ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CE1 Report

System identification

Lebailly Tim
Thammisetty Devakumar

Professor : Karimi Alireza

Tuesday, 30 October 2018

Contents

List of Figures

1 Step and Impulse response

Simulink model is generated with inputs from workspace, includes a saturation block and random number added to the model output and the output is sent to workspace (simout). Figure 1 shows the schematic of simulink model. A sampling time of $0.5s$ is reasonable because the poles of the system are all located at frequencies lower than 1Hz. There is nothing to be identified at higher frequencies (modulus: -80db / decade & phase: constant 0°).

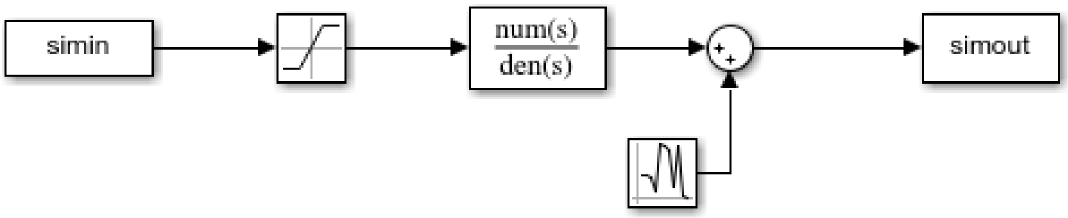

Figure 1: Simulink model schematic

Step response

Inputs for the simulink model are generated with step at 1s with amplitude 0.5

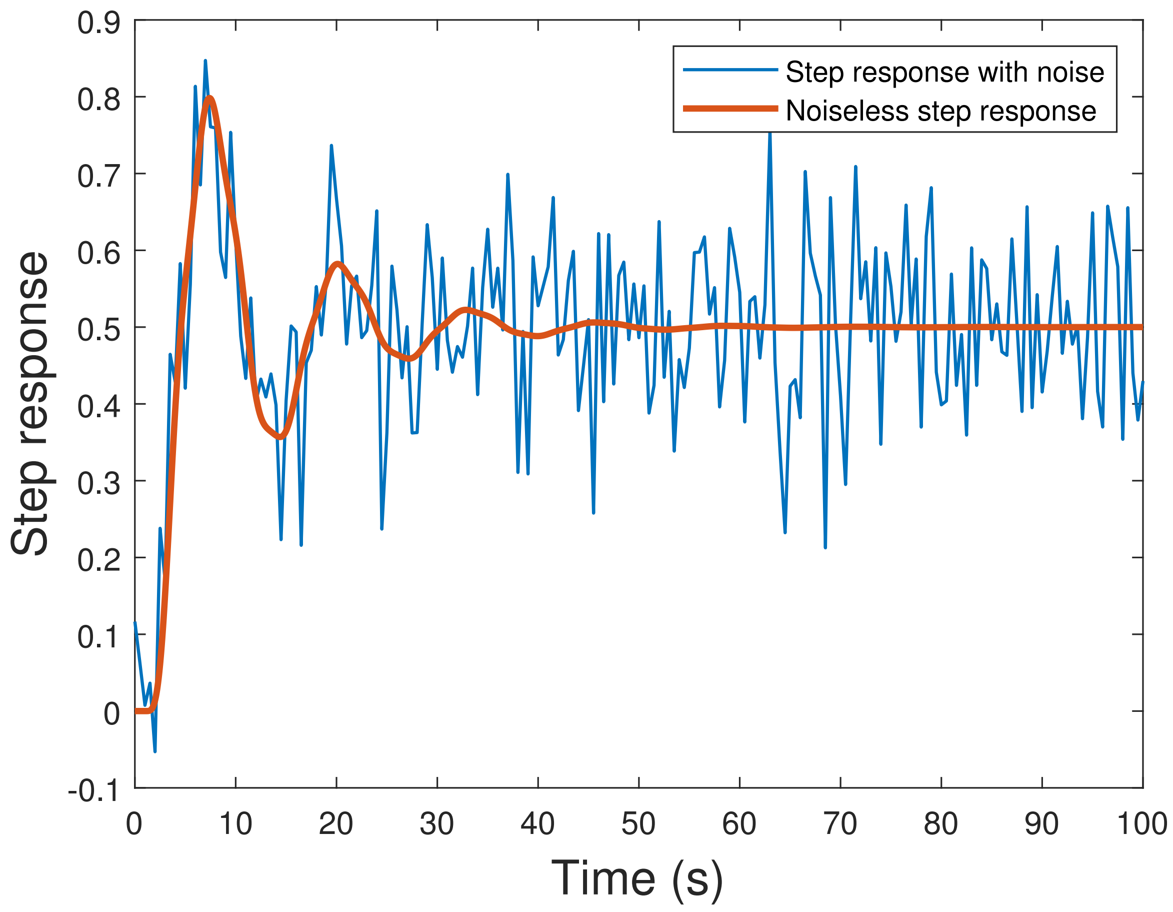

Figure 2: Step response of the given fourth order system with and without noise

Impulse response

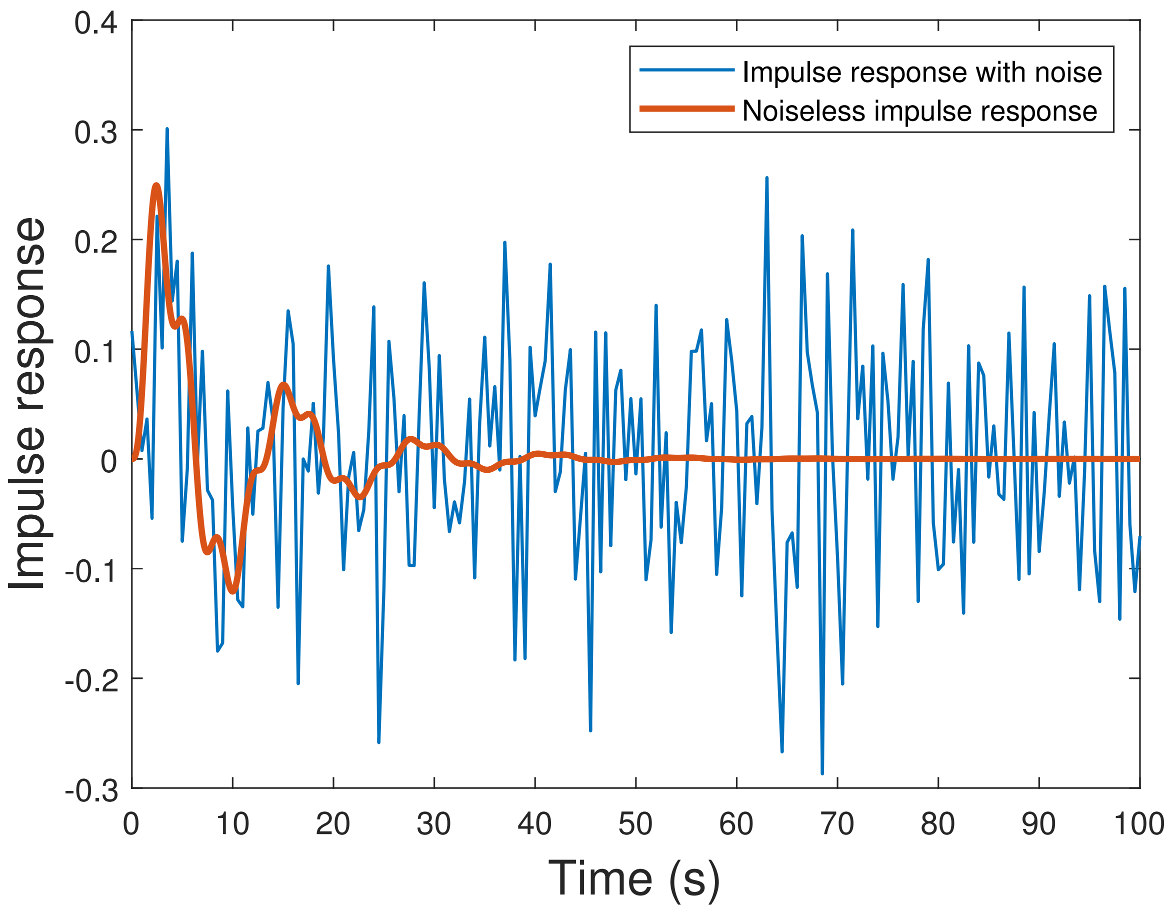

Figure 3: Impulse response of the given fourth order system with and without noise

MATLAB code used for Exercise-1

```matlab
clc
clear all

Te = 0.5;

%% Step response input generation and simulation using simulink plant
simin.time = 0:0.05:100;
idx = find(simin.time == 1);
values = 0.5*double((simin.time >= 1));
simin.signals.values = values';
plot(simin.time, values, 'LineWidth', 2); ylim([0, 1]); xlim([0, 5]);

sim('CE11', simin.time);

% Noiseless step response of the system
sys1 = tf([1], [1, 0.4, 4.3, 0.85, 1])
opt = stepDataOptions('StepAmplitude',0.5);
[y, t, x] = step(sys1, opt, simin.time);
y = [zeros(1, idx) y(1:end-idx)']';

% Plot step response with and without noise and save the plot
figure(); hold on; box on
```

```
plot(simout, 'LineWidth', 1)
plot(simin.time, y, 'LineWidth', 2)
xlabel('Time (s)','FontSize',15)
yl = ylabel('Step response','FontSize',15);
title('Step response with and without noise');
