Homework 1 - Advanced Digital Signal Processing

Marco Ceran

A.A. 2020-2021

1 Introduction

The purpose of this homework is to investigate the correct orders in estimation by MSE analysis, and implement linear filtering in the MATLAB software environment; in particular the goal is to perform system identification of various filters, and to quantify the variation of accuracy, measured with the Mean Square Error metric, for varying lengths of the estimated filter. This body of work covers both SISO (Single-Input and Single-Output) and MIMO (Multiple-Input and Multiple-Output) system identification, with both causal and noncausal systems.

2 Execution

2.1 a.1)

We have a SISO system $y[n] = h[n] * x[n] + w[n]$ where the h[n] filer is characterized by the transfer function:

$$H_1(z) = \frac{1}{1 - 0.9z^{-1}}$$

x[n] is an unitary power zero-mean white gaussian process, while *w[n]* is an AR(1) (autoregressive of order 1) noise process with autocorrelation $r_{ww}[k] = \sigma_w^2(0.95^{|k|})$ and we want to estimate a filter $\hat{h}[n]$ with length p and compare the Mean Square Error vs p for values of Signal to Noise Ratio

$$SNR = \frac{r_{hh}[0]}{\sigma_w^2} = [-20, -10, 0, 10, 20, 30]dB$$

These vectors represent the different values of the SNR:

```
SNRdB = [-20, -10, 0, 10, 20, 30];
SNRlin = 10.^(SNRdB/20);        % divide by 20 instead of 10 because
                                % we multiply the root power quantities
```

Being the autocorrelation of the noise $r_{ww}[k] = \sigma_w^2(0.95^{|k|})$, its value in zero $r_{ww}[0] = \sigma_w^2(0.95^0) = \sigma_w^2$ is the power of the noise. Since the autocorrelation of a signal is the convolution of the signal with the same signal flipped along the time axis, the way to obtain a unitary power noise process with this autocorrelation function is to generate a Gaussian noise vector and pass it through a filter with a transfer function and impulse response (i.e. a causal monolateral exponential impulse response):

$$H_w(z) = \frac{1}{1 - 0.95z^{-1}} \qquad h_w[n] = 0.95^n u[n]$$

and normalize the signal power to 1 after the filtering (its power will be tuned to have the right values of SNR later on). In MATLAB code this operation is implemented as:

```
w0 = randn([Nx 1]);
w(1) = w0(1);
for ii = 2:Nx
    w(ii) = w0(ii) + 0.95*w(ii-1);
end
w0 = w./sqrt(mean(w.^2));   % normalize noise power
```

The w_0 vector refers to the unitary power ($\sigma_w^2 = 1$) noise with the aforementioned autocorrelation function. This vector w_0 will be multiplied to an appropriate factor each time a specific SNR will be needed. In order to compute the *y[n]* process a random Gaussian vector x of length N_x is computed and passed through the $H_1(z)$ filter, implemented with a for loop that iteratively applies the defining linear difference equation with constant coefficients of the filter. The previously computed noise w_0 (multiplied by a term that sets the right SNR, exploiting the fact that the power of a signal is the value of its autocorrelation in zero, and this value is the maximum value of the autocorellation) is then added to the result to yield the desired y vector.
In MATLAB code this process can be expressed as:

```
h = 0.9.^(0:p_val(i)-1)';
x = randn([Nx 1]);
y_noiseless = zeros(Nx, 1);
y_noiseless(1) = x(1);
for ii = 2:Nx
     y_noiseless(ii) = x(ii) + 0.9*y_noiseless(ii-1);
end
w = sqrt(max(conv(h, flip(h))))*w0./SNRlin(k);
y_noise = y_noiseless + w;  % this is our desired y signal
```

In order to perform the identification of the filter's impulse response, given the input x and (noisy) output y signals the following procedure has been used:

$$\hat{\mathbf{h}} = (\mathbf{X}^T\mathbf{C}_{ww}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}_{ww}^{-1}\mathbf{y}$$

with $\mathbf{C}_{ww}$ being the noise covariance matrix and $\mathbf{X}$ being the causal convolution matrix obtained from x as:

$$\mathbf{X} = \begin{bmatrix} x[1] & 0 & ... & 0 \\ x[2] & x[1] & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ x[N-p+1] & x[N-p] & & x[1] \\ 0 & x[N-p+1] & & x[2] \\ 0 & 0 & \ddots & x[N-p] \\ \vdots & \vdots & ... & x[N-p+1] \end{bmatrix}$$

The MSE (evaluated for different values of the est. filter length p) has been computed as the mean of the squared difference between the true filter's impulse response and the estimated filter's one, as following:

$$MSE_h = \frac{1}{l}\sum_{n=1}^{l}(h_1[n] - \hat{h}_1[n])^2$$

Where l is the length of the vector that contains the first l elements of the infinite impulse response of the ideal filter *h[n]*, in general $l \gg p$, $\hat{h}[n]$ is considered to be zero for values of n greater than p, because it's a truncated (FIR) filter of length p; since the true impulse response *h[n]* is exponentially decaying the values of the impulse response at indexes approaching l are so small that the accuracy loss of computing the MSE in this way instead of subtracting the estimated FIR impulse response to the ideal IIR impulse response is really negligible.
An alternative path would be to compute the MSE as the mean of the squared difference of the signal y (the one we computed before) and $\hat{y}[n] = x[n] * \hat{h}[n]$. With this approach the optimum estimated filter's length p that minimizes the MSE may be different from the p that minimizes the MSE_h computed on the impulse responses.
Repeated MSE computation show variations that are due to the particular realizations of the white Gaussian random processes *x[n]* and *w[n]*, so a Montecarlo method is used to extract average values that are more accurate as the number of iterations increases. The MSE that is going to be considered is the average of the MSE values that we find at each iteration.
A goal of this project activity will thus be to find the value of p that minimizes MSE, and to compare the estimated filter's impulse response of that length to the true filter's impulse response, and to also compare the singularities (poles and zeros) of these two systems.

