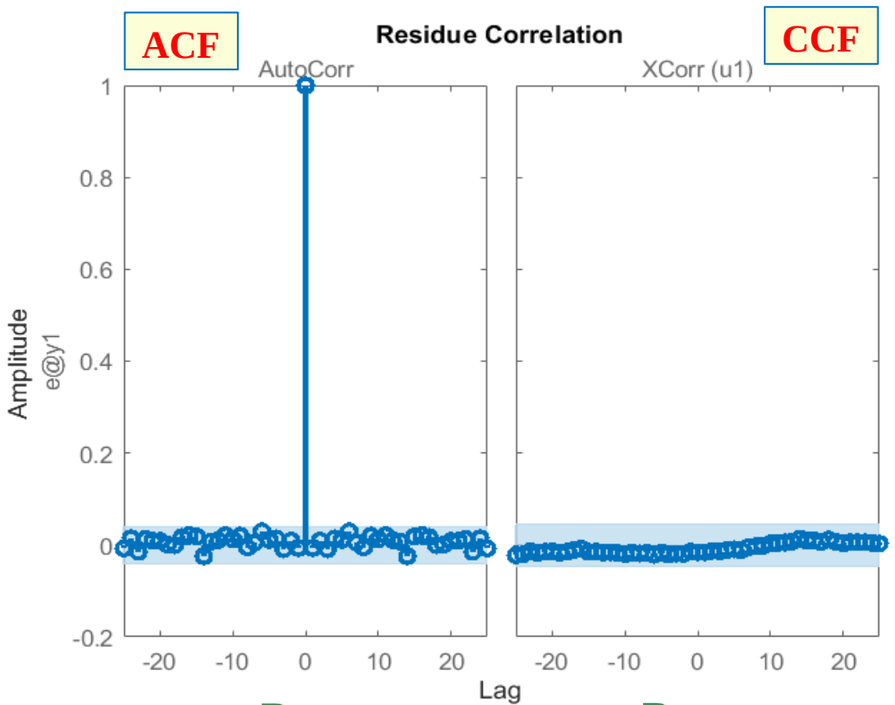
Step 5: Model Validation

Low SNR Dataset

3- Residual Analysis

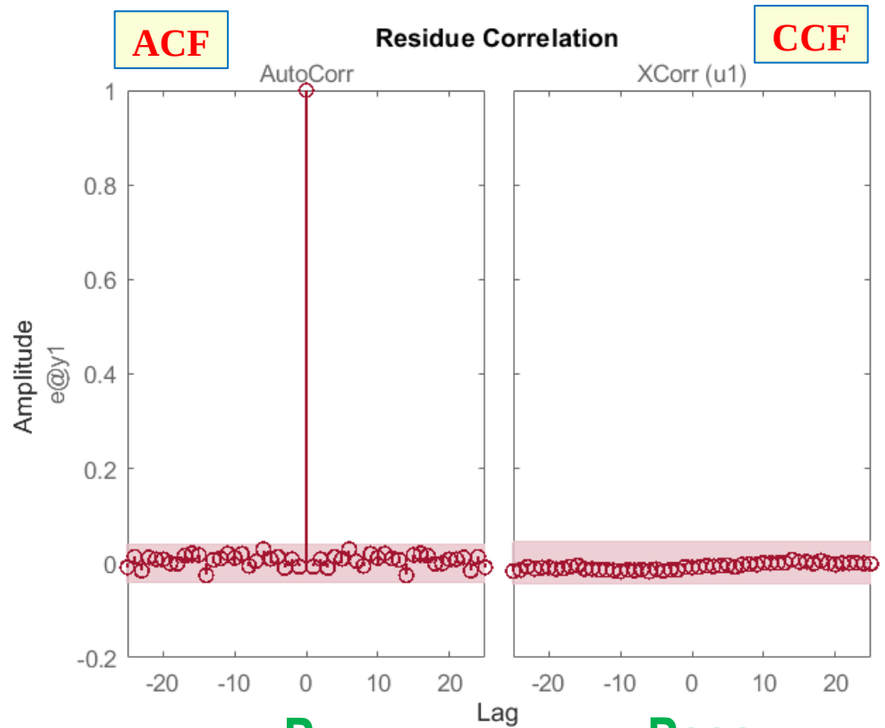
```
resid(sys1,dataL)
```

Second-order model with no zero.



Pass **Pass**

Third-order model with two zeros.



