System identification

Computer Exercise 1
Nonparametric methods

Authors:
Alexander Pisarewski
Bastien Ravot

Professor:
Alireza Karimi

January 7, 2020

Contents

1 Introduction

The objective of the computer exercise sessions is to implement different algorithms learned during the lectures in order to identify a model for a system. This first report focuses on nonparametric identification, thus the goal was to approximate either the impulse response $g(t)$ or the frequency response $G(j\omega)$. Those two responses are linked by the Fourier transform. The identification would allow a controller to be synthesized.

The algorithms were tested against a system with the following transfer function :

$$G(s) = \frac{3 - s}{s^2 + 1.12s + 2}$$

A random number block with a variance of 0.02 and a sample time of $T_e = 0.1$s has been added to the output of the transfer function block in order to simulate noise.

2 Nonparametric identification of a noisy fourth-order system

2.1 Step response

First of all, let us argue why 0.1 second is a reasonable sampling time. Shannon's theorem tells us that if a function $f(t)$ contains no frequencies higher than W [Hz], it is completely determined by giving its ordinates at a series of points placed $1/(2W)$ seconds apart. Thus, with a dominant frequency of $f_d = 0.23$ Hz, a sampling frequency of $f_s = 1/T_e = 10$ Hz allows an almost complete reconstruction of the signal since it is more than 40 times larger. Therefore $T_e = 0.1$s is a good choice.

After implementing our model on Simulink with the data given on the question sheet, it has been possible to simulate the response of the system with a step function as an input. The graph of the latter with respect to time is shown below with and without addition of noise.

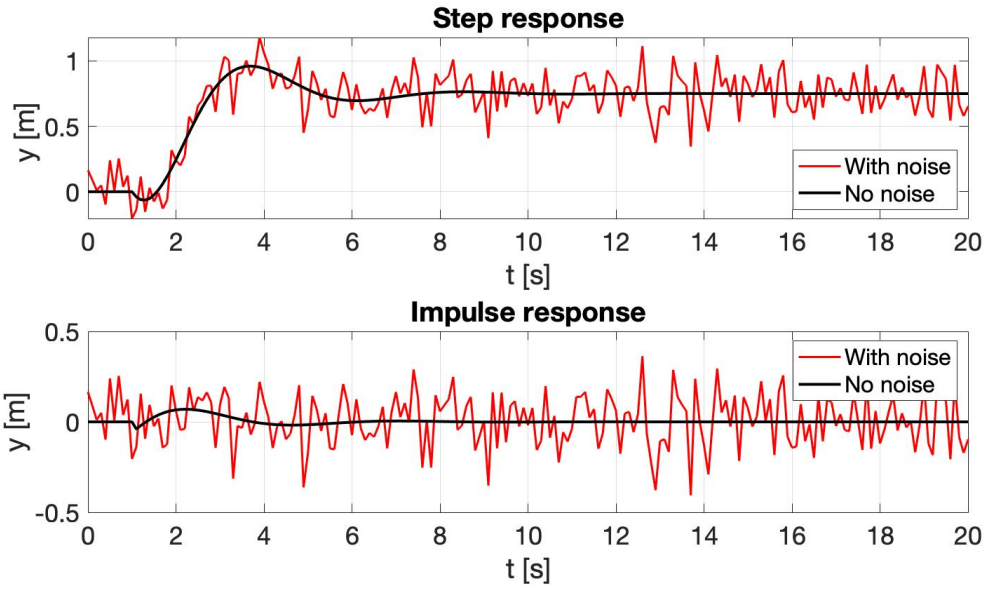

Figure 1: Step and impulse response of the system with addition of noise

The addition of noise makes the response function much harder to read, the overshoot is not distinguishable, neither is the settling time. Moreover, for the impulse response, the addition of noise makes the noiseless response of the system indistinguishable once the noise is added. It is therefore crucial to use methods that are able process out the noise.

2.2 Auto Correlation of a PRBS signal

In order to compute the cross-correlation $R_{uy}(h)$ of two signals, the intcor function has been implemented on Matlab. It assumes both signals u and v to be periodic of periods p_1 and p_2. Moreover, the length N of the shortest signal should be such that $\frac{N}{p_1}$ and $\frac{N}{p_2}$ are integers. This ensures that $u(N+m) = u(m)$, the same holds for v. The code is given in Listing 1.

