Michael Biselx (283812)
Grégoire Lacroix (288605)
Xavier Nal (288275)

ME-421 : System Identification

Computer Exercice 1

Goal

The first computer exercise is meant to demonstrate the use of non-parametric methods for identifying system models.

Exercise 1 : Step response

> *Give the plot of the noiseless unit step/impulse response superimposed on the noisy unit step/impulse response.*

The sampling time is $T_e = 0.1\ s$. This is a valid choice according to the Shannon theorem, as the frequency bandwidth of the system is $f_b = 0.59\ Hz$, and therefore the sampling frequency of $f_s = 1/T_e = 10\ Hz$ is more than sufficient to reconstruct a signal from this system. Indeed, the sampling frequency is almost 20 times larger than the bandwidth of the signal, allowing the system to be identified with much greater precision.

Noise in the Simulink model is simulated using a Random Number block with a variance of 0.01 and a sampling time T_e.

The figures 1 and 2 below show the step and impulse responses of the simulated system over a period of 10 seconds. The curve *y* is the output with noise and the curve *y_clean* is without noise. As we can see, the curve with noise is more difficult to read. While the step response of the system is still visible, the impulse response is completely drowned out by the noise.

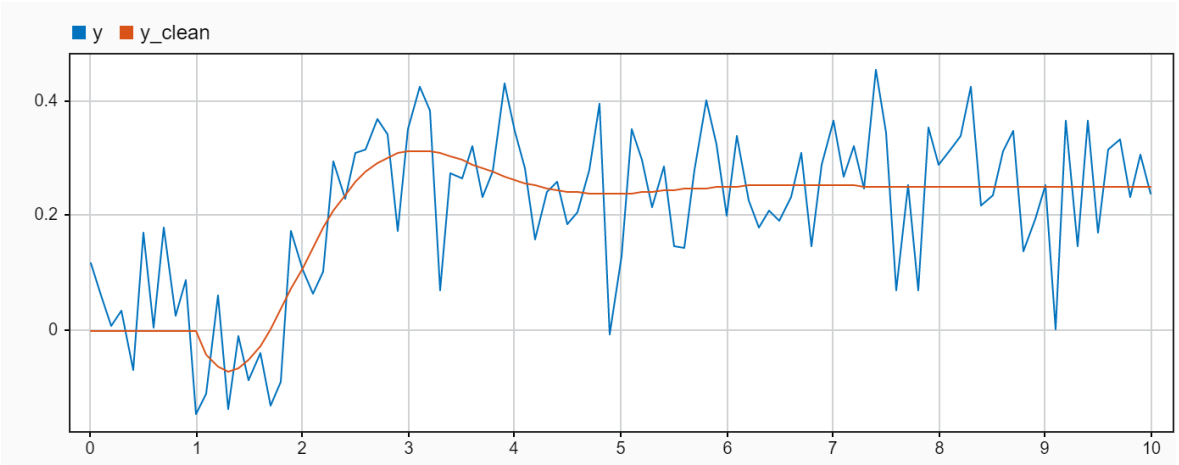

