ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CE-1: NONPARAMETRIC METHODS

SYSTEM IDENTIFICATION ME-421

PROF. ALIREZA KARIMI

Angelo Giovine 368440

Hana Catic 370754

15th April 2024

Contents

1. STEP RESPONSE

1.1

Given this transfer function:

$$G(s) = \frac{1}{s^4 + 0.4s^3 + 4.3s^2 + 0.85s + 1}$$

we create the variables `num` and `den` 1.1

1.2

To understand if the sampling time $T_s = 0.5s$ is reasonable we compute the time constant of the system.

For complex conjugate poles of the form $-\sigma \pm j\omega$, the time constant is related to the real part σ, because the imaginary part ω determines the oscillatory behavior. The time constant τ for a pole at $-\sigma$ is given by $\tau = \frac{1}{\sigma}$.

Let's calculate the time constants for the dominant poles of the given system.

The dominant poles of the system are complex conjugates at approximately $-0.1003 \pm 2.0002j$. The time constants for these dominant poles are both approximately 9.973 seconds.

Given the time constant $\tau \approx 9.973$ seconds, we can calculate the minimum sampling time T_s as:

$$T_s = \frac{\tau}{N}$$

where N is the factor by which the sampling time is faster than the time constant. Choosing N to be 10, for example, we get:

$$T_s = \frac{9.973 \text{ seconds}}{10} = 0.9973 \text{ seconds}$$

This is a conservative estimate, hence $T_s = 0.5s$ is reasonable, so we create `Ts` 1.1.

1.3

For simulate the measurement noise we create a variance variable: `var` 1.1.

1.4

For the input saturation we create: `uplimit` and `lowlimit` 1.1.

1.5

Here is the completed block diagram, which incorporates all the previously mentioned parameters. Additionally, we have chosen to generate the system's noise-free response, which will be utilized in subsequent plots. 1.1 :

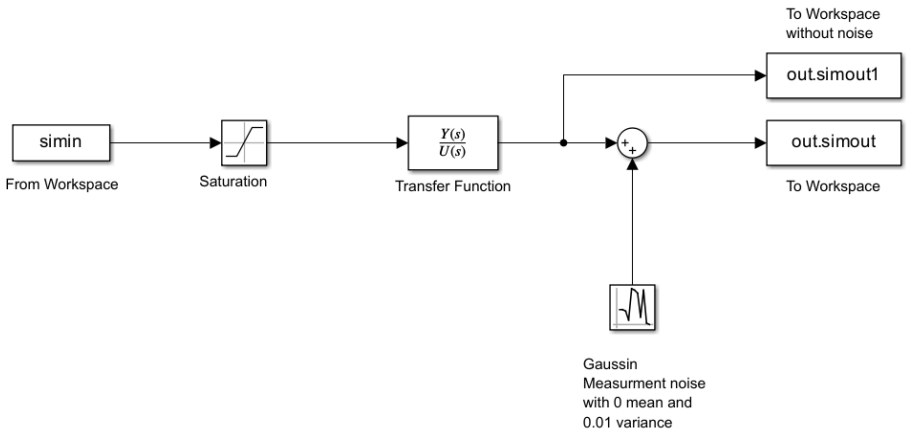

FIGURE 1.1: Block diagram

LISTING 1.1: Parameters

```
% parameters
num = 1;
den = [1 0.4 4.3 0.85 1];
Ts = 0.5;
var = 0.01;
uplimit = 0.5;
lowlimit = -0.5;
```

1.6

Here we want to study the step response of the system, to do so we create the input signal a step of amplitude 0.5 delayed by 1 s (1.2):

LISTING 1.2: Parameters

```
% Apply a step of amplitude uplimit delayed by 1 s
Time = 100;
Discretize_Time = floor(Time/Ts)+1;
step = uplimit*ones(1,Discretize_Time)';
step(1:3,1) = 0;
simin.signals.values = step;
simin.time = transpose(linspace(0,Time,Discretize_Time));
```

1.7

Here we can see the resulting plot, and observing the noiseless system response, the behavior is the expected one, hence the output settles down at a value that is equal to the amplitude of the applied step:

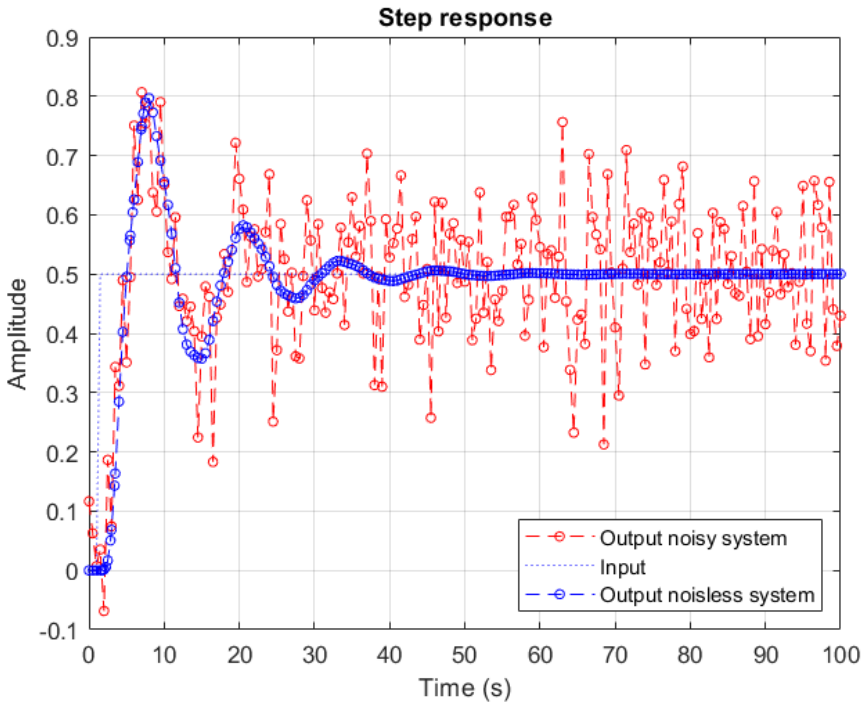

FIGURE 1.2: Step response

Now we want to compute the impulse response of the system, to understand better the behavior of our system we have decide to apply our impulse with 1 second delay:

LISTING 1.3: Apply an impulse

```
%% Apply an impulse
Time = 100;
Discretize_Time = floor(Time/Ts)+1;
imp = zeros(1,Discretize_Time)';
imp(3,1) = 1;
simin.signals.values = imp;
simin.time = transpose(linspace(0,Time,Discretize_Time));
```

Here we can see the resulting plot, once again the noiseless system response help us to check the correctness of our code:

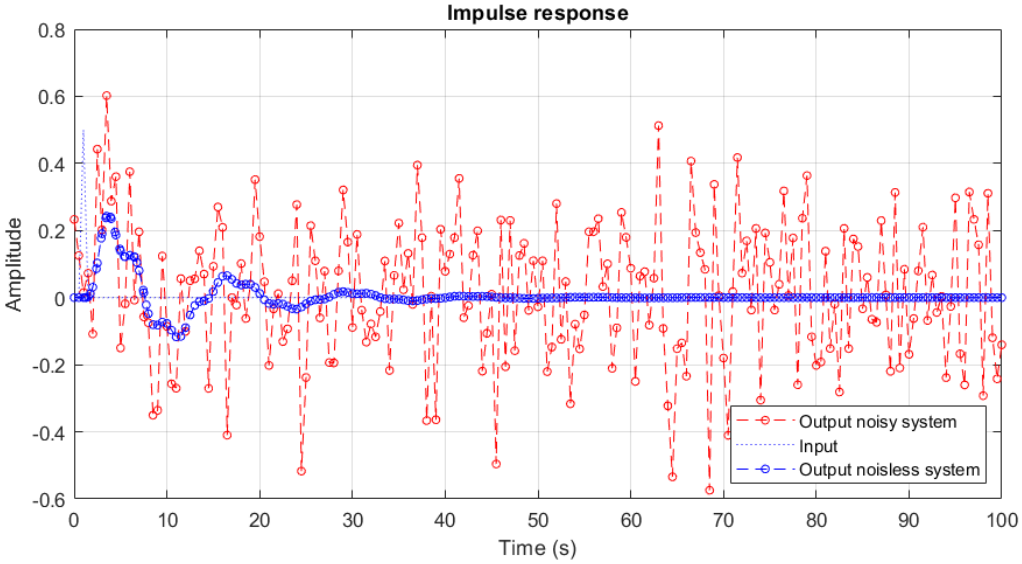

FIGURE 1.3: Impulse response

2. AUTO CORRELATION OF A PRBS SIGNAL

2.1 CROSS CORRELATION

The spectrum of the system is used to determine the degree of excitation of the system that can be achieved with a certain signal. It can be computed as the Fourier transform of the autocorrelation function of the signal. To that end function *[R,h] = intcor(x,y)* is written to compute the cross-correlation between two periodic signals. The function is given in the Appendix A, A.1.

Main assumption used to compute the cross-correlation is periodicity of both signals. However, there is no requirement that they should have the same period, thus the common period of signals is determined and further used. Additionally, since they are periodic, it is assumed that beyond the given interval of the signals, they have the same behaviour and should not be zero padded. In accordance with these assumptions the signals are expanded to be compatible with their common period, using their respective periods. Computation of cross-correlation is done based on the following expression, where M is the common period:

$$R_{uy}(h) = \frac{1}{M} \sum_{k=0}^{M-1} u(k)y(k-h). \tag{2.1}$$

2.2 AUTO CORRELATION OF A PRBS SIGNAL

In order to verify the function A.1 two periods of a PRBS signal with four shift registers are generated using the provided function *x = prbs(n,p)*, as given in listing A.2. Generated PRBS signal is shown in figure 2.1. The number of shift registers and the period of the PRBS signal can be confirmed based on this figure, under the assumption that there is no frequency divider. The maximum number of successive ones in the signal is four, indicating four shift registers. Number of binary numbers that can be generated with four registers is $2^4 = 16$, however, zero is never generated. If a number consisting of only zeros were generated then after that no other number could be generated, thus the period is not 16 but 15. This is also observed in the figure 2.1.

