DSP - Laboratory 2

Daniele Orsuti
daniele.orsuti@phd.unipd.it

November 6, 2021

Abstract

Second laboratory experience [1] for the course Digital Signal Processing (DSP) a.y. 21/22. Topics: FIR filters design.

Before starting with the exercises make sure you have installed **Signal Processing Toolbox** (https://www.mathworks.com/products/signal.html.), and make sure you have headphones handy for the second exercise.

1 Window Method for FIR Filter Design

1.1 Background

The basic idea behind the window design is to choose a proper ideal frequency-selective filter (which always has a noncausal, infinite-duration impulse response) and then to truncate (or window) its impulse response to obtain a linear-phase and causal FIR filter. Therefore, in this method, particular care has to be placed in selecting an appropriate windowing function and an appropriate ideal filter.

In the time domain, windowing consists in multiplying the desired (usually ideal) infinite duration impulse response $h_d[n]$ by a finite duration window (or window function) $w[n]$ to get a soft truncation. The resulting impulse response $h[n]$ of the designed filter is the product

$$h[n] = h_d[n] \cdot w[n] \, , 0 \leq n \leq M-1$$

where M is the number of filter taps. Figure 1 shows a sinc function windowed using a rectangular window, and Figure 2 shows the effects of typical windows (i.e., Rectangular, Barlett, Hanning, Hamming, Blackman) on the magnitude of the frequency response.

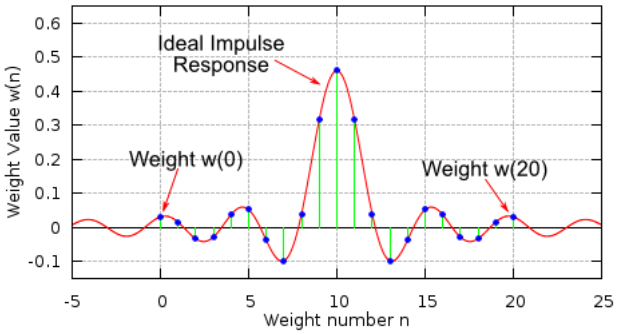

Figure 1: Working principle of time domain window method. *Weights* in the figure correspond to filter coefficients (or taps). In the figure the ideal impulse response is windowed with a rectangular window (i.e., only 21 samples of the ideal response are retained).

[1] All information and material presented here is intended to be used for educational or informational purposes only.

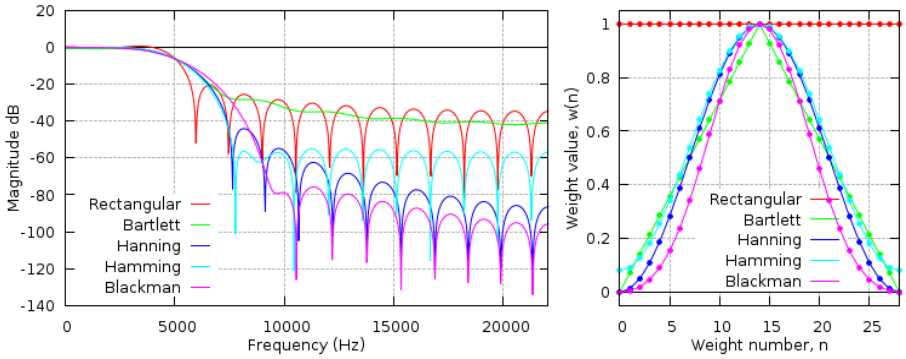

Figure 2: The left figure shows the amplitude of the frequency response of a low pass filter designed through the window method. The different windows shown on the right figure are adopted to perform the windowing operation. The results in term of stop-band attenuation and transition frequency range from pass-band to stop-band can be appreciated. For instance, the rectangular window (in red on the right figure) provides high side lobes level compared to the other windows.

1.2 First task

Design a linear phase FIR lowpass filter with the following specifications (Figure 3 shows filter design specification for a lowpass filter):

- $\omega_p = 0.2\pi$
- $\omega_s = 0.3\pi$
- $R_p = -20log_{10}\frac{1-\delta_p}{1+\delta_p} = 0.25$ dB
- $A_s = -20log_{10}\frac{\delta_s}{1+\delta_p} = 50$ dB

where R_p and A_s are pass-band ripple in dB and stop-band attenuation in dB, respectively. The latter quantities are relative specifications (i.e., in dB) for the filter. Note that in both expressions the denominator is $max|H(e^{i\hat{\omega}})| = 1 + \delta_p$.

