EE4307 Control System Design and Simulation

AY2021/2022 Semester 2

Project

Philippe Andrin Brigger
A0248834U

Contents

1 Introduction

This project aims at getting hands-on experience in system identification. In particular, parametric system identification methods are explored with tools in Matlab. The resulting models could then be used to design and simulate control systems. The steps for modeling are treating the raw data, guessing the model delay and order and then estimating the model parameters. ARX, ARMAX, BJ and OE are the four identification methods which were used. Once a model is fit to the data, the model is validated with a validation dataset. In this report, the modeling results and learning experiences of the project are discussed.

2 Parametric System Identification

In this project a data set is given. A model is fit to the data set with parametric identification methods. This section discusses the process of finding a model and assesses the model quality.

2.1 Data

The data set of this project consists of discrete input and output data points $u[k]$, $y[k]$ which are sampled at a sampling time $T = 0.1\,\text{s}$. The input of the data set is a pseudo-random binary sequence. Additionally, the frequency response of the system is provided. The frequency response is obtained from a spectrum analysis of the input and output data. Since only one data set is given, the data set is split up into two parts. The first half of the data set is defined as the identification set Z while the second half is used as a validation data set Z_v. To prevent the transient part of the system response to influence the model fitting, the first few samples of the data set are not used in the identification data set.

$$Z = [y(50:550)\ u(50:550)]$$
$$Z_v = [y(650:1000)\ u(650:1000)]$$

2.2 Data Treatment

The output data must be analyzed, and if necessary treated, before it can be used for the model fitting. It must be ensured that the output data set doesn't have any outliers or DC gain, as this would make the model fit inaccurate. Figure 1 suggests

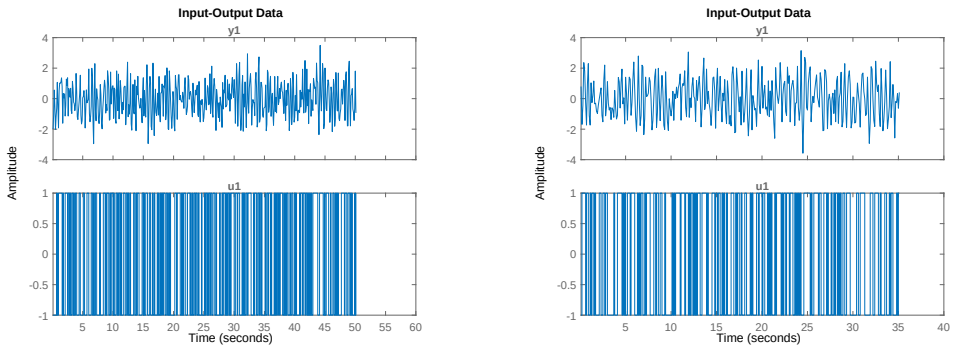


Figure 1: Input and output data for original data set. L: Identification data set. R: Validation data set.

that there are no significant outliers in either data sets. However, both data sets have a slight DC gain (the mean of the output data is not zero). This DC gain is subtracted from the outputs of the data sets. After this treatment, the model fitting can begin.

2.3 Guess of Delay

The first step of finding a model is figuring out the output delay. One way to do this is by plotting the impulse response. Figure 2 suggests that the output has a delay nk of one sample. This assumption is backed up with the system identification

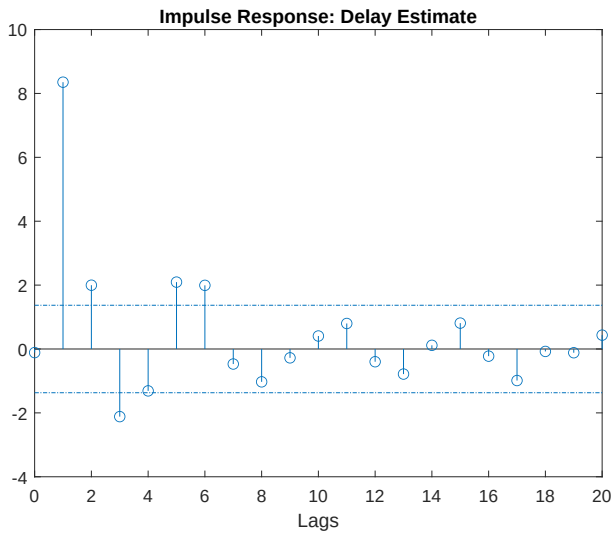


Figure 2: Impulse response of system.

toolbox function *arxstruc* which also identifies a system delay nk of one sample.

$$\begin{aligned} V &= arxstruc(Z, Z_v, struc(2, 2, 1:10)); \\ nn &= selstruc(V, 0) \\ \Rightarrow nk &= 1 \end{aligned}$$

2.4 Guess of Model Order

The next step of finding a model is guessing the model order. One approach is to use the frequency response of the system. The frequency response in fig. 3 has one peak and one valley. This suggests that the model has a 2^{nd} order numerator and 2^{nd} order denominator. A second approach is to use the *arxstruc* and *selstruc* system identification toolbox functions. Figure 4 shows the misfit percentage for varying model orders. Since the misfit is still large for a high order model (a high order model in general should produce a great fit of the identification data, but is prone to over-fitting and possibly doesn't model the system well and is in general more complicated), the initial model order guess from fig. 3 of $na = 2$ and $nb = 2$ is kept.

2.5 Model Parameter Estimation

Once the model order and delay are guessed, the parameters of the model can be estimated. The four model parameter estimation methods used and compared are ARX, ARMAX, BJ and OE.

2.6 Model Quality Assessment

The model fitting is done for one specific data set. The higher the order of the model is, the better the fit will be. This, however, bears the risk of overfitting. In other words, if a different data set is taken, the output of the model will no longer match the output of the data set very well. Therefore, before using a model for control system design purposes, it is important to first validate the different models with a different, validation data set.

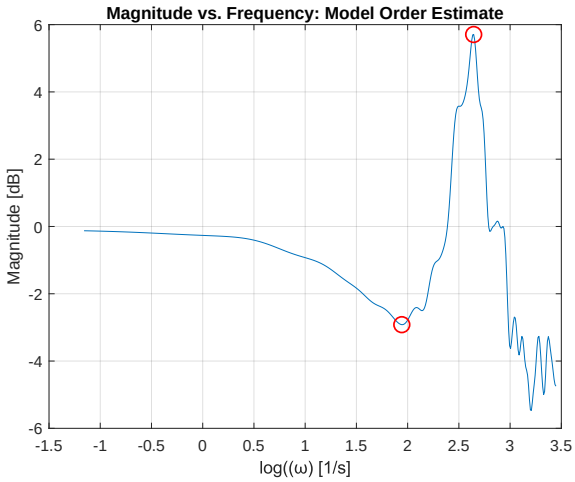


Figure 3: Frequency response of system.

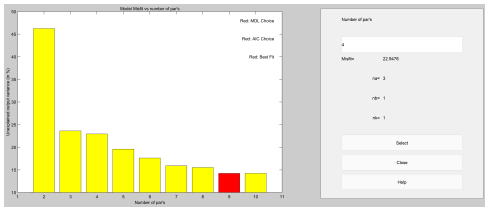


Figure 4: Model order guess with *selstruc*.

2.6.1 Frequency Response Comparison

The first thing that is validated is the frequency response between the model and the validation data set. The frequency response of the smoothened data set shows that the frequency responses of the BJ and OE models are similar to the frequency response of the validation data set (see fig. 5). The ARX model approximates the validation data set decently while the ARMAX model is not accurate enough.

2.6.2 Residual Analysis

Next, the means and variances of the residuals of the four models are compared to assess the quality of the models. Figure 6 shows that the BJ model is the only model where the auto-correlation of the residual and the cross-correlation between residual and input stays within the 99% confidence interval which is shaded in blue. This suggests that the BJ model is the best fit. It is to be noted, that although the frequency response of the OE model is good (see fig. 5), the auto-correlation of the residual is not good. The ARX and ARMAX models have a better residual auto-correlation than the OE model. However, their cross-correlation between residual and input doesn't stay within the confidence interval for all lags. The histograms in fig. 7 support the findings from the auto-correlation and cross-correlation plots that the BJ model is the best fit. The BJ model has a mean closest to zero and the smallest variance of all four models. Although the mean of the OE model is quite small, the data has a large variance which is undesirable. The ARX and ARMAX models also have quite a large variance which is not good. These validation plots underline that the best model fit in this case is obtained with the BJ identification method.

