ME-421: Report on Computer Exercises 1

GROUP 47: Gabriel Tornare, Beryl Yersin

November 2019

1 Introduction

This report reports our answers to the questions on the exercises of the course ME-421: System Identification.

2 CE-1: Nonparametric Methods

2.1 Step response

The Simulink block diagram has been created along the configuration file as shown below

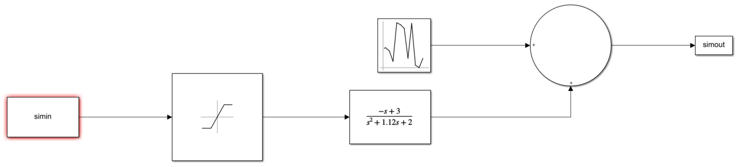
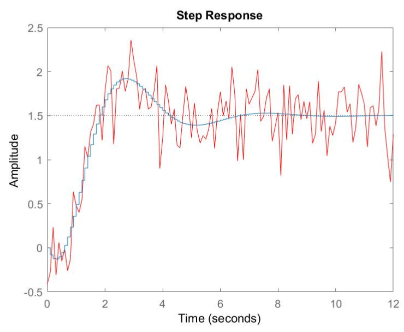


Figure 1: Simulink build

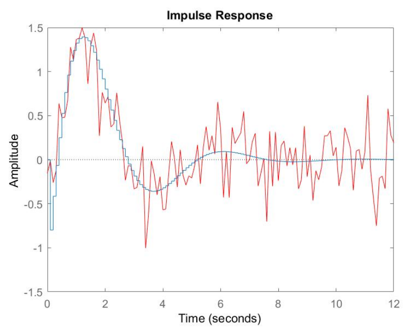
Configuration file CE1-main.m:

```
CE1_1.m

T_e = 0.1;

sys = tf([-1 3], [1 1.12 2]);
sysd = c2d(sys, T_e);

simin.time = T_e*(0:999)';

simin.signals.values = [zeros(10,1); ones(990,1)]/2;
```

```
out = sim('CE1', 'StopTime', '99.9');

hold on
plot(simin.time - 1, out.simout.data*2, 'r')
title("Step response of the system");
step(sysd)

simin.signals.values = [zeros(10,1); ones(10,1); zeros(980,1)]/2;
out = sim('CE1', 'StopTime', '99.9');

figure
hold on
plot(simin.time - 1.5, out.simout.data*2, 'r')
title("Impulse response of the system");
impulse(sysd)
```



(a) Step response of the system.



(b) Impulse response of the system.

Figure 2: Output graphs of the two responses of the Simulink block graph. Red = noisy, blue = noiseless

A scaling has been implemented on the step and impulse responses because as our system cuts the input so that is is comprised between -0.5 and 0.5 (in consequence, a step of one becomes a step of 0.5 and an impulse of a magnitude of 1 becomes an impulse of a magnitude of 0.5), the responses are then also divided by two.

A sampling period of 0.1 is a reasonable chose because if we compute the cutoff frequency of the system, we can see that it is approximately equal to 0.3 hertz, which is more than 10 times lower than the sampling frequency.

2.2 Auto Correlation of a PRBS signal

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

L = length(u);
N = length(y);
M = lcm(L, N);
h = 0:M-1;
R = zeros(M, 1);

u = repmat(u, [M/L, 1]);
y = repmat(y, [M/N, 1]);

for i = h
    R(i+1) = u'*circshift(y, i);
end
R = R/M;
h = h';
end
```

By executing the following file:

```
CE1_2.m

u = prbs(6,4);
[R, h] = intcor(u, u);

plot(h, R)
```

We get the graph on figure 3.

2.3 Impulse response by deconvolution method