legend('Step response with noise', 'Noiseless step response')
pos=get(gca,'Position');
set(gca,'Position',pos + [0.02, 0.04, -0.02, -0.04]);
saveas(gcf, 'ce11_step','epsc')

%% Impulse response input and simulation using Simulink
% Impulse is given at time = 1s
simin.time = 0:0.05:100;
idx = find(simin.time == 1);
values = double((simin.time < -100));
values(idx) = 1;
simin.signals.values = values';
sim('CE11', simin.time);

%% Actual impulse response without noise
sys1 = tf([1], [1, 0.4, 4.3, 0.85, 1]);
[y_imp, t, x] = impulse(sys1, simin.time);
simtime = [zeros(1, idx) simin.time(1:end-idx)]';

%% Plot the impulse response with and without noise
% plot(simin.time, values)
figure();hold on; box on
plot(simout, 'LineWidth', 1)
plot(simin.time, y_imp*Te, 'LineWidth', 2)
xlabel('Time (s)','FontSize',15)
yl = ylabel('Impulse response','FontSize',15);
title('Impulse response with and without noise');
legend('Impulse response with noise', 'Noiseless impulse response')
pos=get(gca,'Position');
set(gca,'Position',pos + [0.02, 0.04, -0.02, -0.04]);
saveas(gcf, 'ce11_impulse','epsc')
```

2 Auto Correlation of a PRBS signal

The autocorrelation function is implemented with a circular shift in order to compute the true autocorrelation of a periodic signal of infinite length. The results can be compared with the MATLAB function $xcorr()$ and the results are the same (in the middle of the output) when $intcorr()$ is given 1 period of a signal, and when $xcorr()$ is given multiple periods of the same signal.

```matlab
% Function that computes the correlation of 2 different signals
% Input:    u: signal 1
%           y: signal 2
% Output:   R: value of correlation function
%           h: index of correlation function
function [R, h] = intcor(u, y)
    R = zeros(2 * length(y), 1);
    for i = 1 : 2 * length(y) + 1
        R(i) = u' * y / length(y);
        y = circshift(y, 1);
    end
    h = (0:2 * length(y))';
end
```