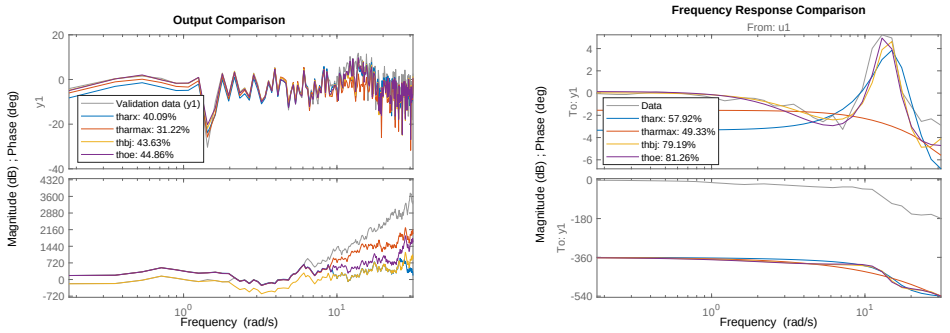


Figure 5: Noisy (L) and smoothened (R) frequency responses of models and validation data.

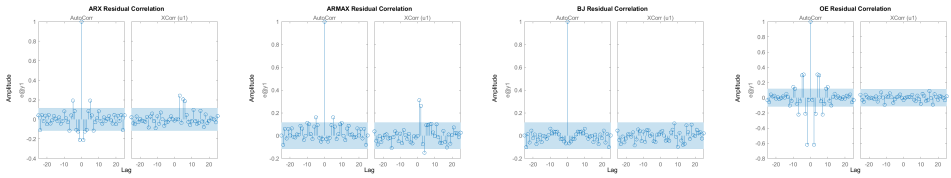


Figure 6: Auto-correlation and cross-correlation plots for different parametric identification methods with 99% confidence region shaded in blue.

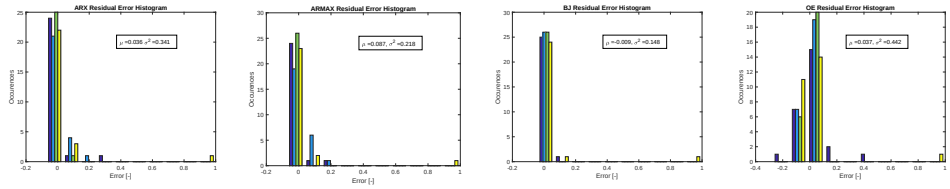


Figure 7: Histogram plot of the residuals of the four identification methods with the residual means and variances.

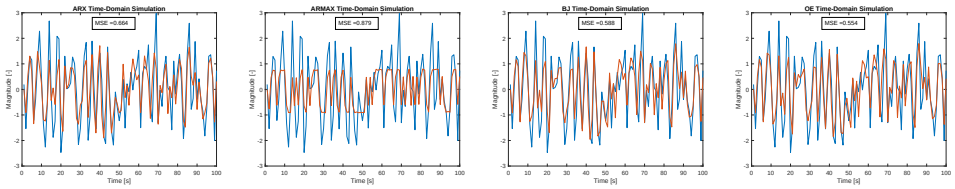


Figure 8: Time domain simulation comparison between model fit and validation data set.

2.7 Time Domain Simulation

A final validation step is to run a time domain simulation of the model with the validation input data and to compare the model output with the simulated validation data set output. Figure 8 shows the time domain simulation output of the model and the validation data set for the four different models. Additionally, the mean squared error (MSE) is given for each identification method. It can be seen that apart from the ARMAX model, all models produce a similar output as the validation set. The mean squared errors of the ARX, BJ and OE models are within a comparable range. Since the residual plots of the OE model and ARX model clearly are worse than the residual plots of the BJ model, the BJ model is the most promising model to use for control system purposes.

3 Conclusion

A parametric system identification approach was used to derive a system model in this project. The four different model identification methods ARX, ARMAX, BJ and OE were used and compared with each other. In this case, the BJ model was deemed as the best fit. It was seen that system identification is as much an art as a science. It is not a-priori clear which model order and identification method parameters should be used to obtain a model which doesn't overfit the identification data set but actually models the true system accurately and reliably. Nevertheless, many tools were acquired in this course with which the identification process can be broken down into different fundamental steps: Data acquisition (for this project the data was already provided), data treatment, delay estimation, model order estimation, parameter estimation and model assessment. Of the above-mentioned steps, especially the parameter estimation step is prone to overfitting and there isn't a clear methodology of choosing appropriate model identification parameter settings (e.g. what the polynomial orders of the BJ model should be). In the end, experience is key to make smart guesses. Luckily, the validation step acts as a safety net step, and can show whether the guesses made in the previous steps are good or whether different parameters should be tested. Clearly, one major challenge which was not encountered in this project is the data acquisition itself. It would be beneficial to extend this project to include a data acquisition step. Acquiring reliable data can be very challenging and cumbersome since the experimental setup directly influences the quality of the data and measurements are always subject to noise. In the end, the quality of the acquired data also limits the quality of the model. To conclude, a system identification engineer must be creative and flexible when acquiring data and trying to fit a model to the acquired data. The balance and uncertainty of the whole process using scientific methods and intuition to choose certain parameters makes this field of engineering very interesting and exciting.

A Appendix

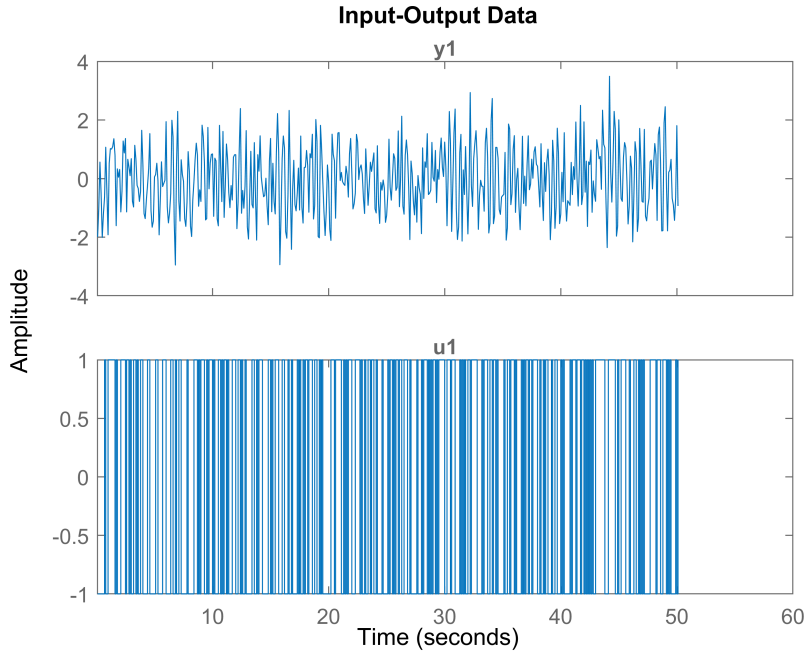
Project 2 EE4307

In this project, a system is identified from a dataset with input and output data. A parametric approach is used to find a system model.

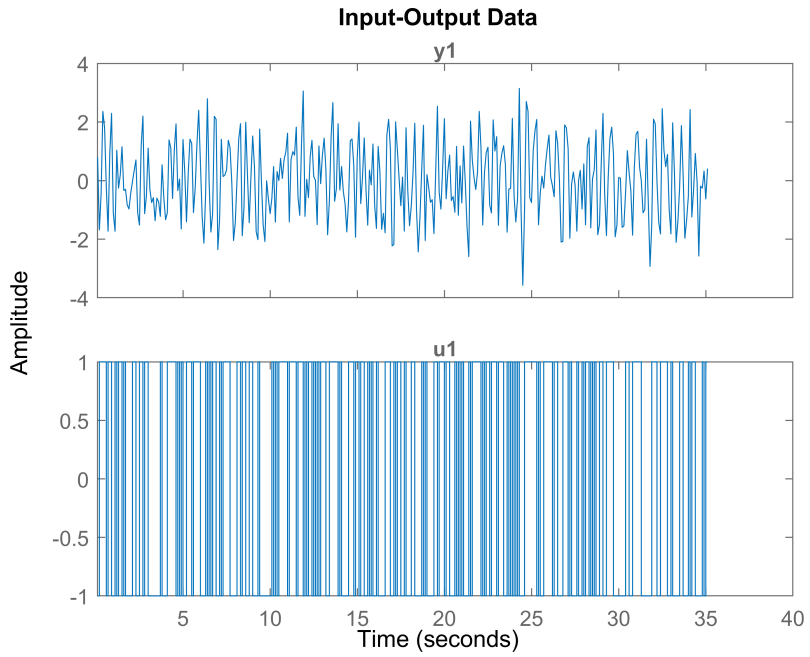
```
% clean up
clear all; close all; clc; fig_num = 1;
```

