System Identification
CE-3: Parametric Identification Methods

Camilla Carta
Michael Spieler

December 19, 2017

1 Identification of a laser beam stabilizing system

For the first part of this exercise the objective was to identify a parametric model for a laser beam stabilizing system, as shown in Figure 1.

Figure 1: Laser beam stabilizing system

The datafile `laserbeamdataN.mat` contained the experimental data coming from the system:

- the input signal `u` which is a PRBS;
- the output signal `y`, which is the beam position;
- a sampling period of $T_e = 0.001s$.

In order to identify an optimal parametric model, different models were studied.

1.1 FIR model identification

First, a finite impulse response model was studied.
Let's assume that the output of the system depends only on past inputs:

$$y(k) = b_1 u(k-1) + b_2 u(k-2) + ... + b_m u(k-m) \tag{1}$$

Then, the z-tranform will give

$$Y(z) = (b_1 z^{-1} + b_2 z^{-2} + ... + b_m z^{-m})U(z) = G(z)U(z) \tag{2}$$

Now, the output of the FIR model can be predicted by

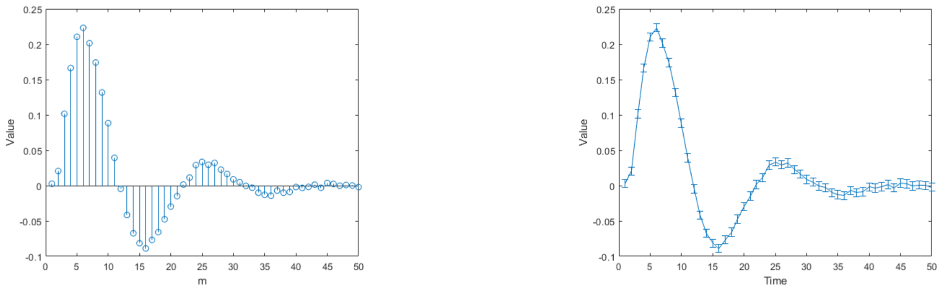
$$\hat{y}(k, \theta) = \phi^T(k)\theta \tag{3}$$

where

$$\phi^T(k) = [u(k-1), u(k-2), ..., u(k-m)] \tag{4}$$

$$\theta^T = [b_1, b_2, ..., b_m] \tag{5}$$

The code for the application of this method can be found in Appendix A. The vector of parameters $\theta^T = [b_1, ..., b_m]$ found can be observed in Figure 2a, while its covariance can be seen in Figure 2b. Finally, the predicted output can be compared to the real output (see Figure 3).



(a) Estimated values of parameters b (b) Covariance of the estimates with a $\pm 2\sigma$

Figure 2: Estimated values of theta and their covariance for FIR method with m=50

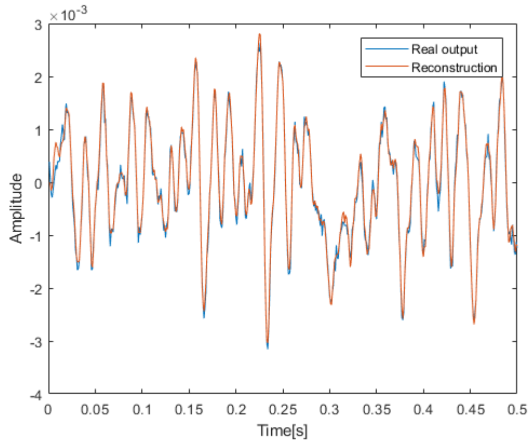


Figure 3: Real output against estimated FIR output

It can be observed that the estimation is quite satisfactory In order to evaluate the performance of the predictor, the loss function can be used, as explained in the course notes at page 109. As such, it can be calculated as $J(\hat{\theta})/N$, with

$$J(\hat{\theta}) = \sum_{k=1}^{N} (y(k) - \hat{y}(k)) \tag{6}$$

where $J(\hat{\theta})$ is the fit criterion. In this case, the loss function was equal to $1.3854 \cdot 10^{-8}$. Nevertheless, the number of parameters was limited at $m = 50$, while m should be theoretically infinite in order to perfectly represent the system.

1.2 ARX model identification

The second model that was studied was a second order ARX model, given by the following predictor:

$$\hat{y}(k, \theta) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) = \phi^T(k)\theta \tag{7}$$

As in the course notes (pp. 71-72), it can be demonstrated that the parameters can be identified using the least square algorithm, which gives

$$\hat{\theta} = \left[\sum_{k=1}^{N} \phi(k)\phi^T(k)\right]^{-1} \left[\sum_{k=1}^{N} \phi(k)y(k)\right] \tag{8}$$

with

$$\phi^T(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)] \tag{9}$$

and

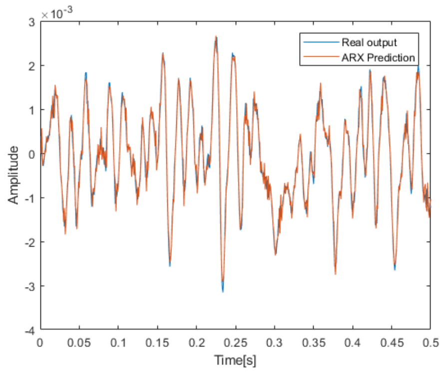
$$\theta^T = [a_1, a_2, b_1, b_2] \tag{10}$$

The code application can be found in Appendix refB.

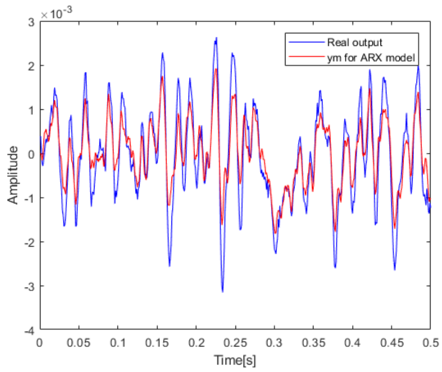
The estimated model was found to be

$$\hat{y}(k) = 1.5483y(k-1) - 0.6484y(k-2) + 0.0183u(k-1) + 0.0731u(k-2) \tag{11}$$

The results for the output prediction and the output for the identified model compared to the real output can be seen in Figure 4. In particular, it can be observed in Figure 4a that the prediction is close to the real output. The loss function, computed as above, was equal to $4.3396 \cdot 10^{-8}$, while the 2-norm of the output error was equal to $1.066782 \cdot 10^{-2}$ when compared to the output corresponding to this simulation (see Figure 4b). We notice that the simulated output behaves much worse than the predicted output.



