System Identification
CE-1: Time-domain methods

Camilla Carta
Michael Spieler

October 23, 2017

1 Step and impulse response

In order to analyze the dynamic characteristics of a system, a second order transfer function (see Equation 1) was given to be simulated with Matlab environment Simulink. Moreover, it had to follow these parameters:

- Sample time: $Te = 0.2s$;
- Measurement noise: $\mu = 0$ and $\sigma^2 = 0.1$;
- Saturation: $-0.5 < INPUT < 0.5$.

We observe that the sampling period $T_e = 0.2s$ is sufficiently small, given that $\omega_0 = 2s^{-1} \ll \frac{2\pi}{T_e} = 31.4s^{-1}$.

$$G(s) = \frac{4}{s^2 + s + 4} \tag{1}$$

The Simulink model is represented in Figure 1.

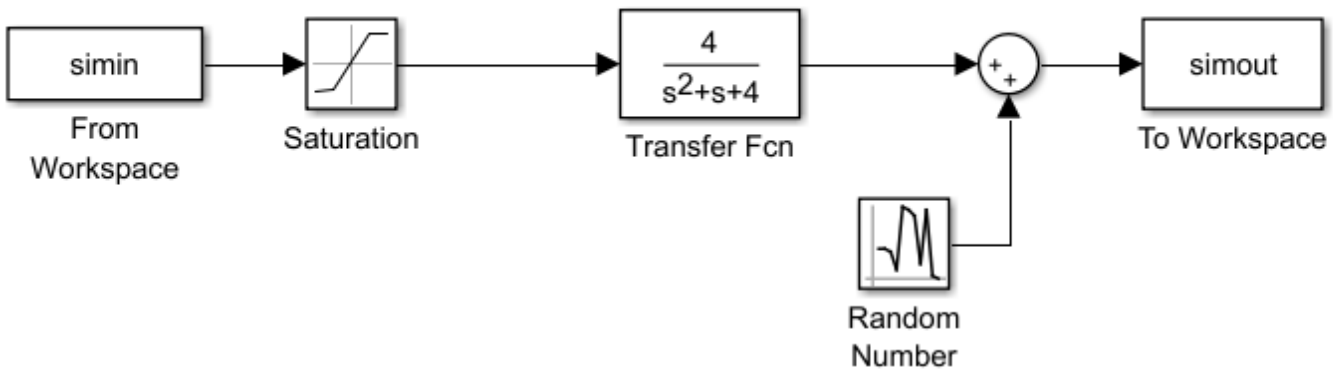


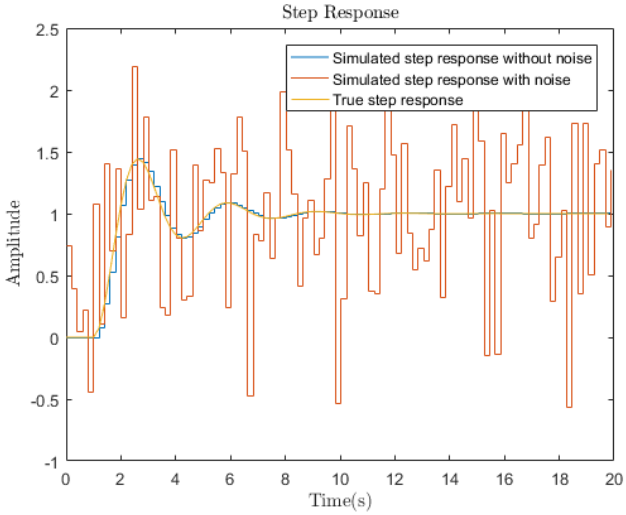
Figure 1: Simulink model

To begin with, the true step and impulse response were calculated, then compared with the simulated system's responses, which can be seen in Figure 2. The step applied in the simulation had a delay of 1s, a length of 50s and an amplitude of 0.5 caused by the Saturation Block; as such, the output was divided by 0.5 in order to estimate the actual response of the simulation. In Figure 2a, the simulated responses (with and without noise) are compared to the real one.