PRBS is a periodic signal, thus the autocorrelation is computed as follows:

$$R_{uu}(h) = \frac{1}{M} \sum_{k=0}^{M-1} u(k)u(k-h) = \begin{cases} a^2, & \text{if } h = 0, \pm M, \pm 2M, ... \\ -\frac{a^2}{M}, & \text{elsewhere} \end{cases}. \tag{2.2}$$

Since PRBS is a periodic function and expressions 2.1 and 2.2 are the identically when $u = y$, it satisfies all of the assumptions of *intcor* function and it can thus be applied. The results are presented in figure 2.2. They are in accordance with expression 2.2, the value of $R_{uu}(h) = 1$, when $h = kM$, where k is an integer and M the period, for all other h the value of $R_{uu}(h) = -0.07$.

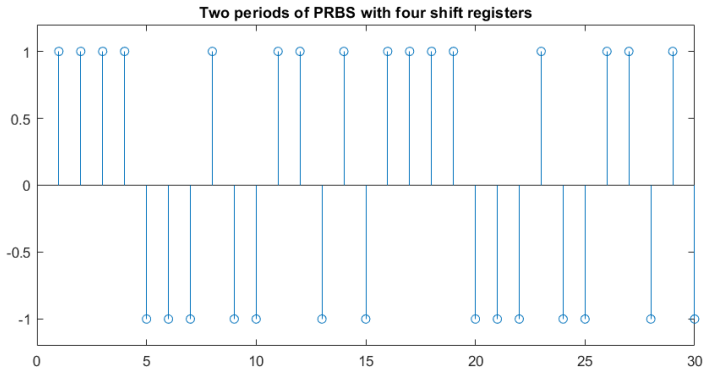

FIGURE 2.1: PRBS signal with four shift registers

Since autocorrelation is a discrete and periodic function, the spectrum is going to be periodic and discrete, respectively. All of the results shown are further verified by examples provided in the course notes.

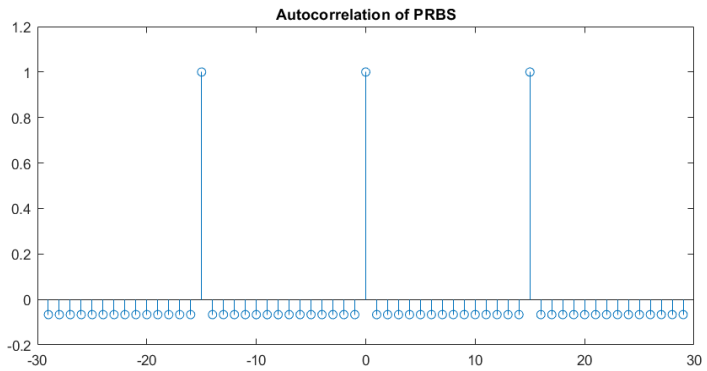

FIGURE 2.2: Autocorrelation of a PRBS signal with four shift registers

Figure 2.3 shows the PRBS of 6 registers and 4 periods, along with the computed autocorrelation.

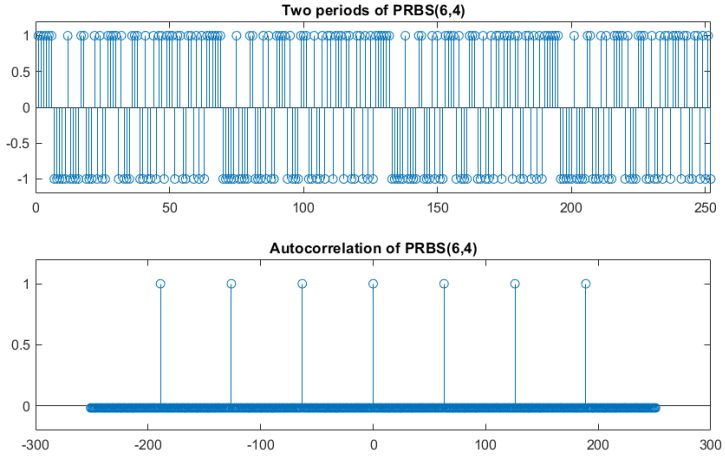

FIGURE 2.3: Autocorrelation of a PRBS(6,4)

3. IMPULSE RESPONSE BY DECONVOLUTION METHOD

3.1

Firstly to compute the impulse response using the deconvolution method, we apply a random signal to our system with a sampling time of $T_s = 0.5s$

LISTING 3.1: Apply a random signal

```
%% Part 1 Apply a random signal
Time = 100;
Discretize_Time = floor(Time/Ts)+1;
randomsig = rand(1,Discretize_Time)';
randomsig = randomsig/2; % to respect the saturation limits
simin.signals.values = randomsig;
simin.time = transpose(linspace(0,Time,Discretize_Time));
```

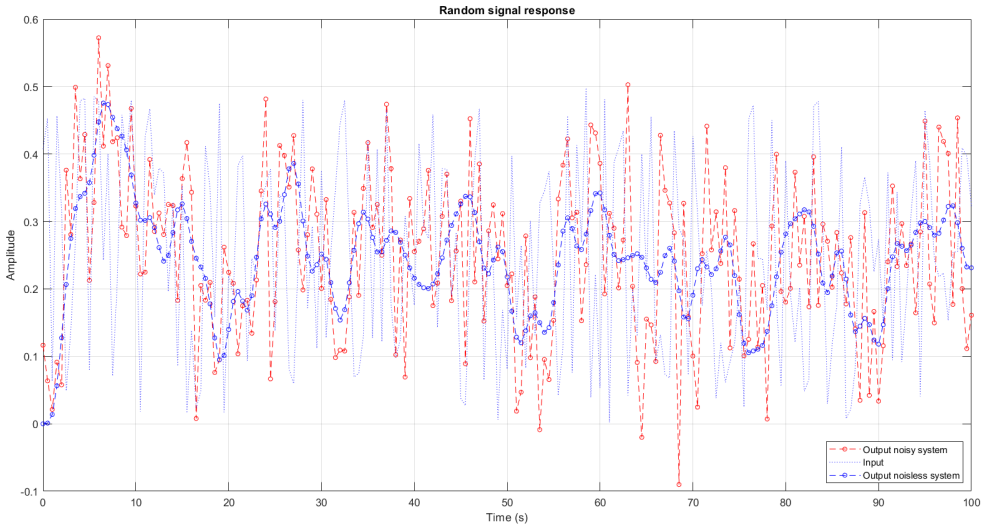

FIGURE 3.1: Random signal response

3.2

The main idea of the deconvolution method is to apply any input $u(k)$ and compute the impulse response $g(k)$. To do that we write the convolution sum in matrix form obtaining:

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} u(0) & 0 & \cdots & 0 \\ u(1) & u(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ u(N-1) & u(N-2) & \cdots & u(0) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(N-1) \end{bmatrix}$$

$$Y = U\Theta \implies \Theta = U^{-1}Y$$

But the asymmetric Toeplitz matrix U is ill-conditioned so Θ cannot be computed. Hence later we will compute the impulse response using two methods. To generate the U matrix, this is the relative MATLAB code:

LISTING 3.2: Toeplitz Matrix

```
%% Part 2 Use toeplitz command to construct the input matrix.
N = length(randomsig);
r = zeros(1,N);
r(1,1) = randomsig(1,1);
c = randomsig';
U = toeplitz(c,r);
```

3.3

LISTING 3.3: time vector

```
%% Part 3
t=0:Ts:(N-1)*Ts;
```

3.4

Now we are going to use the first method to deal with the impossibility of inverting the U matrix, the finite impulse response method. For stable systems (without integrator), impulse response goes asymptotically to zero such that we can suppose $g(k) = 0$ for $k > K$. Then, we can write:

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} u(0) & 0 & \cdots & 0 \\ u(1) & u(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ u(N-1) & u(N-2) & \cdots & u(N-K) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(K-1) \end{bmatrix}$$

or $Y = U_K\Theta_K$, where U_K and Θ_K are the truncated version of U and Θ. Now, the number of equations, N, is more than the number of unknowns K (i.e. U_K is a $N \times K$ dimensional matrix). This problem can be solved in the least squares sense by computing the pseudo inverse of U_K:

$$\Theta_K = U_K^{\dagger}Y = (U_K^T U_K)^{-1}U_K^T Y$$

Hence by observing the impulse response obtained in the previous section of this report, we can choose an adequate K, by choosing $K = 100$ (Almost $50s$). Here the code (3.4): and the relative plots 3.2

LISTING 3.4: 1 Method

```
%% Part 4 Finite impulse response method
K = 100;
Ureduced = U(:,1:(N-K));
Y = simOut.simout.Data;
Ynonoise = simOut.simout1.Data;
Thetareduced = pinv(Ureduced)*Y;
Thetareducednonoise = pinv(Ureduced)*Ynonoise(1:201);
```

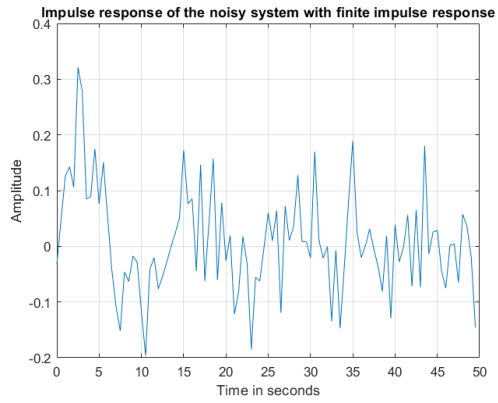

FIGURE 3.2: With noise

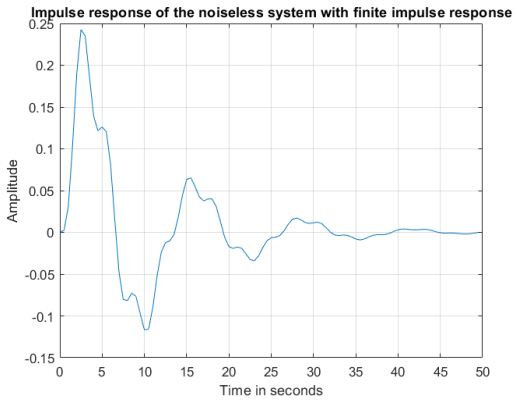

FIGURE 3.3: without noise

3.5

Another approach is to use the regularization approach, let's consider a modified least squares criterion as:

$$J(\Theta) = \mathcal{E}^T\mathcal{E} + \lambda\Theta^T\Theta$$

where $\mathcal{E} = Y - U\Theta$, and λ is a weighting factor for bias-variance trade-off. Then we have:

$$\frac{\partial J}{\partial \Theta} = -2U^TY + 2U^TU\Theta + 2\lambda\Theta = 0 \implies \Theta = (U^TU + \lambda I)^{-1}U^TY$$

A non zero λ makes $U^TU + \lambda I$ a regular matrix, which is invertible and leads to a finite norm Θ. The right value of λ for our intent was found by checking the eigenvalues of the toeplitz matrix and with a bit of trial and error. In the end we have chosen $\lambda = 0.25$. Here the code is given with listing (3.5), and the relative plots 3.4.

