Table of Contents

Filtering & Identification Practical Assignment 2

Roland Varga #5369509
Daniel Barroso #5230543

```
STUDENTID = 5230543;
```

Assignment 1

Sampling frequency

The sampling frequency is chosen to be 100 Hz. This frequency allows to capture the fastest dynamics of the system without sampling an excesive amount of noise.

```
fs1 = 100;
Ts1 = 1/fs1;
```

Input signal

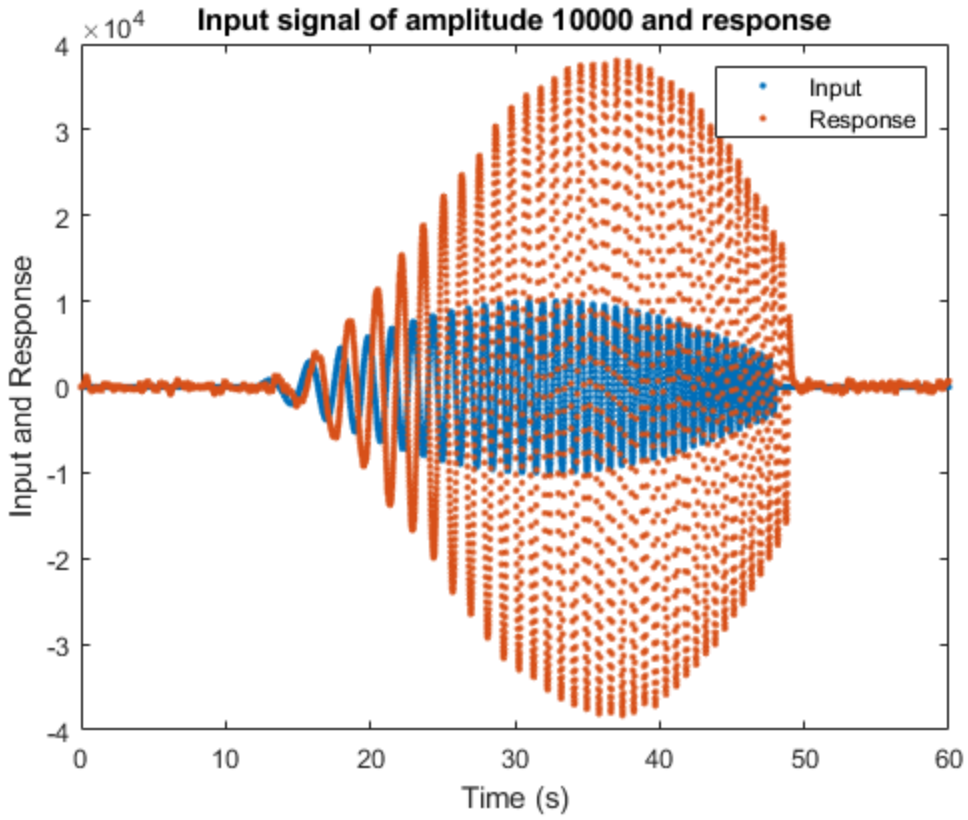
The two parameters taken into account for the design of the input signal are the following: 1.Amplitude: The amplitude of the input signal is designed to have a high signal-to-noise ratio. After checking several inputs, it has been concluded that a signal amplitude in the order of 10^5 is required (Figures 1 and 2). In Figure 1 it can be observed that an input of amplitude 10^4 generates a response with a magnitude comparable to the noise and thus it is not adequate. 2.Persistency of excitation: The persistency of excitation of the input was chosen to be large enough to excite all the system's dynamics. It has been decided that the block size 's' shall be at least 10 times larger than the system order. The system order is initially unkown, so as a first hypothesis it is assumed that it would not be larger than 10. This implies a block size of at least 50. Therefore, the input was designed as a frequency sweep of varying amplitude, with

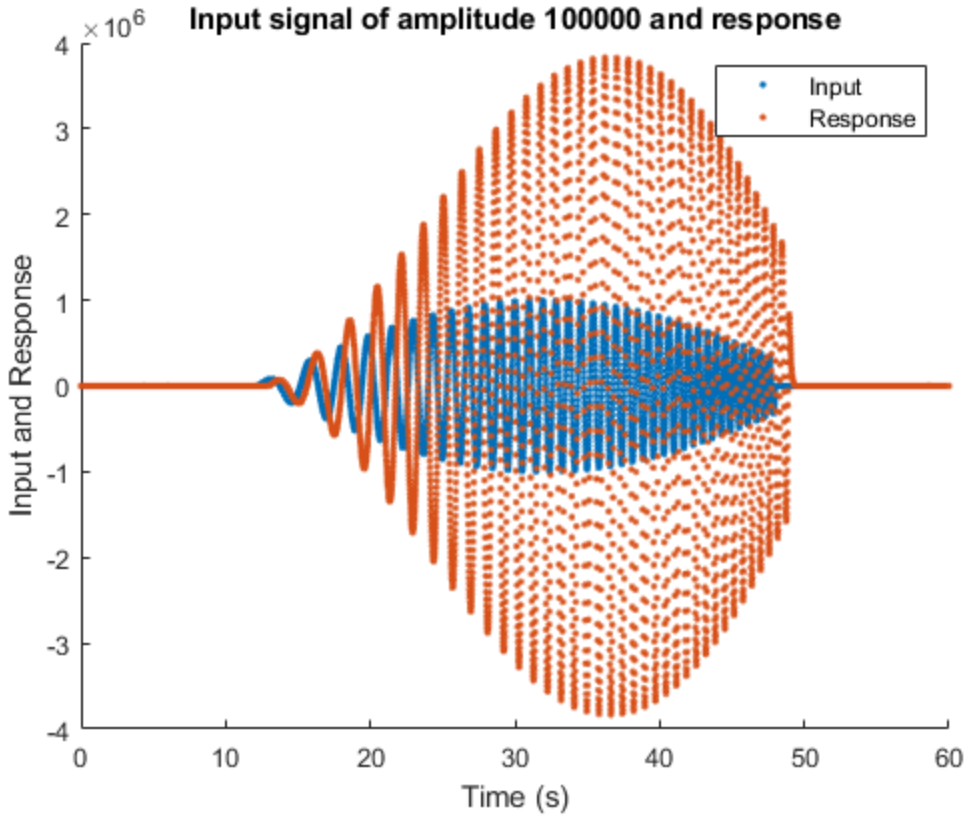
persistency of excitation of order 50, which proved to be enough for the system identification. The input persistency of excitation was checked both by its hankel matrix's rank and by its singular values (Figure 3). By performing several iterations in the identification cycle, it was confirmed that the hypothesis of the model order was correct since all the system dynamics are captured (this is further analyzed in Assigment II in this document)