2.2 a.2)

This case is similar to the previous one, in this case we have to estimate the filter:

$$H_2(z) = \frac{1}{(1 - 0.8e^{-j\pi/4}z^{-1})(1 - 0.8e^{j\pi/4}z^{-1})}$$

That is a AR(2) causal system; the *y[n]* signal is obtained in the same way as before, and the *w[n]* noise term holds the same properties. The estimation is carried out using a causal convolution matrix constructed in the same way as before.

2.3 a.3)

The third system that is going to be estimated is:

$$H_3(z) = H_2^*(1/z) = \frac{1}{(1 - 0.8e^{-j\pi/4}z)(1 - 0.8e^{j\pi/4}z)}$$

It is immediately evident that this system is no more causal, and it is instead anticausal, meaning that the linear difference equation that defines the system requires, in order to compute the value of the output at a certain time instant the knowledge of values at future time in stants.
In order compute the noiseless version of the y signal in MATLAB the following code has been used:

```
x = randn([Nx 1]);
y_noiseless = zeros(Nx, 1);
y_noiseless(end-1:end) = x(end-1:end);
for ii = Nx-2:-1:1
    y_noiseless(ii)=x(ii)+0.8*sqrt(2)*y_noiseless(ii+1)-0.64*y_noiseless(ii+2);
end
```

The iteration that applies the filter starts from the last values and proceeds backwards to the first ones (anticausal system behave just like causal one if you reverse the direction of time arrow).
Another difference compared to the causal cases is in the convolution matrix $\mathbf{X}$ that is obtained as:

$$\mathbf{X} = \begin{bmatrix} 0 & \dots & 0 & x[1] \\ \vdots & & x[1] & x[2] \\ 0 & \vdots & \vdots & \vdots \\ x[1] & & x[N-p] & x[N-p+1] \\ x[2] & & x[N-p+1] & 0 \\ \vdots & & 0 & 0 \\ x[N-p] & \vdots & 0 & 0 \\ x[N-p+1] & \dots & \vdots & \vdots \end{bmatrix}$$

As we can see the convolution matrix for anticausal system has copies of the x input vector running through the secondary diagonal instead of the main one. The noise term is also the same as in the previous cases. The estimated filter's impulse response has been obtained using the same formula as in the previous cases, but the result has been flipped in order to be coherent with the fact that it is the impulse response of an anticausal system:

```
h_est = flip(pinv(X'*cwwinv*X)*X'*cwwinv*y);
```

2.4 a.4)

The fourth and last SISO system is composed of the cascade of the second and third systems, having the transfer function:

$$H_4(z) = H_2(z)H_3(z)$$

$$= \frac{1}{(1-0.8e^{-j\pi/4}z)(1-0.8e^{j\pi/4}z)(1-0.8e^{-j\pi/4}z^{-1})(1-0.8e^{j\pi/4}z^{-1})}$$

This system is clearly noncausal, having both a causal and an anticausal component, and the straightforward way to apply it to a gaussian vector x in MATLAB is to first implement the $H_3(z)$ and then the $H_2(z)$ system one after the other (the order of the linear filters in a cascade configuration is arbitrary, being the cascade configuration of two system commutative):

```
x = randn([Nx+max(p_val) 1]);
y_noiseless = zeros(length(x), 1);
dumm_y = y_noiseless;
dumm_y(end-1:end) = x(end-1:end);
for ii = 3:length(x)
    dumm_y(ii) = x(ii) + 0.8*sqrt(2)*dumm_y(ii-1) - 0.64*dumm_y(ii-2);
end
y_noiseless(1:2) = dumm_y(1:2);
for ii = length(x)-2:-1:1
    y_noiseless(ii) = dumm_y(ii) + 0.8*sqrt(2)*(y_noiseless(ii+1)) -
    0.64*y_noiseless(ii+2);
end
```

The *dumm_y* vector is a throwaway variable that serves the purpose of representing the intermediate output of the $H_2(z)$ system before it is fed to the $H_3(z)$ system.
In order to perform the $H_4(z)$ system identification we can't simply use the identification matrixes seen for the first three filters, because we are now dealing with a noncausal system. The idea is instead to delay the signal x by a quantity equal to half the estimated filter's length before performing the system identification. In this way we can use the causal estimation matrix seen with the first two filters and get accurate results even if the system is not causal. In order to obtain the new desired x and y with the aforementioned characteristics from the ones previously computed this code has been used:

```
y = y(1:end-(p_val(i)-1)/2);
x = x(1+(p_val(i)-1)/2:end);
```

2.5 a.5)

In order to plot the singularities (poles and zeros) of the above systems the `pzplot()` MATLAB function has been used in conjunction with the `tf()` function. The latter accepts as inputs the coefficients of a transfer function or the values of an impulse response and returns a transfer function model whose singularities are then plotted by the former function. All of the four systems seen up to this point are AR (autoregressive) systems, meaning that their singularities are all poles and no zeros, while the estimated impulse responses, being of finite length in order to be represented by MATLAB vectors, characterize FIR filters, and their singularities are thus all zeros.