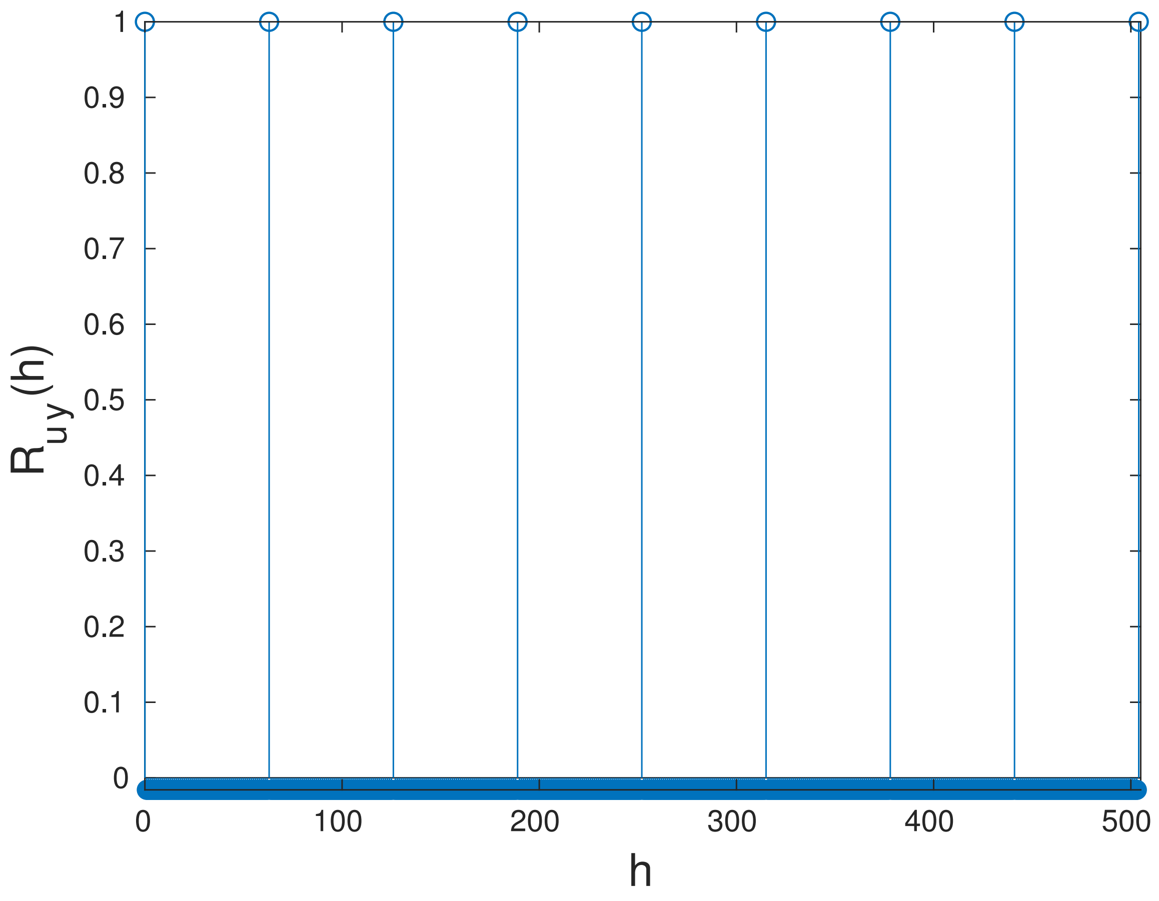

Figure 4: Autocorrelation of PRBS with 6 bit register

3 Impulse response by deconvolution method

The impulse response is computed using the deconvolution method. This consists of solving the following linear system for Θ where Y is the measured output and U is the matrix of shifted inputs.

$$Y = U\Theta \tag{1}$$

Because we know that the impulse reponse Θ only takes non-zero values for t smaller that 50 seconds (based on previous plots), we can truncate the matrix U for the first 100 columns. This makes the linear system of equations better conditioned and we solve it using least-squares.

```
% Input definition
N = 400;
Te = 0.5;
t = 0:Te:(N-1)*Te;
input = rand(length(t), 1) - 0.5;

% Simulink run
var.time = t;
var.signals.values = input;
sim("exo1_simu", t);

% Least squares
U = tril(toeplitz(input));
U = U(:,1 : 50/Te);
g = U \ simout.Data;
```

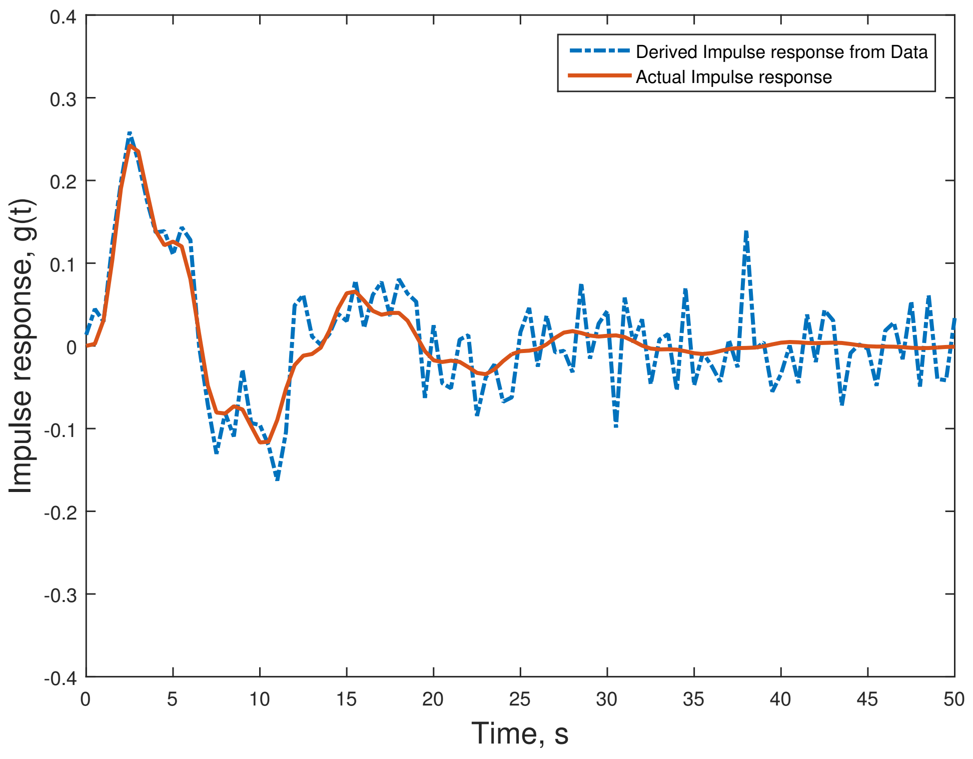

Figure 5: Impulse response comparison

The 2-norm of the error between actual impulse response and identified response is 0.3966 wherein 101 samples are used in the estimation (50s).

4 Impulse response by correlation approach

The impulse response is estimated using correlation approach (intcorr) and by using matlab function xcorr. In both the cases the input used is PRBS signal with p=4 and n=7. Further, the true impulse response of the discretized (c2d) system is computed and the results are compared. The system we are solving is the following:

$$R_{yu} = R_{uu}\Theta \tag{2}$$

Again, for numerical stability, only the first 100 columns of R_{uu} are taken into account and we solve the linear system using least-squares.