(a) Real output against predicted ARX output



(b) Real output against the output for the ARX model

Figure 4: Real output compared to ARX prediction and modeled output

Instrumental Variable method

In order to have unbiased parameter estimated for the ARX model, the vector of instrumental variables can be used, as shown at page 74 for the course notes. As such, the new estimate will be

$$\hat{\theta_{iv}} = \left[\sum_{k=1}^{N} \phi(k)\phi^T(k)\right]^{-1} \left[\sum_{k=1}^{N} \phi_{iv}(k)y(k)\right] \tag{12}$$

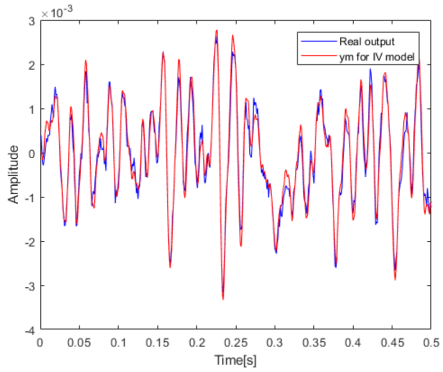
where $\phi(k)$ remains unchanged and $\phi_{iv}^T(k)$ is declared as

$$\phi_{iv}^T(k) = [-y_M(k-1), ... - y_M(k-n), u(k-d-1), ..., u(k-d-m)] \tag{13}$$

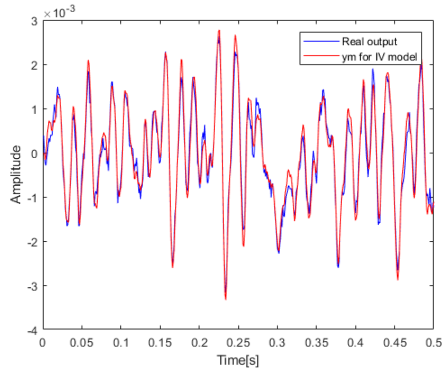
and the noiseless output of this model as

$$y_M(k) = M(q^{-1})u(k) \tag{14}$$

This new model gave a loss function of $2.2195 \cdot 10^{-7}$ and an output error of $4.696157 \cdot 10^{-3}$, which means this model has less precise prediction than the ARX model, but its simulated output is improved. The results can be seen in Figure 5.



(a) Real output against predicted IV output



(b) Real output against the output for the IV model

Figure 5: Real output compared to IV prediction and modeled output

1.3 State-space model identification

The general form of a state-space model is

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned} \tag{15}$$

Following the course notes at pages 78-80, an estimation of the parameters A, B and C can be computed. In particular,

$$Y_r(k) = \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+r-1) \end{bmatrix}, U_r = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+r-1) \end{bmatrix} \tag{16}$$

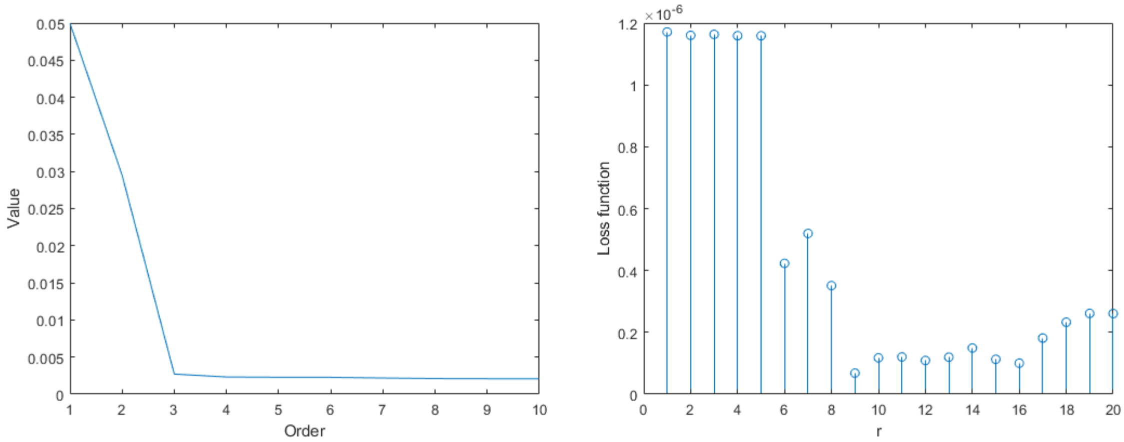
$$\mathbf{Y} = [Y_r(1), Y_r(2), ..., Y_r(N)] \tag{17}$$

$$\mathbf{U} = [U_r(1), U_r(2), ..., U_r(N)] \tag{18}$$

$$\mathbf{U}^{\perp} = \mathbf{I} - \mathbf{U}^T(\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U} \tag{19}$$

$$\mathbf{Q} = \mathbf{Y}\mathbf{U}^{\perp} = O_r\mathbf{X}\mathbf{U}^{\perp} \tag{20}$$

As stated in the course notes, the rank of the matrix **Q** should estimate the order of the system. However, in presence of noise, the rank can over-estimate the order: as such, one should compute the singular values and choose the most representative number. Indeed, the `svd` command was used to find the results shown in Figure 6a. It can be observed that the singular values tend towards 0 for all orders higher than 2. The maximum order n was then chosen equal to 2. The r parameter was then optimized by recursive iteration, that tested for values from 1 to 20. As can be seen in Figure 6b, the optimum loss function was found to be for $r = 9$.