2.6 a.6)

The Power Spectral Densities (PSDs) of the $y_{noiseless}[n] = x[n] * h[n]$ and *w[n]* processes are estimated by computing the fast Fourier transform (FFT) of the *h[n]* impulse responses and taking the sum of the squared real and imaginary parts of the FFT, in MATLAB code:

```
XH = fftshift(fft([h; zeros(length(x)-length(h), 1)]))/Nx;
W =  fftshift(fft(hw*sqrt(max(conv(h, flip(h))))/SNRlin(k)))/Nx;
psdXH = real(XH).^2 + imag(XH).^2;
psdW =  real(W).^2 + imag(W).^2;
fh = (-length(XH)/2: length(XH)/2-1) * (2*pi/length(XH));
```

Since the *x[n]* process is white and Gaussian we expect it to have a power spectral density constant w.r.t. the frequency axis and thus we expect the PSD of $y_{noiseless}[n] = x[n] * h[n]$ to have the same shape as the one of the system H(z).

2.7 a.7)

For the last task of the SISO system part of this homework the frequency response of the Wiener filter that estimates the input *x[n]* from the output *y[n]* will be shown.
The Wiener filter, that is the optimum MMSE estimator for WSS (Wide Sense Stationary) processes, is derived from the statistical properties of the *x[n]* and *y[n]* processes, and since it is the filter that estimates the former from the latter, we expect it to have a PSD inversely proportional to the one of the *h[n]* and $h[n] * x[n]$ processes in high SNR scenarios, while in low SNR scenarios we expect the Wiener filter to have a shape that is inverse to that of the noise PSD. This filter has been computed starting from the definition of Wiener filter and

by applying some simplifications:

$$\hat{A}(\omega) = \frac{\mathbb{E}[X(\omega)Y^*(1/\omega)]}{\mathbb{E}[Y(\omega)Y^*(1/\omega)]}$$

$$X(\omega) = \mathcal{F}\{x[n]\}$$
$$Y(\omega) = X(\omega)H(\omega) + W(\omega) = \mathcal{F}\{y[n]\}$$
$$Y^*(1/\omega) = X^*(1/\omega)H^*(1/\omega) + W^*(1/\omega)$$
$$X(\omega)Y^*(1/\omega) = X^*(1/\omega)X(\omega)H^*(1/\omega) + W^*(1/\omega)X(\omega)$$
$$X^*(1/\omega)X(\omega) = |X(\omega)|^2 = \sigma_x^2$$
$$X(\omega)Y^*(1/\omega) = \sigma_x^2 H^*(1/\omega) + W^*(1/\omega)X(\omega)$$
$$E[X(\omega)Y^*(1/\omega)] = \sigma_x^2 H^*(1/\omega)$$
$$Y(\omega)Y^*(1/\omega) = \sigma_x^2|H(\omega)|^2 + |W(\omega)|^2$$
$$\hat{A}(\omega) = \frac{\sigma_x^2 H^*(1/\omega)}{\sigma_x^2|H(\omega)|^2 + |W(\omega)|^2}$$

In MATLAB environment this has been implemented as:

```
xpower = mean(x.^2);
wpower = mean(w.^2);
H = fftshift(fft([h; zeros(length(y_noiseless)-length(h), 1)]))/length(y_noiseless);
Hconj = conj(H);
Hw = fftshift(fft(hw(1:length(H))))'/length(H);
psdH = abs(H).^2;
psdHw = abs(Hw).^2;
H_wiener = (xpower.*Hconj)./(xpower*psdH + wpower*psdHw);
```

We expect the Wiener filters to have a spectrum that is in high SNR scenarios inverse with respect to those of the $X(\omega)H(\omega)$ processes, while in low SNR scenarios we expect it to have a shape that takes the noise spectrum into account; this method of computing the Wiener filters has yielded good results in all of the 4 SISO systems we're dealing with, as it's shown in the results section.

2.8 b.1)

We are now dealing with a 2 x 2 MIMO system, expressed by:

$$\begin{cases} y_1[n] = h_{11}[n] * x_1[n] + h_{12} * x_2[n] + w_1[n] \\ y_2[n] = h_{21}[n] * x_1[n] + h_{22} * x_2[n] + w_2[n] \end{cases} \tag{1}$$

In which:

$$\begin{cases} h_{11}[n] = h_1[n] \\ h_{12}[n] = h_2[n] \\ h_{21}[n] = h_2[n] \\ h_{22}[n] = h_1[n] \end{cases} \tag{2}$$

where the terms on the right side are the SISO filters' impulse responses seen in the first part of the assignment, the noise terms are (different realization of) the same stochastic process $w[n]$ as in the SISO case, and the $x_1[n]$ and $x_2[n]$ are in the `HW1.mat` file.
The goal is that of performing the identification of a MIMO system for different length of estimated filter p and to evaluate which length minimizes the MSE between the true and estimated filter's impulse response for different values of Signal to Noise Ratio, defined as:

$$SNR = r_{h_{11}h_{11}}/\sigma_w^2$$

In order to perform the estimation of the MIMO impulse response we consider the input and output vectors:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \end{bmatrix}$$

for which the following relation holds true:

$$\begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H_1} & \boldsymbol{H_2} \\ \boldsymbol{H_2} & \boldsymbol{H_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X_1} & \boldsymbol{X_2} \\ \boldsymbol{X_2} & \boldsymbol{X_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{h_1} \\ \boldsymbol{h_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{X_1 h_1 + X_2 h_2} \\ \boldsymbol{X_2 h_1 + X_1 h_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \end{bmatrix}$$

where the uppercase letters inside the matrices represent convolution matrices as defined in the SISO case and thus we define:

$$\mathbf{h} = \begin{bmatrix} \boldsymbol{h_1} \\ \boldsymbol{h_2} \end{bmatrix} \quad \textit{and} \quad \mathbf{X} = \begin{bmatrix} \boldsymbol{X_1} & \boldsymbol{X_2} \\ \boldsymbol{X_2} & \boldsymbol{X_1} \end{bmatrix}$$