Plot of identified impulse response vs true response

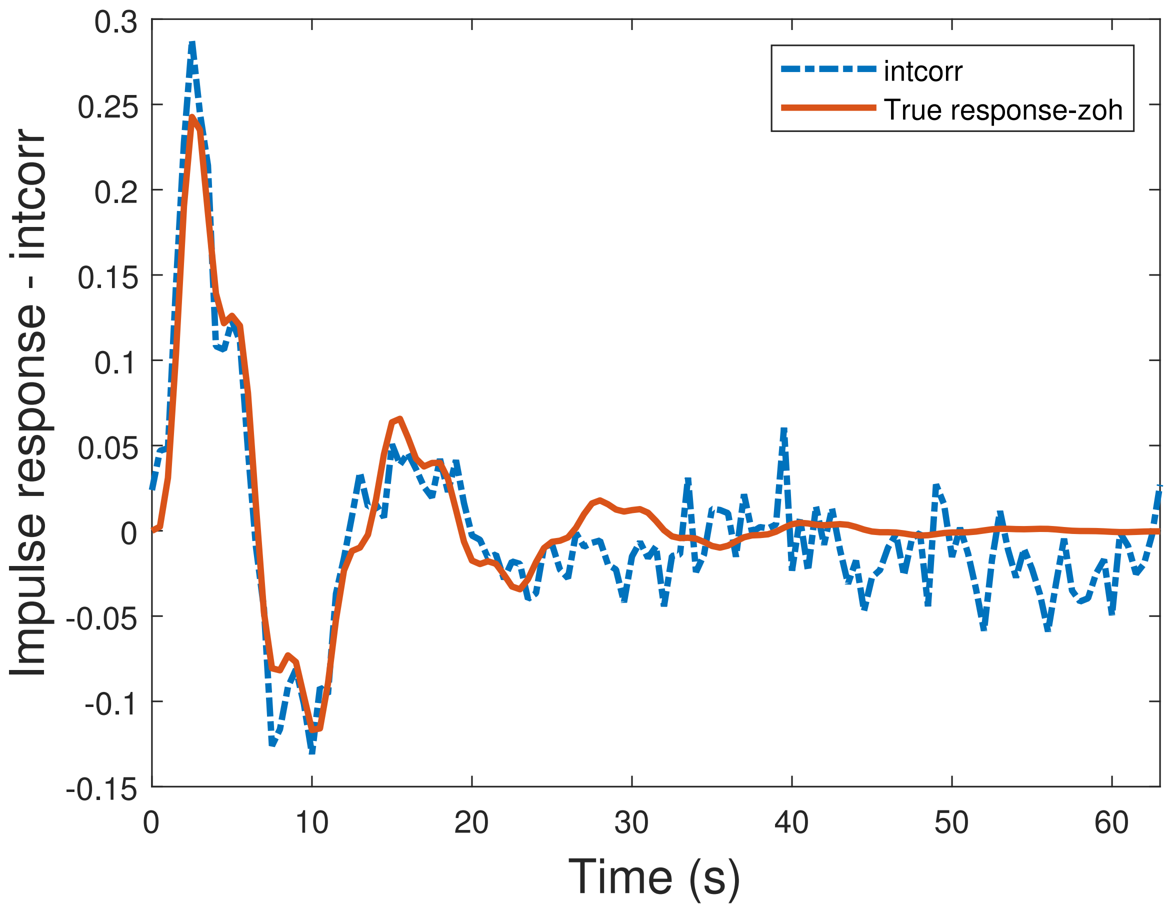

Figure 6: Impulse response using intcorr- comparison with true one

2-norm of the error = **0.2781** for the impulse response estimated using intcorr.

Impulse response with MATLAB xcorr vs true response

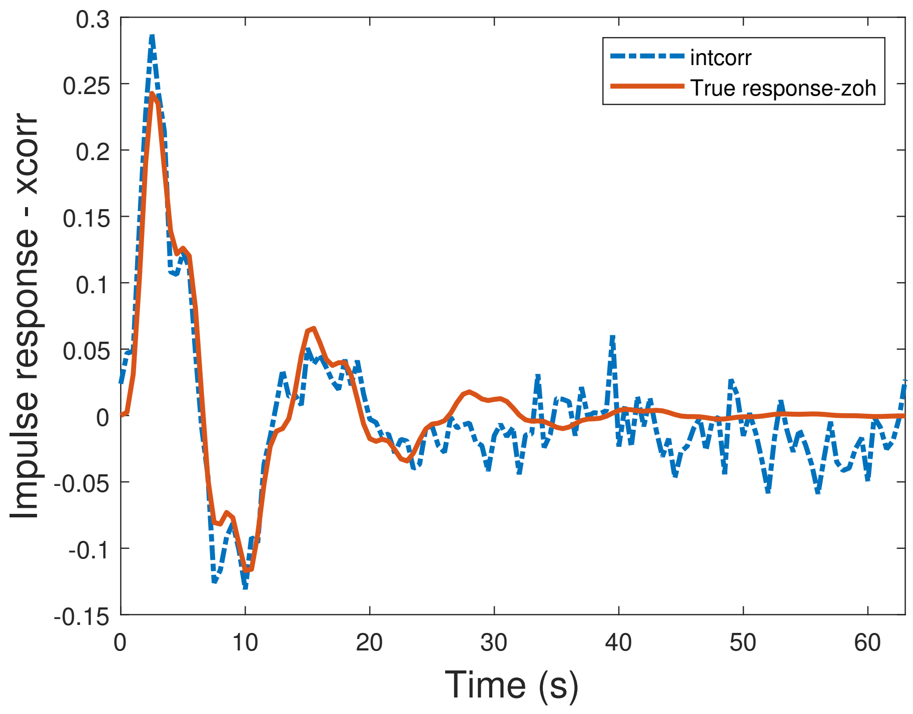

Figure 7: Impulse response using xcorr- comparison with true one

Impulse response estimated using intcorr and xcorr are compared below.
We observed that MATLAB xcorr without additional options (biased, unbiased) adds leading zeros while shifting the input signal before multiplication. However, for periodic signals, this result will not be comparable to intcorr results.

To resolve this issue, we divided the output into 4 periods, used only one period output for the impulse response computation along with entire input (Uprbs) while computing autocorrelation using xcorr. This way, even if the input is shifted, the values would be same as in intcorr biased version. We observe that this method of computing autocorrelation using xcorr is correct for the periodic signals. The results of impulse response derived using intcorr and xcorr with aforementioned method is given below, we observe that both the results match as expected.

However, for the non-periodic signals xcorr can be used as such, without any modifications along with biased and unbiased options. **2-norm of the error** = **0.2781** for the impulse response estimated using xcorr.

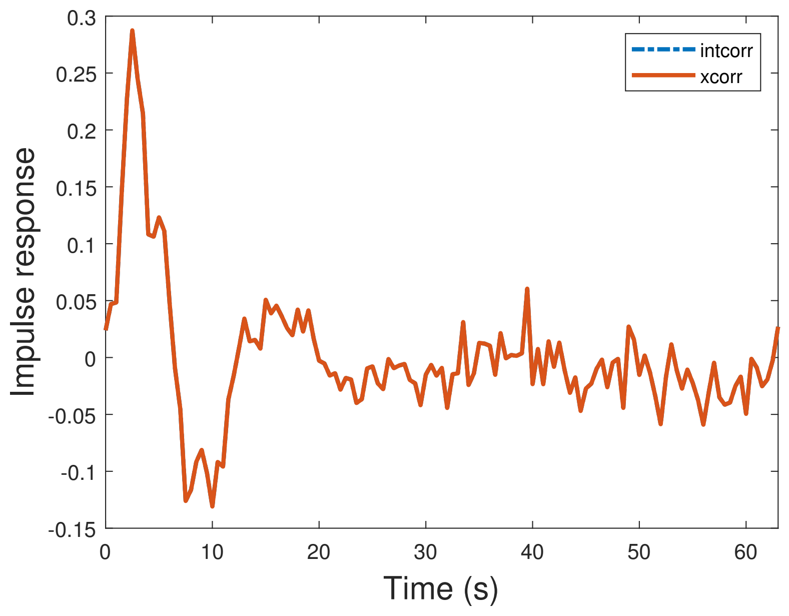

Figure 8: Impulse response comparison using xcorr and intcorr function - Biased estimates