(a) Singular values for the augmented observability matrix

(b) Loss function in function of r

Figure 6: Optimisation of n and r parameters

Subsequently, an estimation of the observability matrix O_r could be constructed by taking the first n columns and the first r rows of **Q**. Then, $\hat{C}$ can be estimated as the first n_y row of O_r and $\hat{A}$ can be computed from the equation

$$[\text{the last } (r-1)n_y \text{ rows of } O_r] = [\text{the first } (r-1)n_y \text{ rows of } O_r] \times \hat{A} \tag{21}$$

The identified parameters are

$$\hat{A} = \begin{bmatrix} 0.2655 & -1.0140 \\ 0.3648 & 1.3732 \end{bmatrix}, \hat{C} = 10^{-3} \cdot \begin{bmatrix} 0.3767 & -0.0184 \end{bmatrix} \tag{22}$$

From these parameters, the estimations for B could be computed as well, by using the least square algorithm

$$\hat{y}(k) = \hat{C}(qI - \hat{A})^{-1}Bu(k) + Du(k) = u_f(k)B \tag{23}$$

where

$$u_f(k) = \hat{C}(qI - \hat{A})^{-1}u(k) \text{ and } \hat{D} = 0 \tag{24}$$

$\hat{B}$ was then computed as

$$\hat{B} = (U_f^T U_f)U_f^T Y = \begin{bmatrix} 10.2157 \\ -286.7666 \end{bmatrix} \tag{25}$$

Finally, the measured output was compared to the state-space model found (see Figure 7). The 2-norm error was equal to $5.846650 \cdot 10^{-3}$.

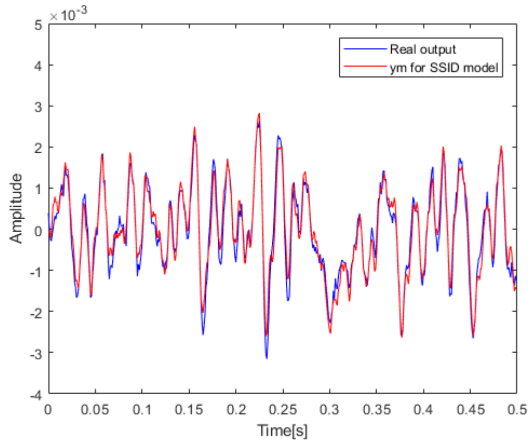


Figure 7: Measured output compared to state-space model output

2 Parametric Identification of a Flexible Link

For the second part the objective was to identify a parametric black-box model for a rotary flexible link, as shown in Figure 8.

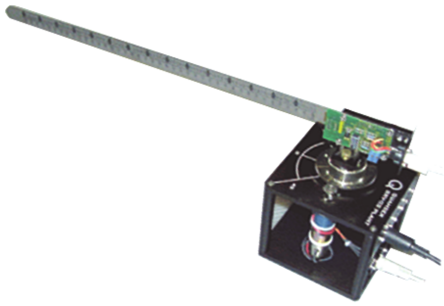


Figure 8: Flexible link system

We were provided with following data:

- the input signal `u` which is a PRBS of length N=2000 which consists of 4 periods.
- the output signal `y`, which is the link deflection measured by a strain gauge.
- a sampling period of $T_e = 0.015s$.

Before all, we normalize the data using the `detrend` MATLAB function to remove the DC component of the measurements.

2.1 Order estimation

The MATLAB code for the order estimation can be found in Appendix D.

2.1.1 Manual order estimation using ARX

Using the `arx` command 10 models of orders from 1 to 10 with $n_k = 1$ and $n_a = n_b = n = 1..10$ were identified and the loss function, which is shown in Figure 9, was calculated for each model. By hand we estimated an order of $n = 6$ (which corresponds to the manually chosen threshold of 0.002 represented by the horizontal red line in Figure 9). Beyond order 6 the loss function stops decreasing significantly.

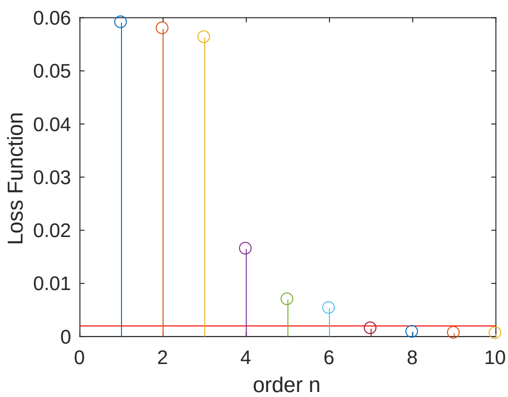


Figure 9: ARX order selection

2.1.2 Order estimation using ARMAX with zero/pole cancellation

The order can be estimated by identifying multiple ARMAX models and plotting the zeros/poles with their confidence intervals. When the confidence intervals of a zero/pole pair intersects, then it can be assumed that they cancel each other out. Thus zero/pole cancellation should occur above the true order n, which we observe for orders above $n = 7$ (shown in Figure 10). It is thus verified that this is very close to the order $n = 6$ estimated above. Therefore we choose the order $n = 7$.