We can at this point estimate the filter's impulse responses as:

$$\hat{\mathbf{h}} = \begin{bmatrix} \hat{\boldsymbol{h}}_1 \\ \hat{\boldsymbol{h}}_2 \end{bmatrix} = (\mathbf{X}^T \mathbf{C}_{ww}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{C}_{ww}^{-1} \mathbf{y}$$

In MATLAB code this has been implemented as:

```
X = zeros(2*Nx + 2*p_val(j) - 2, 2*p_val(j));
for ii = 1:p_val(j)
    X(ii:Nx+ii-1, ii) = x(1:Nx, 1);
    X(p_val(j)+Nx-1+ii:p_val(j)+2*Nx-2+ii, ii) = x(1:Nx, 2);
    X(ii:Nx+ii-1, p_val(j) + ii) = x(1:Nx, 2);
```

```
    X(p_val(j)+Nx-1+ii:p_val(j)+2*Nx-2+ii,
                p_val(j) + ii) = x(1:Nx, 1);
end
h_est = pinv(X'*cwwinv(1:2*Nx + 2*p_val(j) - 2, 1:2*Nx + 2*p_val(j) - 2)*X)*X'
        *cwwinv(1:2*Nx + 2*p_val(j) - 2, 1:2*Nx + 2*p_val(j) - 2)
                *[y(1:Nx+p_val(j)-1, 1);y(1:Nx+p_val(j)-1, 2)];
h1_est = h_est(1:p_val(j));
h2_est = h_est(p_val(j)+1:end);
```

This way of doing identification reduces a MIMO identification problem into the SISO known case.

2.9 b.2)

Now the same MIMO system as before is considered, but the $h_{ij}[n]$ impulse responses are given by:

$$\begin{cases} h_{ij}[n] = \alpha^{|i-j|} e^{-|n|/4} & |n| = 0, 1, 2, 3 \\ h_{ij}[n] = 0 & otherwise \end{cases} \tag{3}$$

The $h_{ij}[n]$ impulse responses and FIR and their length is 7,The $h_{12}[n]$ and $h_21[n]$ impulse response are scaled versions of $h_{11}[n]$ and $h_{22}[n]$ by a factor α. A factor α of 0.6 has been used in the MATLAB implementation. The impulse response vector for $h_{11}[n]$ and $h_{22}[n]$ is:

$$\begin{bmatrix} e^{-3/4} & e^{-1/2} & e^{-1/4} & 1 & e^{-1/4} & e^{-1/2} & e^{-3/4} \end{bmatrix}^T$$

and we can easily derive $h_{12}[n]$ and $h_{21}[n]$ by multiplying this expression by α. In MATLAB the same code has been used for this identification as for the **b.1)** case. Since a FIR filter of length 7 is going to be estimated, we expect that the estimated impulse response that minimizes the MSE is also going to be of length 7.
When plotting the MSE of the estimated impulse responses vs the filter length p, the Mean Square Error will be compared to the asymptotic Cramer Rao Bound (CRB) obtained as:

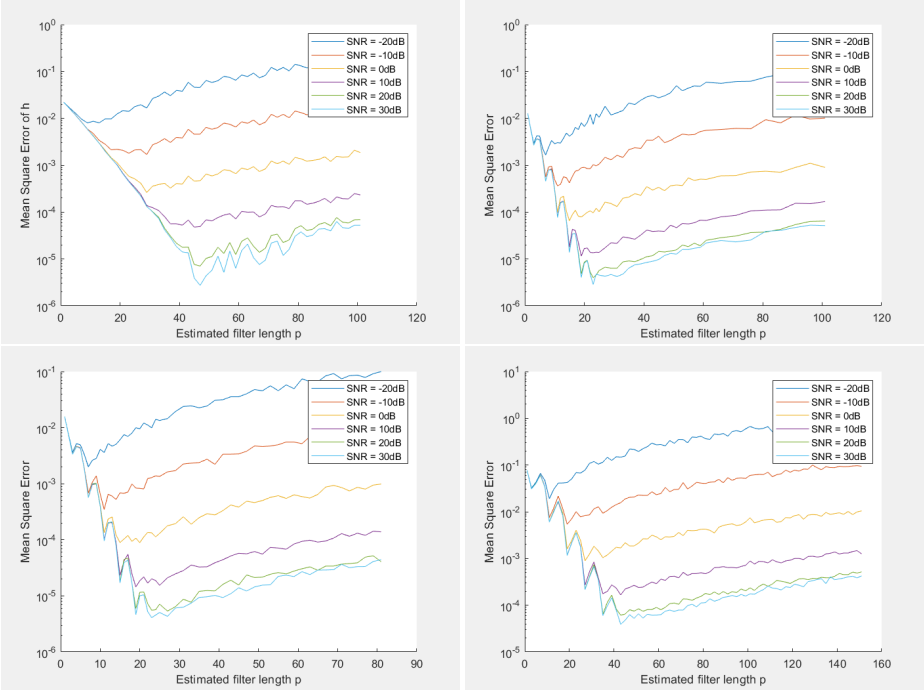
$$CRB = \sigma_w^2 (\mathbf{X}^T \mathbf{C}_{ww}^{-1} \mathbf{X})^{-1} = \frac{r_{hh}[0]}{SNR^2} (\mathbf{X}^T \mathbf{C}_{ww}^{-1} \mathbf{X})^{-1}$$

In the above formula the C_{ww} matrix is to be intended as unitary power, and the correct power of the noise autocorrelation matrix is obtained by multiplying C_{ww} by σ_w^2; in the identification procedure a length of x of 500 has been taken and 20 Monte Carlo iterations have been performed.
We expect of course the MSE to be lower bounded by the CRB.