Fig. 1 : The step response of the the system with and without noise

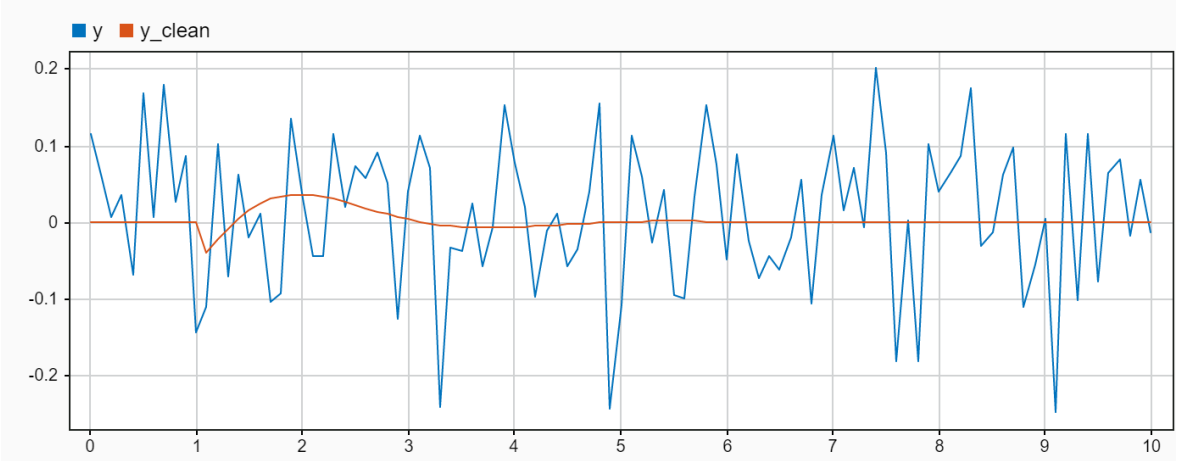

Fig. 2 : The impulse response of the system with and without noise

Exercise 2 : Auto Correlation of a PRBS signal

> *Give the code for the `intcor` function.*
> *Plot the result of the autocorrelation function for `prbs(6,4)`*

The code of the `intcor` function is as follows :

```
function [R, h] = intcor(u, y)

    N = length(u);
    U = fft(u); % don't zero-pad signals ! PRBS is cyclic
    Y = fft(y);

    R = fftshift(ifft(U .* conj(Y))) / N;

    if ~mod(N,2)
        h = linspace(-N/2, N/2-1, length(U));
    else
        h = linspace(1-ceil(N/2), floor(N/2), length(U));
    end
end
```

The figure 3 shows exactly the autocorrelation function of four periods of the PRBS signal with a shift register of length 6. Peaks appear with a period of $M = 2^6 - 1 = 63$ *s* and an amplitude of exactly *1*. Elsewhere the autocorrelation values are all equal to $-M^{-1} = -0.0156$

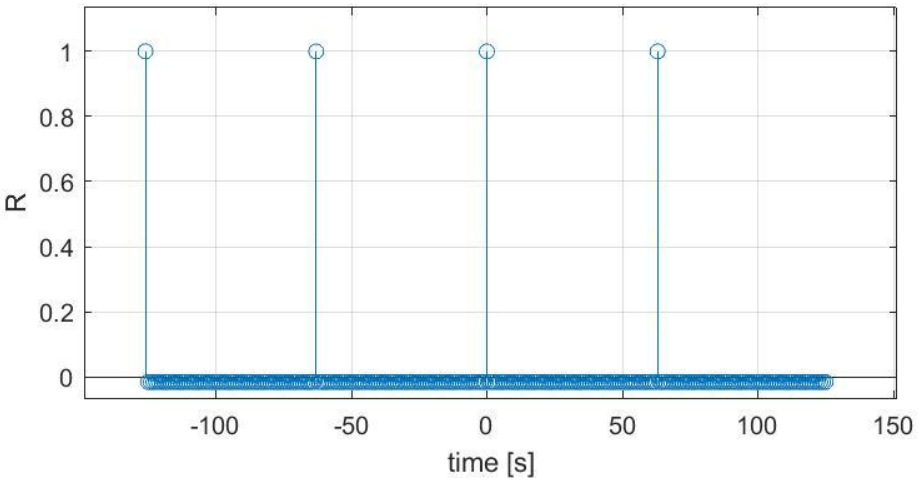

Fig. 3 : autocorrelation of the `prbs(6,4)` signal

Exercise 3 : Impulse response by deconvolution method

Give the code for computing the impulse response by the deconvolution algorithm.
Compare the plot of the identified impulse response with the true impulse response, using two different algorithms (finite impulse response and regularization).
Give the 2-norm of the error.