Pass **Pass**

Both systems pass the Normality test

Identification & Validation of Transfer Function Models

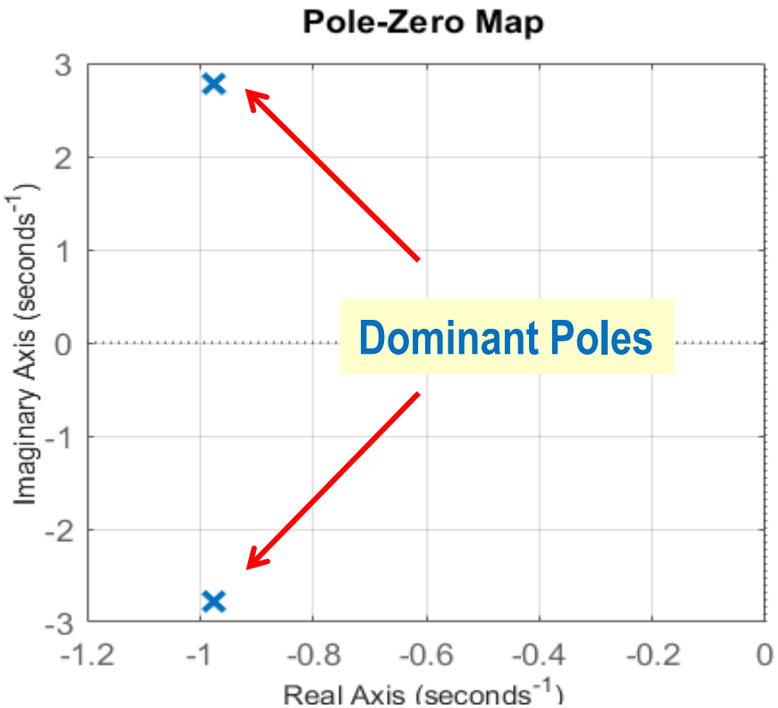
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 6: Check the pole/zero locations & DC-gain for G(s)

Low SNR Dataset

Second-order model with no zero.



Poles: $-0.9755 \pm j2.7853$

DC-gain: $\lim_{s\to 0} G(s) = \frac{5.069}{8.71} = 0.582$

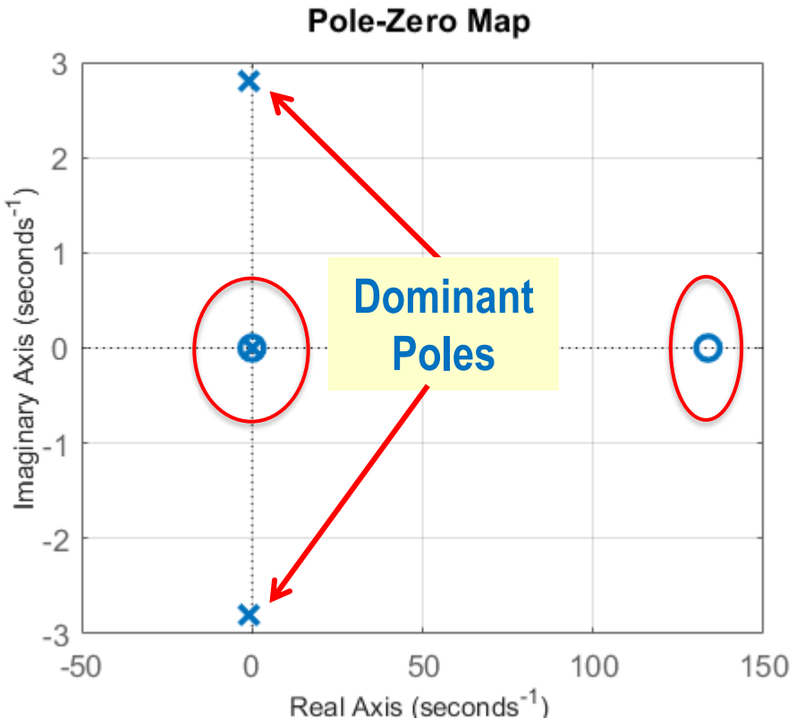
Poles:
$-0.9706 \pm j2.8072$
-0.0156

Zeros:
$+133.6775$
-0.0157

DC-gain:

$$\lim_{s\to 0} G(s) = \frac{0.08048}{0.1374} = 0.586$$

Third-order model with two zeros.



Since there are pole-zero cancellation and one single zero is too far from the dominant poles, the third-order system is over-parameterized. It can be approximated as a second-order system.