3 Practical results

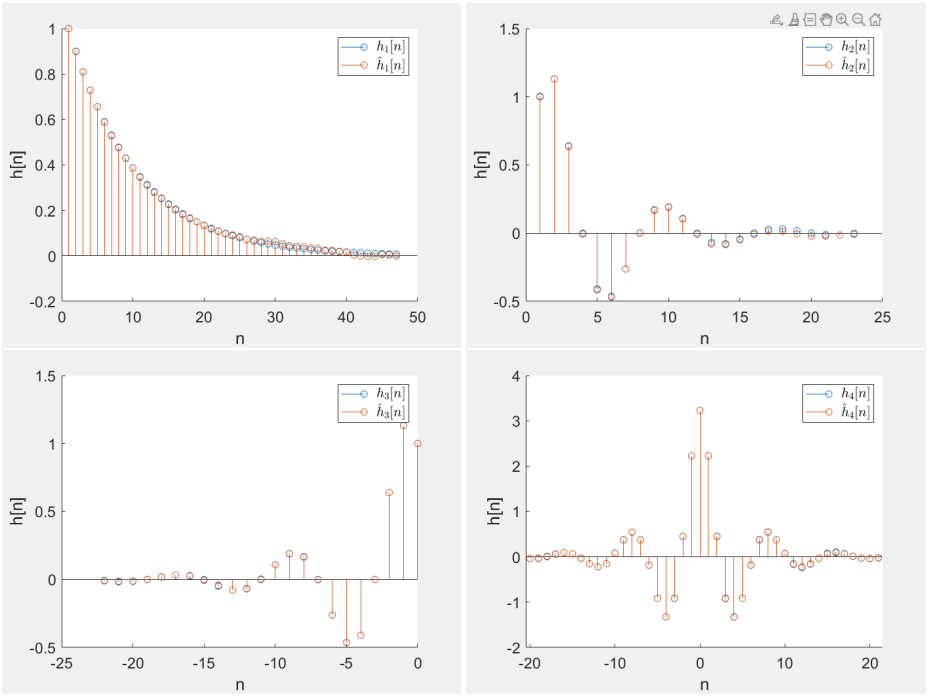
In this section graphs representing the practical results obtained from MATLAB computation of the above illustrated processes will be displayed. The results of

the SISO part will be shown before the MIMO ones, following the same order of the above explaination.

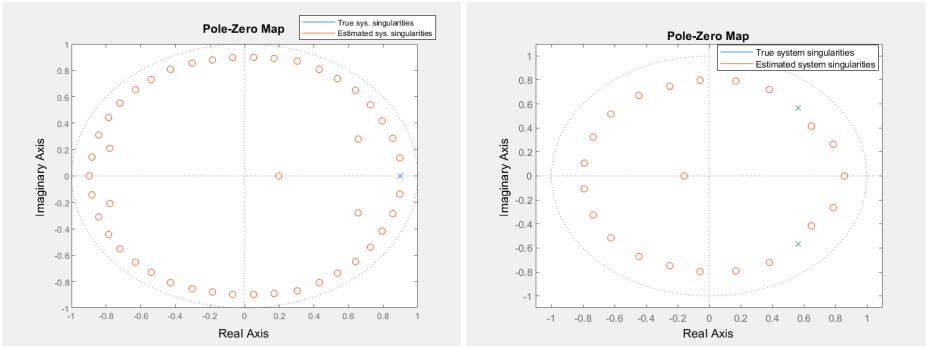


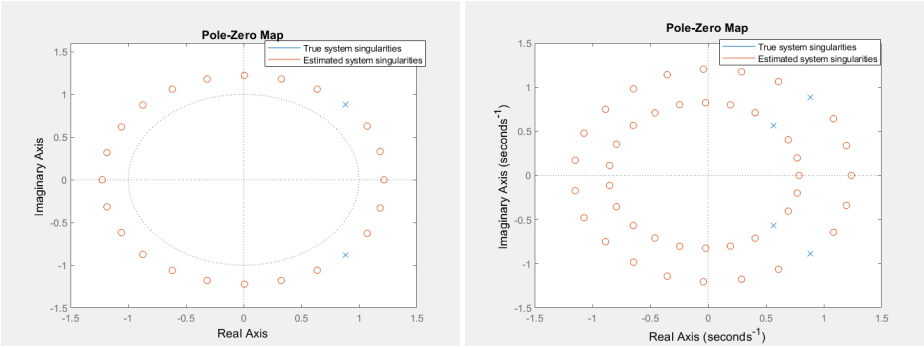
SNR(dB)	30	20	10	0	-10	-20
p^*_{h1}	47	47	45	29	29	9
p^*_{h2}	23	23	19	15	11	7
p^*_{h3}	23	25	19	20	11	7
p^*_{h4}	43	43	43	27	19	11

These four graphs show the MSE evaluated for varying values of the estimated filter length p, in all 4 cases the value of N_x (length of input signal x) is equal to 1500. The number of iterations is equal to 100 for the first 3 filters, while a value of 200 has been used for the fourth one, because the double process of estimation (causal and anticausal) leads to higher variance in the estimated filter results and thus requires a higher number of Montecarlo iterations to settle to a reasonable approximation of the true value of the MSE. For the four filters the optimum filter length p (for SNR = 30 dB) has been found to be respectively 49, 15, 15, 34, the above table reports the optimal filter length for every one of the 4 filter at each of the 6 SNR levels. The fact that $h_2[n]$ ans $h_3[n]$ share the same optimum estimated filter length is due to the fact that they are the same filter, except for a flip along the time axis.