Listing 1: intcor.m computes the correlation function of two periodic signals

```
function [R,h] = intcor(u,y)
p=min([length(u) length(y)]);
R=zeros(1,p);
h=0:p-1;
for H=h
    R(H+1)=sum(u(1:p).*[y(p-H+1:p);y(1:p-H)])/p;
end
end
```

Now let us check our function by computing the autocorrelation of a PRBS signal with the provided prbs function.

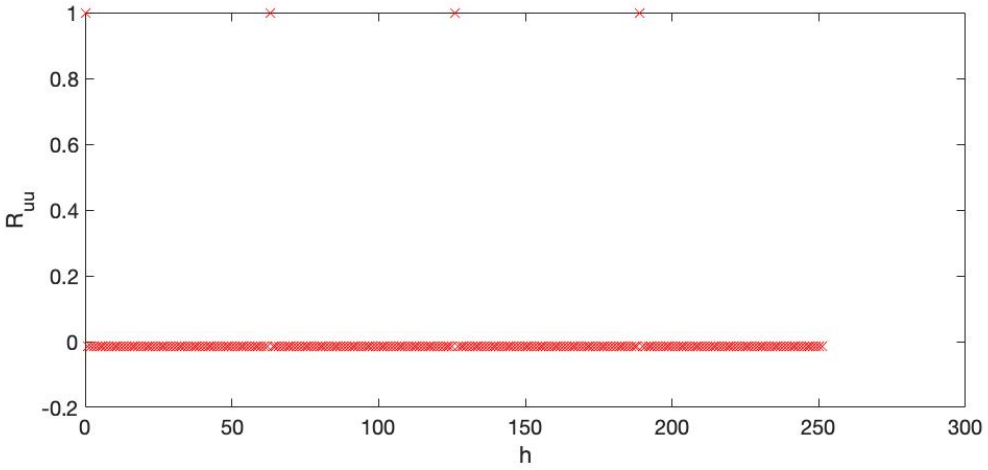

Figure 2: Autocorrelation of the PRBS signal

Figure 2 indeed shows the exact autocorrelation function of the PRBS signal generated with a shift register of length 6. The period of such a signal is $p = 2^6 - 1 = 63$ and its autocorrelation is 1 if $h = i \times p$, $i = 0, 1, 2...$ and $-1/p \approx -0.0159$ elsewhere.

2.3 Impulse response by deconvolution method

The objective of this section is to compute the impulse response of the system in Simulink using the numerical deconvolution method. First, a binary pseudorandom signal, within the range [-0.5 0.5] with a sampling time of Te = 0.1s has been created. Since for stable systems the impulse response goes asymptotically to zero, we can suppose that $g(k) = 0$ for $k > K$. This leads that to a deconvolution system of equations where we have N equations and $K << N$ unknowns.

$$Y = U_K \Theta_K$$

where U_K is a lower triangular Toeplitz matrix. From there, the solution in the least squares sense is: $\Theta_K = (U_K^T U_K)^{-1} U_K^T Y$. The full code is given in Listing 2.

Let us figure out how to choose K in order to reduce the error of our analysis. Figure 3 shows the absolute error between the exact θ, and its approximation θ_K.

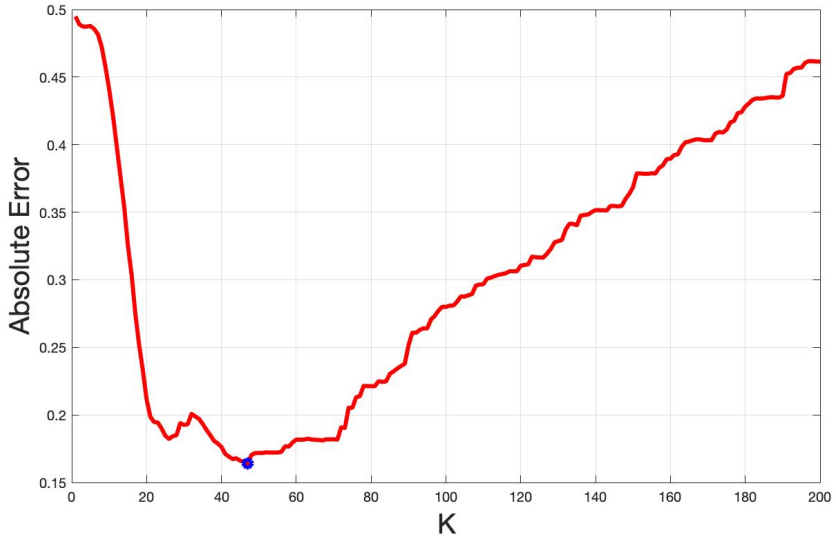

Figure 3: Absolute error $|\theta - \theta_K|$

A minimum error is obtained with $K = 47$ and thus we will pursue our analysis with this number. The 2-norm of the error is then approximately 0.17. Note that due to the random noise at the output of the system the norm of the error is stochastic.