Data Treatment

```
load ee4307_project.mat
uz = u(50:550);
yz = y(50:550);
uzv = u(650:1000);
yzv = y(650:1000);
Z = iddata(yz,uz,T);
ZV = iddata(yzv,uzv,T);

figure(fig_num); fig_num = fig_num + 1; plot(Z);
xlim = [0,50]; saveas(gcf,'original_input_output.pdf');
```




```
figure(fig_num); fig_num = fig_num + 1; plot(ZV);
xlim = [0,35]; saveas(gcf,'original_validation_input_output.pdf');
```

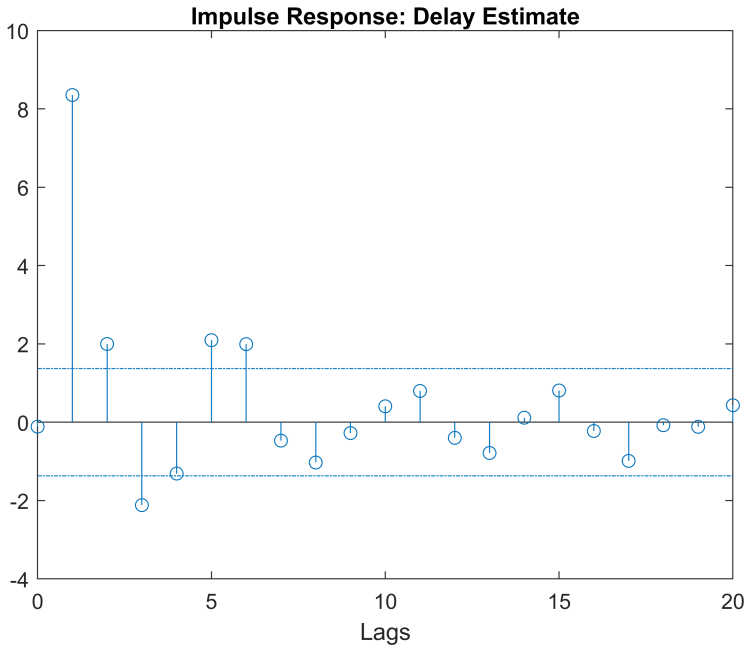



```
ans = -0.0380
ans = 0.0192
```

```
% The mean of the output data should be zero which initially isn't the case.
% Therefore, the mean is subtracted from the output data.
% There are no evident outliers visible.
yz = y(50:550)-mean(yz);
yzv = y(650:1000)-mean(yzv);
Z = iddata(yz,uz,T);
ZV = iddata(yzv,uzv,T);
```

Guessing the Delay

```
figure(fig_num); fig_num = fig_num + 1; ir = cra(Z); title('Impulse Response: Delay Estimate');
```




```
V = arxstruc(Z,ZV,struc(2,2,1:10));
nn = selstruc(V,0);
nk = nn(3);
```

```
nk = 1
```

```
% The delay of the system is 1;
```

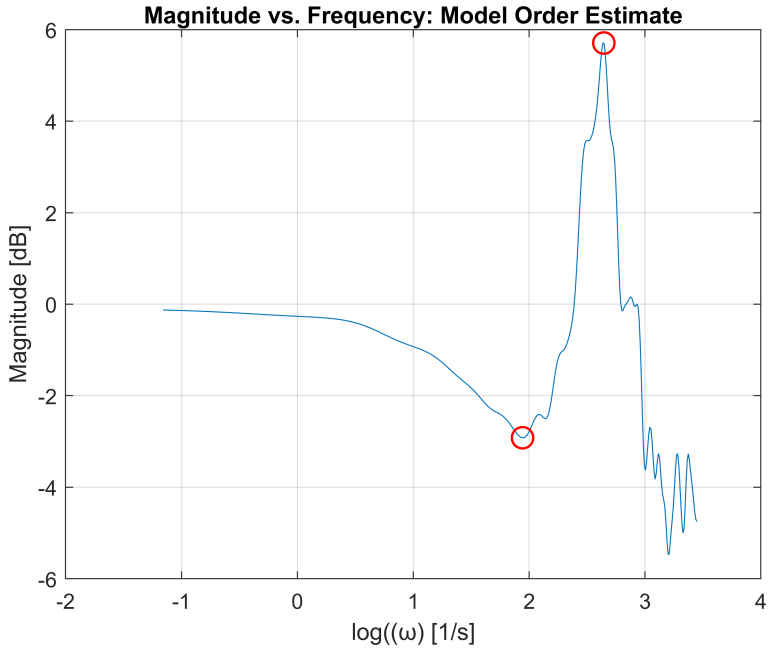
Guessing the Model Order

```
log_w = log(w); % Take logarithm of frequency data
mag_dB = mag2db(mag); % Convert magnitude to dB
figure(fig_num); fig_num = fig_num+1;
plot(log_w,mag_dB)
grid on
title('Magnitude vs. Frequency: Model Order Estimate')
ylabel('Magnitude [dB]')
xlabel('log((ω) [1/s]')
hold on;
[max_val, index] = max(mag_dB);
[min_val, index_min] = min(mag_dB(1:index));
```

```
min_val = -2.9183
index_min = 337
```

```
plot(log_w(index),mag_dB(index),'or','MarkerSize',10,'LineWidth',1.1)
plot(log_w(index_min),mag_dB(index_min),'or','MarkerSize',10,'LineWidth',1.1)
hold off;
```

```
saveas(gcf,'model_order_estimate.pdf')
```




```
% V = arxstruc(Z,ZV,struc(1:5,1:5,nk));
% nn = selstruc(V,0)
% nns = selstruc(V)

% The model fit is bad with selstruc. One peak, one valley.
% Therefore na 2, nb = 2
na = 2; nb = 2;
```

Model Parameter Estimation

```
tharx = arx(Z,[na nb nk]);
tharx = set(tharx,'Ts',T);
present(tharx);
```

```
tharx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 0.08293 (+/- 0.03297) z^-1 + 0.4609 (+/- 0.02316) z^-2

  B(z) = 0.8213 (+/- 0.02635) z^-1 + 0.1166 (+/- 0.03773) z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.
```

```
Status:
Estimated using ARX on time domain data "Z".
Fit to estimation data: 49.24% (prediction focus)
FPE: 0.3503, MSE: 0.342
More information in model's "Report" property.
```

```
tharmax = armax(Z,[na nb 7 nk]);
tharmax = set(tharmax,'Ts',T);
present(tharmax);
```

```
tharmax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 + 0.347 (+/- 0.4212) z^-1 - 0.06832 (+/- 0.08245) z^-2

  B(z) = 0.6887 (+/- 0.01528) z^-1 + 0.3816 (+/- 0.2825) z^-2


  C(z) = 1 - 0.1399 (+/- 0.4212) z^-1 - 1.083 (+/- 0.293) z^-2 - 0.1387 (+/- 0.3153) z^-3 + 0.7298 (+/- 0.1459) z^-4
                                  + 0.3912 (+/- 0.2526) z^-5 - 0.1452 (+/- 0.04724) z^-6 - 0.1463 (+/- 0.08736) z^-

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=7   nk=1
   Number of free coefficients: 11
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Termination condition: Maximum number of iterations reached..
Number of iterations: 20, Number of function evaluations: 202

Estimated using ARMAX on time domain data "Z".
Fit to estimation data: 57.11% (prediction focus)
FPE: 0.2552, MSE: 0.2442
More information in model's "Report" property.
```

```
thbj = bj(Z,[nb 4 4 4 nk]);
thbj = set(thbj,'Ts',T);
present(thbj);
```

```
thbj =
Discrete-time BJ model: y(t) = [B(z)/F(z)]u(t) + [C(z)/D(z)]e(t)
  B(z) = 0.8332 (+/- 0.01842) z^-1 - 0.1077 (+/- 0.06949) z^-2

  C(z) = 1 - 2.609 (+/- 0.1615) z^-1 + 2.351 (+/- 0.4119) z^-2 - 0.6874 (+/- 0.3779) z^-3 - 0.05348 (+/- 0.1267) z^-

  D(z) = 1 - 1.909 (+/- 0.156) z^-1 + 1.76 (+/- 0.2639) z^-2 - 1.174 (+/- 0.2218) z^-3 + 0.3224 (+/- 0.1141) z^-4


  F(z) = 1 - 0.378 (+/- 0.08701) z^-1 + 0.3417 (+/- 0.04226) z^-2 - 0.0139 (+/- 0.05062) z^-3 - 0.2266 (+/- 0.03288)

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   nb=2   nc=4   nd=4   nf=4   nk=1
   Number of free coefficients: 14
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
```

```
Termination condition: No improvement along the search direction with line search..
Number of iterations: 10, Number of function evaluations: 205

Estimated using BJ on time domain data "Z".
Fit to estimation data: 64.19% (prediction focus)
FPE: 0.18, MSE: 0.1702
More information in model's "Report" property.
```

```
thoe = oe(Z,[nb 5 nk]);
thoe = set(thoe,'Ts',T);
present(thoe);
```