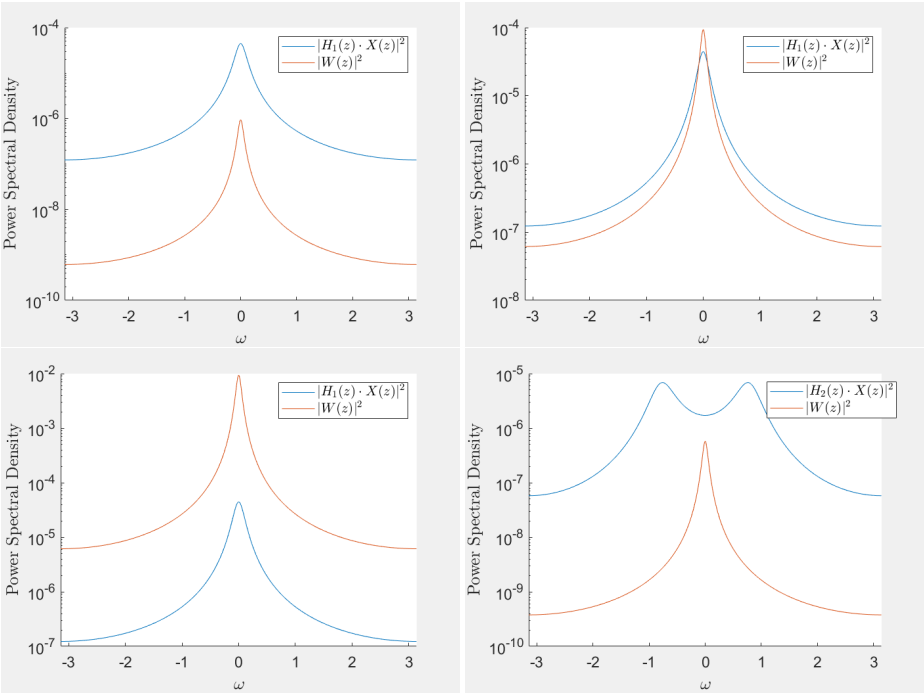


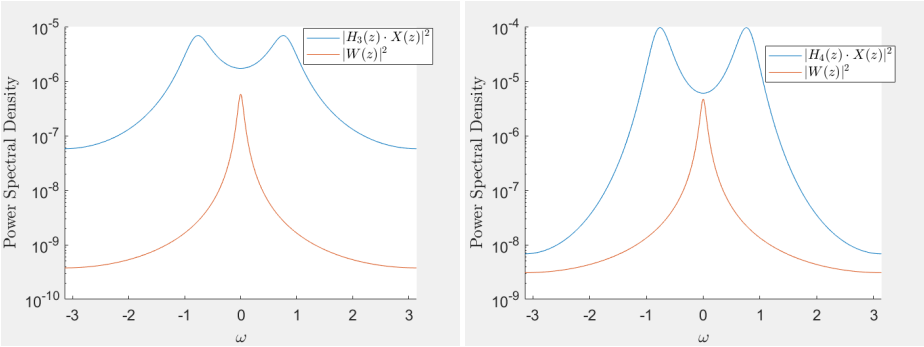
In the above graphs are reported the estimated impulse responses for the four filters (for length p that minimizes the MSE), with the maximum value of SNR (30dB).