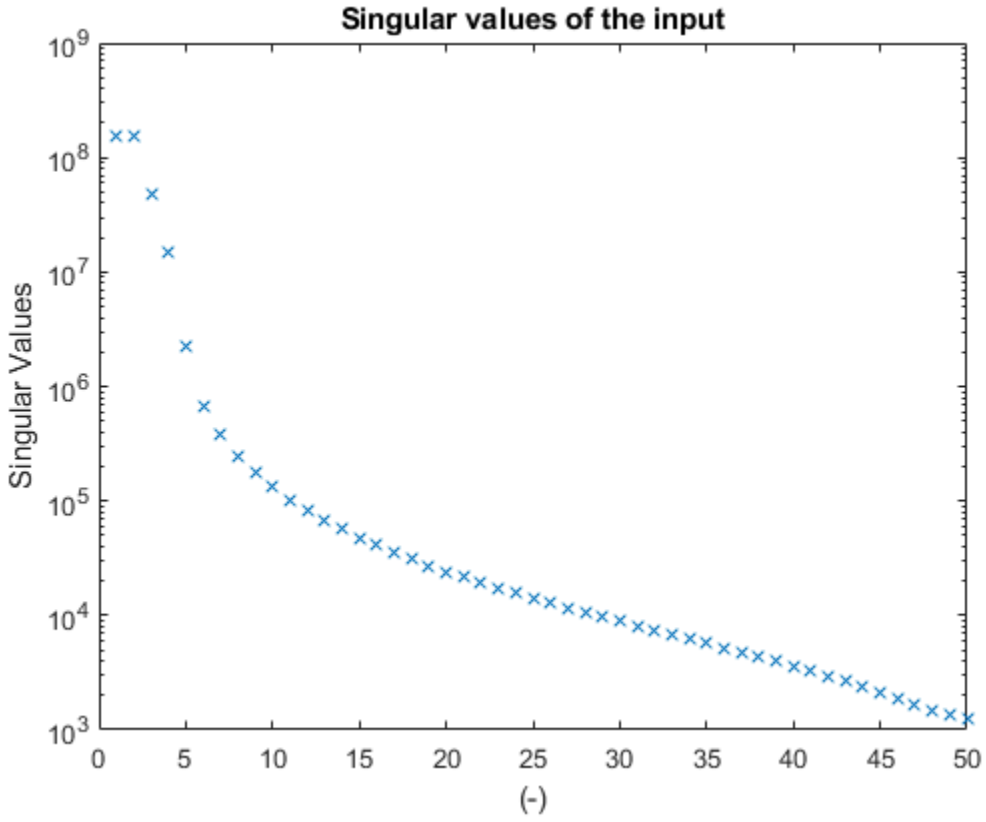
```
%Generate frequency sweep input of order 10^4
omega = 2*pi/4;
sim_time = 60;
t = 0:Ts1:sim_time-Ts1;
t_length = size(t,2);
ramp = linspace(1,4,t_length*0.6);
u = [zeros(1,t_length*0.2)...
    sin(2*pi/(2*40)*t(1:t_length*0.6)).*...
    (10000*sin(omega*(ramp.*t(1:t_length*0.6))))...
    zeros(1,t_length*0.2)];

%Plot input of order 10^4
figure(1)
plot(0:Ts1:sim_time-Ts1,u,'.')
y = spike_filter(exciteSystem(STUDENTID,u,fs1));
hold on;
plot(t,y,'.')
hold off
xlabel ('Time (s)')
ylabel ('Input and Response')
title ('Input signal of amplitude 10000 and response')
legend Input Response

%Generate frequency sweep input of order 10^5
omega = 2*pi/4;
sim_time = 60;
t = 0:Ts1:sim_time-Ts1;
t_length = size(t,2);
ramp = linspace(1,4,t_length*0.6);
u = [zeros(1,t_length*0.2)...
    sin(2*pi/(2*40)*t(1:t_length*0.6)).*...
    (1000000*sin(omega*(ramp.*t(1:t_length*0.6))))...
    zeros(1,t_length*0.2)];


%Plot input of order 10^5
figure(2)
hold on
plot(0:Ts1:sim_time-Ts1,u,'.')
y = spike_filter(exciteSystem(STUDENTID,u,fs1));
plot(t,y,'.')
hold off
xlabel ('Time (s)')
ylabel ('Input and Response')
title ('Input signal of amplitude 100000 and response')
legend Input Response

%Generate hankel matrix of the input
```

```
s = 50;
U_0sN = hankel(u(1:s),u(s:end));
RankU=rank(U_0sN);
fprintf("The rank of the input hankel matrix is: %d \n",RankU);

%Plot singular values of the input
figure(3)
semilogy(svd(U_0sN),'x')
xlabel ('(-)')
ylabel ('Singular Values')
title ('Singular values of the input')
```

