Table of Contents

Filtering & Identification

PracticalAssignment 2 Laurens Hoogenboom #4609638 Ioanna Chanopoulou #5360188

```
clear; close all; clc;
```

Assignment 1

Input signal

We choose the unit step signal as an input because from the simple step response we can derive valuable information that will help us estimate a value for the sampling frequency.

```
N = 1000; % number of measurements
u=ones(N,1);% the unit step signal

% The following plot shows the input signal, you can see that for t>0
 the
% value of the input signal is 1 and that we have discrete time
% measurements, 100 samples per second, fs = 100 Hz => w_s = 2pi*100
 rad/sec.
figure(1)
plot(u,'.' )
xlabel(' Time (s)' )
ylabel(' Input u' )
title(' Input Signal: Unit Step ')
```

Sampling frequency

```
fprintf('*Sampling frequency \n')
% In general, the sampling frequency should be chosen to be fs =
 w_s/2pi,
% with w_s = 10w_B based on an engineering rule.
% The sampling frequency as well as the bandwidth frequency or
 frequency band
% of interest are unknown to us. A good approach when we have to guess
 a
% sampling frequency is sampling at a FAST sampling rate and later on
```

```
% decimate it if it's not suitable.
% At first, we start with a rough estimate of the sampling frequency :
% fs = 100 Hz
fs = 5/(2*pi);
ts = 1/fs;%sampling time
Tfinal = N*ts;%final time
%STUDENTID: the lowest-valued student-id of the students in the group
STUDENTID = 4609638; % 4609638 < 5360188
y = exciteSystem(STUDENTID,u,fs);
t = 0:ts:Tfinal-ts;
% The following plot shows the output response, we can see that there
 are
% some noise spikes
figure(2);plot(t,y);grid on; xlabel('Time (s)');ylabel('Output
 y');title('Step response')
% Afterwards, we inspect the signal's spectrum and try to find the
% bandwidth of interest. In the following plot we observe that the
 signal
% has signal power in the low frequencies. Consequently, we can choose
 the
% bandwidth of interest to be equal to [0,0.3], so fs =
figure(3);periodogram(y,[],[],[],[],1);title('Power spectrum of the
 output signal')

%%%Persistently exciting input signal
fprintf('*Persistency of excitation\n')
% System order at which the input is persistently exciting. We can
 define
% the unit step input sequence so that it is persistently exciting for
% whatever excitement level, by taking advantage of the fact that for
 t < 0
% the step input function is equal to 0.
m = 1; %number of inputs
l = 1; %number of outputs
excitement_level = 100;
s = excitement_level;
u =[zeros(1,excitement_level-1),ones(1,N-excitement_level+1)];
U = hankel(u(1:m*s),u(s*m:end));%block Hankel matrix of the input
%Investigate up to what order the input signal is percistently
 exciting
order_step =2;
i=0;
RANK = zeros(length(u),1);%preallocation
for n = 2:order_step:length(u) %n:model order
    U = hankel(u(1:n),u(n:end));
    RANK(n,1) = rank(U);
end
% The order at which the matrix U stops being full rank is the
 excitement_level = 100
% So, we are providing a persistently exciting input of order 100, the
% plot is following.
figure(4)
t1 = -100:900-1;
plot(t1,u,'.' )
```

```
xlabel(' Time (s)' )
ylabel('u' )
title('Persistently exciting Input u of order 100')
axis([-100 900 -2 2]);

%%%Spike noise
fprintf('*Spike noise\n')
% As already said we are observing some spikes randomly produced. In
 order
% to identify these spikes, we will impose a condition about how
 distant
% the value of the spike is from the value it tends to converge to.
% Afterwards, we are going to use interpolation in order to replace
 the
% values at the spikes with their estimated ones by interpolation. In
 the
% end we are going to use the cleared data.
% We will start by using a condition of about (+/-)0.6 away from the
 steady
% state value which is oscilating around 5.
d1 = 5 +0.6;
d2 = 5-0.6;
j=1;%initialize index for counting the noisy points
for i = 1:length(y)
    if (y(i)>d1 || y(i)<d2)
        tdel(j) = i;%
        ydel(j)= i;%  keep which indeces correspond to spikes
        j=j+1;
    end
end
figure(5)
plot(t,y,'b-')%plot without removal of spikes
hold on
%define the time interval to be replaced by interpolated estimates
j=1;k=1;%initializing
for i = 2:length(tdel)
    if((tdel(i)-tdel(i-1)<=5))
        times{j,k} = t(tdel(i-1));%cell array
        times{j,k+1} = t(tdel(i));
        k=k+1;
    else
        j=j+1;
        k=1;
    end
end
times = t(tdel);
t(tdel) = [];
y(ydel) = [];
[r,~] = size(times);
%Evaluate y at tq
tq = 0:ts:Tfinal-ts;

%Substitute the interpolated values into the previous sequence
y(ydel) = yq;%use the output sequence without the spikes
```

```
plot(tq,yq,'r-');
legend('y','yq');
title('Removed spikes')
hold off
```

Time delays

```
fprintf('*Time delay \n')
%The delay in a system is obtained from the data by estimating the
 system#s
%impulse response. A state-space model generally, has an infinite
 impulse
%response, however, we can approximate that by taking an order m for
 the
%FIR sufficiently large. m should be larger than the time delay
%The calculation of the h_i coefficients of the
%impulse response can be done as follows:
m = 150;% m-th order of FIR
h=toeplitz(u(m+1:N),u(m+1:-1:1))\y(m+1:N);
%Given the h_i, we can visually inspect at which sample instant the
 impulse
%response starts to deviate from zero significantly.In the beginning,
 the
%samplese are almost zero due to the fact that there's a delay in the
 system
%So, now we plot the impulse response parameters and inspect it to see
 for
%how many samples we have a time delay.
figure(6);
plot(1:length(h),h);
xlabel('FIR order m')
%By denoting the time delay by d, we can observe from the plot that at
%sample time 2 there's no extreme deviation from zero. So, the time
 delay
%can be determined to be d =52 and then we can remove the delay as
 follows
d=52;
u=u(1:N-d);
y=y(d+1:N);
```