MATLAB code used for estimation impulse response

```
%% CE 1.4

%% Estimation using correlation approach
% Number of samples per period
N1 = N/Nperiods;

% Select PRBS period for the comuting the impulse response
inp_period = 1;

% Select signals for estimating intcorr
u1 = Uprbs((inp_period -1)*N1+1:inp_period*N1);
y1 = y((inp_period -1)*N1+1:inp_period*N1);

% Compute impulse response using corelation approach
Ryu1 = intcor_yu(y1, u1);
Ruu1 = intcor_yu(u1, u1);
g_intcorr = Ryu1(1:N1)/Ruu1(1);

%% Compute impulse response using xcorr - MATLAB
[Ryu1_x, lagRyu] = xcorr(y1, Uprbs);
[Ruu1_x, lagRuu] = xcorr(u1, Uprbs);
g_xcorr = Ryu1_x(N1:2*N1-1)/Ruu1_x(N1);
```

5 Frequency domain Identification (Periodic signal)

This exercice is comparable to section 3 except that the problem is solved in the frequency domain. Instead of solving equation 1 (which computes an inverse convolution), we take the fourier transform of equation 1 which turns the inverse convolution into a trivial problem:

$$G(e^{j\omega}) = \frac{Y(e^{j\omega})}{U(e^{j\omega})} \tag{3}$$

The solution of equation 4 is the transfer function of the system, also known as the fourier transform of the impulse response.

We simulate our system using a PRBS of 8 bits and 8 periods with gives our input a length of 2040 samples or 1020 seconds. From previous exercises, we know that the impulse response of the system is about 0 after 50 seconds so the system should be in steady state after the first period of the PRBS (127 s). We then take the average of the output over the 7 left over periods.

Our computed bode plot and the real one look similar for low frequencies. This has to do with the fact that our simulink noise adds white noise to every sample at 2Hz. This means that there is no noise for low frequency values. We can thus identify the system very wel at low frequency (SNR is high) but not at higher frequencies (SNR is low).

```matlab
% Definition of variables
bits = 8;
periods = 8;
periods_drop = 1;
u = prbs(bits, periods)/2;
N = length(u);
Te = 0.5;
t = 0:Te:(length(u)-1)*Te;

% Simulation with simulink
var.time = t;
var.signals.values = u;
sim("exo1_simu", t);
y = simout.Data;

% Averaging over multiple periods
M = 2^bits - 1;
yfftMatrix = zeros(M, periods - periods_drop);
ufftMatrix = zeros(M, periods - periods_drop);
for i = 1 : periods - periods_drop
    yfftMatrix(:, i) = fft(y(1 + (i-1) * M : M * i));
    ufftMatrix(:, i) = fft(u(1 + (i-1) * M : M * i));
end

% Fourier transform
Y = mean(yfftMatrix, 2)/M;
U = mean(ufftMatrix, 2)/M;
H = Y./U;
H_positive = H(1:round(M/2)-1);
Y_positive = Y(1:round(M/2)-1);
freqHz = (0:1:M-1)' / (M*Te); % Hz
freqW_postive = 2*pi*freqHz(1:round(M/2)-1); % omega

% System definition
sysReal = tf([1], [1, 0.4, 4.3, 0.85, 1]);
sysReal_d = c2d(sysReal, Te, 'zoh');
sys_computed = frd(H_positive, freqW_postive, Te);
```

```
% Plots
bode(sys_computed); hold on
bode(sysReal_d)
axes_handles = findall(gcf, 'type', 'axes');
legend(axes_handles(3), 'Computed', 'Real','Location', 'NorthWest')
legend(axes_handles(2), 'Computed', 'Real','Location', 'NorthWest')
xlim([0.05  2*3.1])
set(findall(gcf,'type','line'),'linewidth', 2)
```