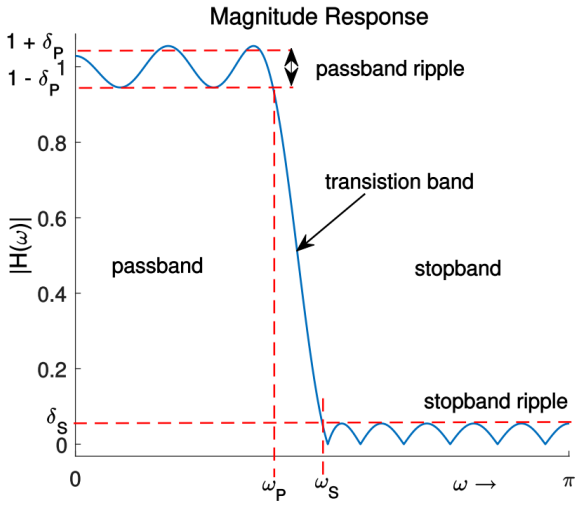

Figure 3: Filter design specifications.$[0\,, \omega_p]$: passband; δ_p: pass-band ripple; $[\omega_s\,, \pi]$: stopband; δ_s: stop-band ripple; $[\omega_p\,, \omega_s]$ transition width $(\Delta\omega)$;

The steps to follow are described in the following:

- We start by designing the filter using a rectangular window and we set $M = 45$. An ideal low pass filter of bandwidth $\omega_c < \pi$ is given by

$$H_d(\hat{\omega}) = H_d(e^{j\hat{\omega}}) = \begin{cases} 1e^{-j\alpha w}, & \text{if } |\hat{\omega}| \leq \omega_c \\ 0, & \text{if } \omega_c < |\hat{\omega}| < \pi \end{cases}$$

where ω_c ($\omega_c = (\omega_p + \omega_s)/2$) is called cut-off frequency and α is the sample delay. The impulse response can be easily shown to be

$$h_d[n] = \frac{\omega_c}{\pi} \frac{\sin \omega_c(n-\alpha)}{\omega_c(n-\alpha)},$$

which is symmetric with respect to α. The impulse response has a sinc shape which is noncausal and infinite in duration. To obtain an FIR filter from $h_d[n]$, one has to truncate $h_d[n]$ on both sides. To obtain a causal and linear-phase FIR filter $h[n]$ of length M, we must have,

$$h[n] = h_d[n] \cdot w_R[n] = \begin{cases} h_d[n], & \text{if } 0 \leq n \leq M-1 \\ 0, & \text{otherwise} \end{cases}, \quad \alpha = \frac{M-1}{2}. \tag{1}$$

where $w_R[n]$ stands for rectangular window. Using Equation 1 determine the impulse response and provide a plot of the frequency response of the designed filter (use *freqz* command https://www.mathworks.com/help/signal/ref/freqz.html). Modify the value of M from 45 to 91 and check how the magnitude of the frequency response changes.

- Now we modify the window function to obtain the required filter specifications. Find δ_p, δ_s from the given specifications of R_p and A_s. Choose an appropriate window function from Figure 4. Remember to update the value of M (evaluate it from the table and round it to the smallest *odd* integer value). For the expression of the selected window function, please refer to: https://en.wikipedia.org/wiki/Window_function.

Window Name	*Transition Width* $\Delta\omega$ *Approximate*	*Exact Values*	*Min. Stopband Attenuation*
Rectangular	$\frac{4\pi}{M}$	$\frac{1.8\pi}{M}$	21 dB
Bartlett	$\frac{8\pi}{M}$	$\frac{6.1\pi}{M}$	25 dB
Hann	$\frac{8\pi}{M}$	$\frac{6.2\pi}{M}$	44 dB
Hamming	$\frac{8\pi}{M}$	$\frac{6.6\pi}{M}$	53 dB
Blackman	$\frac{12\pi}{M}$	$\frac{11\pi}{M}$	74 dB

Figure 4: Window functions characteristics.

- Compare the results obtained using the rectangular $w_R[n]$ window to the one obtained with the window selected on the previous point.

Suggestion: It is a good practice to check numerically that stopband attenuation and passband ripple requirements have been satisfied. You can write a simple MATLAB code which operates on the magnitude of the frequency response to achieve this.