We can see that this value of K is a good compromise. Indeed, a low value of K helps filter out the noise, but the impulse response is also set to 0 sooner.

Figure 4 compares the exact impulse response with the approximation obtained through deconvolution. In this figure, one could notice that the deconvolution method follows pretty much the magnitude of the exact impulse response even though is much noisier. The overshoot is now clearly identifiable, unlike in figure 1. Moreover, once steady state has been achieved, the computed response is no longer disrupted by noise. However, we can observe that it predicts that this steady state is reached much before it is actually the case (the last significant peak is ignored). Even though the error has been minimized, the settling time is underestimated with this value of K.

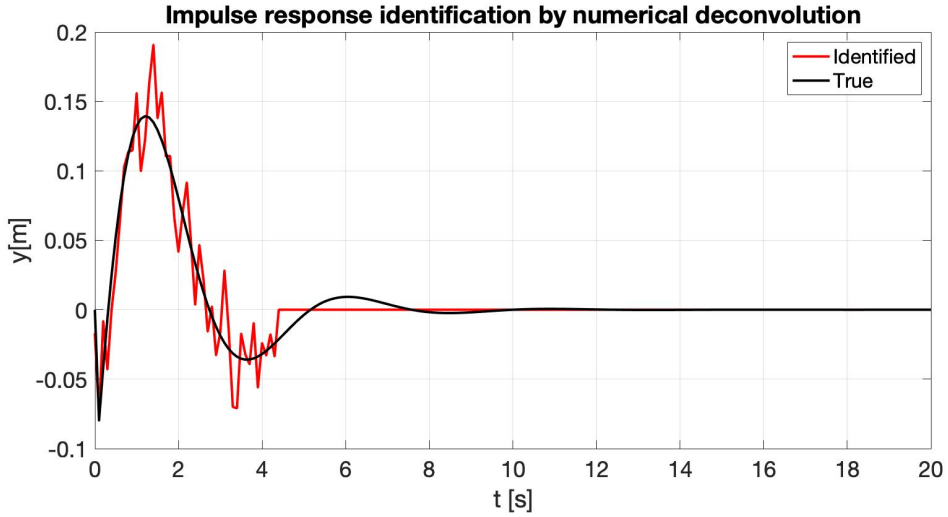

Figure 4: Impulse response obtained through deconvolution

Listing 2: deconvolution.m computes an approximation of the impulse response of the system based on the deconvolution method

```
% Simulink model parameters
Te=0.1;
variance=0.02;
Tmax=49;

% input signal
T=0:Te:Tmax;
N=length(T);
u=rand(N,1)-0.5;
simin.signals.values = u;
simin.time = T;

% simulation
sim('simulinkModel')
Y=simout.data;

% Deconvolution
U=toeplitz(u,[u(1) zeros(1,N-1)]);
theta_dec=U\Y;

K=44; % 44 minimizes error, 100 follows all the way
UK=U(:,1:K);
thetaK=(UK'*UK)\(UK'*Y);
theta_K=[thetaK;zeros(N-K,1)];

% Exact impulse response
sys=tf([-1 3],[1 1.12 2]);
sysd=c2d(sys,Te,'zoh');
theta_exact=impulse(sysd,Tmax)*Te;

% 2-norm of the error
err=norm(theta_K-theta_exact)
```

2.4 Impulse response by correlation approach

Consider the correlation approach to compute the impulse response. This approach is based on the auto-correlation and cross-correlation functions of signals. After applying a PRBS signal (with p = 4 and n = 7) to our system, the previously defined intcor function has been used in order to compute the auto- and cross-correlations. We only used the last period of our input signal in order to analyse the system once it reached steady state, since the transient response is by definition not periodic but will have died out.

Now that the autocorrelation function has been estimated, the impulse response can be obtained by numerical deconvolution. In our situation, since the input is not white noise, the following system of linear equations has to be solved:

$$\begin{bmatrix} \hat{R}_{yu}(0) \\ \hat{R}_{yu}(1) \\ \vdots \\ \hat{R}_{yu}(K-1) \end{bmatrix} = \begin{bmatrix} \hat{R}_{uu}(0) & \hat{R}_{uu}(1) & \dots & \hat{R}_{uu}(K-1) \\ \hat{R}_{uu}(1) & \hat{R}_{uu}(0) & \dots & \hat{R}_{uu}(K-2) \\ \vdots & \vdots & & \vdots \\ \hat{R}_{uu}(K-1) & \hat{R}_{uu}(K-2) & \dots & \hat{R}_{uu}(0) \end{bmatrix} \begin{bmatrix} \hat{g}(0) \\ \hat{g}(1) \\ \vdots \\ \hat{g}(K-1) \end{bmatrix} \tag{1}$$

Solving this system gave us the impulse response shown on Figure 5.