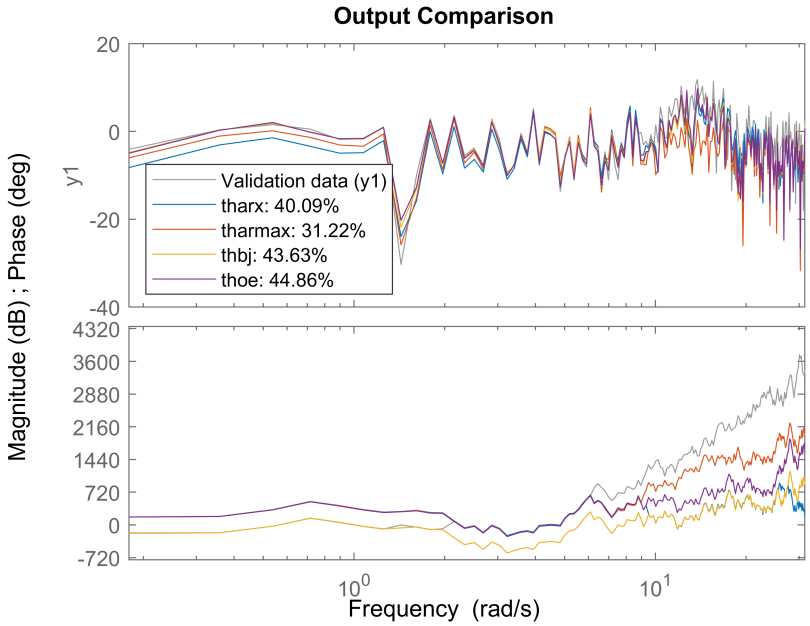
```
thoe =
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.831 (+/- 0.02868) z^-1 - 0.1309 (+/- 0.2656) z^-2


  F(z) = 1 - 0.3919 (+/- 0.3232) z^-1 + 0.3601 (+/- 0.09126) z^-2 - 0.02413 (+/- 0.1062) z^-3 - 0.1883 (+/- 0.03927)
                                                                                              - 0.06714 (+/- 0.08728) z^-

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   nb=2   nf=5   nk=1
   Number of free coefficients: 7
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Termination condition: Near (local) minimum, (norm(g) < tol)..
Number of iterations: 7, Number of function evaluations: 57

Estimated using OE on time domain data "Z".
Fit to estimation data: 41.29%
FPE: 0.4706, MSE: 0.4576
More information in model's "Report" property.
```

```
```

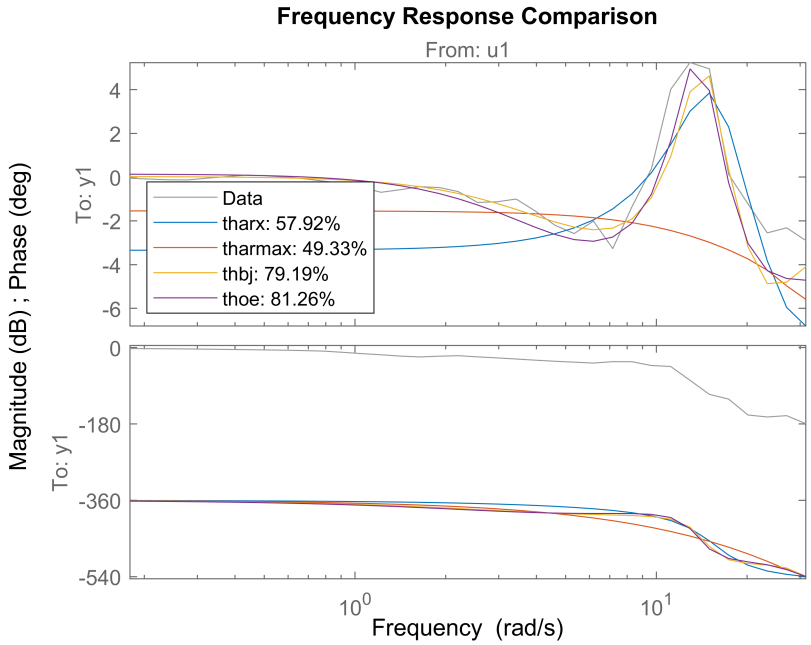
Model Quality Assessment

Frequency Responses

```
figure(fig_num); fig_num = fig_num+1;
compare(fft(ZV),tharx,tharmax,thbj,thoe)
saveas(gcf,'noisy_output_validation.pdf')
L = findobj(gcf,'type','legend');
L.Location = 'southwest';
```

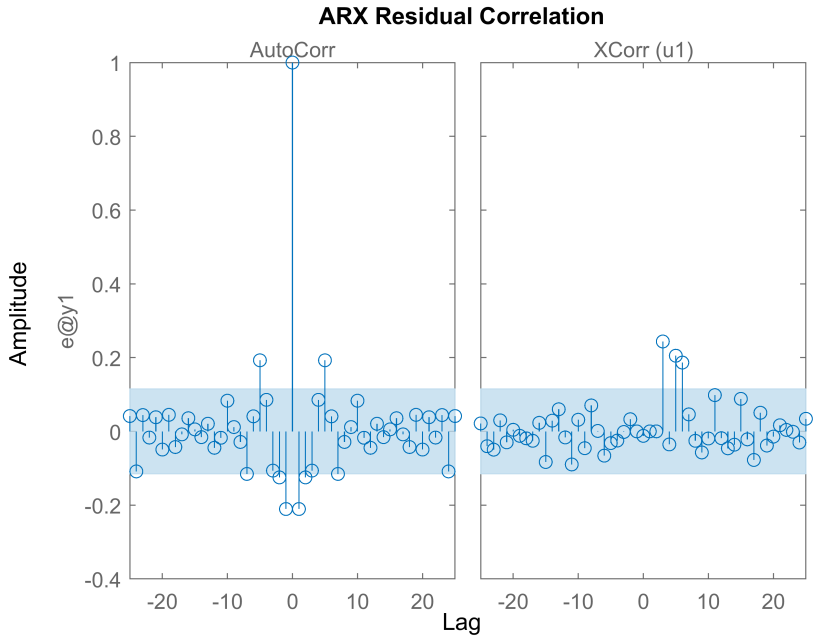



```
figure(fig_num); fig_num = fig_num+1;
compare(spafdr(ZV),tharx,tharmax,thbj,thoe)
saveas(gcf,'smooth_output_validation.pdf')
L = findobj(gcf,'type','legend');
L.Location = 'southwest';
```

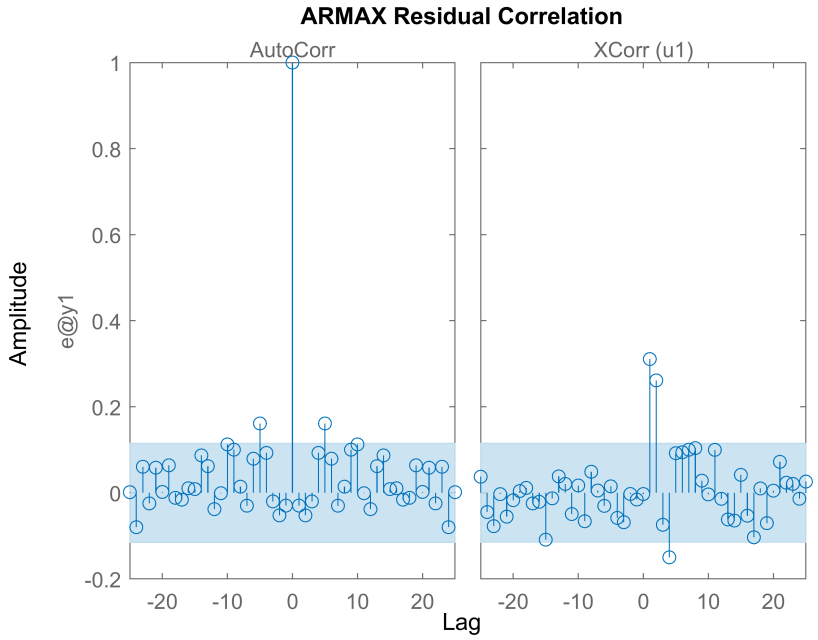


Residual Analysis

```
[E_ARX, R1] = resid(tharx, Z);
[E_ARMAX, R2] = resid(tharmax, Z);
[E_BJ, R3] = resid(thbj, Z);
[E_OE, R4] = resid(thoe, Z);

resid(tharx, Z);
title('ARX Residual Correlation')
saveas(gcf,'arx_residual_correlation.pdf')
```