The rank of the input hankel matrix is: 50



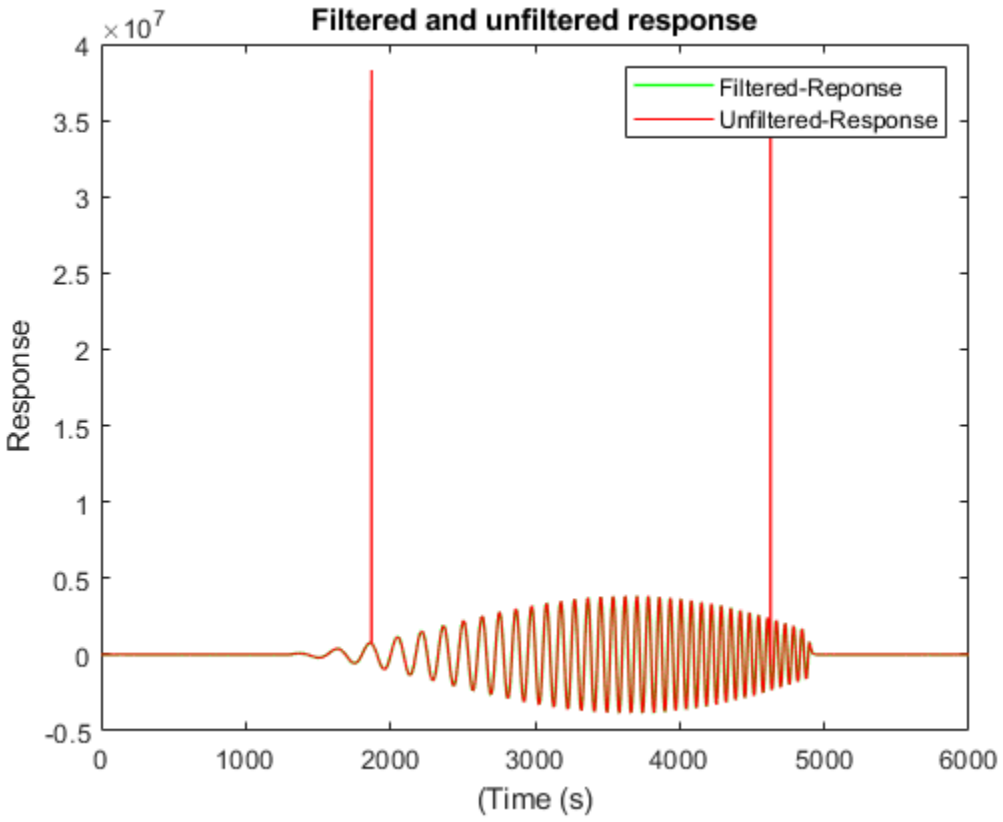

Input signal of amplitude 100000 and response
×10^6
Input
Response
Input and Response
Time (s)


Singular values of the input
Singular Values
(-)

Spikes identification and removal

In order to identify the spikes, the mean and standard deviation of the system response has been calculated. If a sampled point is located out of the range of four times the standard deviation from the mean, it is identified as a spike. In that case, a linear interpolation is performed using the points of the signal that lie before and after the spike. If the spike is located at the end of the signal, such that there are not valid points after the spike, the spike's point are substituted by the last signal value within the allowed deviation range. The code for the spikes identification and removal is located at the end of the document as a separated function called 'spike_filter'.

```
figure(4)
plot(y,'g','LineWidth',1);
hold on;
plot(exciteSystem(STUDENTID,u,fs1),'r')
xlabel ('(Time (s)')
ylabel ('Response')
title ('Filtered and unfiltered response')
legend Filtered-Reponse Unfiltered-Response
```



Time delay

In the second figure it can be observed that the time delay is approximately 0.9 seconds. In order to take this into account for the identification, the response sequence is shifted to match the input signal. Consequently, the last values of the input signal has to be removed so that the input and response have the same lenght.

```
delay = 0.9;
```

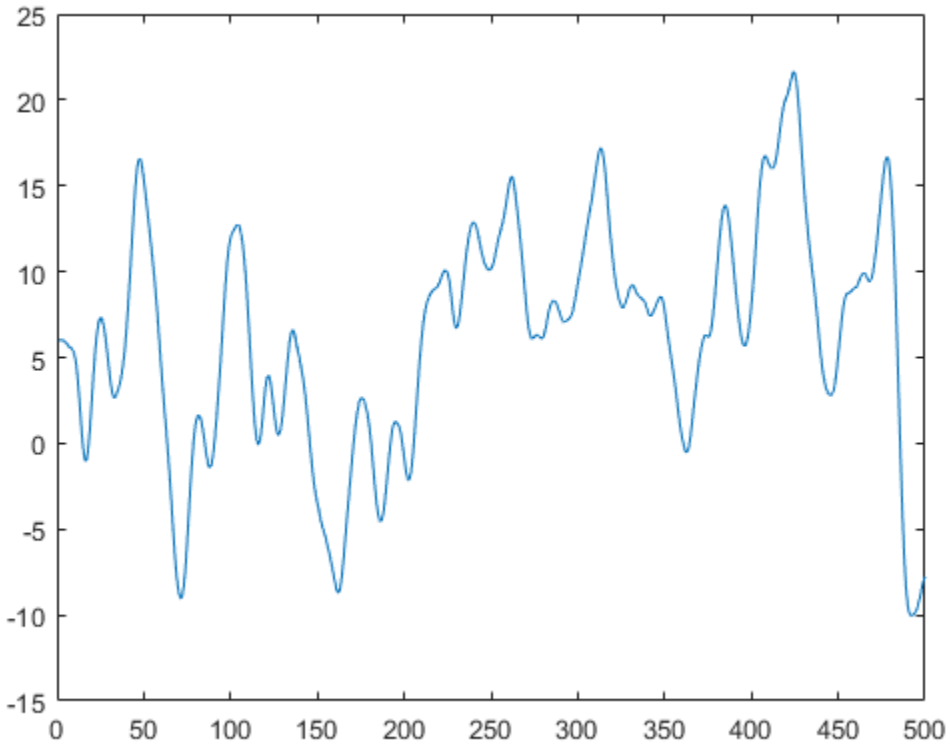
```
delay_samples = 0.9/Ts1;
y_nodelay = y(delay_samples:end);
u_nodelay = u(1:end-delay_samples+1)';
```