LISTING 3.5: 2 Method

```
%% Part 5 regularization
lambda = 0.25;
n = size(U,1);
Theta_reg = (inv(U'*U+lambda*eye(n)))*(U')*(Y);
Theta_reg_nonoise = (inv(U'*U+lambda*eye(n)))*(U')*(Ynonoise);
```

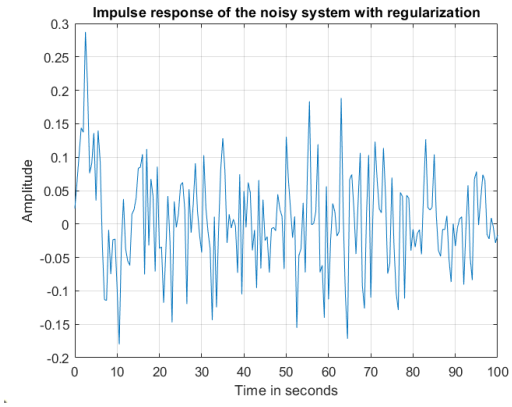

FIGURE 3.4: With noise

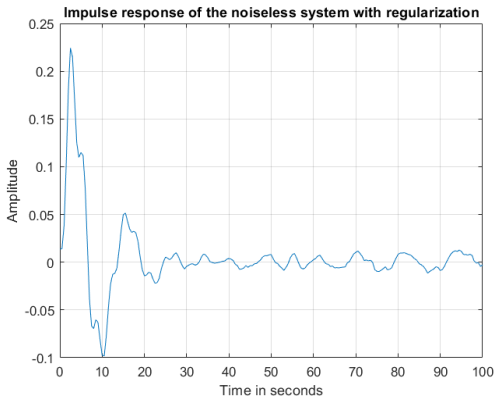

FIGURE 3.5: without noise

3.6

To compare the results we first generate the true impulse response:

LISTING 3.6: True response

```
%% Part 6 Real response
Gdis = c2d(tf(num,den),Ts,'zoh');
True_output = impulse(Gdis, t) * (Ts);
```

Then we plot the different response to compare them:

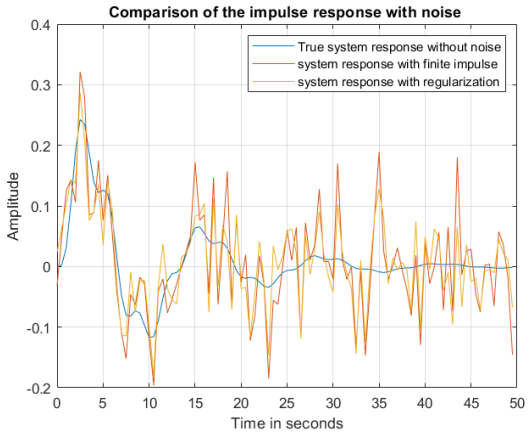

FIGURE 3.6: With noise

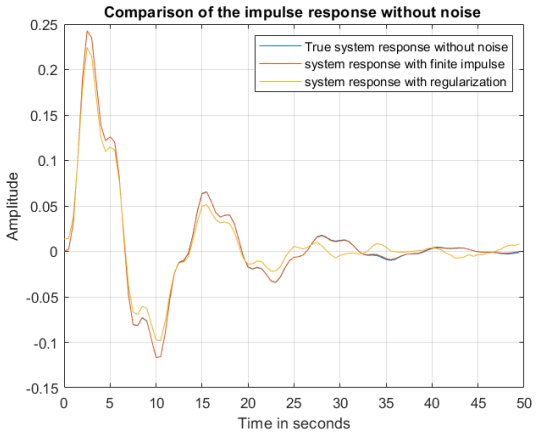

FIGURE 3.7: without noise

We can observe, in the absence of noise (Figure 3.7), an almost perfect match between the true impulse response and the one obtained through finite impulse response, that's due to the fact that after 50 s the output is almost constant and equal to zero, hence by "cutting" the Toeplitz matrix we are preserving all the relevant information about the system response. While in the regularization case (Figure 3.7) the match is not perfect, that is because a finer tune of the λ parameter would have been necessary. In contrast to K, the λ tuning is less obvious, since it has a less intuitive meaning. In the end, by also checking the case in the presence of noise (Figure 3.6), we can see that the difference between the two methods is less evident and both exhibit significant noise.

To be consistent with calculating the 2-norm errors, we have computed them over a time period of K time instants.

LISTING 3.7: Errors

```
%% 2 NORM ERRORS
error_reduced = norm(True_output(1:K)-Thetareduced(1:K));
error_reg = norm(True_output(1:K)-Theta_reg(1:K));
error_reduced_nonoise =
   norm(True_output(1:K)-Thetareducednonoise(1:K));
error_reg_nonoise = norm(True_output(1:K)-Theta_reg_nonoise(1:K));
```

The 2-norm of the errors of the 2 approaches with respect to the true system are listed below:

TABLE 3.1: Error Measurements by Type and Condition

Condition / Approach	Regularization	Finite impulse response
No Noise	0.0992	0.0042
With Noise	0.5835	0.6864

4. IMPULSE RESPONSE BY CORRELATION APPROACH

4.1 PRBS

Input sequence is generated using *prbs* of 4 periods and 7 shift registers and applied to the system given with figure 1.1. The input and output of the noisy system are given in the figure below.

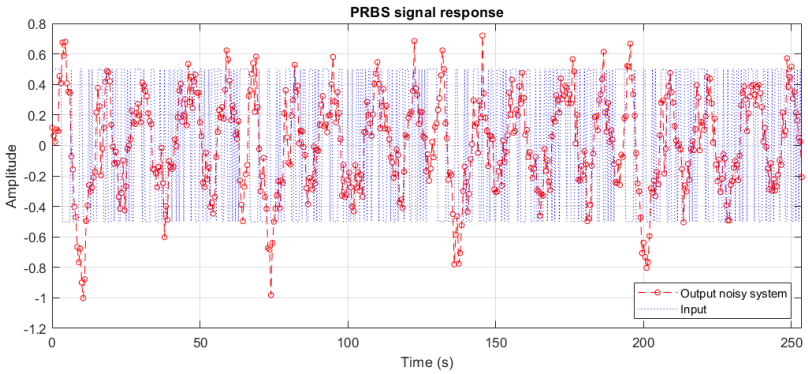

FIGURE 4.1: PRBS input applied to system 1.1

4.2 CONVOLUTION APPROACH

The noisy output of an LTI system is defined by:

$$y(k) = \sum_{j=0}^{\infty} g(j)u(k-j) + d(k). \tag{4.1}$$

Taking into consideration that $u(k)$ is a random signal and one independent of the zero-mean noise $d(k)$, follows:

$$\begin{aligned} E\{y(k)u(k-h)\} &= \sum_{j=0}^{\infty} g(j)E\{u(k-j)u(k-h)\} + E\{d(k)u(k-h)\}, \\ R_{yu}(h) &= \sum_{j=0}^{\infty} g(j)R_{uu}(h-j) + R_{du}(h), \\ R_{yu}(h) &= \sum_{j=0}^{\infty} g(j)R_{uu}(h-j), \\ R_{yu}(h) &= g(h) * R_{uu}(h). \end{aligned} \tag{4.2}$$

When a random input is applied to the system and the output recorded the impulse response $g(h)$ can be computed using numerical deconvolution since the input $u(k)$ is not white, method which has previously

been mentioned. Under the assumption that $g(j) = 0$ for $j \geq K$, follows:

$$R_{yu}(h) = \sum_{j=0}^{K-1} g(j) R_{uu}(h-j)$$

$$\begin{bmatrix} \hat{R}_{yu}(0) \\ \hat{R}_{yu}(1) \\ \vdots \\ \hat{R}_{yu}(K-1) \end{bmatrix} = \begin{bmatrix} \hat{R}_{uu}(0) & \hat{R}_{uu}(1) & \dots & \hat{R}_{uu}(K-1) \\ \hat{R}_{uu}(1) & \hat{R}_{uu}(2) & \dots & \hat{R}_{uu}(K-2) \\ \vdots & \vdots & & \vdots \\ \hat{R}_{uu}(K-1) & \hat{R}_{uu}(K-2) & \dots & \hat{R}_{uu}(0) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(K-1) \end{bmatrix}. \tag{4.3}$$

4.2.1 CROSS-CORRELATION USING *intcor*

Autocorrelation the PRBS signal generated using the function *intcor* which is given in Appendix A, listing A.1, is shown in figure 4.2. The cross-convolution of the output of the simulated system and the input is shown in figure 4.3. The impulse response generated with convolution approach and numerical deconvolution for $K = 120$ is shown in figure 4.4, where K is a hyperparameter and needs to be chosen by hand. It was observed that for $K > 130$, which corresponds to $t > 65s$, the matrix $\hat{R}_{uu}$ becomes numerically singular, thus K should be chosen as the smallest number of samples it takes before the impulse response of the system settles.