1.3 Second task

A band-pass filter can be designed starting from the previous task. The filter specifications are given in Figure 5. Figure 6 provides a pictorial representation of the filter specifications, whereas Figure 7 shows the method to realize a band-pass filter starting from two low pass filters. Proceed as in the previous task to design the two required filter impulse responses; then combine them to obtain the overall impulse response. Note that the filter specifications require to use, for instance, the Blackmann window (use the MATLAB function defined here https://www.mathworks.com/help/signal/ref/blackman.html to obtain the window).
Provide a plot of the impulse response and the frequency response of the designed filter.

$$
\begin{aligned}
&\text{lower stopband edge: } \omega_{1s} = 0.2\pi, \quad A_s = 60 \text{ dB} \\
&\text{lower passband edge: } \omega_{1p} = 0.35\pi, \quad R_p = 1 \text{ dB} \\
&\text{upper passband edge: } \omega_{2p} = 0.65\pi \quad R_p = 1 \text{ dB} \\
&\text{upper stopband edge: } \omega_{2s} = 0.8\pi \quad A_s = 60 \text{ dB}
\end{aligned}
$$

Figure 5: Band-pass filter specifications for second task.

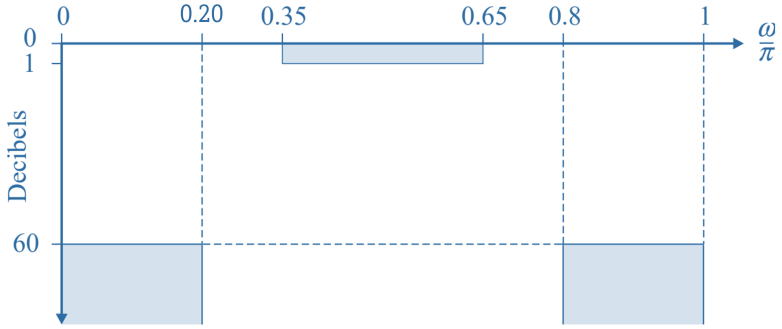

Figure 6: Illustration of the filter specifications for second task.

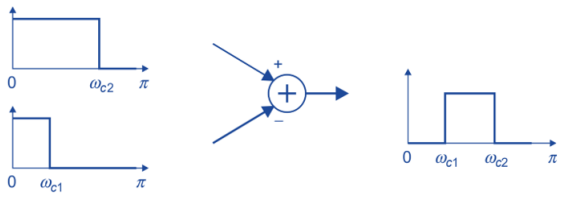

Figure 7: Method to design band-pass filter for second task.

1.4 Third task

Up to now FIR filters have been designed using the window method. Properly selecting the window function allowed to meet the required specifications. However, we would like to gain more flexibility: for instance, we would like to 1) design filters which can meet the required specifications with the lowest order M; 2) specify the band frequencies ω_p and ω_s precisely in the design instead of accepting whatever values we obtain after the design; 3) have asymmetric stop band attenuations when designing band-pass filters; and so on. For linear-phase FIR filters, it is possible to derive a set of conditions for which it can be proven that the design solution is optimal in the sense of minimizing the maximum approximation error (sometimes called the minimax or the Chebyshev error). Filters that have this property are called equiripple filters because the approximation error is uniformly distributed in both the passband and the stopband. This results in lower-order filters. A general equiripple FIR filter design algorithm is the so-called *Parks-McClellan* algorithm, and it incorporates the *Remez* exchange algorithm for polynomial solution. This algorithm is available as a subroutine in MATLAB (https://www.mathworks.com/help/signal/ref/firpm.html).

We now design the FIR filter of the previous task by using the Parks-McClellan algorithm.

- Use the function *firpmord* to retrieve the minimum filter order $N = M - 1$ which meets the required specifications (i.e., the one defined in the second task). *Hints*: please, look at the 'dev' input argument of the function *firpmord*. Below it is provided a FIR design example which requires at least 40 dB of attenuation in the stopbands and less than 3 dB of ripple in the passband.

```
Rp=3; %passband ripple
Rs=40; %stopband ripple
```

```
dev=[10^(-Rs/20) delta_p 10^(-Rs/20)];
```

- Use the function *firpm*, with the value of N obtained in the previous step, to find the filter coefficients.
- Plot the frequency response and check if the specifications are met. If the specifications are not met, correct the value of N (for instance increase it by 2). Compare the filter order used in this task to the one of the previous task.