Identification & Validation of Transfer Function Models

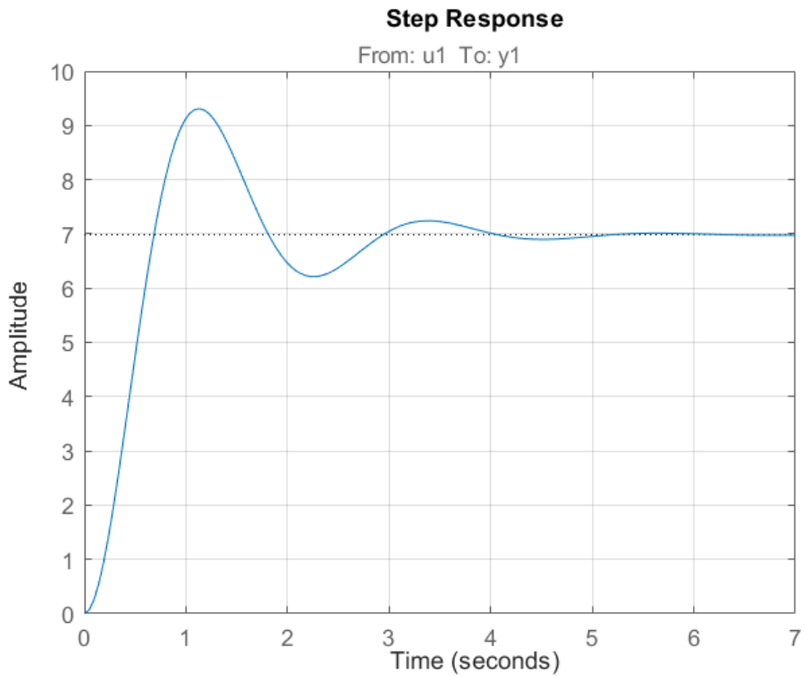
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 6: Check the pole/zero locations & DC-gain for G(s)

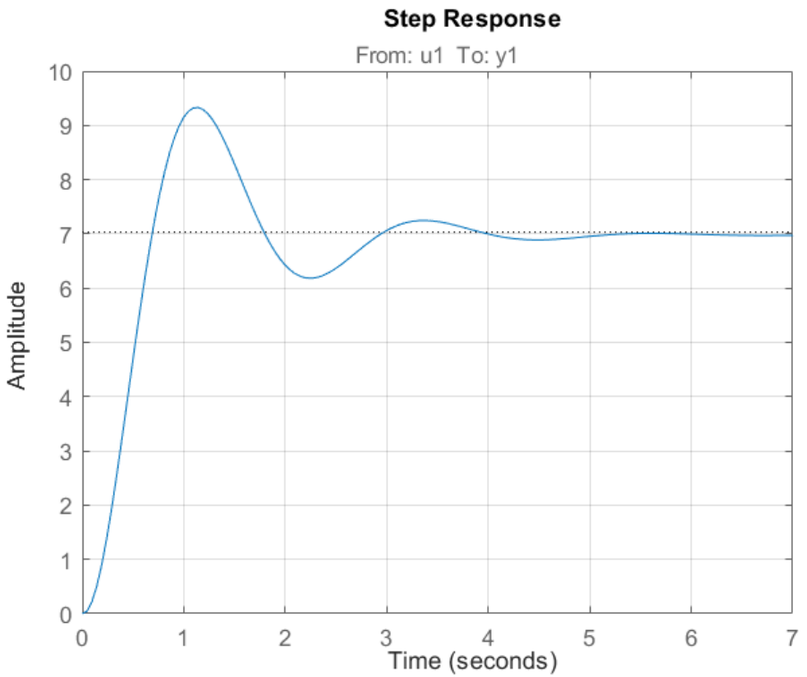
Low SNR Dataset

Second-order model with no zero.



Rise-Time: 0.4630 sec
Settling-Time: 3.7670 sec
Overshoot: 33.27%
Peak-Time: 1.1330 sec

Third-order model with two zeros.



Rise-Time: 0.4618 sec
Settling-Time: 4.5325 sec
Overshoot: 32.70%
Peak-Time: 1.1387 sec

Identification & Validation of Transfer Function Models

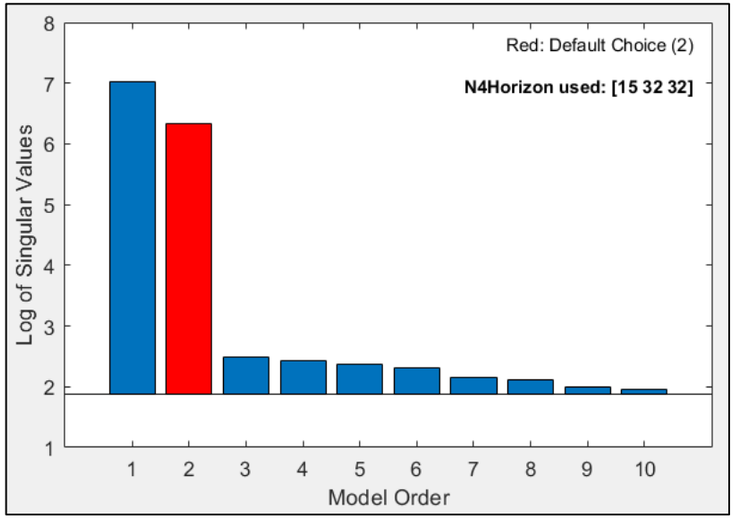
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 3: Order Selection & Delay Estimation