DC-Offset analysis

For the given Student ID, the DC offset varies significantly for each realization. In order to find a mean value for the DC offset, 2000 realizations of the system response to a zero input have been performed. Plotting the mean for every time instant shows that the system does not have a clear DC value. Since the mean values vary around zero, and are highly dependent on the realization, it has been concluded the most appropriate decision is to approximate the DC Offset as 0.

```
for i = 1:2000
    y_all(i,:) = (spike_filter(exciteSystem(STUDENTID,0*10000*...
    ones(5/Ts1,1),fs1)));
end
figure(5)
plot(sum(y_all,1)/2000)
mean(sum(y_all,1)/2000)
```

```
ans =

    6.1774
```

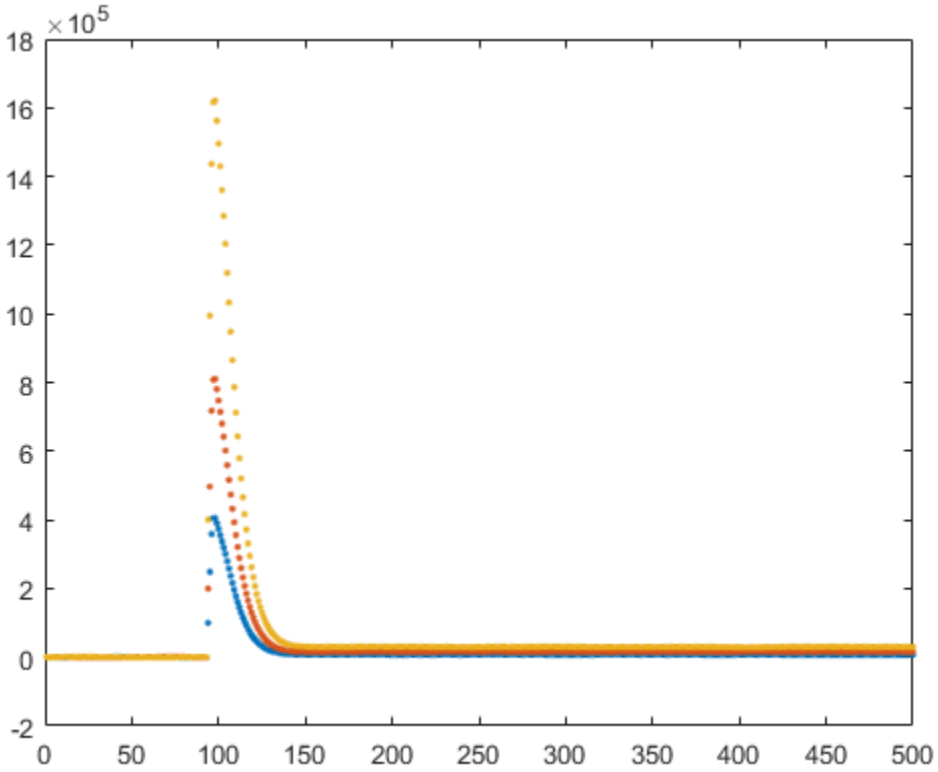


Linearity determination

In order to check if the system is linear, it has been observed if multiplying the input by some constant also causes the response to be multiplied by the same number. As it can be observed in the graph, this is confirmed, since the responses of the inputs multiplied by 1, 2 and 4, show a peak at 4, 8 and 16, respectively.

```
y_lin1 = exciteSystem(STUDENTID,100000*ones(5/Ts1,1),fs1);
y_lin2 = exciteSystem(STUDENTID,2*100000*ones(5/Ts1,1),fs1);
y_lin3 = exciteSystem(STUDENTID,4*100000*ones(5/Ts1,1),fs1);

figure(6)
plot(spike_filter(y_lin1),'.');
hold on
plot(spike_filter(y_lin2),'.');
plot(spike_filter(y_lin3),'.');
hold off
disp(max(spike_filter(y_lin2))/max(spike_filter(y_lin1)))
disp(max(spike_filter(y_lin3))/max(spike_filter(y_lin1)))
```

```
    2.0006

    4.0034
```



Input-Output data for identification and validation

By following the system identification cycle several times, it has been observed that the highest VAF for both the identification and validation is obtained when the training is perfomed on the 60% of the data, and the validation with the remaining 40%. It shall be remarked that the validation is performed using the real system response (including the delay).