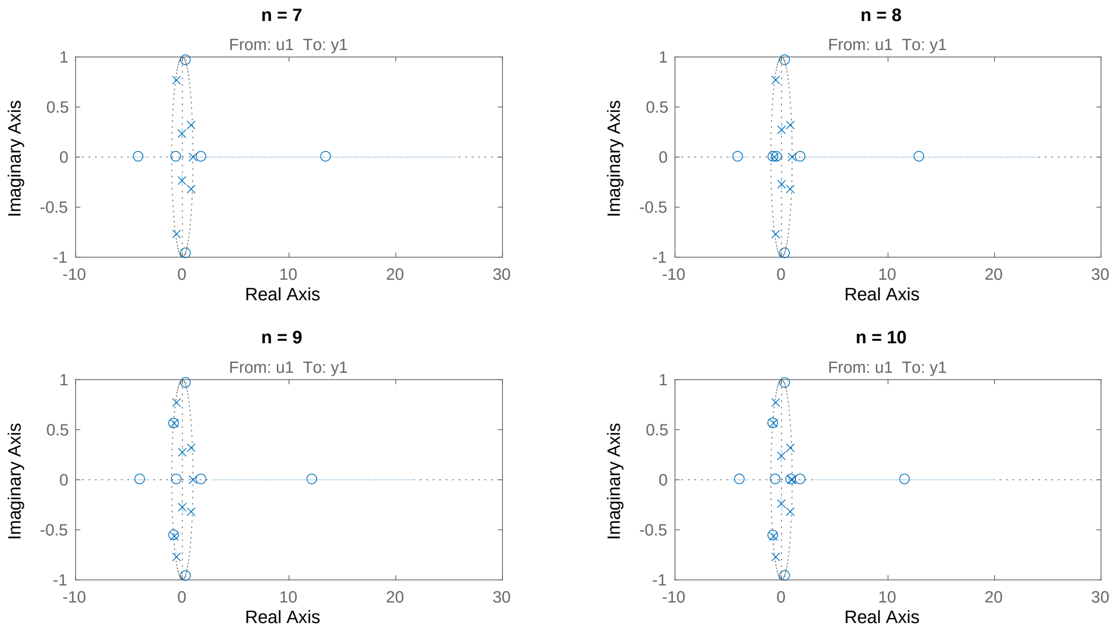


Figure 10: Additional zeros/poles cancel each other out for order $n > 7$.

2.1.3 Estimate the delay

To estimate the delay n_k we identify first an ARX system with $n_a = n_b = 7$ and $n_k = 0$. Then the parameters $b_1 to b_m$ of B were studied and the first n_k coefficients of B that are close to zero were determined. Due to noise, the coefficients are not exactly 0 and thus the confidence interval of the estimated coefficients b_i is studied, as shown in Figure 11. If the 0 lies in the confidence interval then the parameter b_i is considered to be close to zero. Thus we find $n_k = 2$.

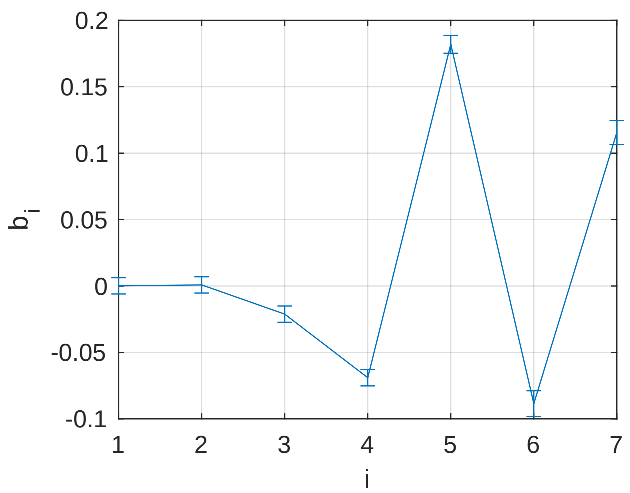


Figure 11: Order selection using ARX

2.1.4 Estimation using `selstruc` command

Finally, MATLAB's `selstruc` command was used to estimate the parameters n_a, n_b and n_k. Figure 12 shows the output. Akaike Information Criterion (AIC) estimate was discarded as an overestimated model order ($n_a = 9$, $n_b = 10$ and $n_k = 1$). Therefore we select by hand the number of parameters as 12, $n_a = 6$, $n_b = 6$ and $n_k = 3$. This result is close to the values obtained above.

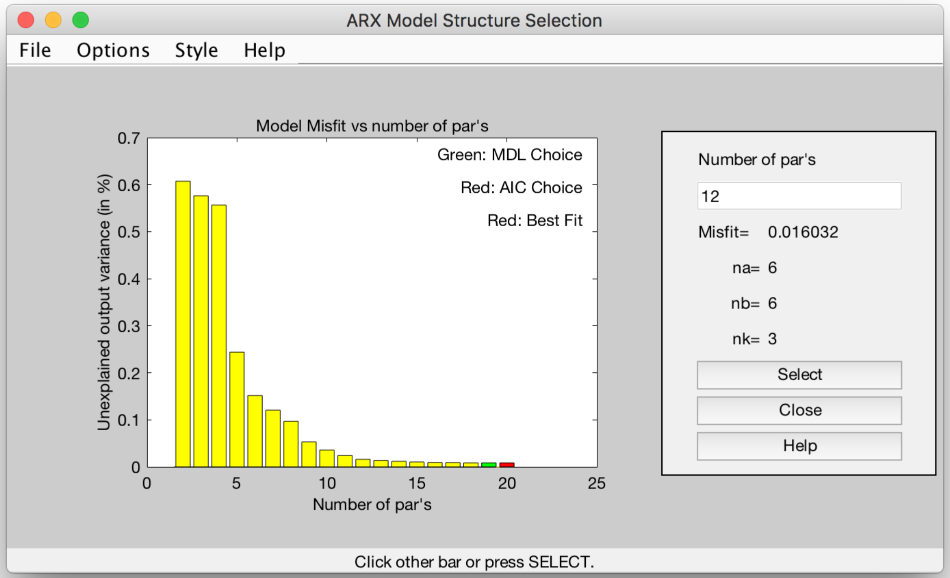


Figure 12: Order estimation using selstruc

2.2 Parametric identification and validation

We divided the data in two equal parts of length N=1000, one for identification and one for validation. We identify 6 different parametric models for our system using following MATLAB methods:

- `arx` for ARX structure
- `iv4` for Instrumental Variables method
- `armax` for ARMAX structure
- `oe` for Output-Error structure
- `bj` for Box-Jenkins structure
- `n4sid` for State-space structure

with following parameters:

- $n_a = n_b = 6$
- $n_k = 3$ delay
- $n_c = n_d = n_a$ for structures with noise model
- $n_f = n_a$ for Box-Jenkins structure
- $n_x = n_a$ for State-space structure

The MATLAB code for the parametric identification can be found in Appendix E.

The results are shown in Figure 13. We observe that the Box-Jenkins structure has the best fit (95.6%) followed by ARMAX (90.6%). This is not surprising, since they are the most complex structures with the highest number of parameters.