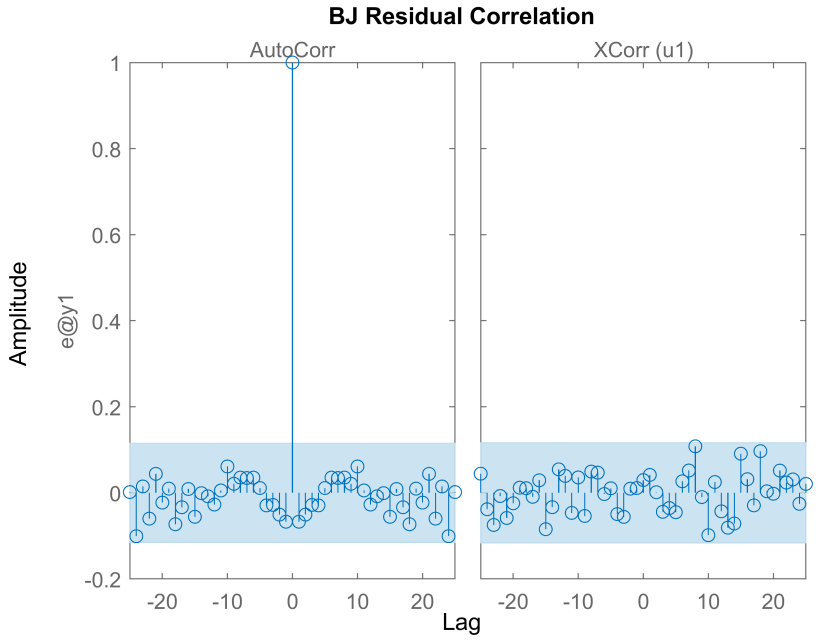



```
resid(tharmax, Z);
title('ARMAX Residual Correlation')
saveas(gcf,'armax_residual_correlation.pdf')
```



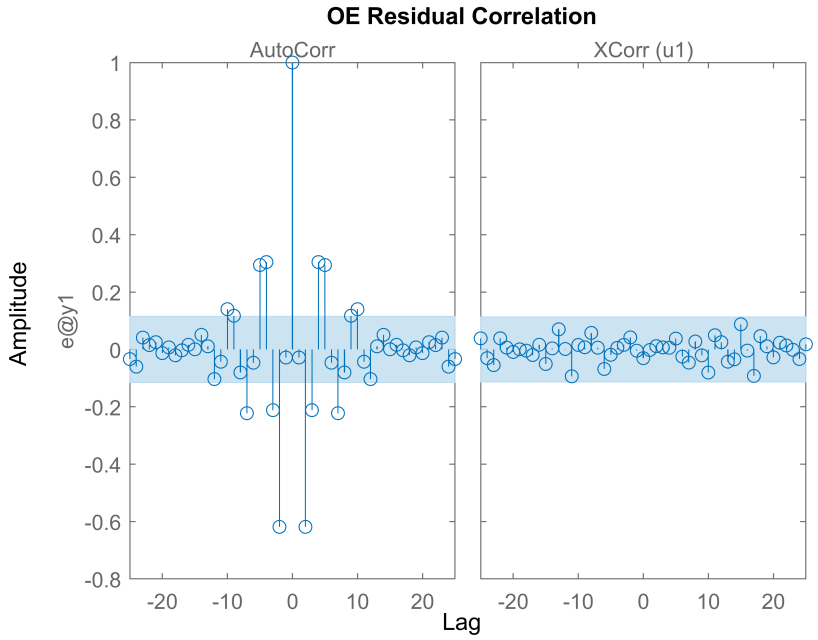

```
resid(thbj, Z);
title('BJ Residual Correlation')
saveas(gcf,'bj_residual_correlation.pdf')
```

10

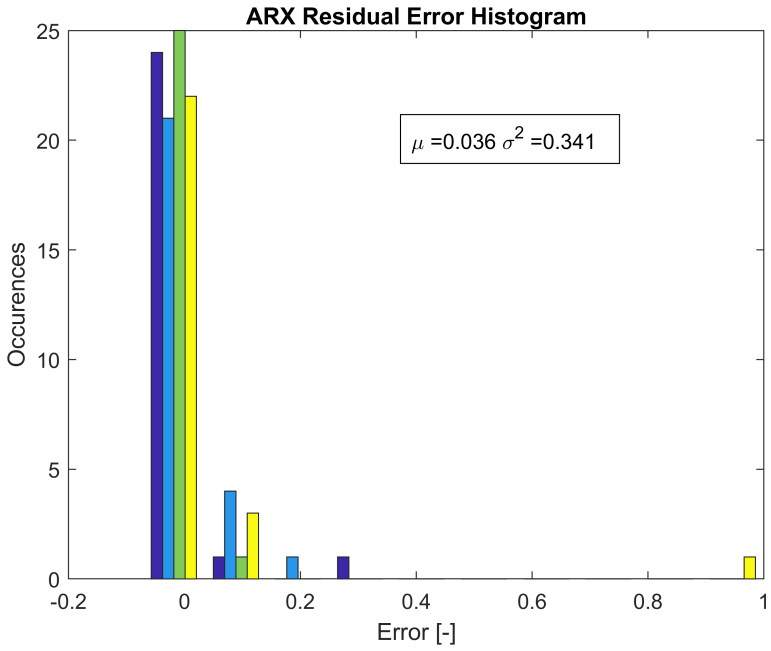



```
resid(thoe, Z);
title('OE Residual Correlation')
saveas(gcf,'oe_residual_correlation.pdf')
```

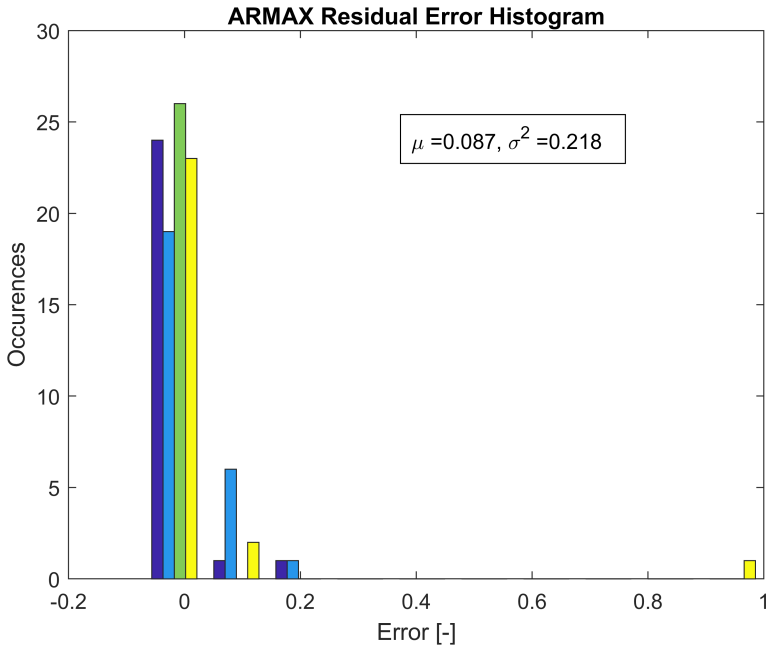
11



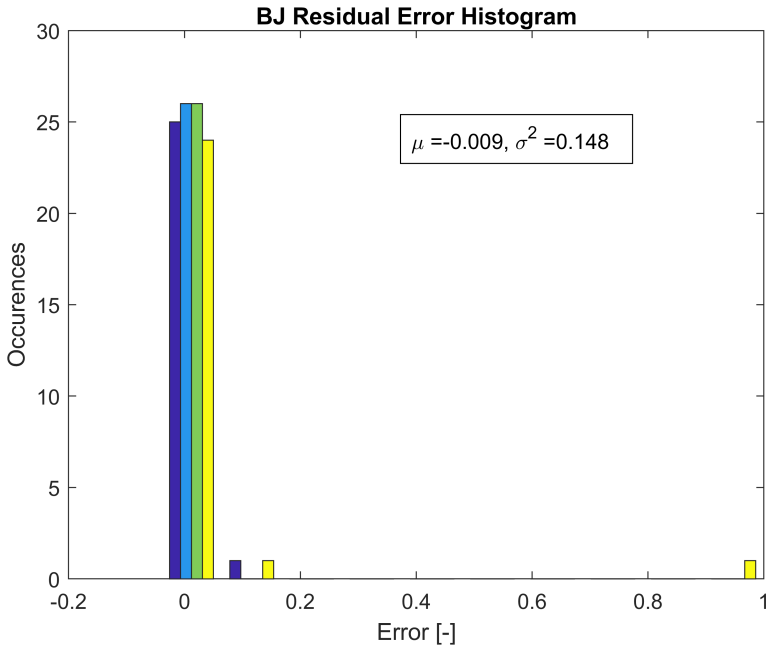

```
hist(R1);
title('ARX Residual Error Histogram')
xlabel('Error [-]')
ylabel('Occurences')
e = mean(E_ARX.y); v = var(E_ARX.y);
str = strcat('\mu = ',num2str(e,'%.3f'),' \sigma^2 = ',num2str(v,'%.3f'));
annotation('textbox',[.5 .5 .3 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'arx_residual_histogram.pdf')
```