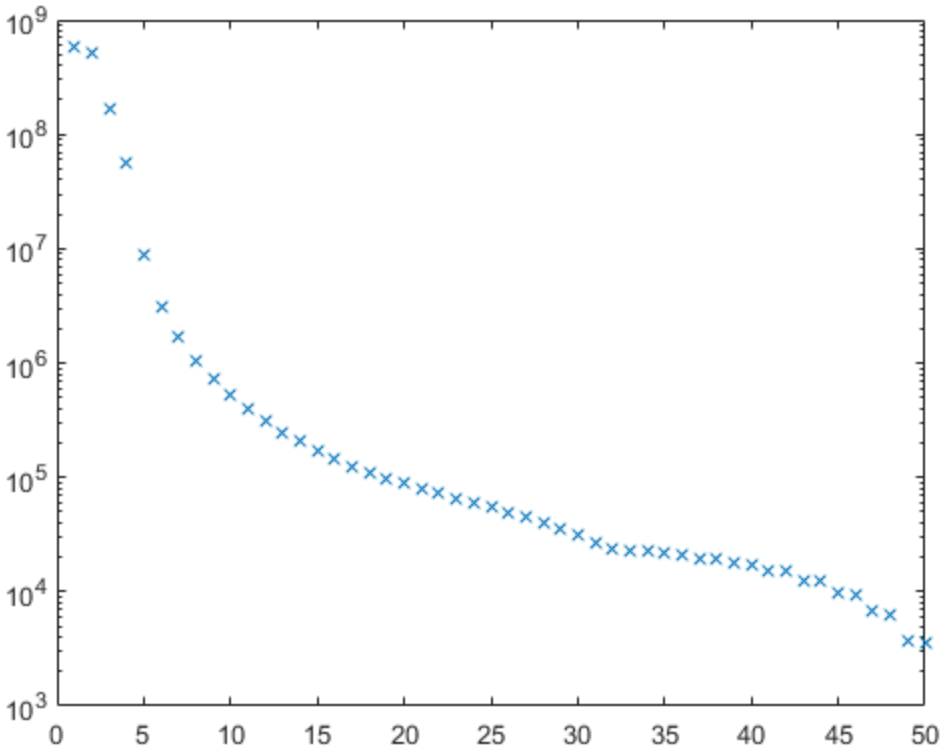
```
%Setting the part of the data used for identificatin and validation
r = 0.6;
lim = floor(size(u_nodelay,1)*r);

%Data without delay for identification

% Input data for identification
u_iden=u_nodelay(1:lim);

% Output data for identification
y_iden=y_nodelay(1:lim);


%Data with delay for validation

% Input data for validation
u_val=u(lim+1:end);

% Output data for validation
y_val=y(lim+1:end);
```

Assignment 2: Identification

Model order determination

In order to determine the order of the system, the plot of the singular values of the system response has been analyzed. In addition to this, the identification and validation VAFs have been taken into account, as well as the residuals autocorrelation and residuals correlation with the input. The figure shows that a large drop takes place after the second, fourth and fifth singular values. After several iterations of the identification cycle anayzing the cases of system model order to be 2, 4 and 5, it has been observed that the VAFs are higher in the later case. Moreover, the residuals autocorrelation and residuals correlation with the input are smaller in this case (this is more deeply addressed in the last section). Therefore, it has been concluded that the model order is 5.

```
Y_0sN = hankel(y(1:s),y(s:end));

figure(7)
semilogy(svd(Y_0sN),'x')
```

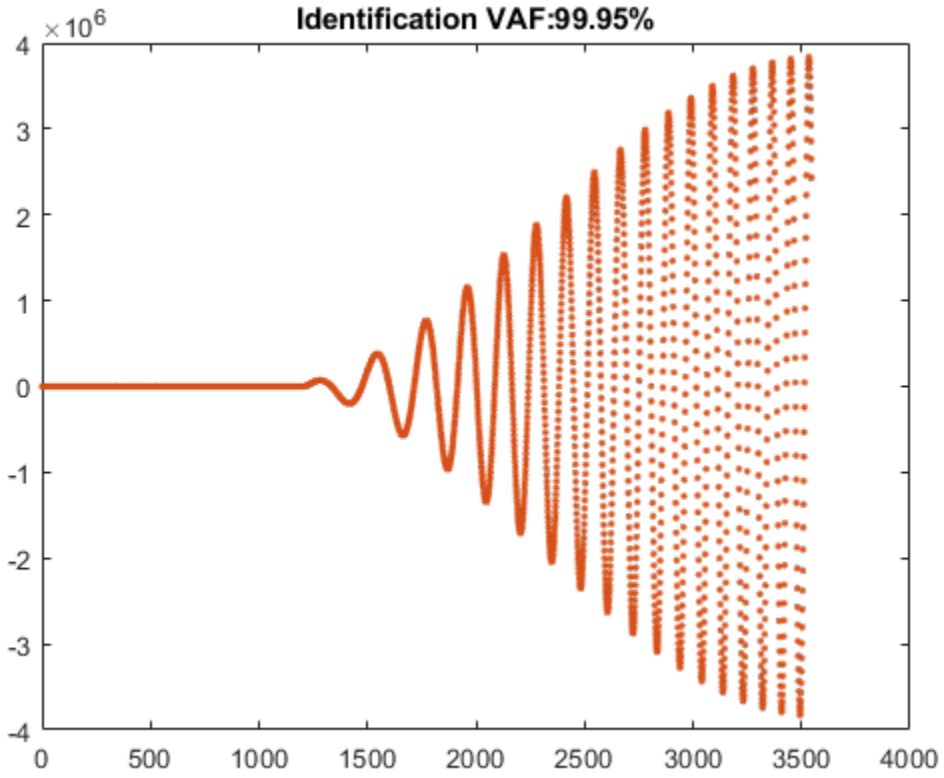


Model Estimation

Several subspace identification methods have been used to perform the identification and validation, being the PO-MOESP algorithm the one providing the best performance. Moreover, since the VAF obtained with this method is larger than 99%, no further algorithms such as error prediction methods have been used after the PO-MOESP.

```
%Identification method
method = 'po-moesp';

%Model order
n = 5;

%Block size
s = 50;

%Identification process
[A,B,C,D,x0,sv] = subspaceID(u_iden,y_iden,s,n,method);

%Simulation with the identified system
[yhat,xhat] = simsystem(A,B,C,D,x0,u_nodelay);

%Calculation of the VAF for identification
vaf(y_iden,yhat(1:lim))

figure(8)
```

```
plot(y_iden,'.')
hold on;
plot(yhat(1:lim),'.')
title 'Identification VAF:99.95%'
```