The models were validated by the whiteness test of the residuals and the cross-correlation of the residuals with the `resid` MATLAB command (see Figures 14 and 15). It can be observed that the best results are given by the Box-Jenkins and the ARMAX models, which validates their performance.

Although the ARMAX is slightly less performing than the Box-Jenkins, its validation is slightly better. Moreover, it is slightly less complex than the Box-Jenkins. For these reasons, it is the preferred model in this case.

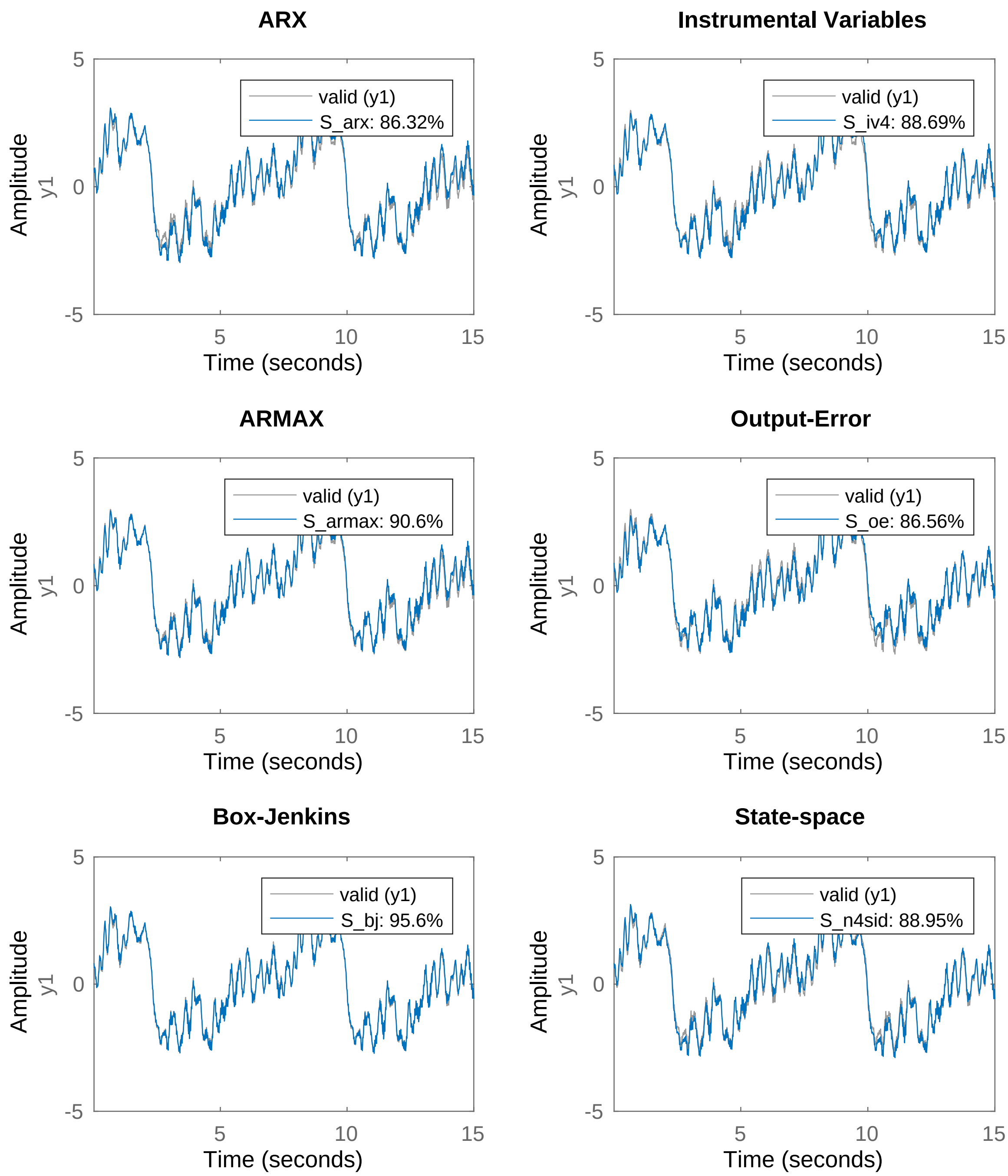


Figure 13: Comparison between the systems and the validation data: the plots display the normalized root mean square measure of the fit in percentage