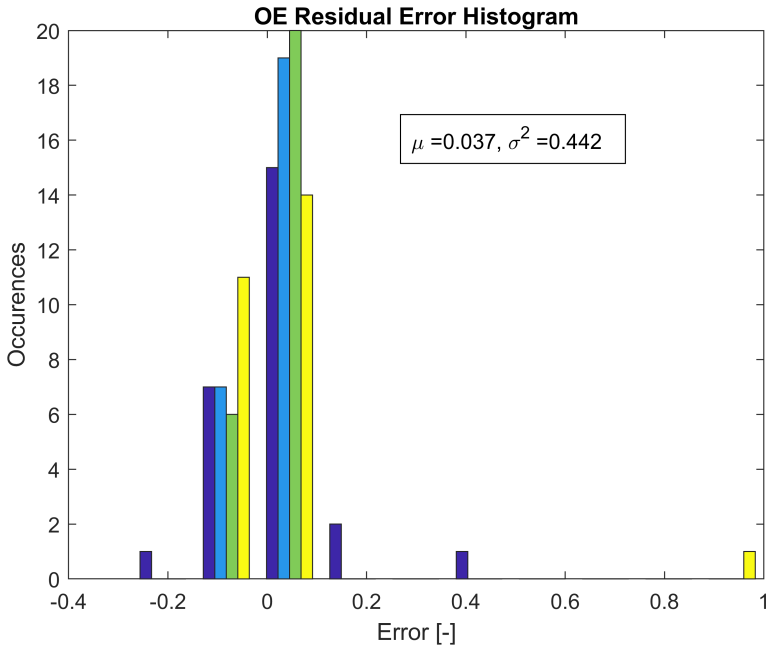

```
hist(R2);
title('ARMAX Residual Error Histogram')
xlabel('Error [-]')
ylabel('Occurences')
e = mean(E_ARMAX.y); v = var(E_ARMAX.y);
str = strcat('\mu = ',num2str(e,'%.3f'),', \sigma^2 = ',num2str(v,'%.3f'));
annotation('textbox',[.5 .5 .3 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'armax_residual_histogram.pdf')
```

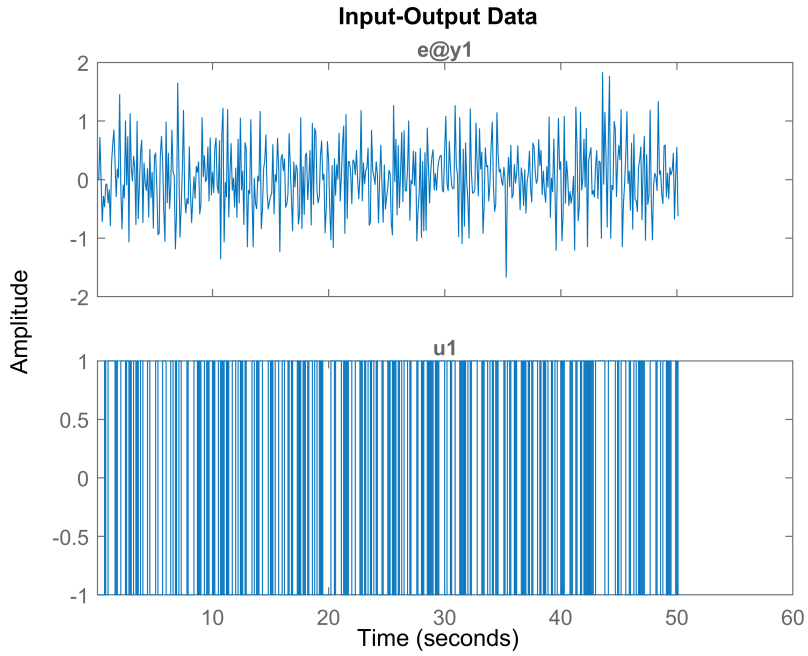



```
hist(R3);
title('BJ Residual Error Histogram')
xlabel('Error [-]')
ylabel('Occurences')
e = mean(E_BJ.y); v = var(E_BJ.y);
str = strcat('\mu = ',num2str(e,'%.3f'),', \sigma^2 = ',num2str(v,'%.3f'));
annotation('textbox',[.5 .5 .3 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'bj_residual_histogram.pdf')
```




```
hist(R4);
title('OE Residual Error Histogram')
xlabel('Error [-]')
ylabel('Occurences')
e = mean(E_OE.y); v = var(E_OE.y);
str = strcat('\mu = ',num2str(e,'%.3f'),', \sigma^2 = ',num2str(v,'%.3f'));
annotation('textbox',[.5 .5 .3 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'oe_residual_histogram.pdf')
```





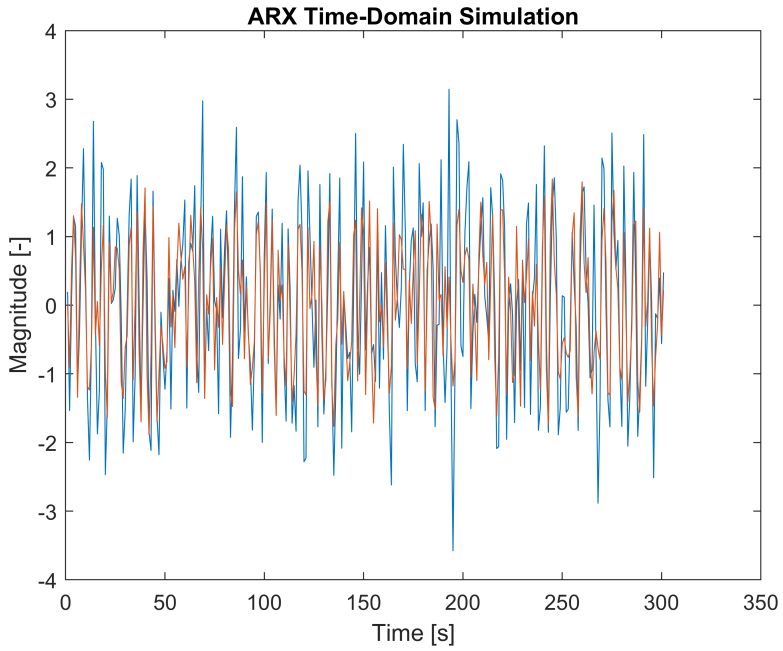

```
e = 0.0363
v = 0.3414
e = 0.0868
v = 0.2180
e = -0.0085
```

16

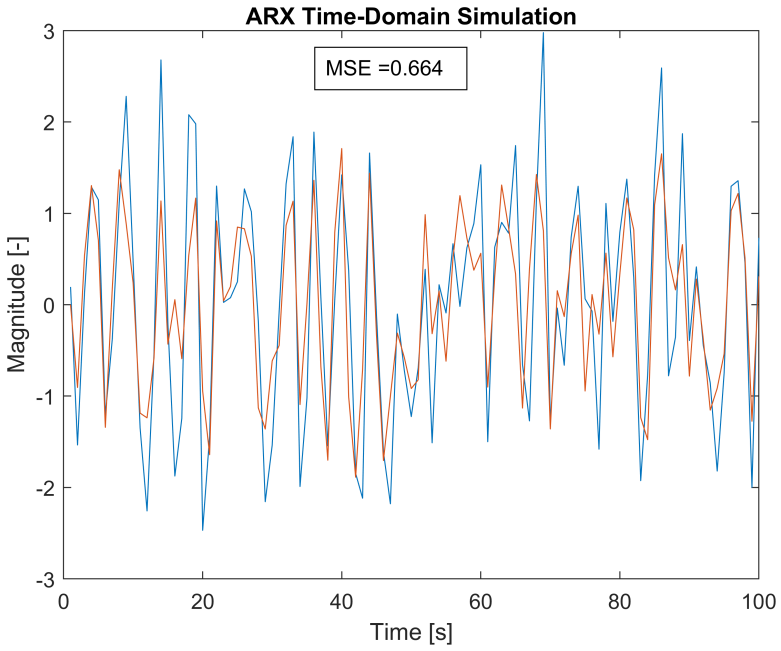
```
v = 0.1478
e = 0.0371
v = 0.4417
```