```
CE1_3.m

T_e = 0.1;
N = 400;
K = 100;
```

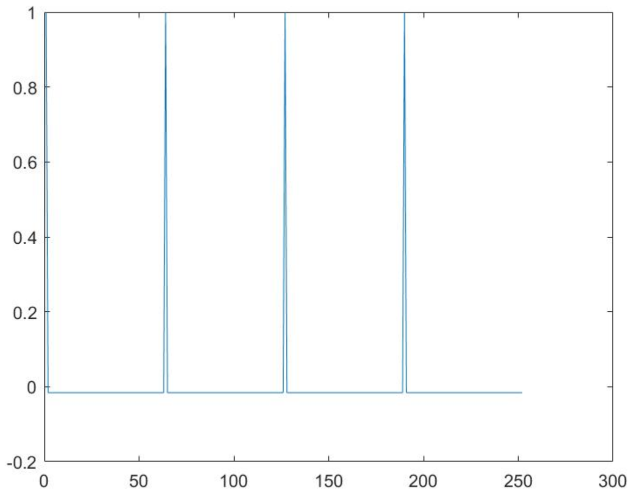


Figure 3: Auto-correlation graph for u = prbs(6, 4)

```
M = 1;

simin.time = T_e*(0:(N-1))';
simin.signals.values = rand(N, 1) - 0.5;
U = toeplitz(simin.signals.values,
[simin.signals.values(1) zeros(1, N-1)]);

out = sim('CE1', 'StopTime', num2str((N - 1)*T_e));
THETA = (U(:,1:K)'*U(:,1:K))\U(:,1:K)'*out.simout.data/T_e;

hold on
plot(simin.time(1:K), THETA, 'r')

sys = tf([-1 3], [1 1.12 2]);
sysd = c2d(sys, T_e);
impulse(sysd, simin.time(1:K))

true_theta = impulse(sysd, simin.time(1:K));
error = sqrt(sum((THETA - true_theta).^2));
```

The result of this code is on figure 4.

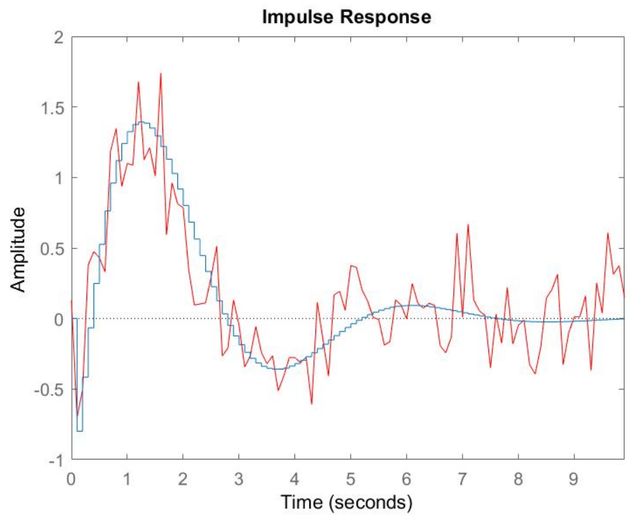


Figure 4: Comparison of the true impulse response (red) with the discrete time system (blue). The 2-norm error of the two responses evolves around 6.2

2.4 Impulse response by correlation approach

```
CE1_4.m

T_e = 0.1;
u = prbs(7, 4)/2;
N = length(u);
simin.signals.values = u;
simin.time = T_e*(0:(N-1))';

K = 120;

out = sim('CE1', 'StopTime', num2str((N - 1)*T_e));
y = out.simout.data;

sys = tf([-1 3], [1 1.12 2]);
sysd = c2d(sys, T_e);
true = impulse(sysd, simin.time(1:K));

R_yu = intcor(y, u);
R_uu = intcor(u, u);
g = toeplitz(R_uu(1:K))\R_yu(1:K)/T_e;
```

```
intcor_error = sqrt(sum((g - true).^2))

hold on
plot(simin.time(1:K), g, 'r')