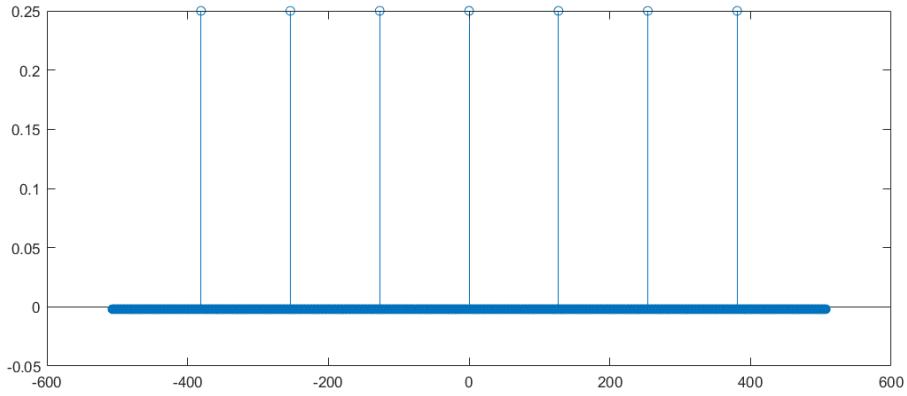

FIGURE 4.2: Autocorrelation of PRBS signal of four periods with seven shift registers

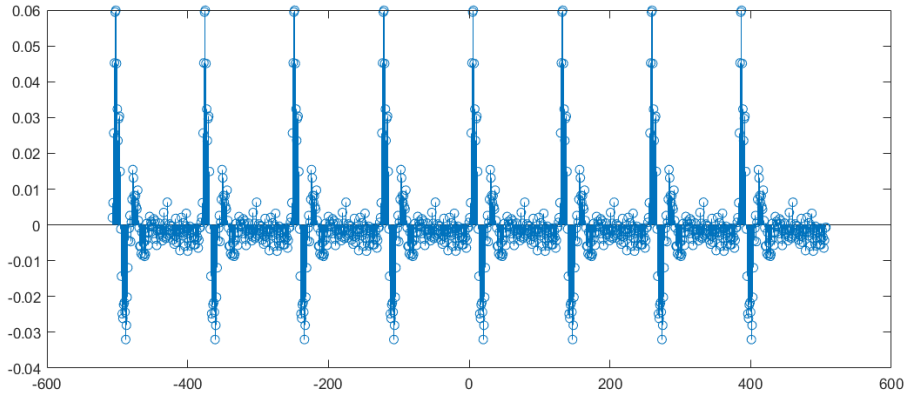

FIGURE 4.3: Cross-correlation of measured output and input

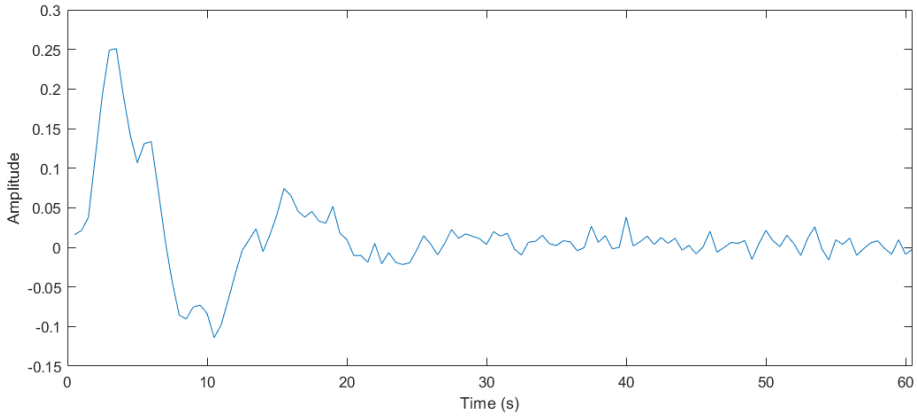

FIGURE 4.4: Impulse response of the system to PRBS signal

4.2.2 CROSS-CORRELATION USING *xcorr*

MATLAB has a function *xcorr* for computation of cross-correlation and autocorrelation. However, it does not produce the same result as the function *intcor*. The results of applying this function on the PRBS signal with optional parameter *biased* is shown with figure 4.5 and with *unbiased* with figure 4.6, when as optional parameter *normalised* is forwarded to the function, the result is normalised such that $R_{uu}(0) = 1$. It is observed that there are significant differences between the biased autocorrelation of PRBS signal in figure 4.5 and that shown in figure 4.2. It is biased because it is not equal to the true value. On the other hand the unbiased estimate as shown in figure 4.6 is a better approximate, but it still has large variance for small $N - h$, because it is based on averaging on a few samples. The *xcorr* function works with a finite number of samples of a signal, even if the signal is periodic, such as the PRBS signal, it assumes that all samples that have been omitted are equal to zero, ie. zero padding, thus the difference between results produced by *intcor* and *xcorr* and furhter difference in the result of convolution approach for approximation of impulse response.

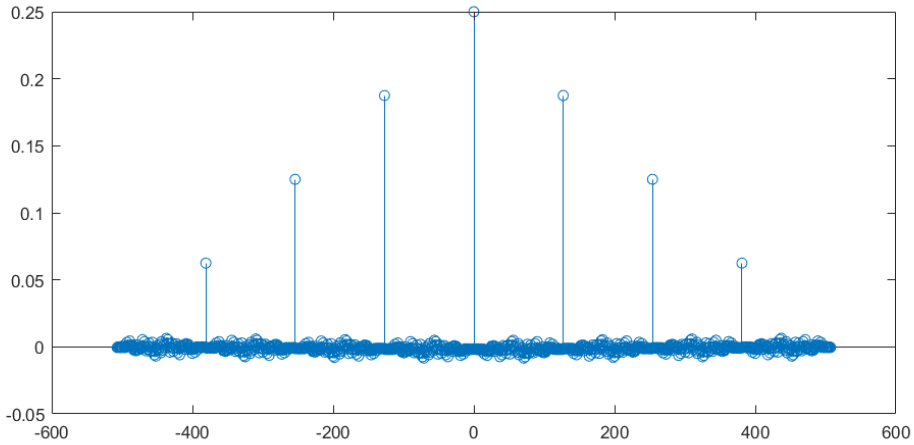

FIGURE 4.5: Biased autocorrelation of PRBS using *xcorr*

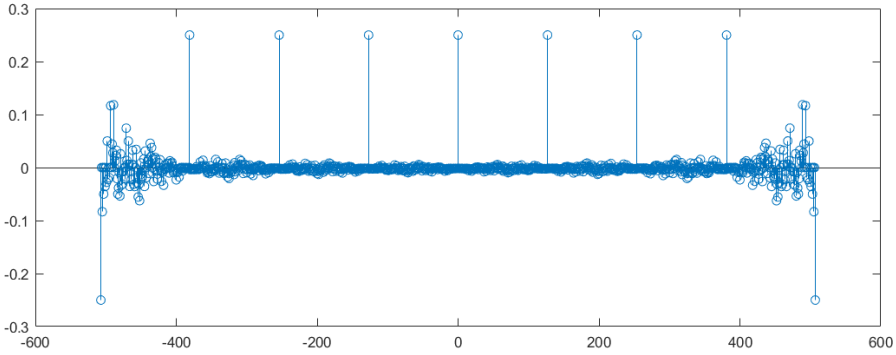

FIGURE 4.6: Unbiased autocorrelation of PRBS using *xcorr*

Similar conclusions can be derived when considering the cross-correlation figures 4.7 and 4.8 generated using the *xcorr* function and figure 4.3 generated using the *intcor* function, which assumes periodicity of the signals.

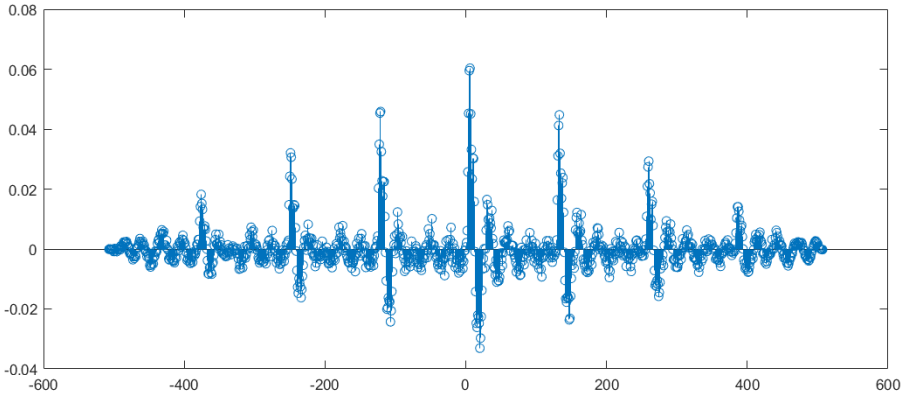

FIGURE 4.7: Biased cross-correlation using *xcorr*

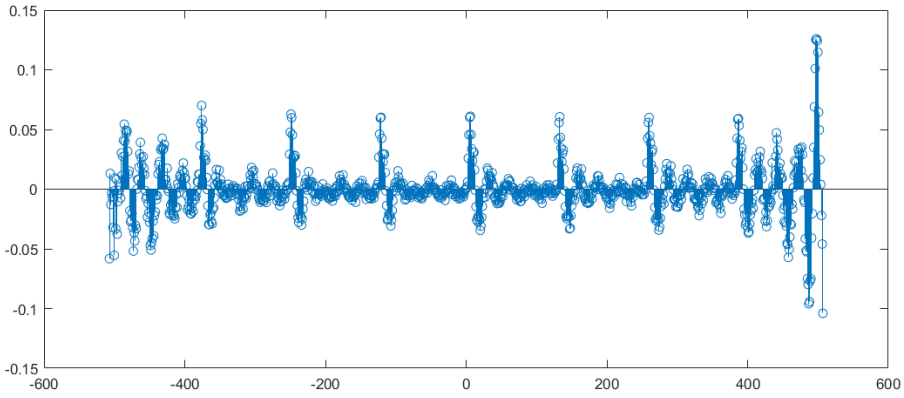

FIGURE 4.8: Unbiased cross-correlation using *xcorr*

4.3 COMPARISON

It can be observed on both figures 4.9 and 4.10 that convolution approach is a good approximation of the impulse response, and unlike methods applied in the previous chapter, it successfully rejects noise, regardless whether there is an assumption on periodicity of the signal or zero padding, biased or unbiased for given $K \leq 125$. Slight differences can be observed between biased and unbiased *xcorr*, where unbiased generates a closer match with impulse response generated with *intcor* and it can be observed that in certain segments they are identical.