Assignment 3: Validation

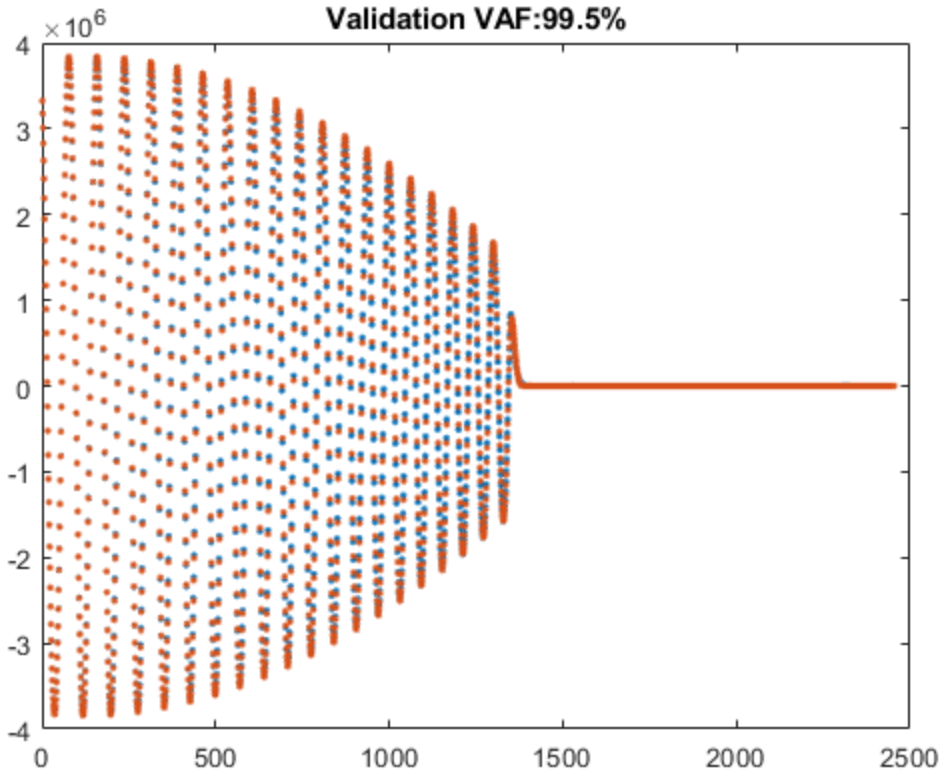
Identification and Validation VAFs

The VAF of the identification reaches 99.98%, while that of the validation is approximately 99.5%. Although this values slightly vary over different realizations, it can be highlighted that the identification VAF is always higher than the validation VAF. This is due to the fact that the identified system is optimized for the identification data, and since it differs from the validation data, the resulting valdation VAF is not as high as for the identification. Nevertheless, it can be remarked that although the input data for the validation contains a sinusoidal of a higher frequency than that of the identification input data, the response in the validation accurately follows the high frequency input. This also supports the choice of the model order as 5, since the higher frequency input does not excite the system in any way that was not taken into consideration during the identification.

```
%Adding the delay to the identified system
delay=zeros(delay_samples-1,1);
yhat_delay=[delay; yhat];

figure(9)
plot(y_val,'.')
```

```
hold on;
plot(yhat_delay(lim+1:end),'.')
hold off;
title 'Validation VAF:99.5%'

vaf(y_val,yhat_delay(lim+1:end))
```