R_yu = xcorr(y, u);
R_uu = xcorr(u, u);
g = toeplitz(R_uu(N:N+K-1))\R_yu(N:N+K-1)/T_e;
xcorr_error = sqrt(sum((g - true).^2))

plot(simin.time(1:K), g, 'g')
impulse(sysd)

legend('intcor', 'xcorr', 'true')
```

The output of this code can be seen on figure 5. The 2-norm errors for intcorr and xcorr are respectively 1.7492 and 1.4151. The 2-norm error of the inter-correlation method is always measured as being higher than the one of the cross-correlation method.

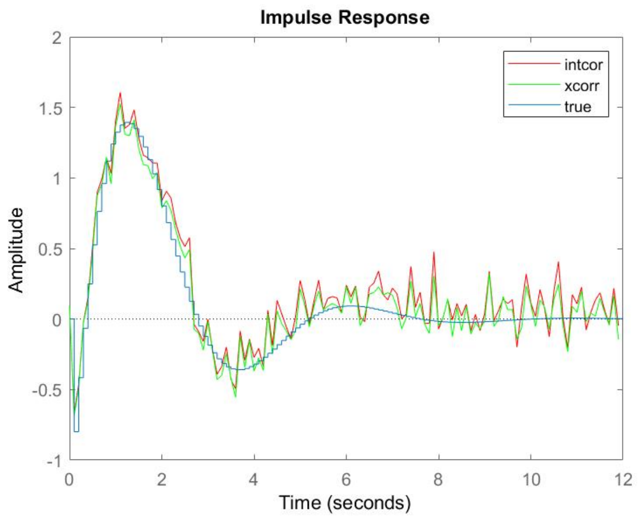


Figure 5: Comparison of the true impulse response of the discrete-time system (blue) with our results using intcor (red) and xcorr (green)

2.5 Frequency domain Identification (Periodic signal)

```
CE1_5.m

T_e = 0.1;
p = 4;
n = 9;
u = prbs(n, p)/2;
N = length(u);
P = N/p;
simin.signals.values = u;
simin.time = T_e*(0:(N-1))';

out = sim('CE1', 'StopTime', num2str((N - 1)*T_e));
y = out.simout.data;

u_hat = zeros(P, 1);
y_hat = zeros(P, 1);
for i = 0:p-1
    u_hat = u_hat + fft(u(1+i*P:(i+1)*P));
    y_hat = y_hat + fft(y(1+i*P:(i+1)*P));
end
u_hat = u_hat/p;
y_hat = y_hat/p;
g_hat = y_hat./u_hat;

x = (2*pi/T_e)/P*(0:P-1);

hold on

sys = frd(g_hat, x);
bode(sys, x)

sys = tf([-1 3], [1 1.12 2]);
sysd = c2d(sys, T_e);
bode(sysd, x)
```

The corresponding Bode diagram is seen on figure 6. We can observe discrepancies at high frequencies, for the system gets more and more disturbed by the noise (whose norm increases with the frequency).

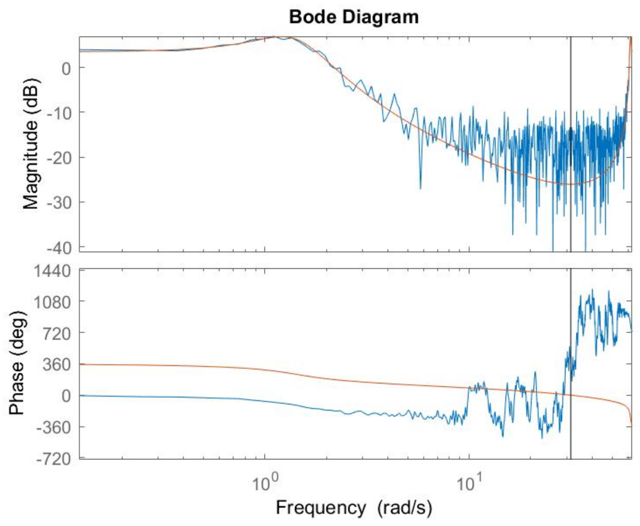


Figure 6: Comparison of the Bode diagrams in amplitude and phase of the true model (red) and the identified one (blue).

2.6 Frequency domain Identification (Random signal)

As explained in slide **31** of chapter **2**, the random signal we apply should excite the system on the largest spectrum possible to obtain the highest possible energy for the excitation signal, and a good solution for that is ”a PRBS signal which has a uniform spectrum”.

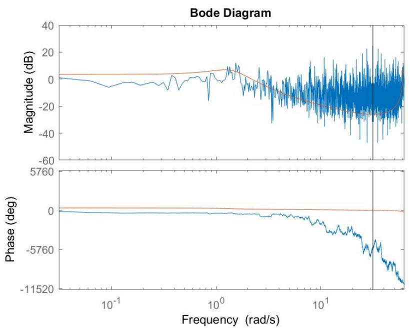
The code works by (un)commenting the filters we want to apply on the signal:

- The type of input signal (here it is always PRBS)
- The window to apply on the correlation function (here Hann window or no window at all)
- The number of groups we want to average on (here can be equal to 1 = no averaging)

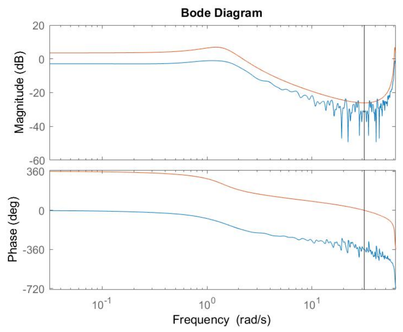
```
CE1_6.m