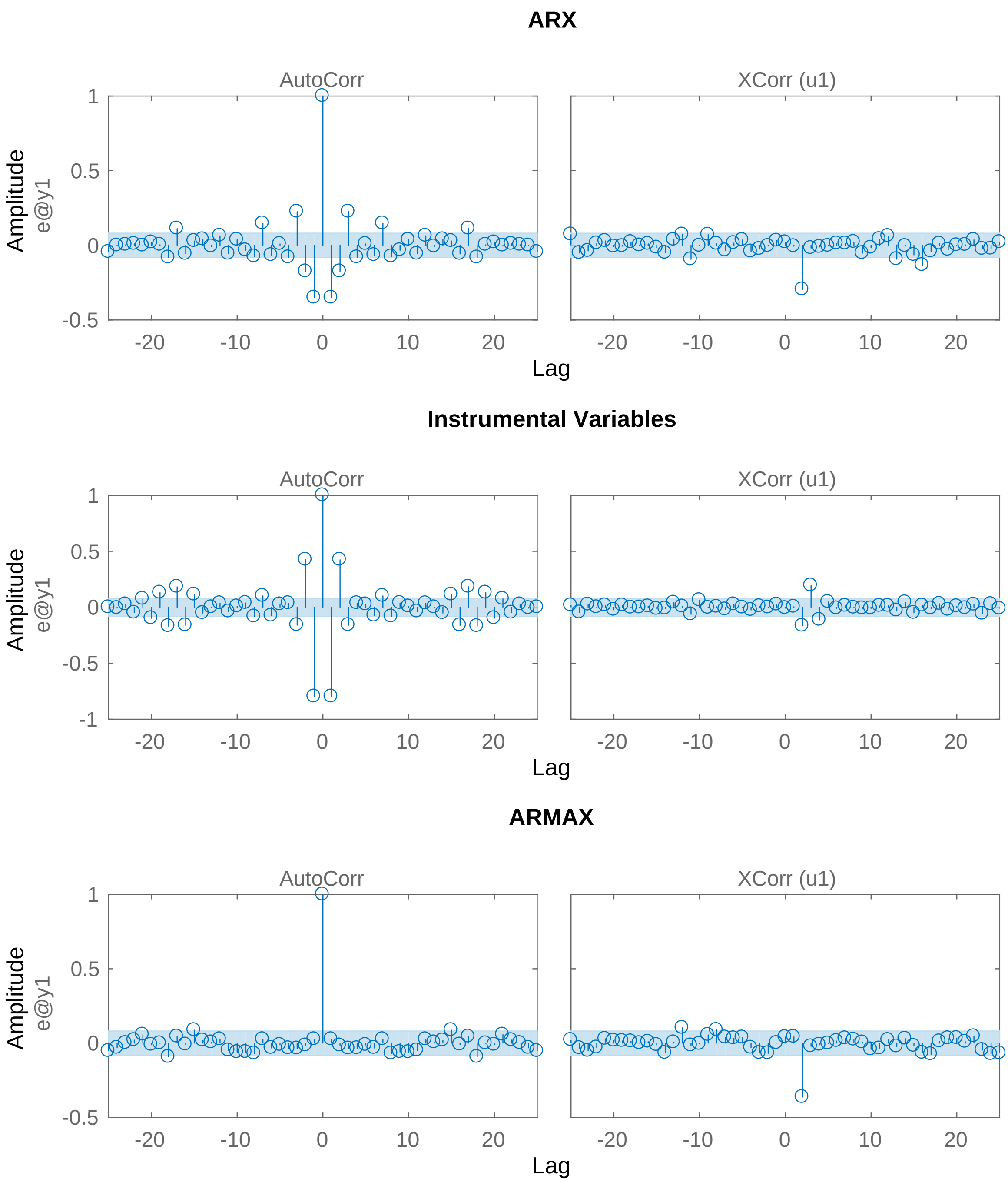


Figure 14: Auto-correlation of the residuals and their cross-correlation with the input signals for ARX, IV and ARMAX. The 99% confidence region marking statistically insignificant correlations is also shown as a patch around the plot X-axis

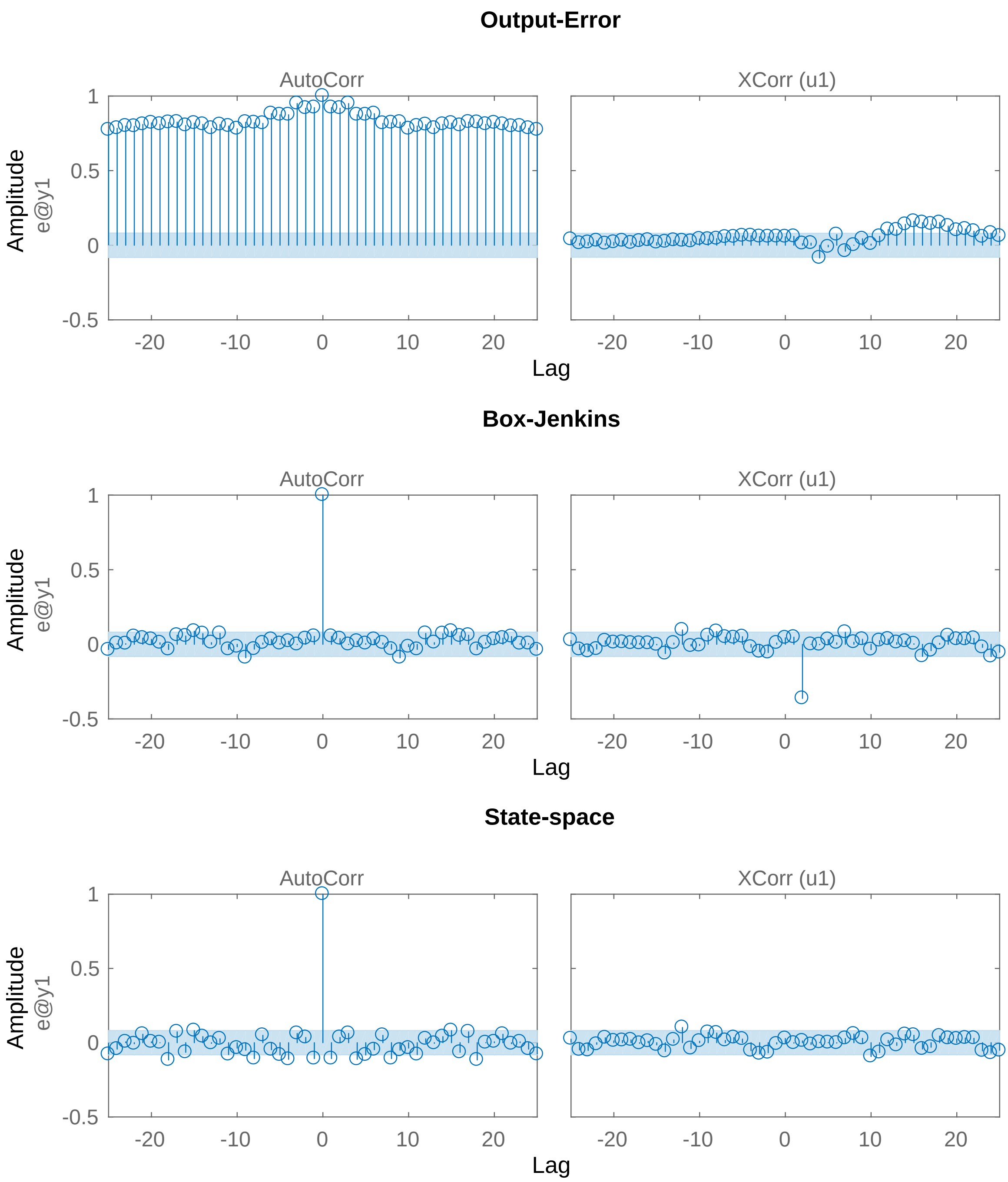


Figure 15: Auto-correlation of the residuals and their cross-correlation with the input signals for OE, BJ and SSID. The 99% confidence region marking statistically insignificant correlations is also shown as a patch around the plot X-axis

The Output-Error structure does not pass the whiteness test since $R_{\epsilon\epsilon}(h) \neq 0$ for $\forall h \neq 0$, which was not expected.