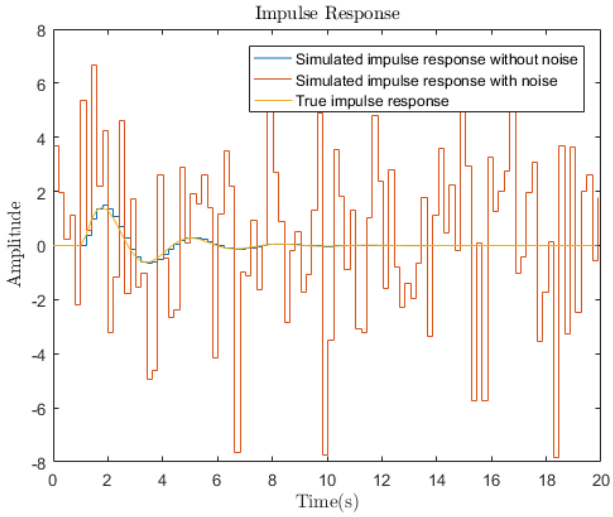
The same procedure was repeated in order to obtain the impulse response of the system (see Figure 2b). As observed before, the input was limited by the Saturation Block, but this time the output was divided by

the saturation limit times the sampling time, so as to take into account the difference between a theoretical Kronecker delta and the empirical one, which has a length of Te and an amplitude set by the saturation (also see page 32-33 of the course book).

The images reveal the sensitivity of this method to noise, which covers the response in a significant way, making impossible the characterization of the system. It was noticed that by reducing the noise variance to 0.01, the second order response was considerably clearer and the steady state more stable.



(a) Real step response against simulated with and without noise



(b) Real impulse response against simulated with and without noise

Figure 2: Step and impulse response

2 Autocorrelation of a PRBS signal

Using the given *prbs.m*, a signal *prbs(n,p)* can be created, with a n-bits register repeated over p periods. Then the function *[R,h] = intcorr(u,y)* was written, which calculates the cross-correlation of same-length signals u and y (see Listing 1).

The function *intcorr* was tested by calculating the autocorrelation R_{uu} of a signal $u = prbs(5,4)$. The result is given by the Figure 3, which is coherent with the results shown at page 22 of the course book.

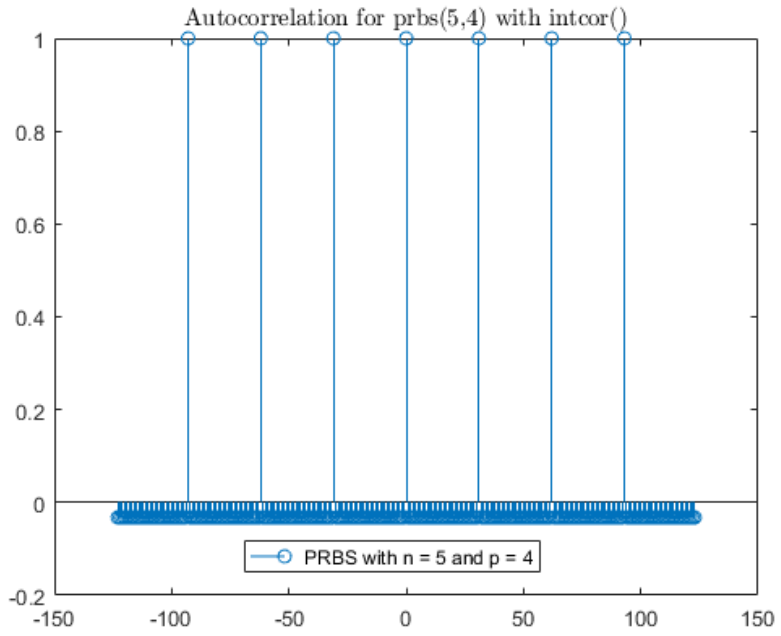


Figure 3: Autocorrelation of prbs(5,4) with the function intcorr()

Listing 1: intcor.m

```
function [R,h] = intcor(u,y)
% correlation function
% supposes that u and y have same length

N = length(u);
R = [];
h = [];

for h0 = -N+1:N-1
    buff = 0;
    for k = 1:N
        buff = buff + 1/N * u(k)*y(mod(k-h0+N-1,N)+1);
    end
    R = [R;buff];
    h = [h;h0];
end
```

3 Impulse response by deconvolution method

The impulse response was calculated here by using the deconvolution method, which is presented at the pages 32-34 of the course book. Using an impulse as input and measuring the output, it is possible to calculate the impulse response of a linear system with the Equations 2 and 3.