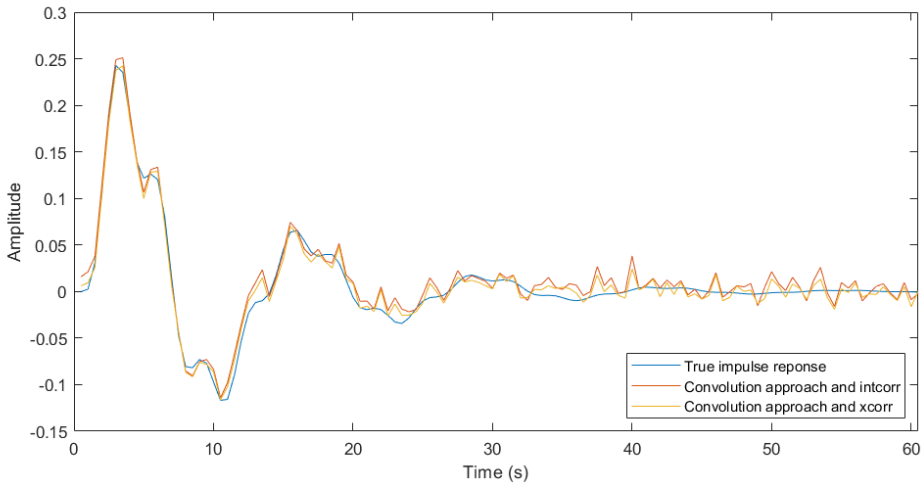

FIGURE 4.9: Comparison of impulse responses, biased *xcorr*

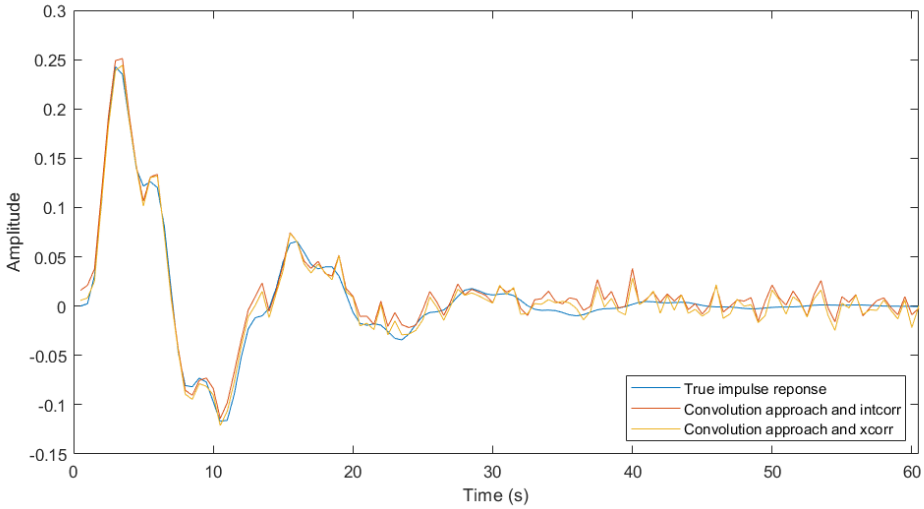

FIGURE 4.10: Comparison of impulse responses, unbiased *xcorr*

The 2-norms of the errors of the impulse response generated using the correlation approach with *unbiased xcorr* and *intcor* functions are given in table 4.1. It is observed that the error depends on the choice of K. For tested $K \leq 120$, the error in case of *unbiased xcorr* function is close to the error in case of *intcor* function, but as K is increased - this is limited by the matrix in expression 4.3 becoming singular, while the error in the case of *intcor* function remains approximately the same as in the previous case, the error

in the case of *ubiased xcorr* function increases five times. The value of K is directly correlated with the sequence of the autocorrelated signal that is used in the correlation approach, this means that as K is increased the sequence starts including parts where the autocorrelation of the signal differs between the two approaches, ie. where the effect of zero-padding in *xcorr* function causes a divergence of the autocorrelation of periodic signals from the real value, the one computed using *intcor*.

TABLE 4.1: Error measurement depending on the choice of K

Function/K	**K = 120**	**K = 125**
intcor	0.1309	0.1305
xcorr	0.1116	0.5387

5. Frequency Domain Identification (Periodic Signal)

In this section, our aim is to identify the system using the Fourier analysis method, exciting the system with a PRBS signal, the advantage of this choice is that the periodicity of the signal allows us to have no truncation error. Moreover, the spectrum of a PRBS is flat for all frequencies different from zero, hence it's good for excitating the system in a large frequency band.

5.1

The first aspect is the generation of the input signal, as stated before we are going to use a PRBS signal, here is the snippet to generate our signal with 9 periods and 8 shift registers and a length $N = 2295$. The system response to this signal is given with Figure 5.1.

Listing 5.1: PRBS generation

```
%% 1 PRBS signal
nprbs = 8;
pprbs = 9;
Uprbs = prbs(nprbs,pprbs);
N = length(Uprbs);
Uprbs = Uprbs/2; % to avoid saturation
% Apply the signal to the simulink system
simin.signals.values = Uprbs;
Time = floor(length(Uprbs))*Ts;
simin.time = (0:Ts:Time -Ts)';
```

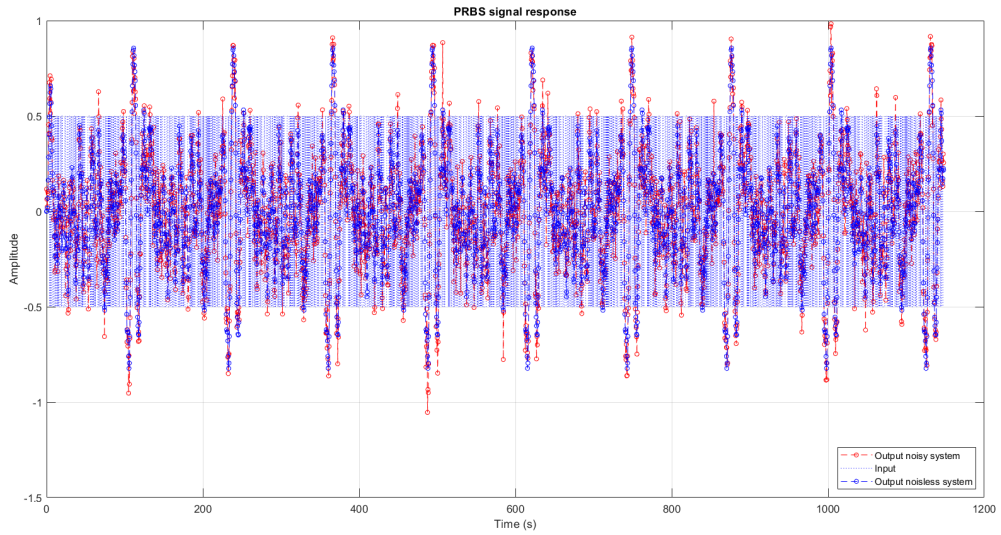

FIGURE 5.1: PRBS response

5.2

A good practice while performing periodic system identification is to have less frequency data with more accuracy. In order to achieve this, the Fourier transform of input and output for each period are computed and the average is taken. The code which executes this is given in the listing below.

LISTING 5.2: FFT

```
%%  2 Fourier transfor of input and output for each period and avg
y = simOut.simout.Data(1:end-2);
y_nonoise = simOut.simout1.Data(1:end-2);
y_res = reshape(y,[],pprbs);
u_res = reshape(Uprbs,[],pprbs);
y_res_nonoise = reshape(y_nonoise,[],pprbs);
u_fft = [];
y_fft = [];
y_nonoise_fft = [];
u_temp = [];
y_temp = [];
y_nonoise_temp = [];

% compute fft for each period
for i = 1: pprbs
    u_temp = fft(u_res(:,i));
    u_fft = cat(2, u_fft, u_temp);
    y_temp = fft(y_res(:,i));
    y_fft = cat(2, y_fft, y_temp);
    y_nonoise_temp = fft(y_res_nonoise(:,i));
    y_nonoise_fft = cat(2,y_nonoise_fft, y_nonoise_temp);
end

```

```
y_fft_avg = mean(y_fft,2)';
u_fft_avg = mean(u_fft,2)';
y_fft_nonoise_avg = mean(y_nonoise_fft,2)';
```

5.3

In order to generate a frequency-domain model we should generate a frequency vector v such as:

$$v = \left[0, \frac{\omega_s}{N}, \frac{2\omega_s}{N}, \ldots, \frac{(N-1)\omega_s}{N}\right], \tag{5.1}$$

where ω_s is the sampling frequency. The code is given in the listing below.

LISTING 5.3: Frequency vector

```
 %% 3 Compute the frequency vector
w_s = 2*pi/Ts;
n = length(u_fft_avg);
v_freq = [0];
v_freq_temp = [];

for j = 1:n-1
    v_freq_temp = j * w_s / n;
    v_freq = cat(2,v_freq,v_freq_temp);
end
```

5.4

Having the frequency vector and the Fourier transform of the input and the output is now possible to construct the frequency domain model using the `frd` command. Particular attention should be given to the dimension of the frequency vector, since the meaningful points for the construction of our model are half of the one we have, that's because for real signals, the `fft` spectrum is a two-sided spectrum, where the spectrum in the positive frequencies is the complex conjugate of the spectrum in the negative frequencies.

LISTING 5.4: Model generation

```
%% 4 Generate the frequency domain vector
G = y_fft_avg ./ u_fft_avg;
G_nonoise = y_fft_nonoise_avg ./ u_fft_avg;
G_model = frd(G(1:n/2),v_freq(1:n/2));
G_model_nonoise = frd(G_nonoise(1:n/2),v_freq(1:n/2));
```

5.5

Finally, we can plot the result of the identification which are given in figure 5.2). It's evident that at high frequencies the model of the noisy system is unreliable, while at low frequencies even in the presence of noise the identified model is very similar to the real noiseless one.