T_e = 0.1;
N = 2000;
p = 1;
%u = normrnd(0, 0.2, N, 1);
%u = rand(N, 1) - 0.5;
u = prbs(8,20);
u=u(1:2000,:);
```

```
%u = prbs(N,p);
M = 100;
%Hann window:
%f = [0.5 + 0.5*cos(pi/M*(0:M)) zeros(1, N-M-1)]';
f = ones(N,1);
%number of blocks:
m = 50;

simin.signals.values = u;
simin.time = T_e*(0:(N-1))';

out = sim('CE1', 'StopTime', num2str((N - 1)*T_e));
y = out.simout.data;

phi_hat_yu = zeros(N/m, 1);
phi_hat_uu = zeros(N/m, 1);

for i = 0:m-1
    phi_hat_yu = phi_hat_yu + fft(intcor(y(1+i*N/m:(i+1)*N/m),
    u(1+i*N/m:(i+1)*N/m)).*f(1:N/m));

    phi_hat_uu = phi_hat_uu + fft(intcor(u(1+i*N/m:(i+1)*N/m),
    u(1+i*N/m:(i+1)*N/m)).*f(1:N/m));
end

phi_hat_yu = phi_hat_yu/m;
phi_hat_uu = phi_hat_uu/m;

G_hat = phi_hat_yu./phi_hat_uu;
x = (2*pi/T_e)/(N/m)*(0:(N/m)-1);

hold on

sys = frd(G_hat, x);
bode(sys, x)

sys = tf([-1 3], [1 1.12 2]);
sysd = c2d(sys, T_e);
bode(sysd, x)
```

The bode diagrams of the different frequency responses of the models (no window, Hann window, averaged) can be found on figures 7 and 8. We can observe that the two methods for improving the results (Hann window, averaging) output less noisy bode diagrams than when no window and no averaging are applied. The Hann window is good for reducing the noise at high frequencies without losing some data like the averaging method, which is however simpler to implement but needs a quantity of data large enough to avoid too much data being averaged and having a big impact on the true value curve of the frequency response, as seen in figure 8b.

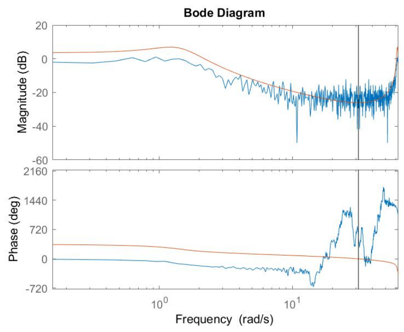


(a) Bode diagram with no modification

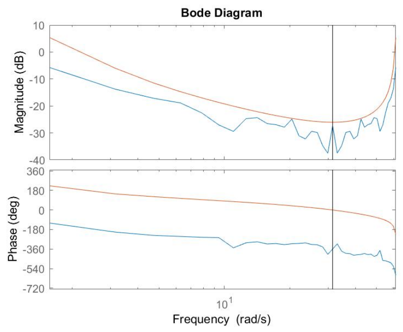


(b) Bode diagram with a Hann window

Figure 7: Bode diagrams for no modification and Hann window



(a) m=5



(b) m=50

Figure 8: Bode diagrams when averaging over m groups