Notice that the correlation approach using the function xcorr or intcor gives similar results which ensures the accuracy of our intcor function. However it can be noticed that once steady state has been reached, xcorr gives better results than intcorr, this can be illustrated with the 2 norm of both errors. On one side xcorr has an error of approximately $err_{xcorr} = 0.20$ and on the

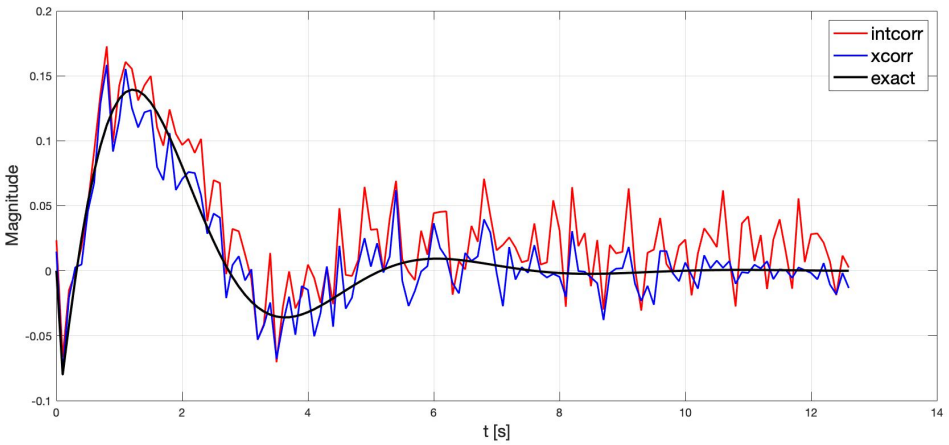

Figure 5: Impulse response obtained by correlation approach

other, intcor has an error estimated to $err_{intcorr} = 0.33$. This is explained by the fact that xcorr gives a biased estimate of the correlations, which is advantageous here because the correlations are very noisy, imprecise for large lags when they should anyway be small.

Interestingly, the errors are still slightly higher than the one obtained by the deconvolution approach. This may be because in the algorithm of the deconvolution approach the settling time was tuned to minimize the error. In the correlation approach this was not done because the period of the input signal was already such that the impulse response would be computed for just a little longer than the settling time.

Note that the PRBS excitation gives us results which are fairly good because the PRBS has an autocorrelation function which is very close to that of white noise except for the fact that it is periodic [1].

The codes used are given in Listings 3 and 4.

Listing 3: correlation_intcor.m approximates the impulse response of the system using the correlation approach and the intcor function

```
% Simulink model parameters
Te=0.1;
variance=0.02;

% input signal
Uprbs=prbs(7,4)*0.5;
Tmax=(length(Uprbs)-1)*Te;
T=0:Te:Tmax;
N=length(T);
simin.signals.values = Uprbs;
simin.time = T;

% simulation
sim('simulinkModel')
Y=simout.data;

% Retain only the last period
u=Uprbs(3*(2^7-1)+1:end);
v=Y(3*(2^7-1)+1:end);

% Correlation
Ruu=intcor(u,u);
Rvu=intcor(v,u);
```

```
24
25 % Deconvolution
26 M=toeplitz(Ruu);
27 g_intcorr=M\Rvu';
28 g_intcorr=[g_intcorr;zeros(N-length(g_intcorr),1)]; % Padding with zeros
29
30 % Exact impulse response
31 sys=tf([-1 3],[1 1.12 2]);
32 sysd=c2d(sys,Te,'zoh');
33 theta_exact=impulse(sysd,Tmax)*Te;
34
35 % 2-norm of the error
36 err_intcor=norm(g_intcorr-theta_exact)
```