The deconvolution algorithms can be applied using the following code :

```
% setup input
u       = u_sat - 2*u_sat*rand(size(time));
U       = toeplitz(u, [u(1), zeros(1, length(time)-1)]);

% simulate system
simin  = timetable(time, u);
simout = sim('model1');

% finite impulse response deconvolution :
K       = 200;
Uk      = U(:, 1:K);
impulse_deconv1 = Uk \ simout.y.Data;

% regularization deconvolution :
lambda = 10;
impulse_deconv2 = (U.'*U + lambda*eye(size(U))) \ (U.' *
simout.y.Data);
```

The regularization method results in a much longer signal than the finite response, so, for a fair comparison, the regularization method result is truncated to the length *K*. For the chosen parameters of $K = 100$ and $\lambda = 1.2$, the 2-norm of the error for the finite response deconvolution is 0.08, and for the regularization deconvolution it is 0.13, so quite a bit worse (as can be seen in fig. 4).

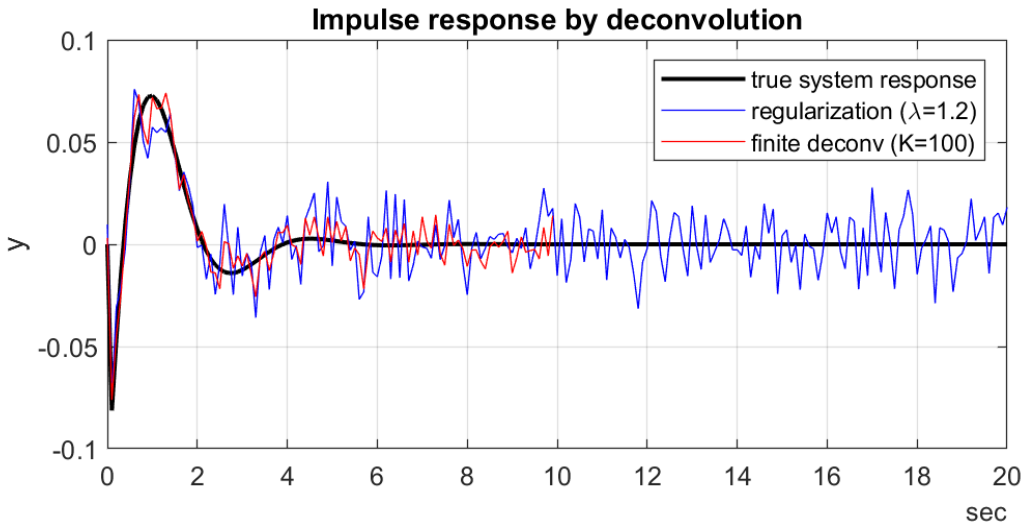

Fig. 4 : The identified impulse responses using the numerical deconvolution methods.

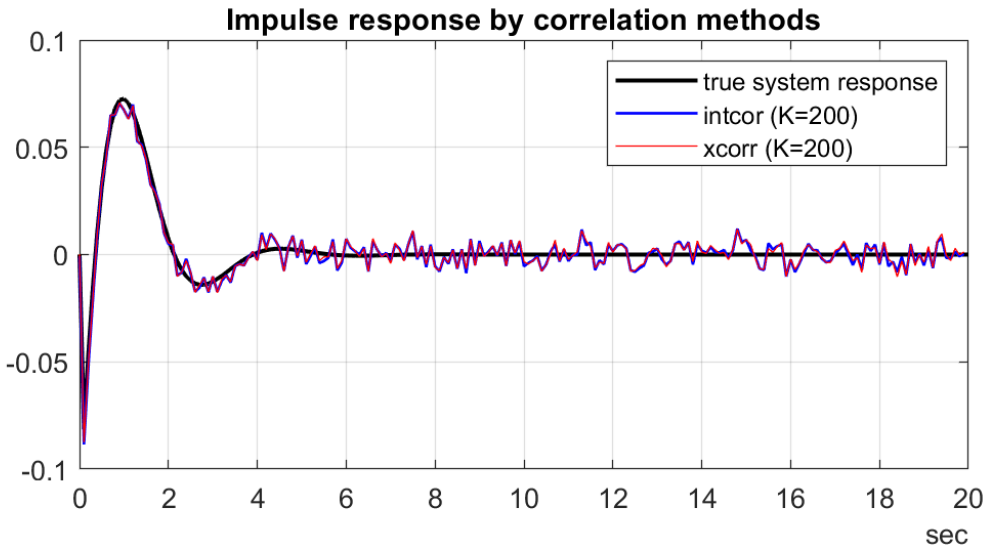

Fig. 5 : impulse responses identified using correlation methods. The result is quite a bit better than the one found using numerical deconvolution.