High SNR Dataset

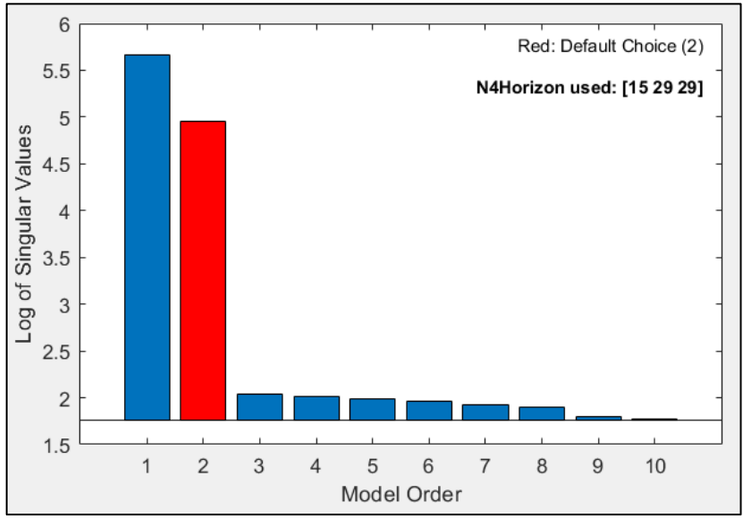
```
n4sid(dataH,1:10);
```



Selected order = 2

Low SNR Dataset

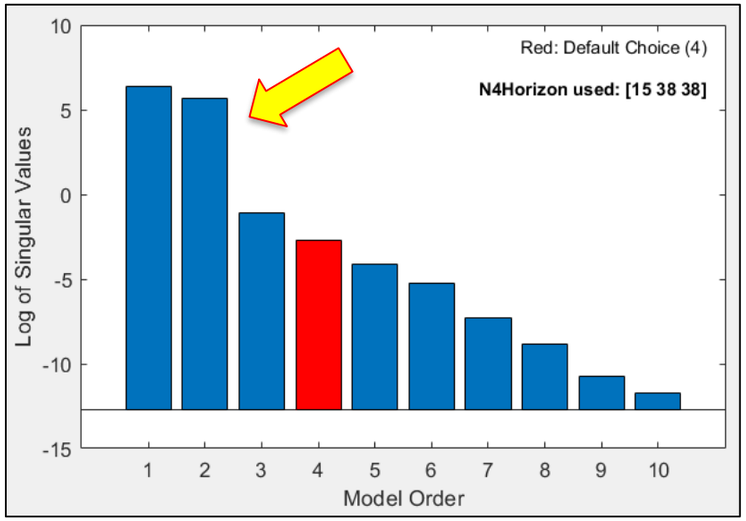
```
n4sid(dataL,1:10);
```



Selected order = 2

Noise-free Dataset

```
n4sid(data,1:10);
```



Recommended order = 4
(Possibility of the second-order model)

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 4: Select the Model Structure & Identify the Transfer Function Model

Noise-free Dataset

Fourth-order model with three zeros.

```
np = 4;
nz = 3;
sys4 = tfest(datai,np,nz,0)
```

```
sys4 =

     -0.01879 s^3 + 4.789 s^2 + 82.79 s + 77
  ------------------------------------------
  s^4 + 18.28 s^3 + 56.22 s^2 + 171.7 s + 131.5


Fit to estimation data: 99.94%
FPE: 1.936e-05, MSE: 1.921e-05
```

$$G_4(s) = \frac{-0.01879s^3 + 4.789s^2 + 82.79s + 77}{s^4 + 18.28s^3 + 56.22s^2 + 171.7s + 131.5}$$

Second-order model with no zero.

```
np = 2;
nz = 0;
sys5 = tfest(datai,np,nz,0)
```

```
sys5 =

          5.052
  --------------------
  s^2 + 1.981 s + 8.622


Fit to estimation data: 99.71%
FPE: 0.0004462, MSE: 0.0004447
```

$$G_5(s) = \frac{5.052}{s^2 + 1.981s + 8.622}$$

Identification & Validation of Transfer Function Models

Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation (Model Validity Criteria)

Noise-free Dataset

```
present(datai)
```

Fourth-order model with three zeros.

```
sys4 =

  -0.01879 (+/- 0.001061) s^3 + 4.789 (+/- 0.6271) s^2 + 82.79 (+/- 167.9) s + 77 (+/- 165.3)
  ---------------------------------------------------------------------------------------------
  s^4 + 18.28 (+/- 33.05) s^3 + 56.22 (+/- 98.15) s^2 + 171.7 (+/- 351.6) s + 131.5 (+/- 282.4)

Fit to estimation data: 99.94%
FPE: 1.936e-05, MSE: 1.921e-05
```

Second-order model with no zeros.

```
sys5 =

                  5.052 (+/- 0.002549)
  ------------------------------------------------
  s^2 + 1.981 (+/- 0.001207) s + 8.622 (+/- 0.003655)

Fit to estimation data: 99.71%
FPE: 0.0004462, MSE: 0.0004447
```

- The fit percent, FPE and MSE are all in good range for both models.
- However, the fourth-order model has large confidence intervals.