Remove the DC offset from the output

```
fprintf('*DC offsets \n')
%The offsets are unknown, so a way to remove them is by subtracting
%estimates of them from the output data. An estimation of the offsets
 is
%the calculation of the sample mean of the measured sequences. In
 order to
%accomplish that, we are going to use a function that detrends data by
%subtracting the sample mean, the "detrend" function.
yd=detrend(y,'constant');% yd: detrended output sequence

%%%Determine linearity
```

```
fprintf('*Determine linearity \n')
% In order to determine the lineearity of the system we can implement
% differently scaled step input sequencies and see what we get as an
% output.
% If the output is the same response just differently scaled, then the
% system is linear, otherwise it is not.
a1 =2;%scaling factor
figure(7);
u1 =a1*[zeros(1,excitement_level-1),ones(1,N-excitement_level+1)];
y1 = exciteSystem(STUDENTID,u1,fs);
subplot(3,1,1)
plot(t,y1);grid on; xlabel('Time (s)');ylabel('Output
 y');title('Response using 2*u')
a2 = 3;
u2 =a2*[zeros(1,excitement_level-1),ones(1,N-excitement_level+1)];
y2 = exciteSystem(STUDENTID,u2,fs);
subplot(3,1,2)
plot(t,y2);grid on; xlabel('Time (s)');ylabel('Output
 y');title('Response using 3*u')
a3 = 5;
u3 =a3*[zeros(1,excitement_level-1),ones(1,N-excitement_level+1)];
y3 = exciteSystem(STUDENTID,u3,fs);
subplot(3,1,3)
plot(t,y3);grid on; xlabel('Time (s)');ylabel('Output
 y');title('Response using 5*u')

%%%Identification and Validation data
fprintf('*Identification and Validation data\n')
%Usually, we devide the measurements into an identification part
%and a validation part. The identification is performed on the first
%part(here we take the first 2/3 of all samples) and the validation on
 the
%second part (the remaining 1/3 of all samples).
u_id=u(1:floor(2*N/3),:);%identification input set
y_id=y(1:floor(2*N/3),:);%identification output set

u_val=u(floor(2*N/3)+1:N,:);%validation sets
y_val=y(floor(2*N/3)+1:N,:);
```

```
*Sampling frequency
*Persistency of excitation
*Spike noise

Unrecognized function or variable 'yq'.

Error in PA_2 (line 125)
y(ydel) = yq;%use the output sequence without the spikes
```

Assignment 2: Identification

```
%Since we cannot use a white-noise sequence as an input signal, the
%identification methods that can be used are PI-MOESP and PO-MOESP. We
 will
%use PO-MOESP here.
```

```
method = 'po-moesp';
%To determine the order, we look for a gap between a set of dominant
 sin-
%gular values that correspond to the dynamics of the system and a set
%of small singular values due to the noise.
n = 15;%non-zero singular values
s = 100;
fprintf('\n Model order:%6i',n)
[A,B,C,D,x0,sv] = subspaceID(u_id,y_id,s,n,method);
figure(8);
semilogy(sv,'x');
%predicted output
y_hat_id = simsystem(A,B,C,D,x0,u);
%identification VAF
dy = y - y_hat_id;
VAF_id = max(0, 1 - ((1/length(y))*sum(dy.^2))/((1/
length(y))*sum(y.^2)));
%??Should we use PEM as well????
A0=A;
B0=B;
C0=C;
D0=D;
x00=x0;
maxiter = 20; %iterations
K0 = zeros(length(C0),l);
[Abar,Bbar,C,D,K,x0] = pem(A0,B0,C0,D0,K0,x00,y,u,maxiter); %PEM

%improved VAF
y_hat_pem = simsystem(Abar,Bbar,C,D,x0,u); %predicted output ???
%Define VAF
dy = y - y_hat_pem;
VAF_pem = max(0, 1 - ((1/length(y))*sum(dy.^2))/((1/
length(y))*sum(y.^2)));
```

Assignment 3: Validation

```
%%%VAF for the identification data with the VAF for the validation
 data
%Now, predict the output with the validation set

yhat_val = simsystem(A,B,C,D,x0,u_val);

%VAF
dy = y_val - yhat_val;
VAF_val = max(0, 1 - ((1/length(y_val))*sum(dy.^2))/((1/
length(y_val))*sum(y.^2)));

%%%Auto-correlation of the residuals
epsilon1  =  y_id - y_hat_id;% error-vector between model output and
 measured output for the identification data set
epsilon2 = y_val - yhat_val;
ac1 = xcorr(epsilon1,epsilon1,d);%autocorrelation
ac2 = xcorr(epsilon2,epsilon2,d);
```

```
%a plot of the autocorrelation follows
figure(9);plot(-d:d,ac1);hold on;plot(-d:d,ac2);hold off
legend('Autocorrelaion_1','Autocorrelaion_2')

%%%Cross-correlation between the residuals and the inputs
 xc1 = xcorr(epsilon1,u_id,d);
 xc2 = xcorr(epsilon2,u_val,d);
%A plot can be made using the following commands
figure(10);plot(-d:d,xc1);
```

Published with MATLAB® R2020a