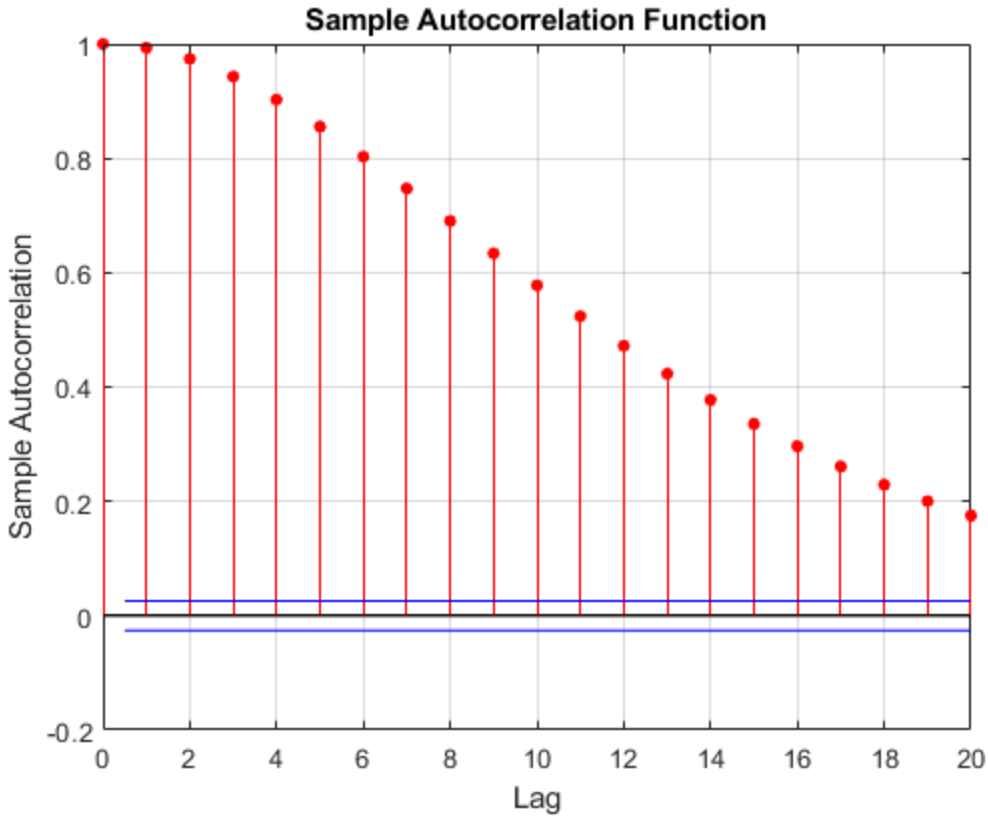
```
ans =

   99.9749
```

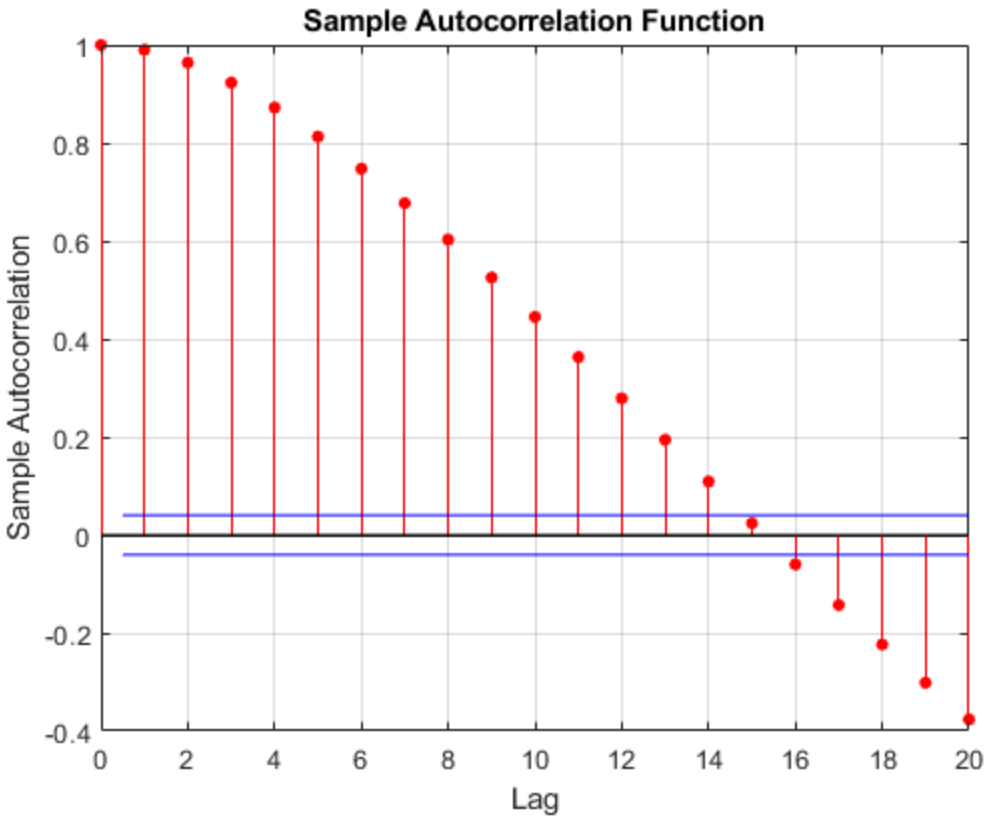


Residuals auto-correlation

The residuals auto-correlation indicates the amount of information present in the error and thus not captured in the identified model, and has been extensively used in this work. Since the analyzed model exciteSystem presents noise, its autocorrelation has been analyzed in the first place. As shown in the first graph, it can be observed that it is not a zero-mean white noise and thus it has some embedded information. In order to address the identification performance, the used reasoning is as follows: if the identified model was perfect and all the usefull information was captured, the residuals auto-correlation should match the auto-correlation of the noise present in the system. The data used for calculating the residuals auto-correlation has been the validation data. Looking at the residuals auto-correlation over the different iterations of the id-cycle has allowed to determine the model order. More precisely, a model order of 5 generates a residuals auto-correlation that is close to that of the noise, as shown on the second figure.

```
y_zeroinput = spike_filter(exciteSystem(STUDENTID,0*u,fs1));
figure(10)
autocorr(y_zeroinput)

figure(11)
res = y_val-yhat_delay(lim+1:end);
autocorr(res)
```