Identification & Validation of Transfer Function Models

Example 5

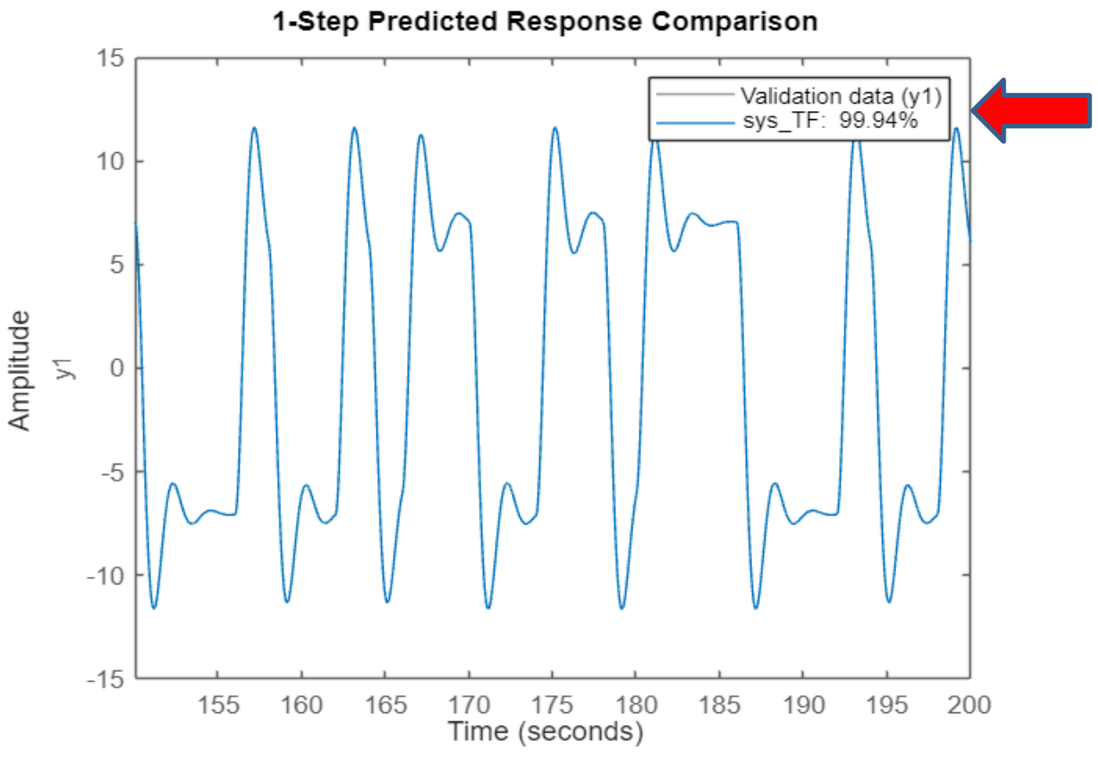
This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation (Cross-Validation)

Noise-free Dataset

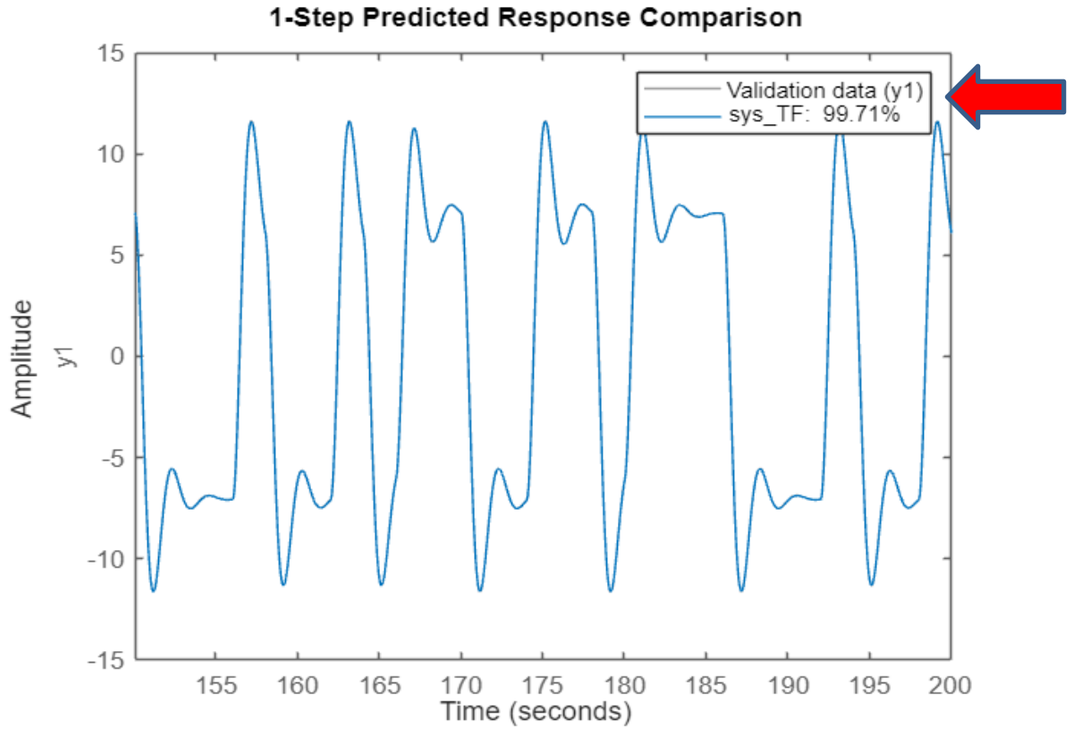
Fourth-order model with three zeros.

```
compare(datav,sys4,1)
```