1.5 Fourth task - (Optional): filtering sampled sinusoidal signal with noise

In this application, a digital FIR filter removes noise in a signal that is contaminated by noise existing in a broad frequency range. For example, such noise often appears during the data acquisition process. In real-world applications, the desired signal usually occupies a certain frequency range. We can design a digital filter to remove frequency components other than the desired frequency range. In a data acquisition system, we record a 500 Hz sine wave at a sampling rate of 8,000 samples per second ($f_s = 8000$ Hz). The signal is corrupted by broadband noise $r[n]$:

$$x[n] = 1.4141 \cdot \sin(2\pi \cdot \frac{500n}{8,000}) + r[n]\,.$$

In MATLAB $x[n]$ and $r[n]$ read:

```
T = 1/fs;
n = 0:1:249;
r = sqrt(0.1)*randn(1,250);
x = sqrt(2)* sin(2*pi*500*n*T)+r;
```

As we will see, lowpass filtering will remove noise in the range 1,000 Hz to 4,000 Hz , and hence the signal to noise power ratio will be improved. Assuming that the desired signal has a frequency range of only 0 to 800 Hz, we can filter noise from 800 Hz and beyond. A lowpass filter would complete such a task, use one of the low pass filter designed in the previous tasks. The filter specifications reads:

- Passband frequency range: 0 Hz to 800 Hz with the passband ripple less than 0.02 dB
- Stopband frequency range: 1 kHz to 4 kHz with 50 dB attenuation

Plot the time domain signal and its frequency response before and after filtering (use the *'filter'* command for the filtering operation).

2 FIR Notch filters for tones removal in audio signals

Caveat: Please use headphones when listening to the audio signal. It is highly suggested to start with a low volume to avoid the sound bothering you.

In this second exercise we are going to design a simple FIR filter (actually, a cascade of FIR filters) to remove annoying sounds (sinusoidal tones) from an audio signal. Although the impulse response of the filter is very simple, we can still achieve the desired denoising operation. An alternative approach would be to refer to the previous exercise to design a band stop filter (willing students can enjoy trying out), hereafter the straightforward approach (with simple impulse response) is adopted. Figure 8 shows the typical shape of the magnitude of the frequency response of a notch filter used to attenuate a particular frequency.

Objective: Write a MATLAB program that removes unwanted tones from an audio file. The file *'SunshineSquare.wav'* has had some unwanted tones added. Our job is to remove the tones so you can hear the message in a clearer way.

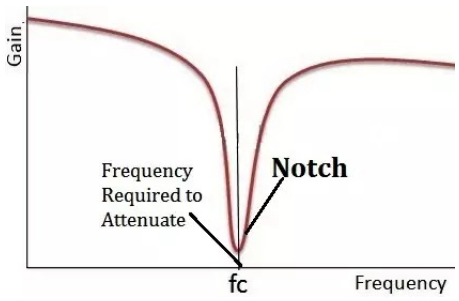

Figure 8: Example: magnitude of the frequency response of a notch filter.

Approach: There at two steps needed to remove the tones. First, determine the frequencies of the interfering tones, and second, filter out those frequencies.

To read the audio file and plot the spectrogram of it (https://www.mathworks.com/help/signal/ref/spectrogram.html). Use the following code.

```
[xx, fs] = audioread('SunshineSquare.wav');
figure;
spectrogram(xx,fs);
```

Note, fs is the sampling rate of the wav file and is important to specify it.

A weighted three-point averager is enough to remove one frequency at a time. Given the impulse response:

$$h[n] = [1, A, 1]$$

find the frequency response $H(z) = H(e^{j\hat{\omega}})$ in terms of A (see the first task). Find the values of A needed to remove each of the unwanted frequencies. Once you have the correct values, this code can be used to remove one frequency at a time:

```
hh = [1, AA, 1];
yy = filter(hh, 1, xx);
```

You will have to fill in your values for **AA**. You can check the frequency response of your filter by using freqz:

```
ww = -pi:pi/100:pi; % 1/100 is the frequency step
HH = freqz(hh, 1 ,ww); % 1 = a
plot(ww ,20*log10(abs(HH)));
```

Hint: You will have to use multiple filters. Once you have it working, combine those filters into one filter.

2.1 First task

From the spectrogram plot determine the frequencies associated with the sinusoidal disturbing tones.