Time-Domain Simulation

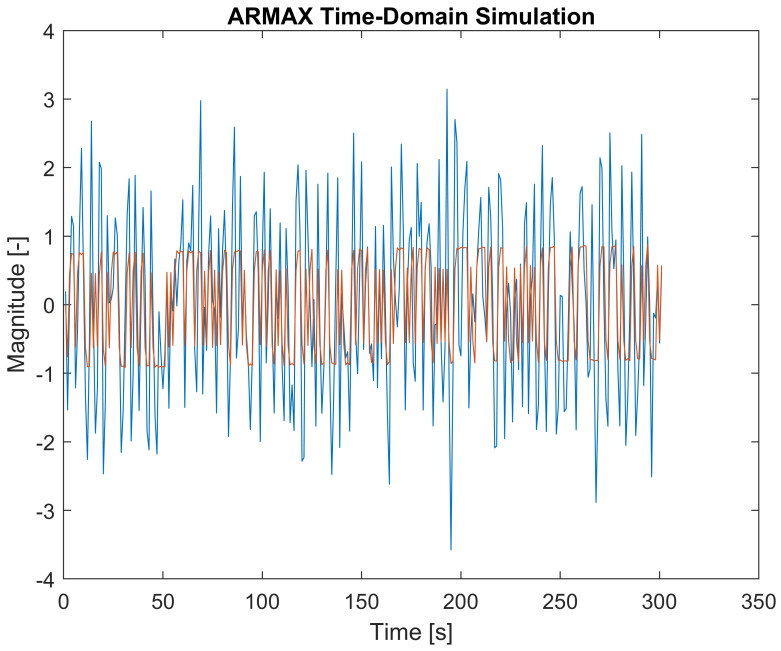
```
ux = detrend(u(700:1000));
yx = detrend(y(700:1000));
ysim = sim(ux,tharx);
figure(fig_num); fig_num = fig_num+1;
plot([yx ysim])
title('ARX Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
```



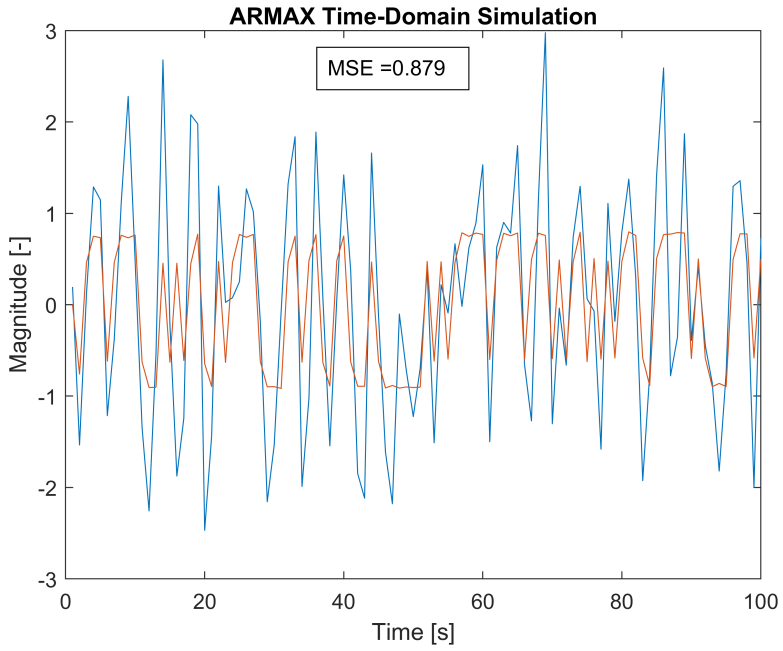

```
figure(fig_num); fig_num = fig_num+1;
plot([yx(1:100) ysim(1:100)])
title('ARX Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
str = strcat('MSE = ',num2str(mean((yx - ysim).^2),'%.3f'));
annotation('textbox',[.41 .6 .2 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'arx_time_domain_simulation.pdf')
```



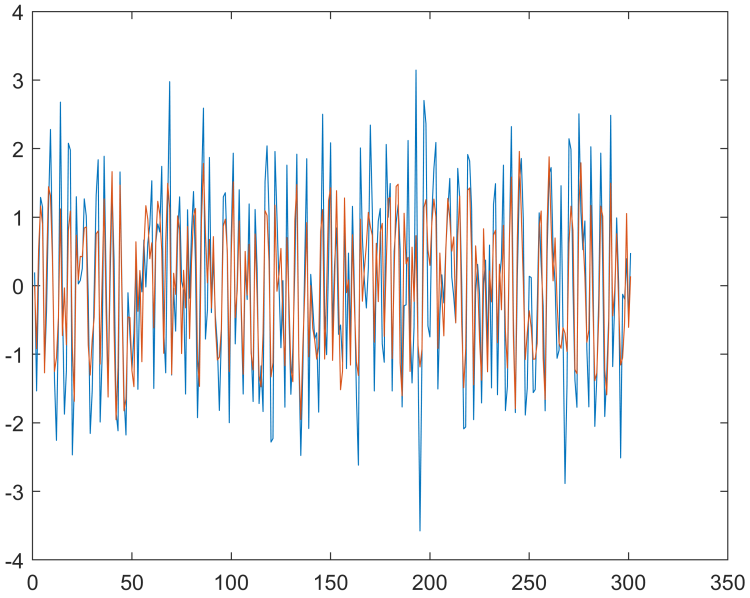

```
ysim = sim(ux,tharmax);
figure(fig_num); fig_num = fig_num+1;
plot([yx ysim])
title('ARMAX Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
```




```
figure(fig_num); fig_num = fig_num+1;
plot([yx(1:100) ysim(1:100)])
title('ARMAX Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
str = strcat('MSE = ',num2str(mean((yx - ysim).^2),'%.3f'));
annotation('textbox',[.41 .6 .2 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'armax_time_domain_simulation.pdf')
```

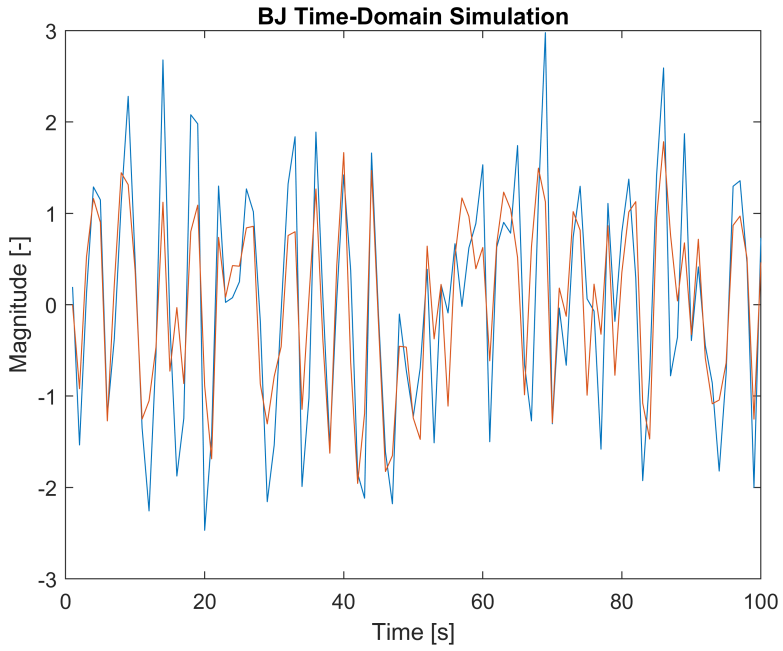



```
ysim = sim(ux,thbj);
figure(fig_num); fig_num = fig_num+1;
plot([yx ysim])
```