Second-order model with one zero.

```
compare(datav,sys5,1)
```



Identification & Validation of Transfer Function Models

Example 5

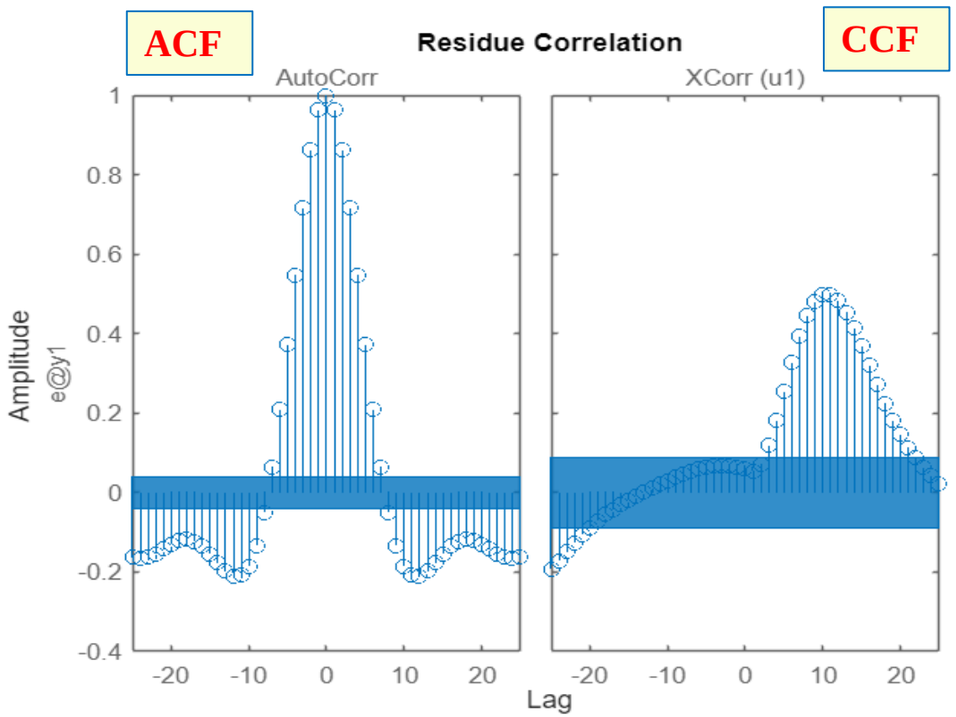
This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 5: Model Validation (Residual Analysis)

Noise-free Dataset

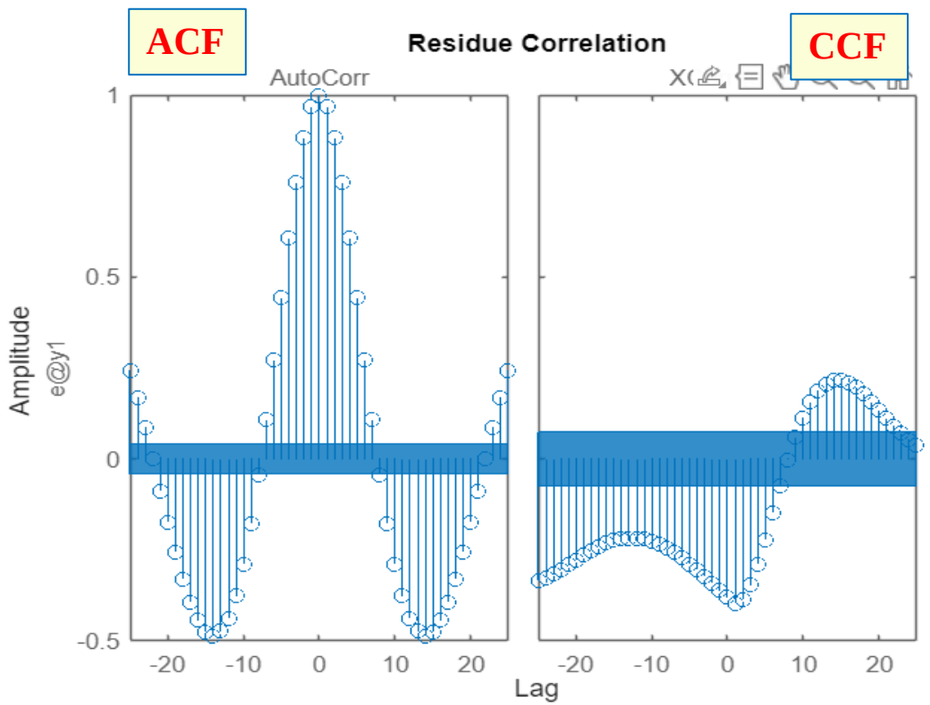
Fourth-order model with three zeros.

```
resid(sys4,data)
```



Second-order model with no zero.

```
resid(sys5,data)
```



Residual analysis is not applicable for noise-free dataset.