Appendices

A ce3_2.m

```
clear all
clc
close all

laserbeamdata = load('laserbeamdataN.mat');
y = laserbeamdata.y;
u = laserbeamdata.u;
Te = 1e-3; % sampling time
%%
N = length(y);
m = 50;
Phi = toeplitz(u, [u(1);zeros(N-1,1)]);

% truncate
Phi = Phi(:,1:m);

% least squares
% theta = pinv(Phi'*Phi)*Phi'*y;
theta = Phi\y
figure
stem(theta)
xlabel('m')
ylabel('Value')

%%
% reconstruct
yhat = Phi*theta;

x_ax = Te*[1:length(yhat)];
% compare
figure
plot(x_ax,y);
hold on
plot(x_ax,yhat);
hold off

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','Reconstruction')

% loss function
J = sum((y - yhat).^2)/N

%%

% covariance of parameters theta

var_hat = J/(N-m);

cov = var_hat*pinv(Phi'*Phi);
two_sigma = 2*sqrt(diag(cov));

figure
errorbar(theta, two_sigma)

xlabel('Time')
ylabel('Value')
```

B ce3_3.m

```
clear all
close all
clc

laserbeamdata = load('laserbeamdataN.mat');
y = laserbeamdata.y;
u = laserbeamdata.u;
Te = 1e-3; % sampling time

N = length(y);

% ARX model: yhat(k) = -a1*y(k-1) -a2*y(k-2) +b1*u(k-2) +b1*u(k-2)
% A(z) = 1 + a1*z^-1 + a2*z^-2
% B(z) = b1*z^-1 + b2*z^-2
% G(q^-1) = (q^-d * B(q^-1) / A(1^-1))

% n=2, m=2, d=0
% phi(k) = [-y(k-1), ..., -y(k-n), u(k-d-1), ..., u(k-d-m)]'
% Phi = [phi(1); phi(2); ..., phi(N)]
% theta = [a1 a2 b1 b2]'
n=2;
m=2;
d=0;

% least squares
Phi = toeplitz([0;-y(1:N-1)], [0; 0]);
Phi = [Phi, toeplitz([0;u(1:N-1)], [0; 0])];
% theta_hat = inv(Phi'*Phi)*Phi'*y;
theta_hat = Phi\y


%%

% reconstruct
yhat = Phi*theta_hat;

% compare
x_ax = Te*[1:length(yhat)];

figure
plot(x_ax,y);
hold on
plot(x_ax,yhat);
hold off

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','ARX Prediction')

% loss function
J_pred = sum((y - yhat).^2)/N



%% lsim
% G = z^-d*B(z^-1)/A(z^-1)
A = [1 theta_hat(1:n)'];
B = [theta_hat(n+1:end)'];
ARX_model = tf(B, A, Te);

yARX = lsim(ARX_model,u);

% compare
plot(x_ax,y,'b');
```

```
hold on
% plot(yhat,'r');
hold on
plot(x_ax,yARX,'r')

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','ym for ARX model')


norm2 = norm(yARX - y, 2);
fprintf('Norm-2 of the error: %E\n', norm2);
%% Instrumental Variable method
yM = yARX;

% least squares
Phi_iv = toeplitz([0;-yM(1:N-1)], [0; 0]);
Phi_iv = [Phi_iv, toeplitz([0;u(1:N-1)], [0; 0])];
%theta_hat_iv = Phi_iv\y;

theta_hat_iv = pinv(Phi_iv'*Phi)*Phi_iv'*y;

% reconstruct
yhat_iv = Phi_iv*theta_hat_iv;

x_ax = Te*[1:length(yhat)];

figure
plot(x_ax,y);
hold on
plot(x_ax,yhat_iv);
hold off

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','IV Prediction')

J_pred_iv = sum((y-yhat_iv).^2)/N

%% lsim
% G = z^-d*B(z^-1)/A(z^-1)
A_iv = [1 theta_hat_iv(1:n)'];
B_iv = [theta_hat_iv(n+1:end)'];
IV_model = tf(B_iv, A_iv, Te);

yIV = lsim(IV_model,u);

% compare
figure
plot(x_ax,y,'b');
hold on
% plot(yhat,'r');
hold on
plot(x_ax,yIV,'r')

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','ym for IV model')

norm2_iv = norm(yIV - y, 2);
fprintf('Norm-2 of the error: %E\n', norm2_iv);

%% compare
plot(yIV,'g');
hold off

```

```
legend('Real output','ARX output','IV output')
```

C ce3_4.m

```
clear all
close all
clc

laserbeamdata = load('laserbeamdataN.mat');

y = laserbeamdata.y;
u = laserbeamdata.u;
Te = 1e-3; % sampling time

%%
N = length(y);
for r = 1:9; %optimizes error

    Y = [];
    U = [];

    for k=1:N-r
        Y = [Y, y(k:k+r)];
        U = [U, u(k:k+r)];
    end


    %% U orthogonal
    U_orth = eye(N-r,N-r) - U'*pinv(U*U')*U;

    Q = Y*U_orth;

    thres = 0.01;
    n = sum(svd(Q) > thres)

    figure
    plot(svd(Q))
    xlabel('Order')
    ylabel('Value')

    Or = Q(1:r,1:n);

    C = Or(1,:);

    A = pinv(Or(1:end-1,:))*Or(2:end,:);

    %%

    z = tf('z',Te);

    F = C*inv((z*eye(size(A)))-A);

    U_f = [];

    for k = 1:size(F,2)
        U_f = [U_f lsim(F(k),u)];
    end

    %%
    B = pinv(U_f'*U_f)*U_f'*y;

    yhat = U_f*B;

    %loss function
```

```
    J(r) = sum((y-yhat).^2)/N

    %%
    figure
    plot(y), hold on
    plot(yhat)

    legend('real','approx')

end

figure
stem(J)

xlabel('r')
ylabel('Loss function')

%% Model
t = 0:Te:Te*(length(y)-1);
[tf_a, tf_b] = ss2tf(A, B, C, 0);
sys_ss = tf(tf_a, tf_b, Te, 'variable', 'z');
y_hat = lsim(sys_ss,u,t);

norm2 = norm(y_hat-y);
fprintf('Norm-2 of the error: %E\n', norm2);

% compare
figure
plot(t,y,'b');
hold on
% plot(yhat,'r');
hold on
plot(t,y_hat,'r')

axis([0 0.5 -0.004 0.005])

xlabel('Time[s]')
ylabel('Amplitude')
legend('Real output','ym for SSID model')
```