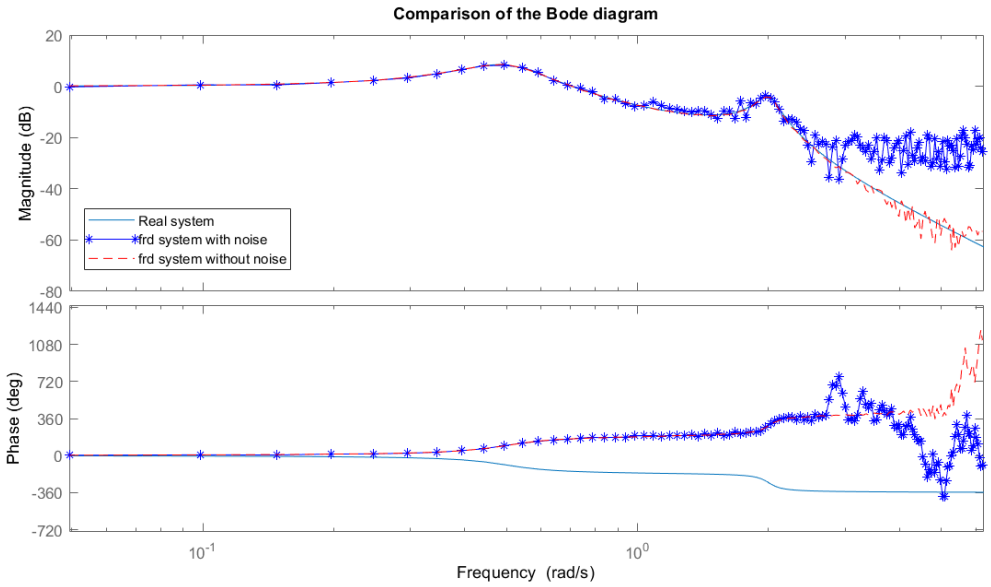

FIGURE 5.2: Bode comparison

6. FREQUENCY DOMAIN IDENTIFICATION (RANDOM SIGNAL)

6.1 SPECTRAL ANALYSIS

Spectral analysis is a method based on correlation approach when the applied input signal is a random signal. Assuming that the system is LTI, it is described by the following expression:

$$y(t) = g(t) * u(t) + d(t). \tag{6.1}$$

Under the assumption that the random signal applied to the system is uncorrelated with the disturbance signal, following relation is satisfied:

$$R_{yu}(h) = g(h) * R_{uu}(h). \tag{6.2}$$

After applying Fourier transform:

$$\Phi_{yu}(\omega) = G(e^{j\omega})\Phi_{uu}(\omega). \tag{6.3}$$

Thus, the frequency response can be obtained by:

$$G(e^{j\omega}) = \frac{\Phi_{yu}(\omega)}{\Phi_{uu}(\omega)}. \tag{6.4}$$

This relation does not depend on the noise, thus it should be possible to identify a better, more accurate model. However, the cross-correlation of the output and input signal and the autocorrelation of the output signal can only be estimated, since the signals are random, and computation of true value up to numerical errors is not possible since it would require an infinite number of samples. Similarly, Fourier transform is also only an estimate contaminated by truncation error and inaccurate estimates of the correlation functions. The estimate of the frequency response is given with relation given below.

$$\hat{G}(e^{j\omega_n}) = \frac{\hat{\Phi}_{yu}(\omega_n)}{\hat{\Phi}_{uu}(\omega_n)} = \frac{\sum_{h=0}^{N-1} \hat{R}_{yu}(h)e^{-jh\frac{2\pi n}{N}}}{\sum_{h=0}^{N-1} \hat{R}_{uu}(h)e^{-jh\frac{2\pi n}{N}}} \tag{6.5}$$

6.1.1 CHOICE OF THE RANDOM INPUT SIGNAL

The random signal that is applied to the system can be binary or multi-level, generated with Gaussian or uniform distribution. Multi-level signals with levels which all have a higher magnitude than the constraint on the input signal will be cut off to the maximum magnitude, effectively a binary signal with magnitude equal to the maximum. If any of the levels applied to the system has a magnitude smaller than the maximum, it will have energy smaller than the energy of a binary signal with maximum magnitude. With this it is concluded that binary signal will have highest energy. Since it is a binary signal, both Gaussian and uniform distribution will yield the same results. Thus binary signal with uniform distribution is considered as the excitation signal with highest energy and applied to the system for frequency domain identification.

6.2 RESULTS

Results of spectral analysis applied to the whole dataset are given in figure 6.1. It is observed that there is noise and significant difference between the identified model and the true model in the higher frequency range. This comes from the measurement error, noise in the output signal, and the previously mentioned truncation error from both the estimation of correlations and Fourier transform. Note that all results correlation functions are computed using *unbiased xcorr*. It is not appropriate to use the *intcor* functions since the chosen excitation signal is not periodic but random. In the case of *biased xcorr* the results are good even without windowing and averaging and applying these methods does not improve accuracy. In order to show the effect of these methods *unbiased xcorr* is used.

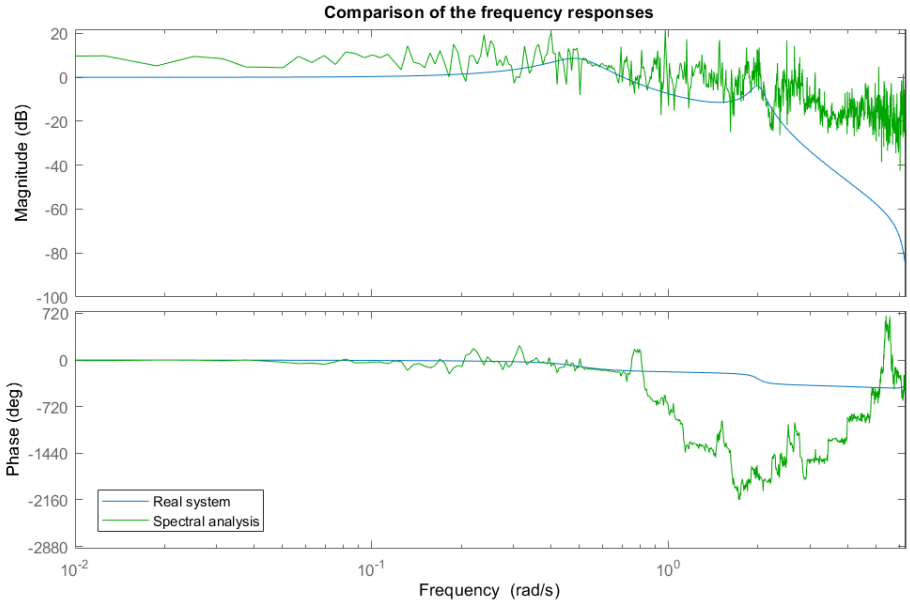

FIGURE 6.1: Comparison of frequency response of the true model and the identified model obtained by spectral analysis

6.2.1 WINDOWING

The large variance for large $|h|$ observed in figure 6.1 can be reduced by windowing. The windowed cross-spectral density function is in this case defined as:

$$\hat{\Phi}_{yu,f}(\omega_n) = \sum_{h=0}^{N-1} \hat{R}_{yu}(h) f(h) e^{-j2\pi nh/N}. \tag{6.6}$$

Windowing applied in the h domain improves the estimate of correlation function, it is smaller than the length of correlation function and padded with zeros to remove the imprecise values of the estimate, caused by computing the correlation of a random signal based on a finite sequence. The applied windows are Hann given with 6.7 and Hamming given with 6.8.

$$f_{hann}(h) = \begin{cases} 0.5 + 0.5\cos \pi h/M, & h = 0, 1..., M \\ 0, & h > M \end{cases}. \tag{6.7}$$

$$f_{hamming}(h) = \begin{cases} 0.54 + 0.46\cos \pi h/M, & h = 0, 1..., M \\ 0, & h > M \end{cases}. \tag{6.8}$$

The effect of Hann window on the cross-spectral density function is given with figure 6.2. The value of M is chosen based on the correlation functions, such that it keeps the significant part of the estimates and removes imprecise parts, windowing should not cut information on the estimate of the correlation functions. According to this $M = 125$.

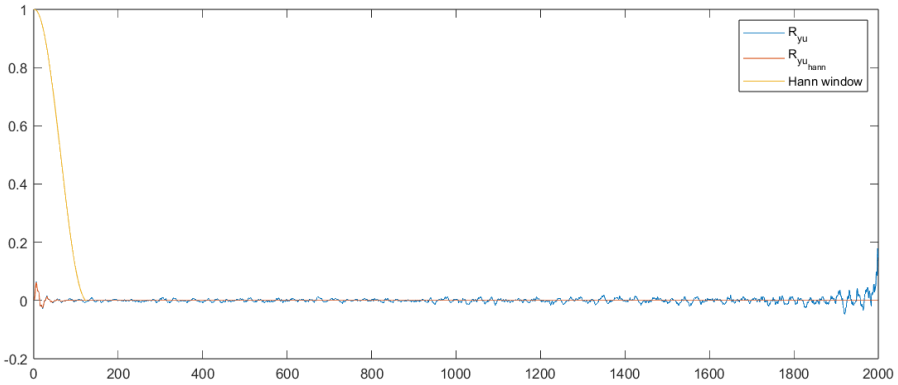

FIGURE 6.2: Cross-spectral density function before and after applying Hann window

It can be observed in figure 6.3 that windowing improves the identified model in the higher frequency, even though the noise persist. Additionally, almost no difference is observed between applying Hann and Hamming window.