Identification & Validation of Transfer Function Models

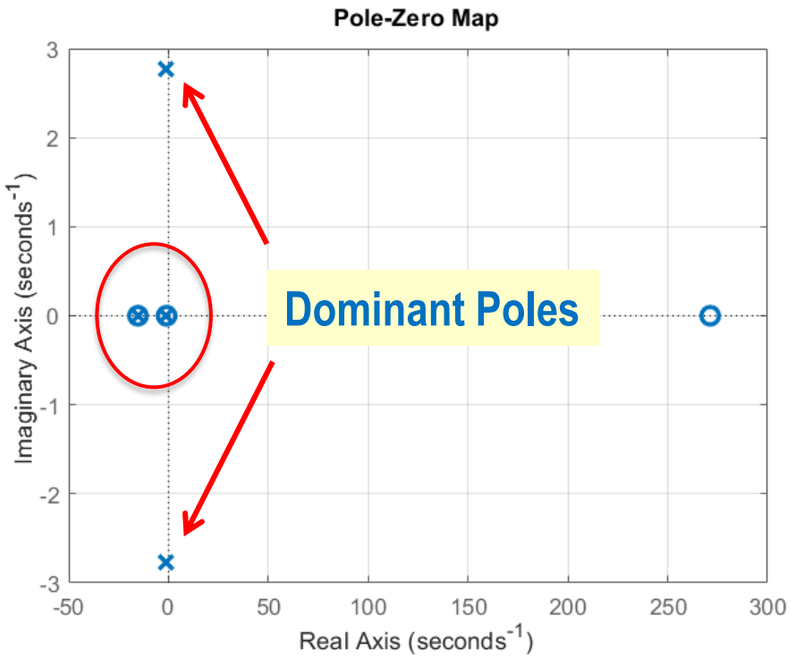
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

Step 6: Check the pole/zero locations

Noise-free Dataset

Fourth-order model with three zeros.

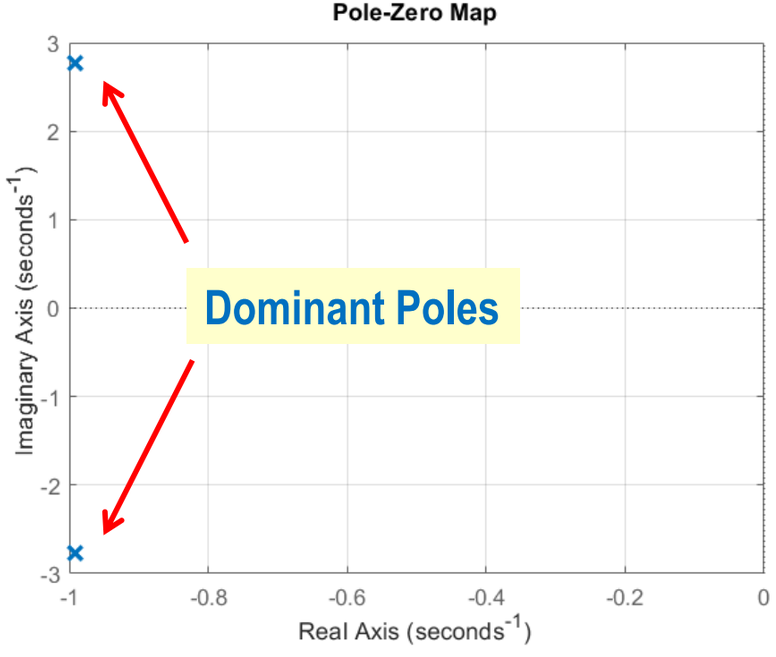


Poles:
$-0.994 \pm j2.775$
-15.31
-0.989

Zeros:
$+271.24$
-15.32
-0.987

Pole-zero cancellation represents an **over-parametrized** model. The fourth-order model can be approximated as a second-order model based on the dominant poles.

Second-order model with no zero.