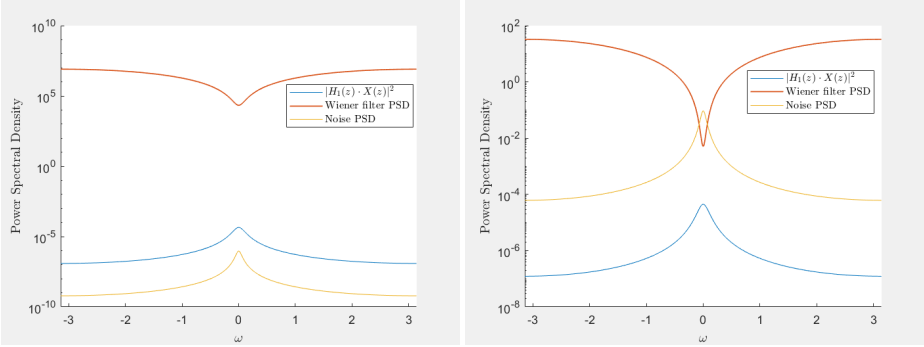


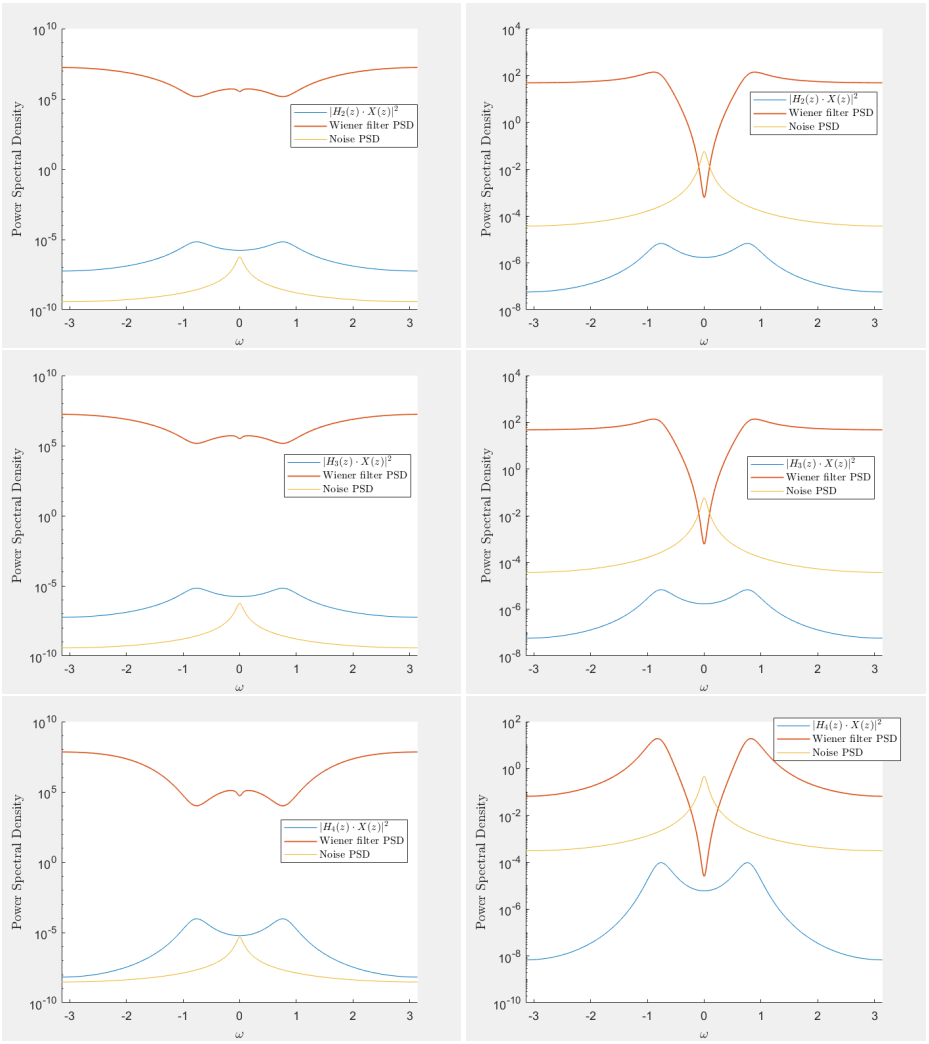
The poles (of the true systems) and zeros (of the estimated ones) are displayed, also for the optimum filter lenghts and the highest SNR. The number of poles of the true systems depends on the degree of the polynomial at the denominator of the transfer function, while the number of zeros of the estimated ones is $N_{zeros} = p - 1$ where p is the filter length, hence the number of zeros is equal to the order of the numerator polynomial in the transfer function. We can notice that zeros of the estimated filter response tend to be distributed with equal distance in circle around the pole(s) of the true filter. This is due to the fact that the p zeros' positions are the complex roots of the polynomial of order p whose indexes are (ideally) the first p elements of the true impulse response *h[n]*; zeros' positions differing from that circumference are due to the estimate of the $\hat{h}[n]$ filter being not perfect.





These six graphs represent the power spectral densities of the noise w (that is in fact the same in all 4 cases, except for power levels due to different values of SNR) and of the process $y_{noiseless} = x[n] * h[n]$; the first three graphs all represents the signal vs noise PSD plots of the first filter with values of SNR equal to 30, 10, -10 dB respectively. In the first cases the signal and noise PSDs have similar shape, due to both the signal and the noise being obtained by filtering a white Gaussian process with an AR(1) system, only difference between the two being position of the pole (0.9 for the signal and 0.95 for the noise). In the last 3 PSD plots only the higher SNR (30 dB) is shown, and the noise PSD is orders of magnitude lower than the signal's one as a consequence of that. The second and third signal share the same PSD, because the time axis flip that differentiates them doesn't have effects on the power spectral densities. In the fourth case the psd of the $y_{noiseless}$ signal is the sum of the second and third cases because of $H_4(z)$ being the cascade of $H_2(z)$ and $H_3(z)$ filters.





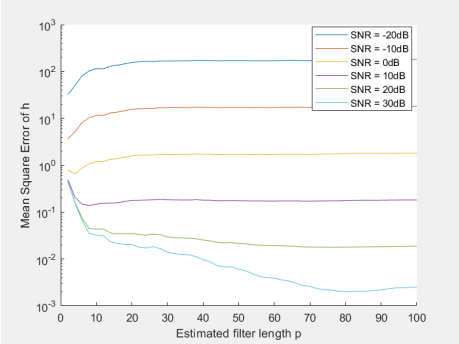
Here are pictured the Wiener filters that estimate the input signal *x[n]* from the output one *y[n]* in the four cases, both with 30 dB of SNR and with -20 dB of SNR (left and right pictures respectively) . These Wiener filters estimates get the inputs of the SISO system that have been analyzed from their output, thus having a PSD that is roughly the inverse compared to the $y_{noiseless}[n]$ ones in high SNR cases.

For the first MIMO identification problem there are four filters to be estimated, $h_{11}[n]$, $h_{22}[n]$, $h_{12}[n]$, $h_{21}[n]$, and the following relations hold:

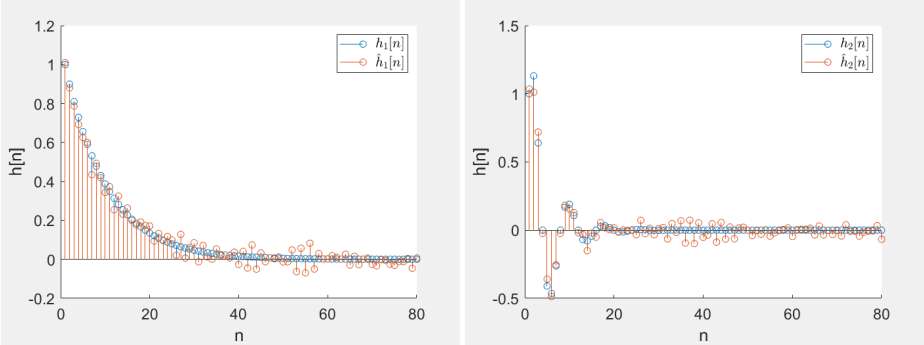
$$\begin{cases} h_{11}[n] = h_{22}[n] \\ h_{12}[n] = h_{21}[n] \end{cases} \tag{4}$$

reducing the number of impulse responses to be estimated to 2. The MSE of the

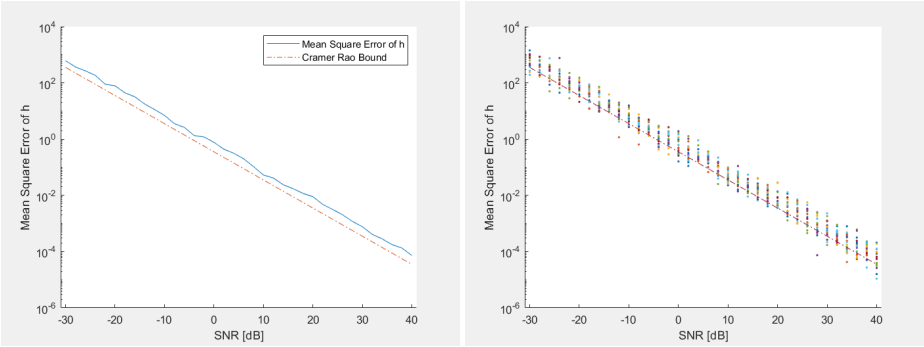
estimated impulse responses has been evaluated jointly for the 2 filters, so that there is an unique optimum value p that minimizes the MSE for both the filters:



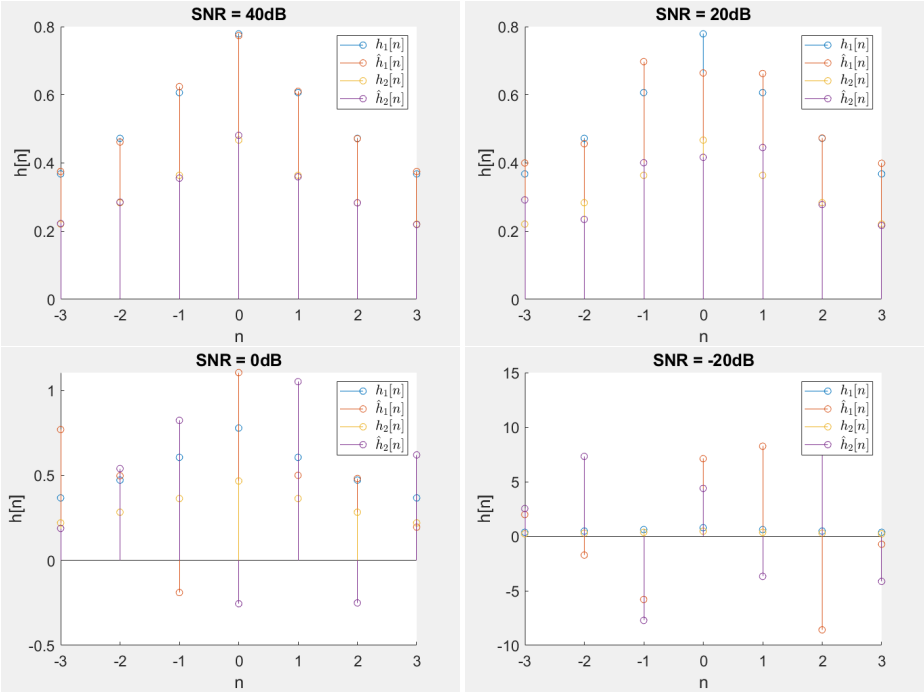
This is the graph that represents the MSE vs p for different values of SNR, and the value p that minimizes it for SNR = 30dB is p = 44, and here are reported the estimated impulse responses and the true ones for p = 44:



Even if the filters are the same as the first two ones in the SISO cases, the estimation is less precise, probably due to the length p being a compromise between the best filter length for the first impulse response and the second one, and estimating more filters at once may introduce inaccuracies in the results.
The second MIMO problem deals with the estimation of a MIMO filter that has finite length that equals 7, and instead of finding the optimum estimated filter's length p, the goal is to perform the identification for various values of SNR (the range from -40 dB to 50 dB has been chosen) and to compare the MSE with the CRB in these different cases.



These two graphs show this comparison, in the first case the MSE is displayed as a mean of the values obtained in the 20 iterations that have been performed in the Montecarlo simulation, while in the second graph the MSE values have been represented with a scatter plot that better captures the dispersion around the mean values. The first graph shows how the MSE is close to the CRB, meaning that the estimation is accurate.



These four graphs show the true impulse response of *h1[n]* vs the estimated filters for different values of SNR (-20, 0, 20 and 40 dB respectively) and the estimated ones, and it is clear how the estimation becomes more accurate for increasing values of the signal to noise ratio, .

4 Conclusion

This body of work has shown how to perform system identification by representing signals and impulse responses with vector and matrices, and how this method can be used in several different cases (SISO, MIMO, causal, anticausal and noncausal systems) and how systems identificated in this way compare with Wiener filters. In the MIMO case has been shown that this method of performing identification can yield results that are really close to the CRB.

5 Bibliography

- Spagnolini U., *Statistical Signal Processing in Engineering*, Wiley, 2018
- Luise M., Vitetta G. M., *Teoria dei Segnali*, Milano, McGraw-Hill, 2009