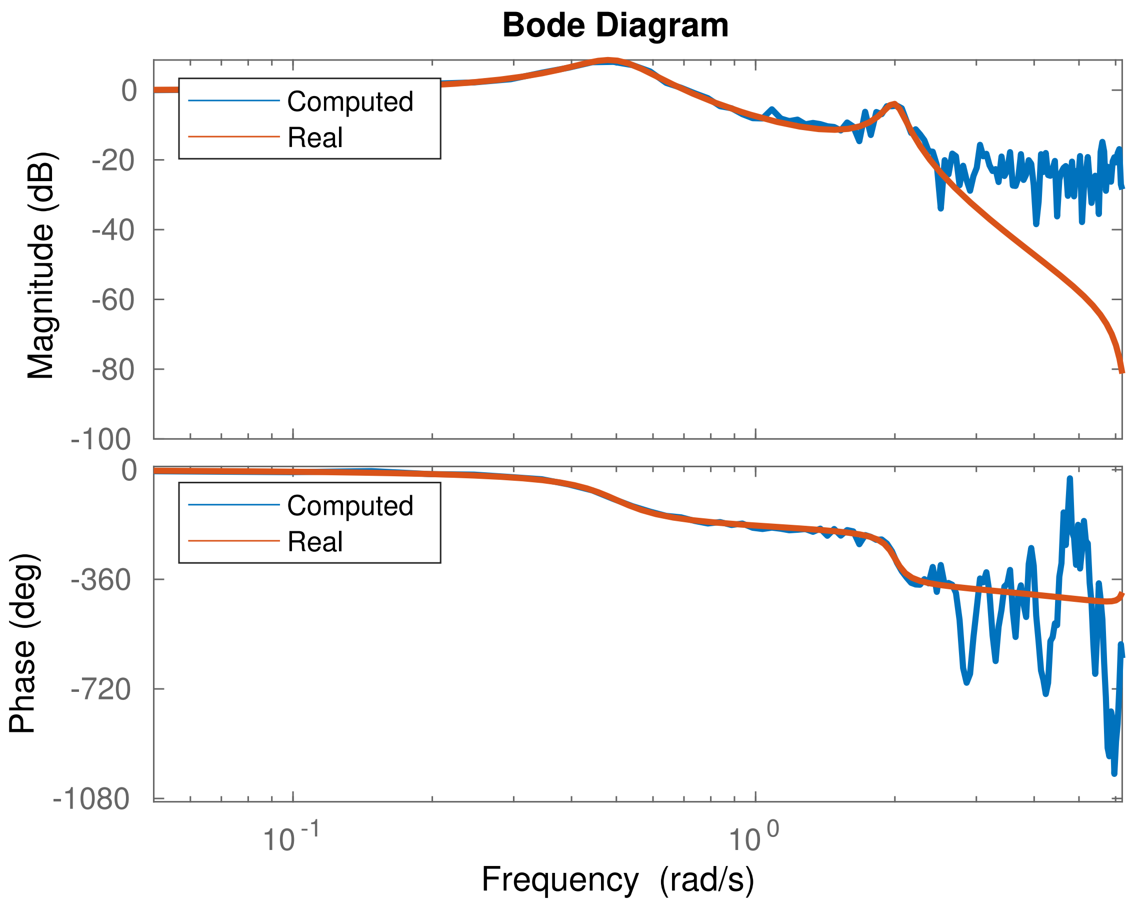

Figure 9: Bode plot of identified system vs real system

6 Frequency domain Identification (Random signal)

This exercice is comparable to section 4 except that the problem is solved in the frequency domain. Instead of solving equation 2 (which computes an inverse convolution), we take the fourier transform of equation 2 which turns the inverse convolution into a trivial problem:

$$G(e^{j\omega}) = \frac{\Phi_{yu}(\omega)}{\Phi_{uu}(\omega)} \tag{4}$$

```matlab
% Variable definition
u = rand(2000, 1) - 0.5; % uniform noise
N = length(u);
Te = 0.5;
t = 0:Te:(length(u)-1)*Te;

% Simulation with simulink
var.time = t;
var.signals.values = u;
sim("exo1_simu", t);
y = simout.Data;

% Averaging over multiple periods
periods = 10;
M = N / periods;
PI_yu_matrix = zeros(M, periods);
PI_uu_matrix = zeros(M, periods);
for i = 1 : periods
    up = u((1 + (i-1) * M : M * i));
    yp = y((1 + (i-1) * M : M * i));
    R_uu = xcorr(up, up);
    R_uu = R_uu(M:end);
    R_yu = xcorr(yp, up);
    ham = hamming(length(R_yu));
    R_yu = ham .* R_yu;
    R_yu = R_yu(M:end);
    PI_yu_matrix(:, i) = fft(R_yu);
    PI_uu_matrix(:, i) = fft(R_uu);
end

% Fourier transform
PI_yu = mean(PI_yu_matrix, 2);
PI_uu = mean(PI_uu_matrix, 2);
H = PI_yu./PI_uu;
H_positive = H(1:round(M/2)-1);
freqHz = (0:1:M-1)' / (M*Te); % Hz
freqW_postive = 2*pi*freqHz(1:round(M/2)-1); % omega

% System definition
sysReal = tf([1], [1, 0.4, 4.3, 0.85, 1]);
sysReal_d = c2d(sysReal, Te, 'zoh');
sys_computed = frd(H_positive, freqW_postive, Te);

% Plots
bode(sys_computed); hold on
bode(sysReal_d)
axes_handles = findall(gcf, 'type', 'axes');
legend(axes_handles(3), 'Computed', 'Real','Location', 'NorthWest')
legend(axes_handles(2), 'Computed', 'Real','Location', 'NorthWest')
xlim([0.05 2*3.1])
set(findall(gcf,'type','line'),'linewidth', 2)
```

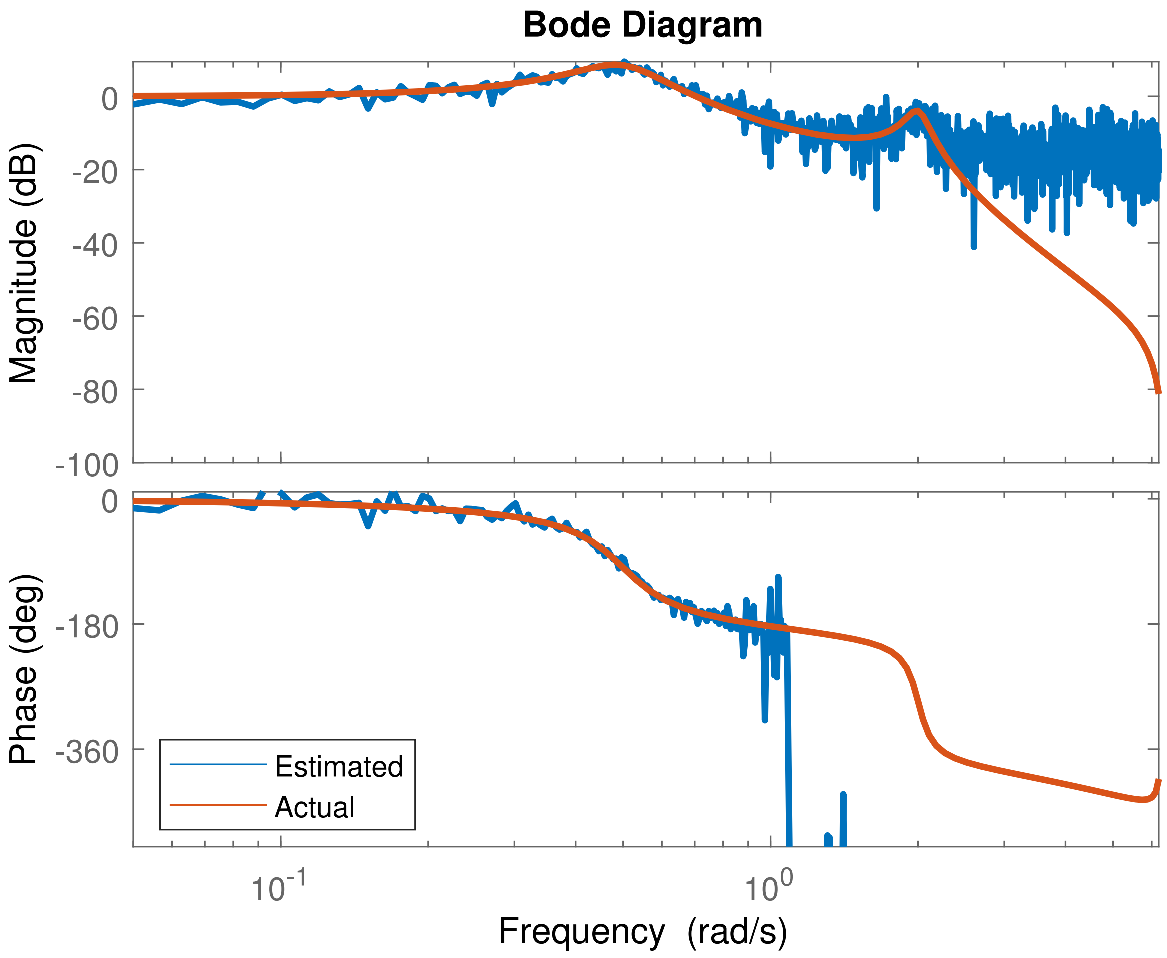

Figure 10: Bode plot of identified system vs real system using the entire data (Without averaging)

The solution of equation 4 is the transfer function of the system, also known as the fourier transform of the impulse response.