Exercise 4 : Impulse response by correlation approach

Give the code for computing the impulse response using intcor. What about the code when xcorr is used?
Show a plot of the identified impulse responses compared with the true one, and give the 2-norm of the errors.
Comment on the different methods based on the 2-norm of the errors.

The code for computing the impulse response using `intcor` and `xcorr` is very similar. It is as follows :

```
% compute impulse response using intcor
[Ryu_i, ~  ] = intcor(simout.y.Data, u);
[Ruu_i, h_i] = intcor(u, u);

idx_i  = find(h_i>=0, 1) : find(h_i>=(K-1),1);
RUUk_i = toeplitz(Ruu_i(idx_i), Ruu_i(idx_i).');
impulse_deconv_intcor = RUUk_i \ Ryu_i(idx_i);

% compute impulse response using xcorr
[Ryu_x, ~  ] = xcorr(simout.y.Data, u, 'biased');
[Ruu_x, h_x] = xcorr(u, u, 'biased');

idx_x  = find(h_x>=0, 1) : find(h_x>=(K-1),1);
RUUk_x = toeplitz(Ruu_x(idx_x), Ruu_x(idx_x).');
impulse_deconv_xcorr = RUUk_x \ Ryu_x(idx_x);
```

The impulse responses are identified in figure 5. The parameter K can be arbitrarily chosen, as long as it is less than the length of repetition of the PRBS signal. If K is greater than the repetition length of the PRBS (which is 255), `intcor` gets confused and returns nonsense. However, since `xcorr` zero-pads the signal, this is not an issue, and the PRBS is not treated as a repeating signal.
For a K = 200 < `len(prbs)`, both methods give an extremely similar 2-norm error of 0.066 (`intcor`) and 0.065 (`xcorr`). For K = 300 > `len(prbs)`, the `xcorr` deconvolution gives an error of 0.09, while the `intcor` deconvolution gives an error of 0.32, which is huge.

Exercise 5 : Frequency domain Identification (Periodic signal)

Give the code to identify the frequency-domain model using Fourier Analysis on a `prbs`*-excited system.*
Plot the Bode diagram of the identified model, and compare it to the true one.

The model can be identified using Fourier Analysis with the following code, where we average over `P = 8` periods :

```
u = u_sat * prbs(8, P);                          % PRBS input signal
N = length(u)/P;

[...]                                            % do simulation

FU = mean(fft(reshape(u, N, P), [], 1), 2); % fft of input signal
FY = mean(fft(reshape(y, N, P), [], 1), 2); % fft of output signal

FR = FY./FU;                                     % response

f = linspace(0, 2*pi*(N-1)/N/Te, N);             % frequency vector

freq_model = frd(FR, f);                         % frequency-domain model
```

The figure 6 shows the frequency-domain model which was identified using the Fourier Analysis of a PRBS-excited system output. While a little noisy at higher frequencies, it is in fact a fairly good model of the system.