Listing 4: correlation_xcorr.m approximates the impulse response of the system using the correlation approach and the xcorr function

```
1  % Simulink model parameters
2  Te=0.1;
3  variance=0.02;
4
5  % input signal
6  Uprbs=prbs(7,4)*0.5;
7  Tmax=(length(Uprbs)-1)*Te;
8  T=0:Te:Tmax;
9  N=length(T);
10 simin.signals.values = Uprbs;
11 simin.time = T;
12
13 % simulation
14 sim('simulinkModel')
15 Y=simout.data;
16
17 % Retain only the last period
18 u=Uprbs(3*(2^7-1)+1:end);
19 v=Y(3*(2^7-1)+1:end);
20
21 % Correlation
22 Ruu_x=xcorr(u,u);
23 Rvu_x=xcorr(v,u);
24
25 % Deconvolution
26 M_x=toeplitz(Ruu_x((length(Ruu_x)+1)/2:end));
27 g_xcorr=M_x\Rvu_x((length(Ruu_x)+1)/2:end);
28 g_xcorr=[g_xcorr;zeros(N-length(g_xcorr),1)]; % Padding with zeros
29
30 % Exact impulse response
31 sys=tf([-1 3],[1 1.12 2]);
32 sysd=c2d(sys,Te,'zoh');
33 theta_exact=impulse(sysd,Tmax)*Te;
34
35 % 2-norm of the error
36 err_xcorr=norm(g_xcorr-theta_exact)
```

2.5 Frequency domain Identification (Periodic signal)

So far we have identified the impulse response of the system. Let us now attempt to identify the frequency response of this same system.

To do so, one can rely on Fourier analysis. Indeed, for linear time-invariant (LTI) continuous time systems it holds that:

$$G(j\omega) = \frac{Y(j\omega)}{U(j\omega)} \tag{2}$$

where Y and U are the Fourier transforms of y (the response of the system) and u (the input signal).

For LTI discrete-time system, as our Simulink simulation, we have

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

where Y and U are now the Fourier series of y and u. Thus we can identify $G(e^{j\omega})$ but not $G(j\omega)$, which leads to a sampling error. As the Shannon theorem is here almost verified (the response at frequencies higher than the cutoff frequency is almost zero), this error should not be too high, we should have $G(e^{j\omega}) \approx G(j\omega)$.

In order to be able to compute the Fourier transform of the input signal exactly, a periodic input signal was used. In particular, a PRBS signal was chosen because it excites all frequencies (it is similar to white noise while being deterministic and periodic). To obtain a signal length of approximately 2000, several combinations of the length the shift register n and number of periods p are possible. The length of the shift register determines the resolution of the identification. Indeed, due to the Fourier series, the frequency vector associated with the identification is $\left[\ 0, \quad \omega_s/N, \quad 2\omega_s/N, \quad \ldots \quad (N-1)\omega_s/N\ \right]$ with $N = 2^n - 1$.

Thus a higher n leads to a better resolution, but it also leads to a smaller number of periods. This is problem because averaging is done over the periods to try and eliminate the effect of noise. The first two periods are discarded so that the transient response of the system does not effect the results (the response of the noiseless system is periodic which avoids the truncation error). Averaging is done over the remaining periods, and the more periods we average over the more we reduce the effect of noise. We found $n = p = 8$ to be a suitable pair. It gives the results plotted on figure 6.