2.2 Second task

Let's find the impulse response expression. If we want to eliminate a sinusoidal input signal, then we actually have to remove two signals, since

$$x[n] = \cos(\hat{\omega}_0 n) = \frac{1}{2}\left(e^{j\hat{\omega}_0}\right)^n + \frac{1}{2}\left(e^{-j\hat{\omega}_0}\right)^n .$$

Each complex exponential can be removed with a first-order FIR filter, and then the two filters would be cascaded to form the second-order nulling filter that removes the cosine. The first complex exponential signal is nulled by a filter with system function

$$H_1(z) = H_1(e^{j\hat{\omega}}) = 1 - z_1 z^{-1}$$

where $z_1 = e^{j\hat{\omega}_0}$ because $H_1(z_1) = 0$ at $z_1 = z_1$. Similarly $H_s(z) = 1 - z_2 z^{-1}$ removes the second complex exponential. By cascading the two filters, we obtain:

$$H(z) = H_1(z) \cdot H_2(z) = \cdots = 1 - 2\cos(\hat{\omega}_o) z^{-1} + z^{-2}\,. \tag{2}$$

The final results in Equation 2 follows from straightforward algebraic operations (it is suggested to check the final result).

Normalize Equation 2 to obtain a unit DC gain frequency response (DC corresponds to $\hat{\omega} = 0$ or equivalently to $z = e^{j\hat{\omega}} = 1$).

2.3 Third task

Design the filters with adequate coefficients. Plot the overall frequency response obtained by cascading the designed filters.

2.4 Fourth task

Filter the audio signal, plot the spectrogram of it and reproduce the original and filtered signal. Does the filtering operation produce satisfactory results ?

3 Differentiator filter (Optional)

In the previous laboratory session (UPC decoding exercise) we used a first order difference filter to determine the transitions from black bars to white spaces in the bar code. We now design a differentiator FIR filter using the window method, starting from the ideal impulse response, then we use the designed filter to evaluate the derivative of a known signal. The frequency response of an ideal differentiator is given by

$$H_d(e^{j\hat{\omega}}) = \begin{cases} j\hat{\omega}, & \text{if } 0 \le \hat{\omega} \le \pi \\ -j\hat{\omega}, & \text{if } -\pi < \hat{\omega} < 0 \end{cases},$$

the ideal impulse response can be obtained performing the inverse Fourier transform:

$$h_d[n] = \begin{cases} \frac{\cos \pi(n-\alpha)}{(n-\alpha)}, & \text{if } n \neq \alpha \\ 0, & \text{if } n = \alpha \end{cases}, \quad 0 \le n \le M-1\,, \alpha = \frac{M-1}{2}\,. \tag{3}$$

Such a differentiator cannot be realized exactly in practice. Thus approximate implementations are used. Figure 9 shows the frequency responses of different types of differentiators.

Besides the ideal response, we can observe the first-order difference and central difference responses. Their input-output relationship are given by $y[n] = x[n] - x[n-1]$ and $y[n] = 0.5x[n] - 0.5x[n-2]$, respectively. Notice that both filters approximate the ideal response very well at low frequencies. Actually, a differentiator is usually designed to be used at low frequency because the high response at high frequency will enhance noise. This makes the central differentiator a good candidate. We now proceed with the window method and check the frequency response of the designed filter.

3.1 First task

Use Equation 3 using the Hamming window to design the differentiator, with $M = 21$. Plot the impulse/frequency response of the designed filter.

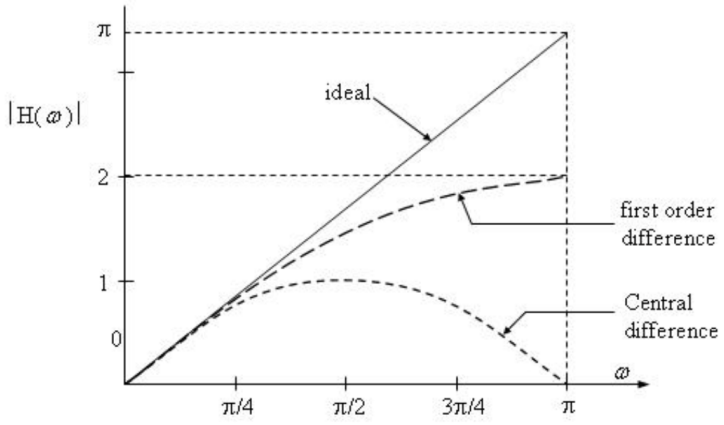

Figure 9: Different types of differentiators.

3.2 Second task

Use the designed filter to filter a signal with a known derivative (e.g., a sinusoidal signal). Plot on the same figure the expected derivative (analytical expression) and the actual derivative (obtained through filtering). Use *'filter'* command to perform the filtering operation. Remove the delay introduced by the FIR filter. Note how the transient response affects the output signal.

3.3 Third task

Truncate the ideal impulse response using a rectangular window instead of the Hamming window. Compare the frequency response and the output of the filter with the ones obtain using a Hamming window.