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

$$\Theta = U^{-1}Y \tag{3}$$

As the Toepliz matrix U is ill-conditioned so that Θ cannot be computed, it is necessary to use the finite impulse response by assuming that $g(k) = 0$ for $k \geq K$, as a stable system's impulse response should converge to zero.

As such, the Equations 2 and 3 can be rewritten as

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} u(0) & 0 & \cdots & 0 \\ u(1) & u(0) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ u(N-1) & u(N-2) & \cdots & u(N-K) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(K-1) \end{bmatrix} \tag{4}$$

$$\Theta_K = (U_K^T U_K)^{-1} U_K^T Y_K \tag{5}$$

where U_K, Y_K and Θ_K are truncated versions for a dimensions K to be established.

In Figure 4, the estimated impulse response after truncation with K = 60 can be observed, which corresponds to a simulation of 12s. The 2-norm error was found equal to 2.6326. It can be noticed that the method is sensitive to noise.

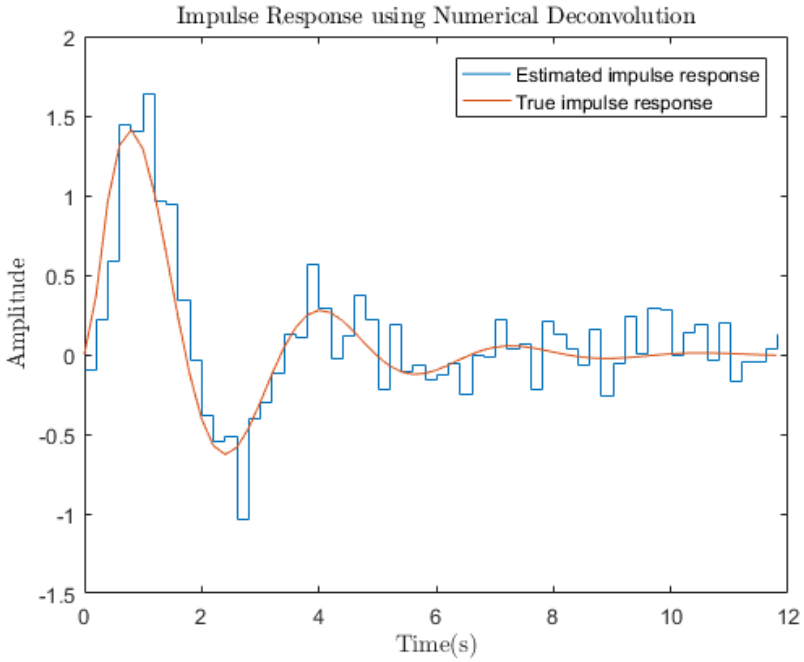


Figure 4: Impulse response calculated with the numerical deconvolution method against the real response.

Listing 2: Numerical deconvolution

```
Te = 0.2;
sim_time = 70;
N = sim_time/Te;
amplitudeLimit = 0.4;

% random input signal
a = -amplitudeLimit; % scaling to avoid saturation
b = amplitudeLimit;
u = a + (b-a).*rand(N,1);

% simulation
simin = struct();
simin.signals = struct('values', u);
simin.time = linspace(0,N*Te, N);
sim('ce1_1_sim')

% deconvolution
U = toeplitz(u, [u(1);zeros(N-1,1)]);
k = 60;
Uk = U(:,1:k); % truncate U

```

```
theta = pinv(Uk)*simout/Te;

figure
stairs(simin.time(1:k), theta(1:k));hold on;
G = tf([4],[1,1,4]);
G_discrete = c2d(G, Te, 'zoh');
[G_impulse, G_time] = impulse(G_discrete, sim_time-Te);
plot(G_time(1:k), G_impulse(1:k));


title('Impulse Response using Numerical Deconvolution','Interpreter','latex')
legend('Estimated impulse response','True impulse response')
xlabel('Time(s)','Interpreter','latex')
ylabel('Amplitude','Interpreter','latex')

% error
[H_impulse, H_time] = impulse(H, simin.time(k));
error = norm(H_impulse - theta)
```