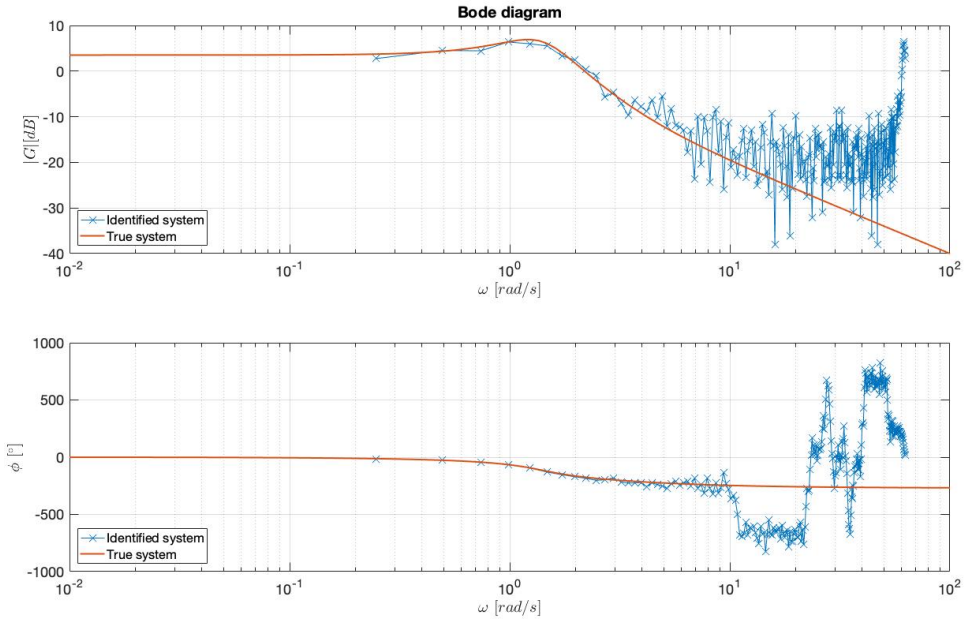

Figure 6: Bode diagram of the system identified using Fourier analysis

The full code is given in Listing 5

Listing 5: fourier.m computes the approximate frequency response of the system using Fourier analysis

```
% Simulink parameters
Te=0.1;
variance=0.02;

w_s=2*pi/Te;

% Input signal
n=8;
p=8;
Uprbs=prbs(n,p)*0.5;
Tmax=(length(Uprbs)-1)*Te;
T=0:Te:Tmax;
N=2^n-1;
simin.signals.values = Uprbs;
simin.time = T;

% simulation
sim('simulinkModel')
Y=simout.data;

% DFT
full_fft_Y=fft(reshape(Y,[],p));
fft_U=fft(Uprbs(1:N));

% discard the first 2 periods and average
av_fft_Y=mean(full_fft_Y(:,p-2:end),2);

freq_vec=0:w_s/N:w_s*(N-1)/N; % frequency vector

% Identification
identified_sys=frd(av_fft_Y./fft_U,freq_vec,Te);
[mag,phase,w]=bode(identified_sys);
```

2.6 Frequency domain Identification (Random signal)

Another method was implemented in order to identify the frequency response of the system. This other frequency domain identification algorithm is based on spectral analysis.

The input signal will be a random signal. In order to obtain the highest possible energy of excitation, we chose to use a binary random signal, which can only take the values [-0.5 0.5] that correspond to the saturation of the system. The energy of excitation being given by $\sum_{k=0}^{N} u^2(k)$, no higher value can be obtained with any other random signal.

The procedure is as follows. First, the autocorrelation of the input signal R_{uu} as well as the cross-correlation between the output and input signal R_{yu} were computed. Then the discrete Fourier transform of those functions was taken to obtain the power spectral density functions ϕ_{uu} and ϕ_{yu} respectively. Finally, the estimated transfer function is :

$$\hat{G} = \frac{\phi_{yu}}{\phi_{uu}} \tag{4}$$

The result is shown on figure 7.

To improve the results, one can apply a window to the correlation functions before computing the discrete Fourier transform. The window is chosen such that the correlations for high values of $h > M$ are set to 0. Here a Hann window was used so that the decay to zero is smooth. Windowing helps reducing the effect of noise because it has a smoothing effect, which gets more pronounced as M is decreased. However decreasing M also decreases the resolution, which is why a value of 200 was chosen for our problem.

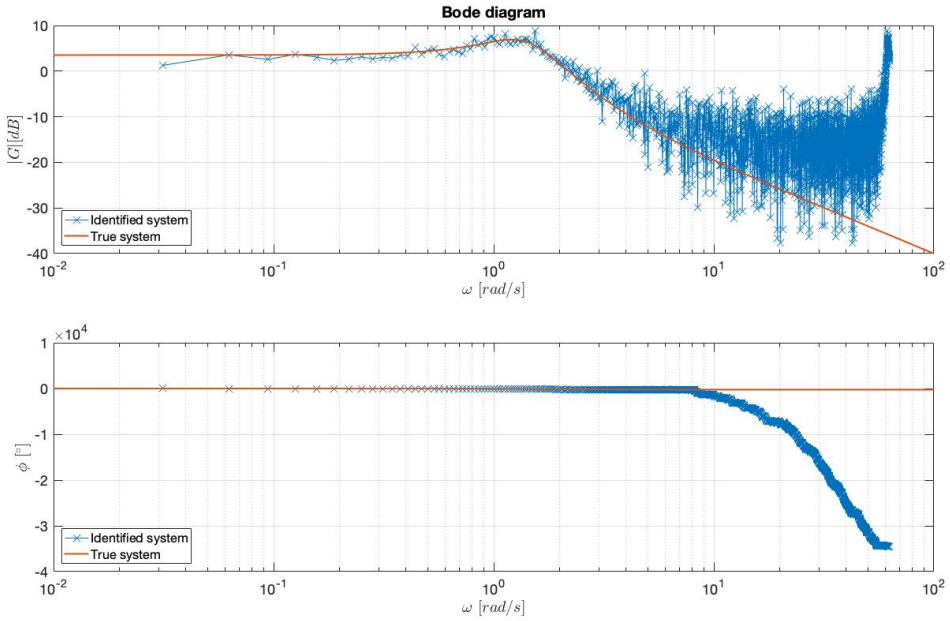

Figure 7: Bode diagram of the system identified using spectral analysis

Note that using a biased estimate when computing the correlation functions is equivalent to using an unbiased estimate with triangular windowing (with a window size corresponding to the length of the signal), which is why we found it to give better results.