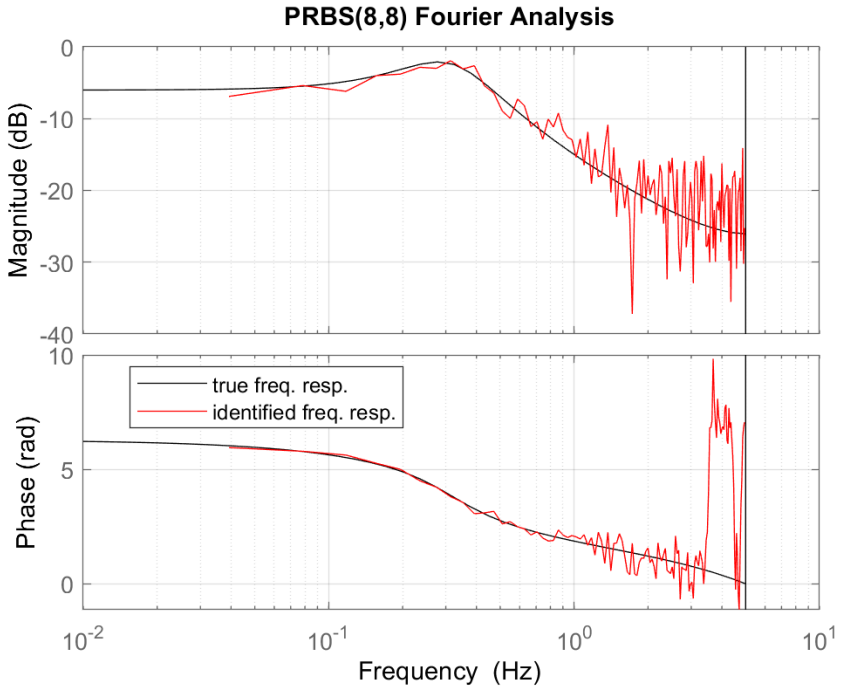

Fig. 6 : frequency-domain model identified using a `prbs` *signal and Fourier analysis*

Exercise 6 : Frequency domain Identification (Random signal)

Give the code to identify the frequency-domain model using Fourier Analysis on a random signal-excited system.
Show plots of the identified models using the methods :
- *Truncation (no window)*
- *Hann or Hamming window*
- *Averaging*

Compare the identified models to the true one.

The code to identify the frequency-domain model using spectral analysis on a random signal excited-system is as follows :

```
u = randi([0,2*u_sat], [N,1]) -u_sat;           % random input
signal

[...]                                           % do simulation

% Frequency response using truncated spectral analysis
[PSD_U, f] = intpsd(u, u, [], Te);
[PSD_Y, ~] = intpsd(y, u, [], Te);

G1 = PSD_Y./PSD_U;
freq_model1 = frd(G1(1:floor(N/2)), f(1:floor(N/2)));

% Frequency response using windowed spectral analysis
win = hann(N);                                  % window to use
[PSD_UW, f] = intpsd(u, u, win, Te);
[PSD_YW, ~] = intpsd(y, u, win, Te);

G2 = PSD_YW./PSD_UW;
freq_model2 = frd(G2(1:floor(N/2)), f(1:floor(N/2)));

% Frequency response using truncated averaged spectral analysis
M = 10;                                         % number of epochs
uu = reshape(u, [], M);
yy = reshape(y, [], M);
[PSD_UU, f] = intpsd(uu, uu, [], Te);
[PSD_YY, ~] = intpsd(yy, uu, [], Te);
PSD_UM = mean(PSD_UU, 2);                       % average the inputs
PSD_YM = mean(PSD_YY, 2);                       % average the outputs

G3 = PSD_YM./PSD_UM;
freq_model3 = frd(G3(1:floor(N/M/2)), f(1:floor(N/M/2)));
```

`intpsd` is the inter-correlation power spectral density calculated according to `intcor` :

```
function [S, f] = intpsd(u, y, win, Ts)
    N = length(u);
    U = fft(u);                              % don't zero-pad signals
    Y = fft(y);

    if isempty(win)
        S = fft(u) .* conj(fft(y))/N;    % fast psd without
windowing
    else
        R = intcor(u, y);                % re-use intcor
        S = fft(ifftshift(win.*R));      % psd of windowed intcor
    end

    f = linspace(0, 2*pi*(N-1)/N/Ts, N);
end
```

Comparison of the identified models (figure 7) reveals that simply truncating the data is (unsurprisingly) not a very good method for identifying the system. The spectral leakage is quite large and there is a lot of noise. Using a Hann window does reduce the leakage to the point where we can clearly identify the resonant mode of the system, although it's still not great at higher frequencies. Averaging over several epochs (in this example: 10 epochs), however, seems to noticeably improve the quality of the obtained model - especially concerning the phase, which becomes continuous.
In figure 8, we compare the impulse responses found by the various methods, and see that the results are fairly decent for the windowed and averaged models.