Poles:
$-0.991 \pm j2.764$

Identification & Validation of Transfer Function Models

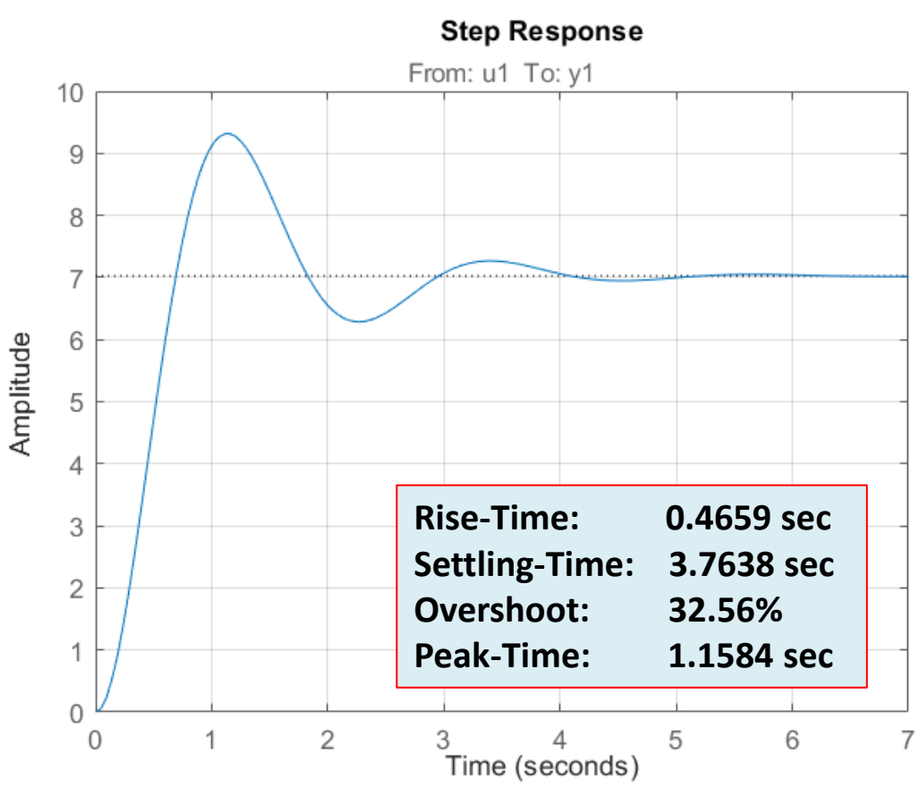
Example 5

This example shows how to identify and validate a CT transfer function model from input-output data, and effect of SNR on the quality of the estimated model.

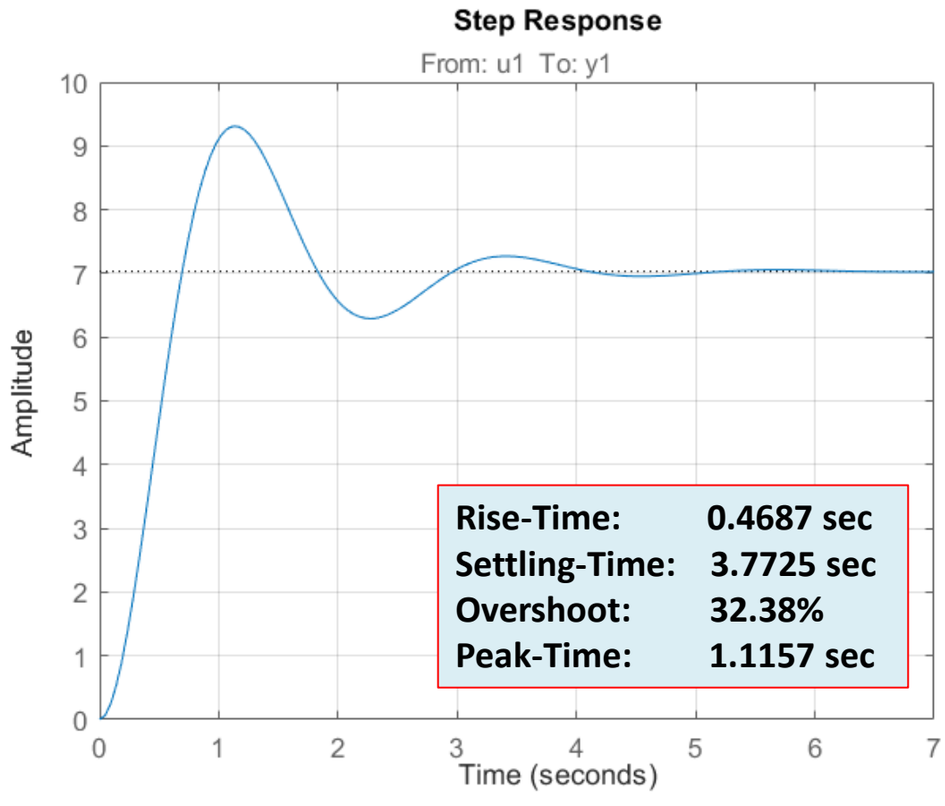
Step 7: Check the Step Response and Time-domain Specifications

Noise-free Dataset

Fourth-order model with three zeros.



Second-order model with no zero.



THANK YOU

HUMBER

WE ARE HUMBER