The new Bode diagram is plotted on figure 8.

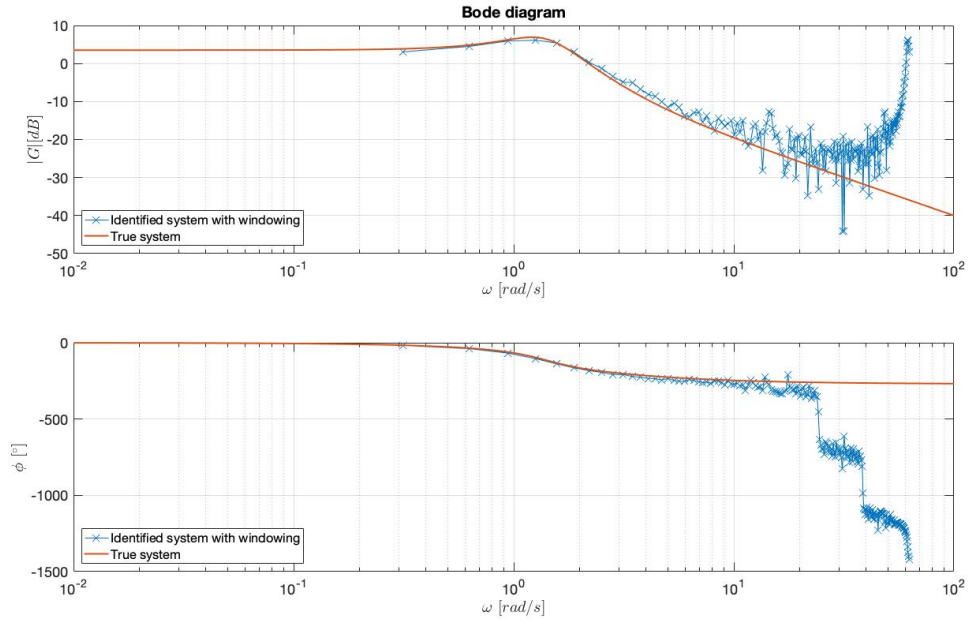

Figure 8: Bode diagram of the identified system with windowing

It can be seen that the resolution is sufficient to identify the resonant mode, and that compared to figure 7 the effect of noise has been very significantly reduced. However it could not be eliminated, especially at high frequencies.

To try and improve even further the identification one can cut the data into m groups of equal length before proceeding. All of these data sets are processed as described above. The results of the discrete Fourier transforms are averaged between the groups, before computing the approximate transfer function. This will decrease the variance of the results when an unbiased estimate of the correlation functions is used.

A last Bode diagram is obtained, please see figure 9.