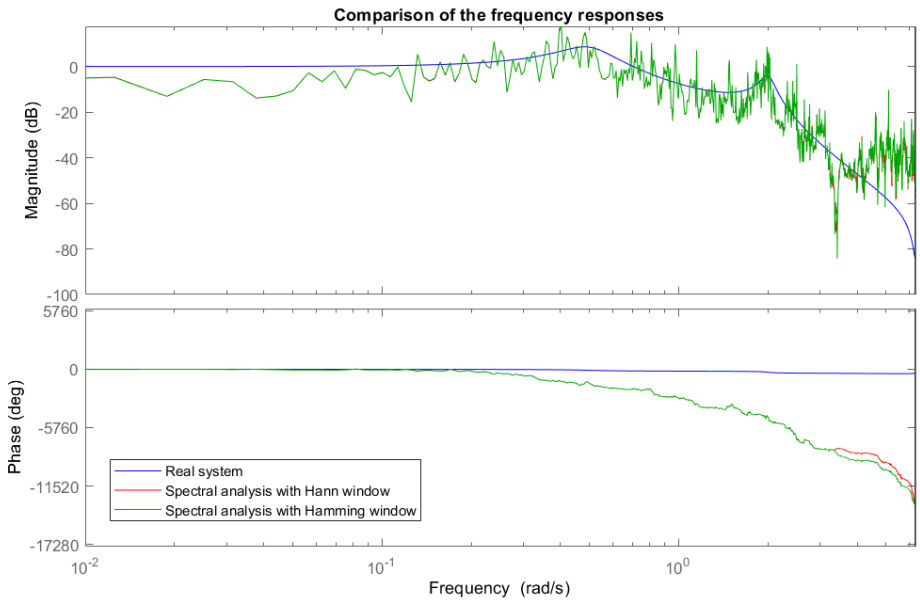

FIGURE 6.3: Comparison of frequency response of the true model and the identified model obtained by spectral analysis with windowing

6.2.2 AVERAGING

Unbiased estimate can be improved by averaging, as it reduces the variance of estimates while preserving unbiasedness. For m sets of input/output the unbiased estimate of spectral density functions are improved by averaging:

$$\begin{aligned}\overline{\Phi}_{uu}(\omega_n) &= \frac{1}{m}\sum_{i=1}^{m}\Phi_{uu,i}(\omega_n),\\ \overline{\Phi}_{yu}(\omega_n) &= \frac{1}{m}\sum_{i=1}^{m}\Phi_{yu,i}(\omega_n),\\ G(e^{j\omega_n}) &= \frac{\overline{\Phi}_{uu}(\omega_n)}{\overline{\Phi}_{yu}(\omega_n)}.\end{aligned} \tag{6.9}$$

The results of averaging after dividing the original set in $m = 10$ sets of input/output data are given with figure 6.4. It is observed that there is less variance across all frequencies and the system model in the mid-frequencies is improved, however there is still a significant error in the higher frequency range.

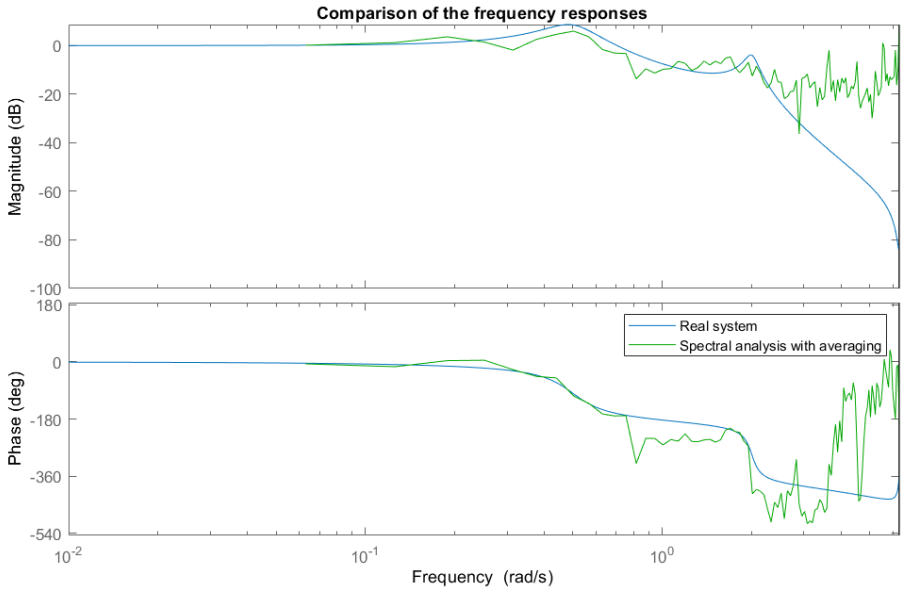

FIGURE 6.4: Comparison of frequency response of the true model and the identified model obtained by spectral analysis with averaging

6.2.3 COMPARISON

Results of spectral analysis are compared with the true frequency model in the figure 6.5. It can be observed that windowing and averaging both improve the frequency response of the identified model in different aspects. Averaging reduces overall noise, while windowing improves estimates of cross-spectral density for higher frequencies and thus the identified model, and follows the true model the best. Both averaging and windowing improve the model, and would seem reasonable to apply both windowing and averaging to achieve the best model with spectral analysis. The code is given in Appendix, A.4.

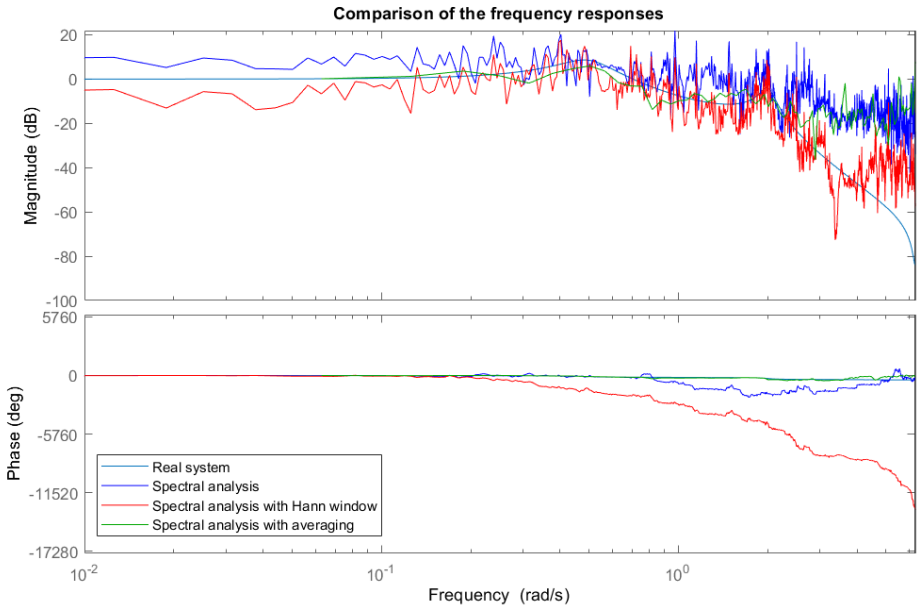

FIGURE 6.5: Comparison of frequency response of the true model and the identified models

A. APPENDIX

LISTING A.1: Cross correlation function for periodic signals

```
function [R,h] = intcor(u,y)
    % Cross correlation function for periodic signals u and y


    lenu = length(u);
    leny = length(y);
    M = lcm(lenu, leny); % least common multiplier - the common
       period of signals u and y

    % Make the signals compatible, not assuming zero padding
    u = repmat(u, M/lenu);
    y = repmat(y, M/leny);

    R = zeros(2*M-1,1);

    % Compute cross correlation of periodic signals
    for h = -(M-1):1:(M-1)
        newY = circshift(y, h); % Shifting of the signal, not
           assuming zero padding
        for k = 1:1:M
            R(h+M, :) = R(h+M, :) + 1/M*u(k,:)*newY(k,:);
        end
    end

    h = -(M-1):1:(M-1);
end
```

LISTING A.2: Autocorrelation of PRBS signal

```
%% Autocorrelation of a PRBS signal
x = prbs(4,2); % PRBS signal, four shift registers, two periods
[R,h] = intcor(x,x); % Autocorrelation

%% Plot figures
figure;
stem(1:1:30, x);
title('Two periods of PRBS with four shift registers');
ylim([-1.2,1.2]);
figure;
stem(h,R);
```

```
12 title('Autocorrelation of PRBS');
13 ylim([-0.2,1.2]);
```