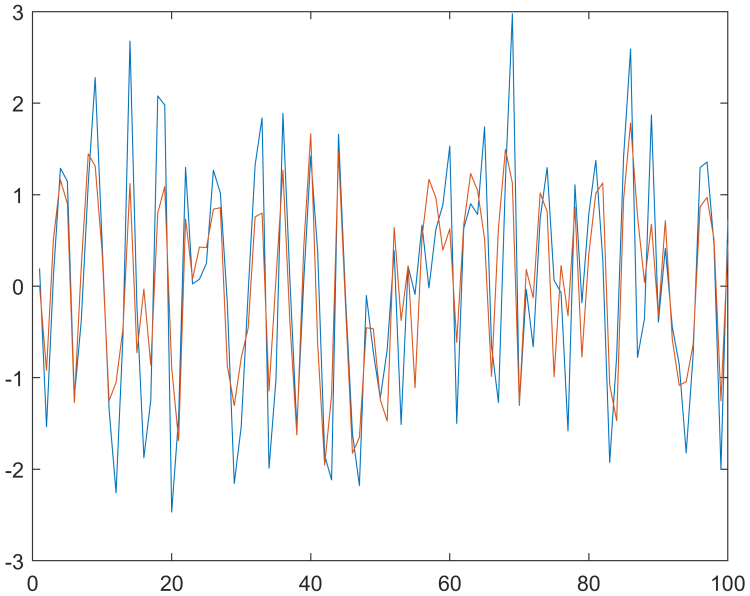



```
plot([yx(1:100) ysim(1:100)])
title('BJ Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
```

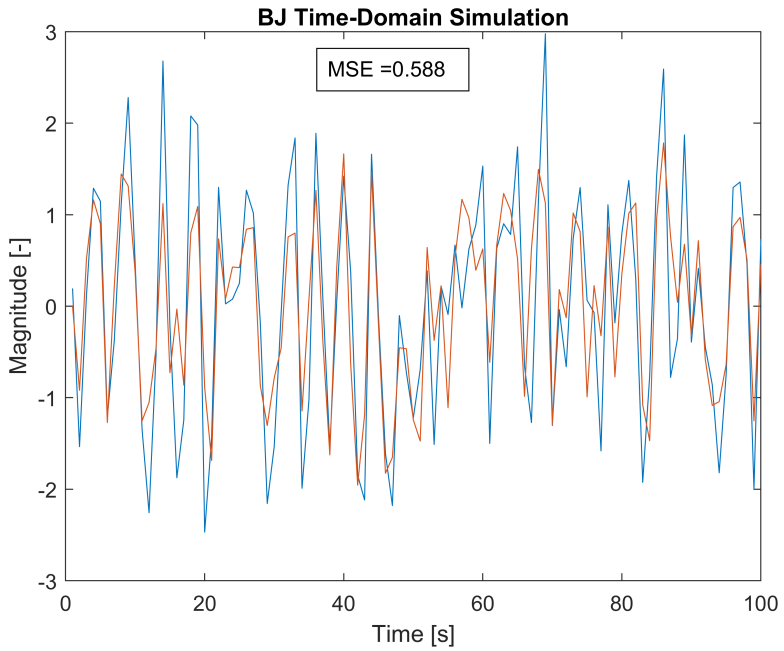
21



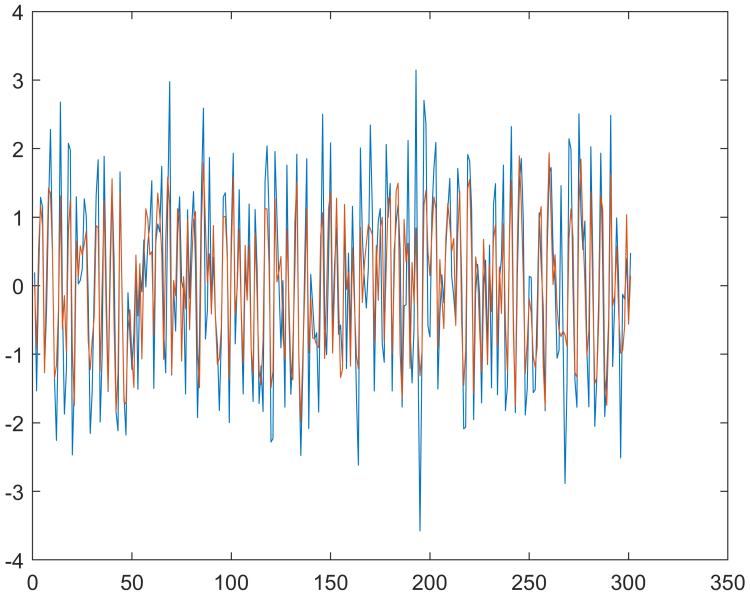

```
figure(fig_num); fig_num = fig_num+1;
plot([yx(1:100) ysim(1:100)])
```



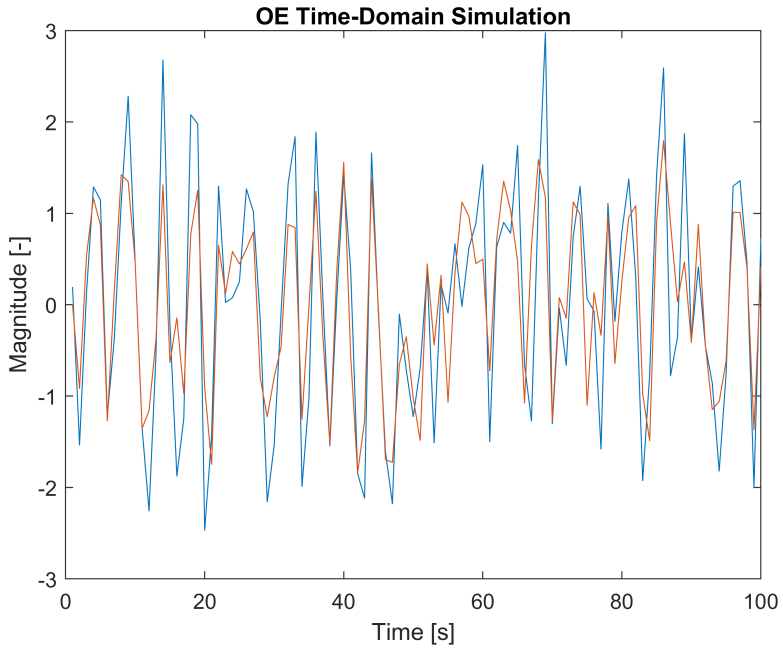
```
plot([yx(1:100) ysim(1:100)])
title('BJ Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
str = strcat('MSE = ',num2str(mean((yx - ysim).^2),'%.3f'));
annotation('textbox',[.41 .6 .2 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'bj_time_domain_simulation.pdf')
```



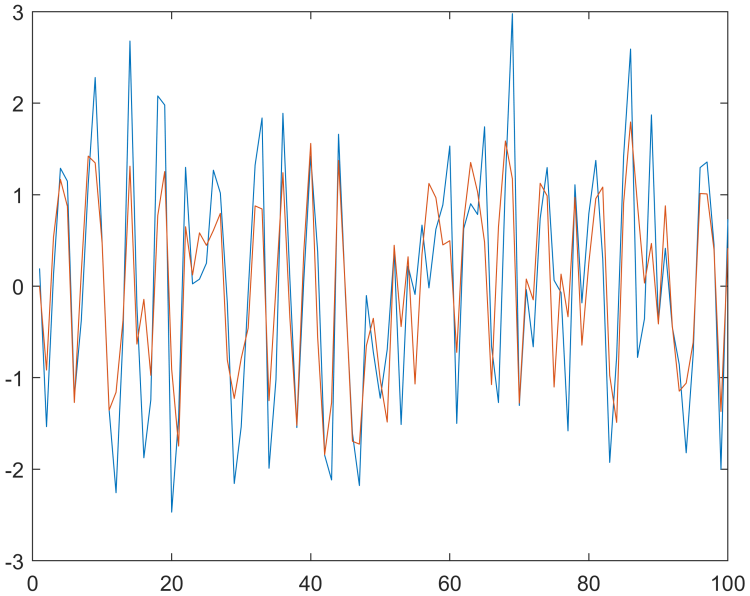

```
ysim = sim(ux,thoe);
figure(fig_num); fig_num = fig_num+1;
plot([yx ysim])
```




```
plot([yx(1:100) ysim(1:100)])
title('OE Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
```




```
figure(fig_num); fig_num = fig_num+1;
plot([yx(1:100) ysim(1:100)])
```



```
plot([yx(1:100) ysim(1:100)])
title('OE Time-Domain Simulation')
xlabel('Time [s]')
ylabel('Magnitude [-]')
str = strcat('MSE = ',num2str(mean((yx - ysim).^2),'%.3f'));
annotation('textbox',[.41 .6 .2 .3],'String',str,'FitBoxToText','on')
saveas(gcf,'oe_time_domain_simulation.pdf')
```