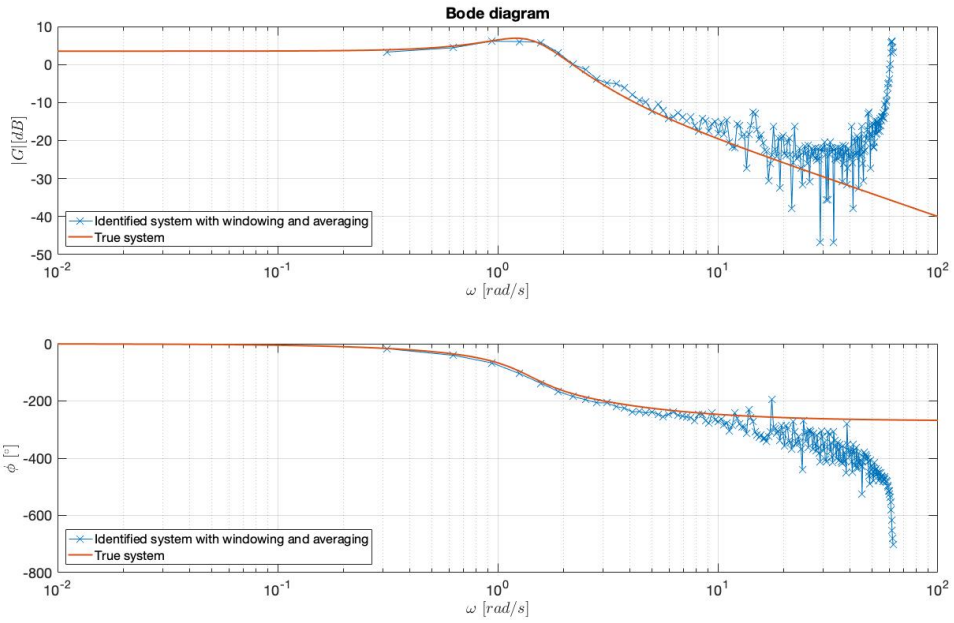

Figure 9: Bode diagram of the identified system with windowing and averaging

The results are unfortunately not very different from those obtained using only windowing. The phase estimation is slightly better in the sense that it the divergence to very negative values appears later, for higher frequencies.

The code described above is given in Listing 6

Listing 6: spectral.m approximates the frequency response of the system based on spectral analysis

```
% Simulink parameters
Te=0.1;
variance=0.02;

w_s=2*pi/Te;

% Input signal
N=2000;
U_rand=unidrnd(2,N,1)-1.5;
Tmax=(N-1)*Te;
T=0:Te:Tmax;
simin.signals.values = U_rand;
simin.time = T;

% simulation
sim('simulinkModel')
Y=simout.data;

```

```
% Correlation functions
Ruu=xcorr(U_rand,U_rand,'biased');
Ruu=Ruu((length(Ruu)+1)/2:end);
Ryu=xcorr(Y,U_rand,'biased');
Ryu=Ryu((length(Ryu)+1)/2:end);

% DFT
phi_uu=fft(Ruu);
phi_yu=fft(Ryu);

freq_vec=0:w_s/N:w_s*(N-1)/N;

identified_sys=frd(phi_yu./phi_uu,freq_vec,Te);
[mag,phase,w]=bode(identified_sys);

%% Windowing
M=round(N/10); % size of the window
window=hann(2*M);
window=window(length(window)/2+1:end);

% Correlation with windowing
phi_uu=fft(Ruu(1:M).*window);
phi_yu=fft(Ryu(1:M).*window);

freq_vec=0:w_s/M:w_s*(M-1)/M;

identified_sys=frd(phi_yu./phi_uu,freq_vec,Te);
[mag_w,phase_w,w_w]=bode(identified_sys);

%% Windowing and averaging
m=4; % number of sets the data will be cut into
N=N/m;
M=200; % size of the window
window=hann(2*M);
window=window(length(window)/2+1:end);
phi_uu=zeros(M,m);
phi_yu=zeros(M,m);

for k=0:m-1
    % Correlation for a given set
    Ruu=xcorr(U_rand(1+k*N:(k+1)*N),U_rand(1+k*N:(k+1)*N),'unbiased');
    Ruu=Ruu((length(Ruu)+1)/2:end);
    Ryu=xcorr(Y(1+k*N:(k+1)*N),U_rand(1+k*N:(k+1)*N),'unbiased');
    Ryu=Ryu((length(Ryu)+1)/2:end);
    % DFT for the set
    phi_uu(:,k+1)=fft(Ruu(1:M).*window);
    phi_yu(:,k+1)=fft(Ryu(1:M).*window);
end
% averages of the DFTs
phi_uu=mean(phi_uu,2);
phi_yu=mean(phi_yu,2);

freq_vec=0:w_s/M:w_s*(M-1)/M;

identified_sys=frd(phi_yu./phi_uu,freq_vec,Te);
[mag_w_a,phase_w_a,w_w_a]=bode(identified_sys);
```

3 Conclusion

To conclude, the implemented algorithms allowed us to obtain a large amount of characteristics of the analysed system. Two main classes of methods have been focused on; the methods in the time and the frequency domain. The time-domain methods allow us to identify the impulse response while the frequency-domain methods identify the frequency response (magnitude and phase).

As seen these latter can be determined with different methods which all have their advantages and drawbacks. They all contain errors such as the sampling, measurement and truncation errors in the Fourier analysis.

Moreover, as noticed in the report, changing several parameters in the algorithms allow us to improve the obtained results. Indeed, changing the settling time in the deconvolution approach changes drastically the results. Similarly spectral analysis requires windowing (with a well-chosen window size) to reduce the effects of noise.

4 References

[1] Prof. Karimi. System identification. Course slides.

[2] Dr. Alireza Karimi. System identification, Fall 2019. Course notes.