LISTING A.3: Impulse response by correlation approach

```
1  clear all
2  close all
3  clc
4
5  % Parameters
6  num = 1;
7  den = [1 0.4 4.3 0.85 1];
8  Ts = 0.5;
9  var = 0.01;
10 uplimit = 0.5;
11 lowlimit = -0.5;
12
13 %% 1.4.1 Generate an input sequence Uprbs
14 p = 4;
15 n = 7;
16 Uprbs=prbs(n,p);
17 Uprbs = Uprbs/2; % to avoid saturation
18
19 % Apply the signal to the simulink system
20 simin.signals.values = Uprbs;
21 Time = floor(length(Uprbs))*Ts;
22 simin.time = (0:Ts:Time - Ts)';
23
24 % Simulate the system with the PRBS signal
25
26 simOut= sim("CE114simulink_22b.slx");
27 plot(simOut.simout, '--ro', 'MarkerSize', 4, 'DisplayName', 'Output
      noisy system');
28 hold on
29 plot(simin.time, simin.signals.values, ':', 'Color',
      'blue','MarkerSize', 5, 'DisplayName', 'Input')
30 xlabel('Time (s)')
31 ylabel('Amplitude')
32 legend('Location','southeast')
33 title('PRBS signal response')
34 xlim([0 simin.time(end)])
35 grid on
36
37 %% 1.4.2 compute the impulse response of the discrete-time system
      g(k) using the correlation approach
38
39 K = 120;
40 N = length(Uprbs);
41 y = simOut.simout.Data(1:N); % Simulate response of the system
42 y_nonoise = simOut.simout1.Data(1:N);
```

```
43
44 [Corr_uu,h_uu] = intcor(Uprbs,Uprbs); % Autocorrelation of input, Ruu
45 [Corr_yu, h_yu] = intcor(y,Uprbs); % Cross correlation of output and
      input, Ryu
46 [Corr_yu_nonoise, h_yu_nonoise] = intcor(y_nonoise,Uprbs); % Cross
      correlation of noiseless output and input, Ryu
47
48 figure
49 stem(h_uu,Corr_uu)
50 figure
51 stem(h_yu,Corr_yu)
52
53 Corr_N = length(Corr_yu);
54
55 idx = find(h_uu == 0); % Find Ruu(0)
56
57 % Impulse respinse using numerical deconvolution, since input is not
      white,
58 % it is assumed that g(j) = 0 for j >= K
59 r = Corr_uu(idx:idx+K)';
60 R = toeplitz(r);
61 G = R\Corr_yu(idx:idx+K);
62
63 % Plot
64 figure;
65 plot([1:size(G)]*Ts,G);
66 xlabel('Time (s)')
67 ylabel('Amplitude')
68 xlim([0 size(G,1)*Ts]);
69
70 %% 1.4.3 Compute the impulse response using xcorr function of Matlab
71
72 [xCorr_uu, xh_uu] = xcorr(Uprbs,Uprbs, 'unbiased'); %
      Autocorrelation of input, Ruu
73 [xCorr_yu, xh_yu] = xcorr(y,Uprbs, 'unbiased'); % Crosscorrelation
      of output and input, Ryu
74 [xCorr_yu_nonoise, xh_yu_nonoise] = xcorr(y_nonoise, Uprbs,
      'unbiased'); % Crosscorrelation of noiseless output and input, Ryu
75
76 figure;
77 stem(xh_uu, xCorr_uu);
78 figure;
79 stem(xh_yu, xCorr_yu);
80
81 idx = find(xh_uu == 0); % Find Ruu(0)
82
83 % Impulse respinse using numerical deconvolution, since input is not
      white,
84 % it is assumed that g(j) = 0 for j >= K
```

```
r = xCorr_uu(idx:idx+K)';
R = toeplitz(r);
xG = R\xCorr_yu(idx:idx+K);

% Plot
figure;
plot([1:size(xG)]*Ts,xG);
xlabel('Time (s)')
ylabel('Amplitude')
xlim([0 size(xG,1)*Ts]);

%% 1.4.4 Comparison

t=0:Ts:(N-1)*Ts;
Gdis = c2d(tf(num,den),Ts,'zoh');
True_output = impulse(Gdis, t) * (Ts); % True impulse response

% Plot
figure;
plot([1:size(G)]*Ts,True_output(1:size(G)), 'DisplayName', 'True
    impulse reponse')
hold on
plot([1:size(G)]*Ts, G, 'DisplayName', 'Convolution approach and
    intcorr')
hold on
plot([1:size(xG)]*Ts, xG, 'DisplayName', 'Convolution approach and
    xcorr')
legend('Location', 'southeast')
xlabel('Time (s)')
ylabel('Amplitude')
xlim([0 size(G,1)*Ts]);

% 2-norm of the errors
intcor_error = norm(True_output(1:size(G)) - G);
xcorr_error = norm(True_output(1:size(G)) - xG(1:size(G)));
```

LISTING A.4: Spectral analysis

```
clear all
close all
clc

% Parameters
num = 1;
den = [1 0.4 4.3 0.85 1];

Ts = 0.5;

G_real = tf(num,den);
G_real = c2d(G_real,Ts, "zoh");
```

```
13
14 var = 0.01;
15 uplimit = 0.5;
16 lowlimit = -0.5;
17
18 %% Part 1 apply a random signal
19
20 N = 2000;
21 Time = N*Ts;
22 Discretize_Time = floor(Time/Ts);
23
24 randomsig = randi([0 1],N, 1) - 0.5; % Random signal that has the
      highest excitation of binary with uniform distribution
25 u = randomsig;
26
27 simin.signals.values = u;
28 simin.time = transpose(linspace(0,Time,Discretize_Time));
29
30 % Simulate and get the output plot
31 figure;
32 simOut= sim("CE116simulink_22b.slx");
33 plot(simOut.simout, '--ro', 'MarkerSize', 4);
34 hold on
35 plot(simin.time, simin.signals.values, ':', 'Color',
      'blue','MarkerSize', 5 )
36 plot(simOut.simout1, '--bo', 'MarkerSize', 4);
37 hold on
38 plot(simin.time, simin.signals.values, ':', 'Color',
      'blue','MarkerSize', 5 )
39 xlabel('Time (s)')
40 ylabel('Amplitude')
41 legend({'Output noisy system','Input', 'Output noisless
      system'},'Location','southeast')
42 title('Random signal response')
43 grid on
44 %% Part 2 Compute the frequency-response of the model using the
      spectral analysis method
45
46 y = simOut.simout.Data(1:end-1);
47 y_nonoise = simOut.simout1.Data(1:end-1);
48
49 % random input signal is not periodic thus intcor function can not
      be used
50 % so unbiased xcorr function is used
51
52 % Computing autocorrelation and cross correlation using unbiased
      xcorr
53 [Ryu, hyu] = xcorr(y, u, "unbiased");
54 [Ruu, huu] = xcorr(u, u, "unbiased");
```

```
55
56  idx_yu = find(hyu == 0); % Find Ryu(0)
57  idx_uu = find(huu == 0); % Find Ruu(0)
58
59  % Estimate the Fourier Transform
60  Ryu_fft = fft(Ryu(idx_yu:end));
61  Ruu_fft = fft(Ruu(idx_uu:end));
62
63  % Compute the frequency vector
64  N = length(Ryu_fft);
65  ws = 2*pi/Ts;
66  freq = 0:ws/N:(N-1)*ws/N;
67
68  % Compute G
69  G = Ryu_fft ./ Ruu_fft;
70  G_model = frd(G, freq, Ts);
71
72  %% Part 3 Improve results by Windowing
73
74  M = 125;
75
76  % Apply Hann Window
77  hann_window = hann(2*M);
78  Ryu_hann = zeros(size(Ryu(idx_yu:end)));
79  Ryu_hann(1:1+M) = Ryu(idx_yu:idx_yu+M) .* hann_window(M:end);
80
81  Ryu_hann_fft = fft(Ryu_hann);
82  Ruu_hann_fft = Ruu_fft;
83
84  G = Ryu_hann_fft ./ Ruu_hann_fft;
85  G_hann_model = frd(G, freq, Ts);
86
87  figure
88  plot(hyu(idx_yu:end),Ryu(idx_yu:end), 'DisplayName', 'R_{yu}')
89  hold on
90  plot(hyu(idx_yu:end),Ryu_hann, 'DisplayName', 'R_{yu_{hann}}')
91  plot(1:1:M+1, hann_window(M:end), 'DisplayName', 'Hann window')
92  legend
93
94  % Apply Hamming Window
95  M = 125;
96  hamming_window = hamming(2*M);
97  Ryu_hamming = zeros(size(Ryu(idx_yu:end)));
98  Ryu_hamming(1:1+M) = Ryu(idx_yu:idx_yu+M) .* hamming_window(M:end);
99
100 Ryu_hamming_fft = fft(Ryu_hamming);
101 Ruu_hamming_fft = Ruu_fft;
102
103 G = Ryu_hamming_fft ./ Ruu_hamming_fft;
```

```
104 G_hamming_model = frd(G, freq, Ts);
105
106
107 %% Part 4 Improve results by averaging
108
109 m = 10;
110
111 exp_len = N/m;
112 average_Ryu_fft = zeros(exp_len,1);
113 average_Ruu_fft = zeros(exp_len,1);
114
115
116 for i = 1:m
117     % Dividing simulated data into m different experiments
118     exp_u = u((i-1)*exp_len + 1:i*exp_len);
119     exp_y = y((i-1)*exp_len + 1:i*exp_len);
120
121     % Computing autocorrelation and cross correlation using unbiased
          xcorr
122     [exp_Ryu, exp_hyu] = xcorr(exp_y, exp_u, "unbiased");
123     [exp_Ruu, exp_huu] = xcorr(exp_u, exp_u, "unbiased");
124
125     idx_yu = find(exp_hyu == 0); % Find Ryu(0)
126     idx_uu = find(exp_huu == 0); % Find Ruu(0)
127
128     % Compute Fourier Transform
129     exp_Ryu_fft = fft(exp_Ryu(idx_yu:end));
130     exp_Ruu_fft = fft(exp_Ruu(idx_uu:end));
131
132     % Compute average of Fourier Transform
133     average_Ryu_fft = average_Ryu_fft + 1/m *exp_Ryu_fft;
134     average_Ruu_fft = average_Ruu_fft + 1/m*exp_Ruu_fft;
135 end
136
137 % Compute frequency vector
138 N = exp_len;
139 ws = 2*pi/Ts;
140 exp_freq = 0:ws/N:(N-1)*ws/N;
141
142 % Compute G
143 G = average_Ryu_fft ./ average_Ruu_fft;
144 G_average_model = frd(G, exp_freq, Ts);
145
146 %% Part 5 Comparison
147
148 figure
149 bode(G_real, G_model, 'g', freq);
150 legend('Real system', 'Spectral analysis')
151 title('Comparison of the frequency responses')
```

```
152 xlim ([10^-2, (2*pi/Ts)*0.5]);
153
154 figure
155 bode(G_real,'b', G_hann_model, 'r', G_hamming_model, 'g', freq);
156 legend('Real system', 'Spectral analysis with Hann window',
       'Spectral analysis with Hamming window')
157 title('Comparison of the frequency responses')
158 xlim ([10^-2, (2*pi/Ts)*0.5]);
159
160
161 figure
162 bode(G_real, freq)
163 hold on
164 bode(G_average_model, 'g', exp_freq);
165 legend('Real system', 'Spectral analysis with averaging')
166 title('Comparison of the frequency responses')
167 xlim ([10^-2, (2*pi/Ts)*0.5]);
168
169
170 figure
171 bode(G_real, G_model,'b', G_hann_model, 'r', freq);
172 hold on
173 bode(G_average_model, 'g', exp_freq);
174 legend('Real system', 'Spectral analysis', 'Spectral analysis with
       Hann window', 'Spectral analysis with averaging')
175 title('Comparison of the frequency responses')
176 xlim ([10^-2, (2*pi/Ts)*0.5]);
```