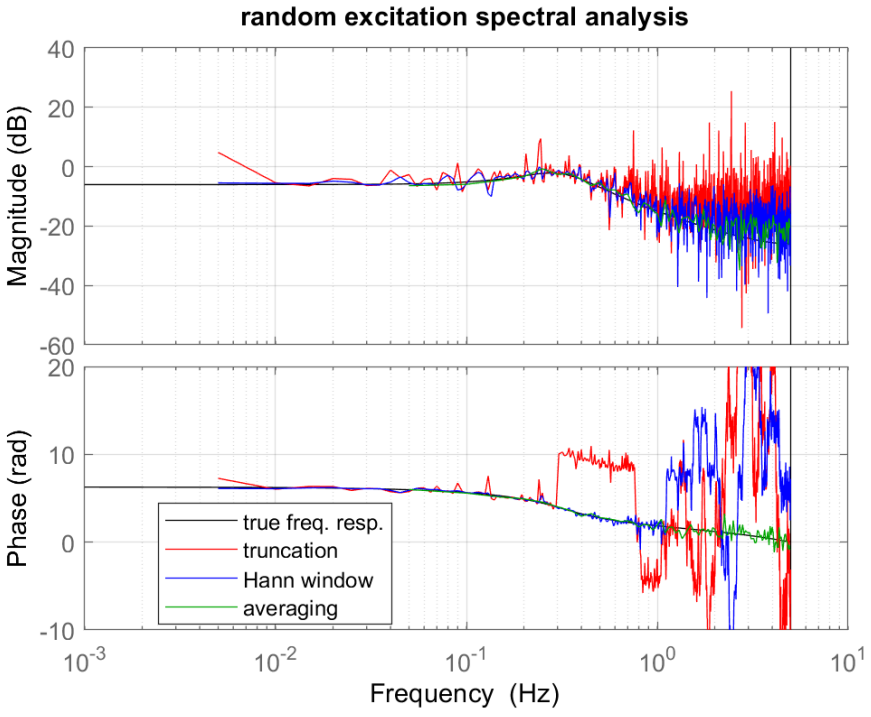

Fig. 7 : the different frequency-domain models identified using spectral analysis.

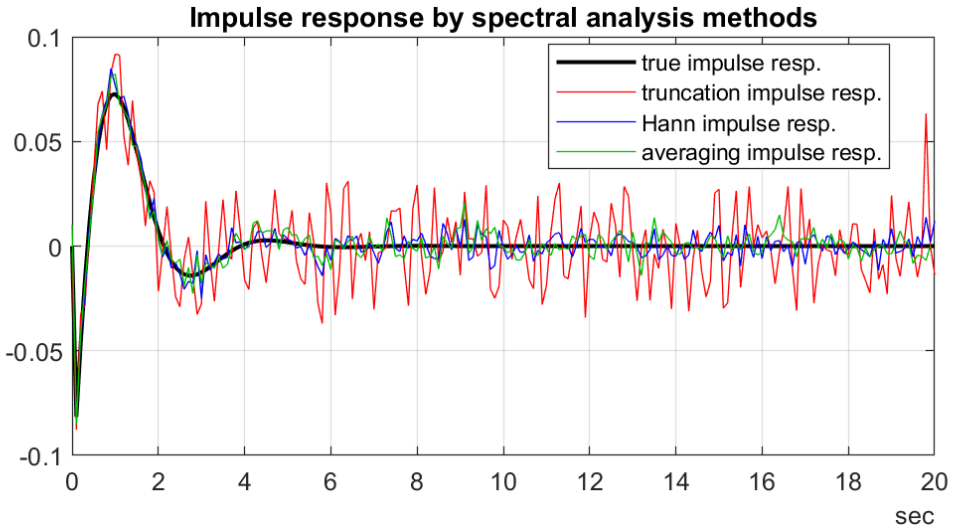

Fig. 8 : the impulse responses of the identified frequency-domain models.