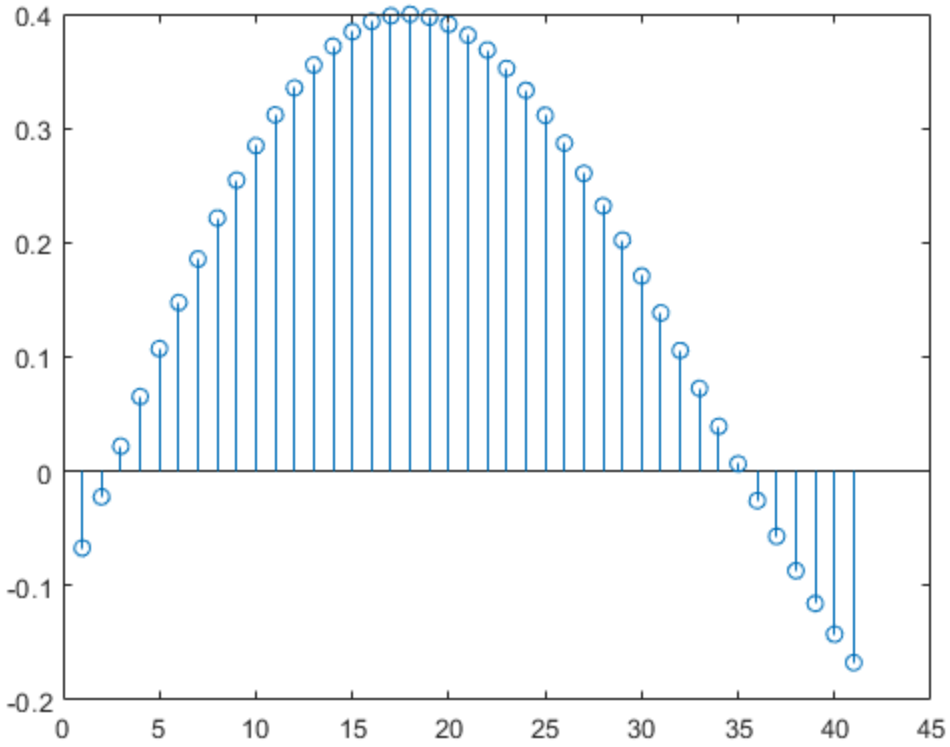




Cross-correlation between input and residuals

It is expected that the cross-correlation between the residuals and the input is generally small.

```
figure(12)
corr=crosscorr(res,u_val);
stem(corr)
```



Functions

```
function [A,B,C,D,x0,sv] = subspaceID(u,y,s,n,method)
% Function INPUT
% u         system input (matrix of size N x m)
% y         system output (matrix of size N x l)
% s         block size (scalar)
% n         model order (scalar)
% method    method (string e.g. 'moesp')
%
% Function OUTPUT
% A         System matrix A (matrix of size n x n)
% B         System matrix B (matrix of size n x m)
% C         System matrix C (matrix of size l x n)
% D         System matrix D (matrix of size l x m)
% x0        Initial state (vector of size n x one)
% sv        Singular values (vector of size n x one)

    switch method
        case 'moesp'
            % Computation of the Hankel matrices
            UY = [hankel(u(1:s),u(s:end)); hankel(y(1:s),y(s:end))];

            % LQ factorization to get L22
            [~,R] = qr(UY',0);
```

```
            L = R(1:2*s,1:2*s)';
            L22 = L(size(L,1)/2+1:end,size(L,2)/2+1:end);

            % SVD of L22
            [U,S,~] = svd(L22,0);

        case 'pi-moesp'
            % Computation of the Hankel matrices
            U = hankel(u(1:2*s),u(2*s:end));
            Up = U(1:s,:);
            Uf = U(s+1:2*s,:);
            Y = hankel(y(1:2*s),y(2*s:end));
            Yp = Y(1:s,:);
            Yf = Y(s+1:2*s,:);

            % LQ factorization to get L32
            [~,R] = qr([Uf' Up' Yf'],0);
            L = R(1:3*s,1:3*s)';
            L32 = L(2*s+1:end,s+1:2*s);

            % SVD of L32
            [U,S,~] = svd(L32,0);

        case 'po-moesp'
            % Computation of the Hankel matrices
            U = hankel(u(1:2*s),u(2*s:end));
            Up = U(1:s,:);
            Uf = U(s+1:2*s,:);
            Y = hankel(y(1:2*s),y(2*s:end));
            Yp = Y(1:s,:);
            Yf = Y(s+1:2*s,:);

            % Constructing the instrumental variable
            Z = [Up;Yp];

            % LQ factorization to get L32
            [~,R] = qr([Uf; Z; Yf]',0);
            L = R(1:4*s,1:4*s)';
            L32 = L(3*s+1:end,s+1:3*s);

            % SVD of L32
            [U,S,~] = svd(L32,0);
    end

    % Storing singular values in output 'sv'
    sv = diag(S);
    %semilogy(sv,'x')

    % Computing A by linear least squares
    A = U(1:s-1,1:n)\U(2:s,1:n);

    % Extracting C
    C = U(1,1:n);
```

```
    % Construction of the phi matrix transpose
    m = size(u,2);
    l = size(y,2);
    phi_t = zeros(size(y,1),n+n*m+l*m);
    for k=0:size(y,1)-1
        phi_t1 = C*(A^k);
        phi_t2 = zeros(1,n*m);
        for j=0:k-1
            phi_t2 = phi_t2 + kron(u(j+1),C*(A^(k-1-j)));
        end
        phi_t3 = u(k+1);
        phi_t(k+1,:) = [phi_t1 phi_t2 phi_t3];
    end

    % Linear least squares to get the solution vector
    sol = phi_t\y;

    % Extracting information from the vector
    x0 = sol(1:n);
    B = sol(n+1:end-1);
    D = sol(end);
end


function y_filt = spike_filter(y)

    % Calculating the signal mean and standard deviation

    std_dev = std(y);
    m = mean(y);

    y_filt = y;

    % Initialize the range of points of a spike (ws), including the
 valid
    % signal values before and after the spike

    ws_start = 0;
    ws_end = 0;

    for i=1:size(y,1)
       if (y(i)>m+std_dev*4)
           if (ws_start == 0)
               ws_start = i-1;
           else
               % Nothing to do
           end
       else
           if (ws_start ~= 0)
               ws_end = i;

               % Doing the linear interpolation
               x = [ws_start; ws_end];
               v = [y(ws_start); y(ws_end)];
```

```
                xq = ws_start:ws_end;
                vq1 = interp1(x,v,xq);

                % Substituting with the correct values
                y_filt(xq) = vq1;

                ws_start = 0;
                ws_end = 0;
            end
        end
    end

    %Case in which the spike is located at the end of the signal:
 spike
    %points are directly subsituted by the previous allowed value.
    if (ws_start ~= 0)
        y_filt(ws_start:end) = y_filt(ws_start);
    end

end

function v = vaf(y, y_est)
    % Variance Accounted For (VAF) | Percentage value (%)
    % y     : measured output
    % y_est : estimated output

    v = var(y - y_est) / var(y) ;
    v = 100 * ( 1 - v );

    if ( v < 0 )
        v = 0;
    end
end
```

```
ans =

   99.9999
```

Published with MATLAB® R2020a