To improve our accuracy, we use a hamming window. The hamming window is particularly useful for unbiased autocorrelation function as the variance for large translation h is very big. The bode plots shown are based on a biased autocorrelation so the effect of the window is less proeminent but still observable on the phase plot.

Our transfer function $G(e^{j\omega})$ is also computed with the average of R_{yu} over 10 periods. We can do this because the input is pure noise and there should not be a big transient effet. It is worth noting that we could drop the first x values of u and y to make sure no transient is taken into account but for the sake of simplicity, we omit that. We can observe that the variance is largely reduced.

As for exercice 5, the system is badly identified for high frequencies. The cause is the same (low SNR at high frequencies) as the system doesn't change.

The input signal we are using is a uniform binary random signal. It is the one that achieves the highest exitation as it has the highest variance and uses the full range of the interval.

The system is best identified using the "biased" correlation, using averaging and using a hamming window because. These techniques reduces the variance of the measured signal and thus smoothens the identified signal although it leads to a biased estimate (it is not really noticeable on the plots).

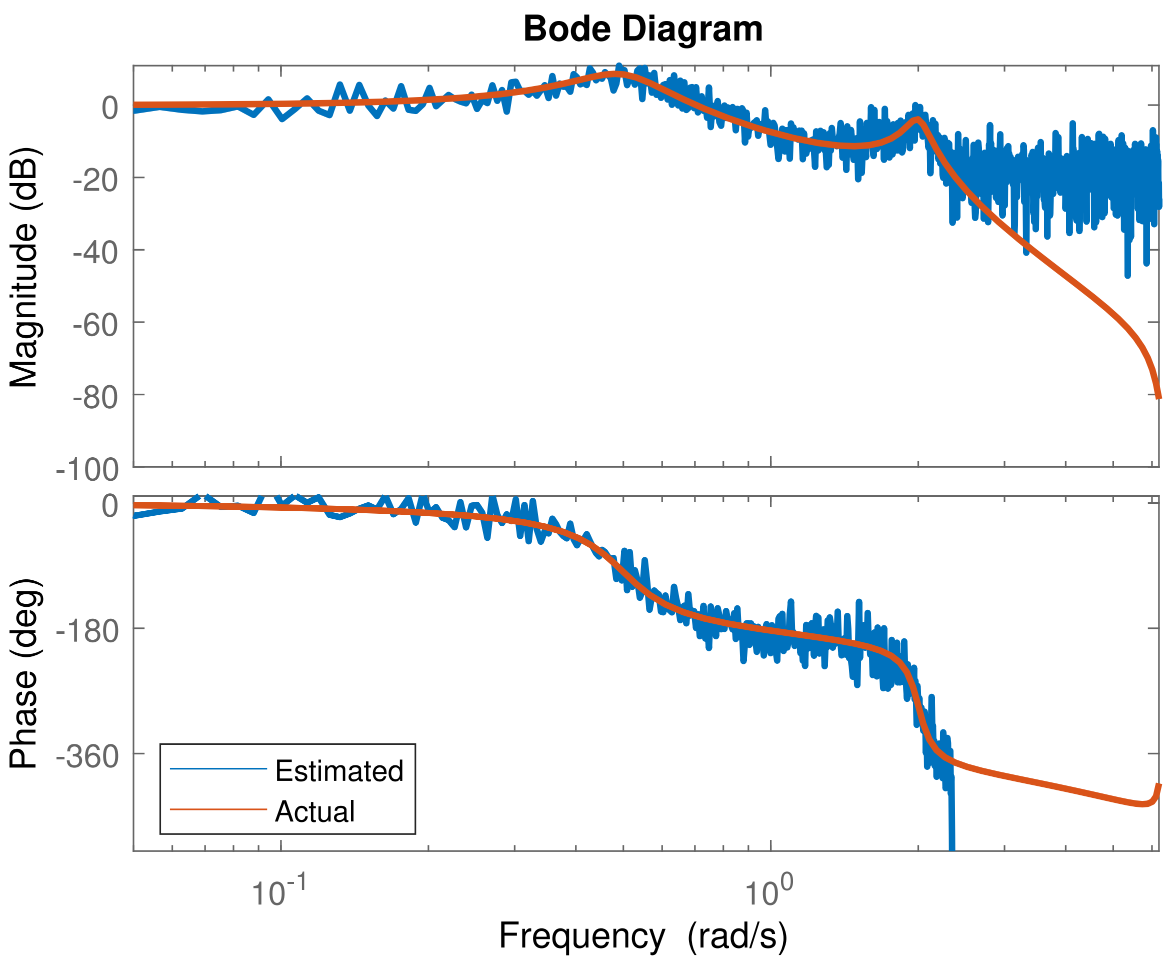

Figure 11: Bode plot of identified system vs real system by applying hamming window