4 Impulse response by correlation approach

The last method tested was the correlation approach, which can be found at the pages 35-39 of the course book. Using the cross-correlation Ryu and the autocorrelation R_{uu}, it can be calculated that:

$$\begin{bmatrix} \hat{g}(0) \\ \hat{g}(1) \\ \vdots \\ \hat{g}(K-1) \end{bmatrix} = \begin{bmatrix} \hat{R}_{uu}(0) & \hat{R}_{uu}(1) & \cdots & \hat{R}_{uu}(K-1) \\ \hat{R}_{uu}(1) & \hat{R}_{uu}(0) & \cdots & \hat{R}_{uu}(K-2) \\ \vdots & \vdots & \cdots & \vdots \\ \hat{R}_{uu}(K-1) & \hat{R}_{uu}(K-2) & \cdots & \hat{R}_{uu}(0) \end{bmatrix} \begin{bmatrix} \hat{R}_{yu}(0) \\ \hat{R}_{yu}(1) \\ \vdots \\ \hat{R}_{yu}(K-1) \end{bmatrix} \tag{6}$$

The Equation 6 is valid if u(k) is not white noise and if the assumption is made that $g(h) = 0$ for $h \geq K$.

For this method, the previously found function *intcorr()* was tested against the Matlab implementation of the cross-correlation *xcorr()*. The results can be seen in Figure 5. The 2-norm error for the *intcorr()* was equal to 2.9374, while the one for *xcorr()* was equal to 3.0937.

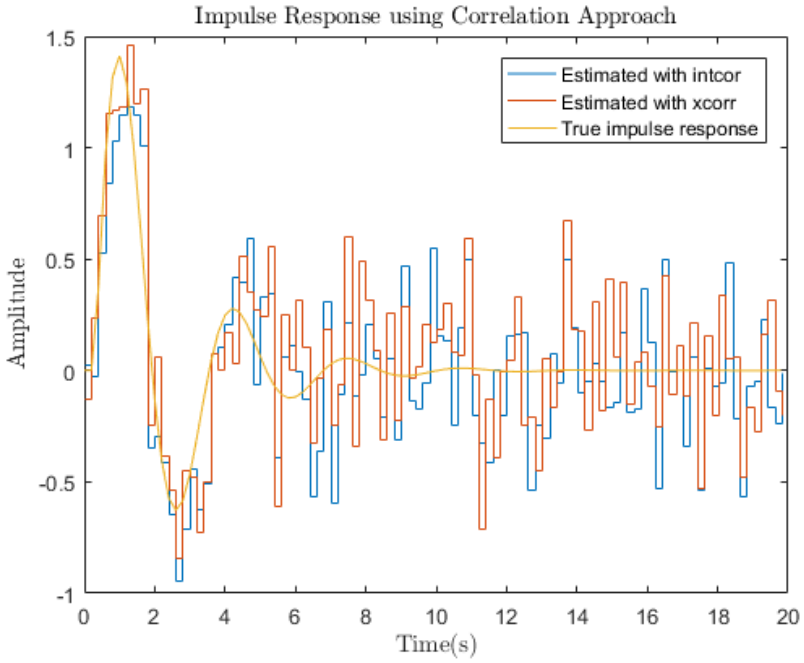


Figure 5: Impulse response by the mean of the correlation approach calculated with intcor() and xcorr() against the real response.

Listing 3: Correlation approach

```
% input signal
Uprbs = 0.5* prbs(7,4);
Te = 0.2; % sample time
N = length(Uprbs);
sim_time = N*Te;

% simulation
simin = struct();
simin.signals = struct('values', Uprbs);
simin.time = linspace(0,sim_time, N);
sim('ce1_1_sim')

% extract one period of the signals
N = length(Uprbs)/4;
Uprbs = Uprbs(1:N);
Y = simout(end+1-N:end)/Te;
time = simin.time(1:N);

% Correlation approach using intcor
Ruu = intcor(Uprbs, Uprbs);
Ryu = intcor(Y, Uprbs);
U = toeplitz(Ruu);
theta_intcor = pinv(U)*Ryu;

% Correlation approach using xcorr
Ryu = xcorr(Y, Uprbs);
Ruu = xcorr(Uprbs, Uprbs);
U = toeplitz(Ruu);
theta_xcorr = pinv(U)*Ryu;

G = tf([4],[1 1 4], 'InputDelay', Te);
Z = c2d(G, Te, 'zoh');
[G_impulse, G_time] = impulse(Z, time(end));

% error
err_intcor = norm(theta_intcor(1:127) - G_impulse)
err_xcorr = norm(theta_xcorr(1:127) - G_impulse)
```