D ce3_5.m

```
clear all
close all
clc

flexibleData = load('CE.mat');

%% Initialisation
u = flexibleData.u;
y = flexibleData.y;

[N,M] = size(y);

Te = 0.015;

%% Create Data Object

data = iddata(y,u,Te);

[data_d,T] = detrend(data);

%plot(data,data_d)

```

```
%% Order estimation with ARX
nc = 0; nd = 0; nf = 0; nk = 1;
thres = 0.002;
order_arx = 0;

figure
plot([0 10], [thres, thres], 'r')
ylabel('Loss Function')
xlabel('order n')
hold on
for n =1:10
    orders = [n n nk];
    SYS = arx(data_d, orders);
    stem(n,SYS.Report.Fit.LossFcn), hold on

    if(SYS.Report.Fit.LossFcn > thres)
        order_arx = order_arx + 1;
    end
end
order_arx

figure
na = order_arx; nb = order_arx;
SYS = arx(data_d, [na nb 1]);
errorbar(SYS.b, SYS.db*2)

%% Validation with ARMAX
nc = 0; nd = 0; nf = 0; nk = 1;
order_armax = 0;
figure
for n =7:10
    orders = [n n n nk];
    SYS = armax(data_d, orders);
    %stem(n,SYS.Report.Fit.LossFcn), hold on

    if(SYS.Report.Fit.LossFcn > thres)
        order_armax = order_armax + 1;
    end
    subplot(2,2,n-6)
    h = iopzplot(SYS);
    title('n = '+string(n))
    showConfidence(h,2)
end

%% Find order from zero/pole plot
order_armax = 7
%% estimate delay nk
na = order_armax; nb = order_armax; nc = order_armax;
SYS = arx(data_d, [na nb 0]);
lower = SYS.b - 2*SYS.db;
upper = SYS.b + 2*SYS.db;
test = lower.*upper <= 0; % test if 0 within 2 sigma
nk = 0;
for i = test
    if i == 0
        break
    else
        nk = nk+1;
    end
end
nk
figure
errorbar(SYS.b, SYS.db*2)
grid on
ylabel('b_i')
```

```
xlabel('i')

%% Plot Zero/Pole and their confidence interval
figure
h = iopzplot(SYS)
showConfidence(h,2)

%% Divide data
N1 = N/2;

data = iddata(y(1:N1),u(1:N1),Te);
valid = iddata(y(N1+1:end),u(N1+1:end),Te);

%% Compare

NN = struc(1:10,1:10,0:10);
V=arxstruc(data,valid,NN);
[NN, Vm] = selstruc(V, 'PLOT');
NN
```

E ce3_5_2.m

```
clear all
close all
clc

flexibleData = load('CE.mat');

%% Initialisation
u = flexibleData.u;
y = flexibleData.y;

[N,M] = size(y);

Te = 0.015;

%% Divide data
N = N/2;

data = iddata(y(1:N),u(1:N),Te);
valid = iddata(y(N+1:end),u(N+1:end),Te);

data = detrend(data);
valid = detrend(valid);

%%
na = 6;
nb = 6;
nk = 3;
nc = na;
nd = na;
nf = na;
nx = max(nb + nk, na);

figure
S_arx = arx(data, [na nb nk]);
subplot(3,2,1)
compare(valid, S_arx)
title('ARX')

S_iv4 = iv4(data, [na nb nk]);
subplot(3,2,2)
compare(valid, S_iv4)
title('Instrumental Variables')
```

```

S_armax = armax(data, [na nb nc nk]);
subplot(3,2,3)
compare(valid, S_armax)
title('ARMAX')

S_oe = oe(data, [nb nf nk]);
subplot(3,2,4)
compare(valid, S_oe)
title('Output-Error')

S_bj = bj(data, [nb nc nd nf nk]);
subplot(3,2,5)
compare(valid, S_bj)
title('Box-Jenkins')

S_n4sid = n4sid(data, nx);
subplot(3,2,6)
compare(valid, S_n4sid)
title('State-space')


printpdf(gcf, 'ce3_5_2_system_compare.pdf', 1, 1.5)

%%

figure
S_arx = arx(data, [na nb nk]);
subplot(3,1,1)
resid(valid, S_arx)
title('ARX')

S_iv4 = iv4(data, [na nb nk]);
subplot(3,1,2)
resid(valid, S_iv4)
title('Instrumental Variables')

S_armax = armax(data, [na nb nc nk]);
subplot(3,1,3)
resid(valid, S_armax)
title('ARMAX')
printpdf(gcf, 'ce3_5_2_system_resid1.pdf', 1, 1.5)


figure
S_oe = oe(data, [nb nf nk]);
subplot(3,1,1)
resid(valid, S_oe)
title('Output-Error')

S_bj = bj(data, [nb nc nd nf nk]);
subplot(3,1,2)
resid(valid, S_bj)
title('Box-Jenkins')

S_n4sid = n4sid(data, nx);
subplot(3,1,3)
resid(valid, S_n4sid)
title('State-space')

printpdf(gcf, 'ce3_5_2_system_resid2.pdf', 1, 1.5)
```