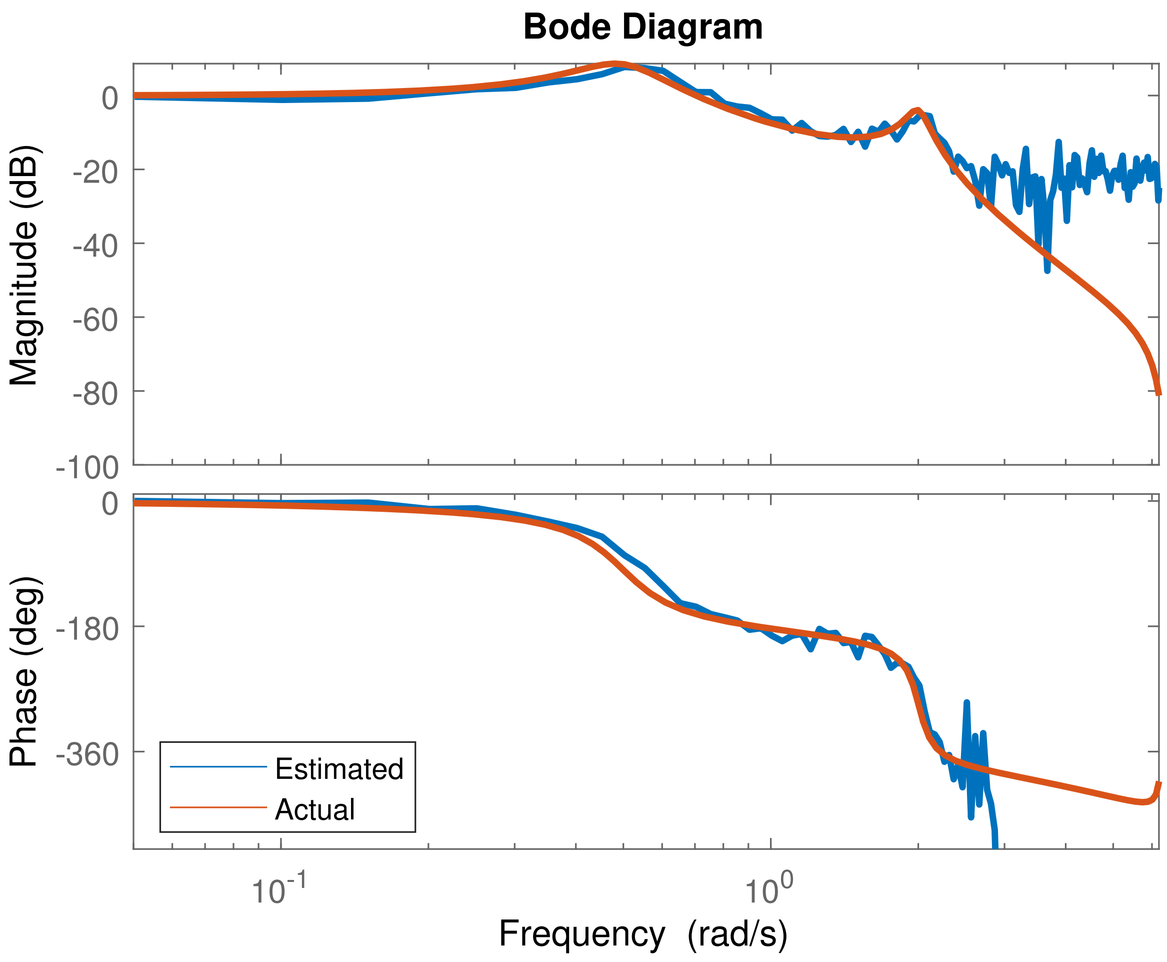

Figure 12: Bode plot of identified system vs real system with averaging over 8 chunks

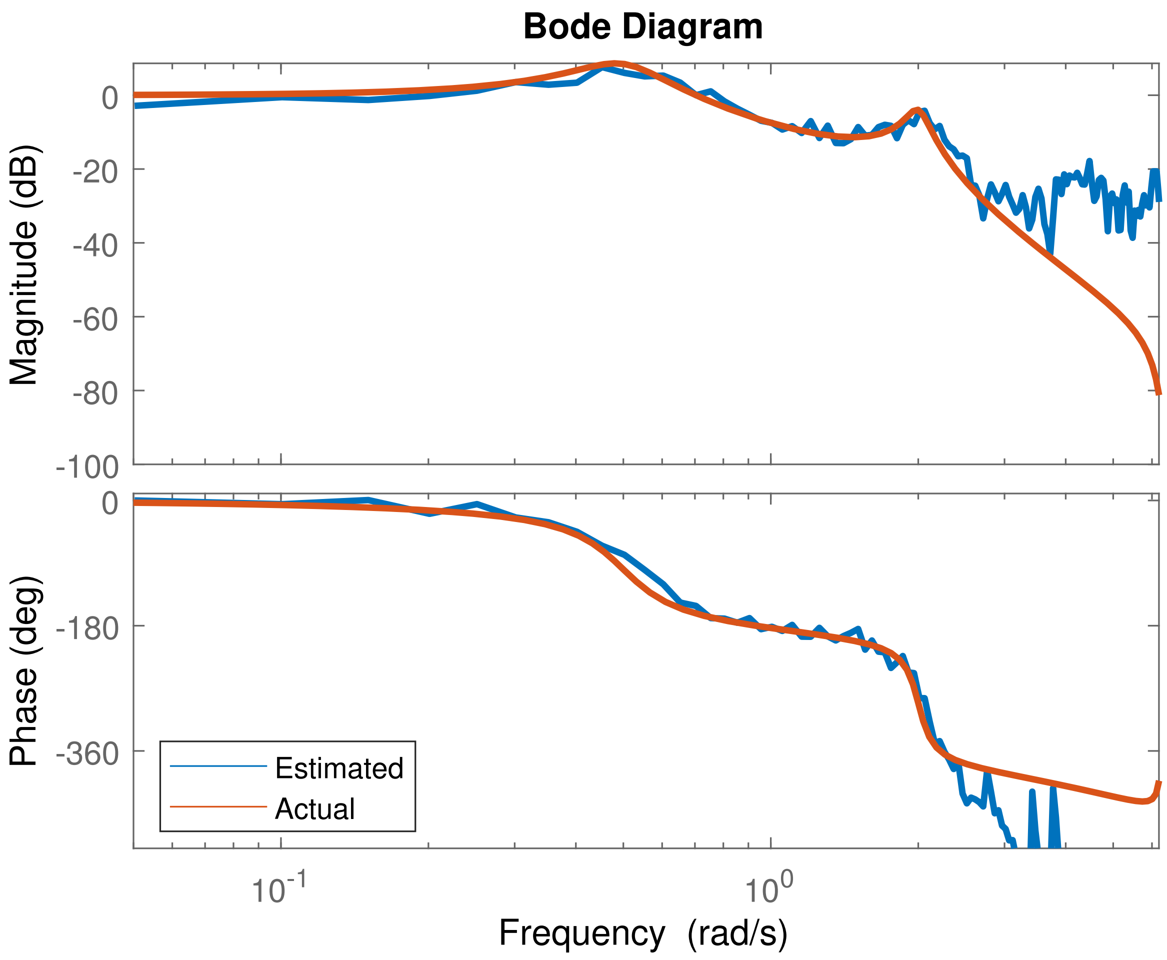

Figure 13: Bode plot of identified system vs real system with averaging over 8 chunks along with hamming window

7 Conclusion

The system identification of a real output with noise is done using various methods presented in class. We note the following,

- The step and impulse response of system is studied in exercise 1 in presence of output noise. It is observed that response is noisy. Hence, the output can be used for understanding the efficacy of system identification methods.
- PRBS input signal is used for exciting the system and system identification is carried out using non-parametric methods for time-domain and frequency domain.
- Impulse response identification of noisy output signal, excited by PRBS input using the deconvolution method and correlation method closely matches with the noiseless impulse response of same system. However, we note that the identified impulse response doesn't go to zero and remains noisy where the actual response is zero. This is primarily due to the noise in the system output.
- Frequency domain identification is carried out using periodic signal and random signal. In both cases the identified frequency response is very closely matching with the actual response. However, at higher frequencies, response could not be identified primarily due to the sampling time of 0.5s.

Hence, the time-domain and frequency domain identification of system is studied using various methods and it is observed that identified response matches very closely with the actual response.