



Principi di Bioingegneria

A.A. 2022/23

Lezione 13

Analisi dei segnali biomedici
Esercitazione 3

Vincenzo Catrambone, PhD
vincenzo.catrambone@unipi.it

Analisi in frequenza dei segnali fisiologici

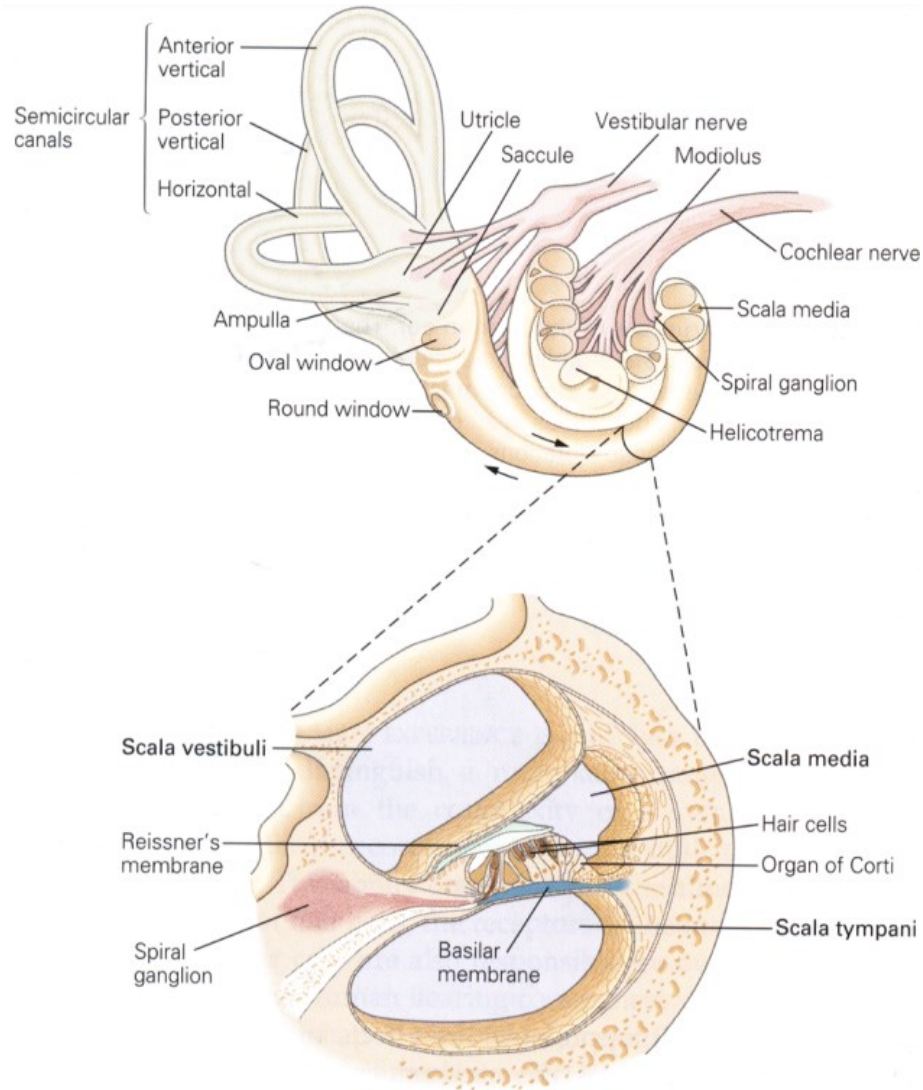
Filtering

Filtri FIR e IIR

Filtering in MATLAB

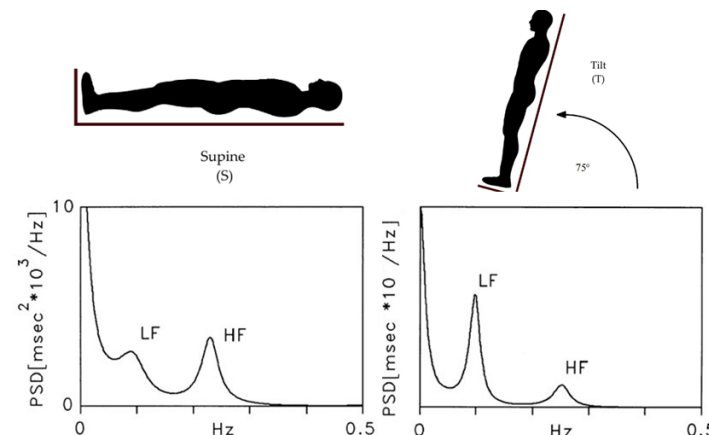
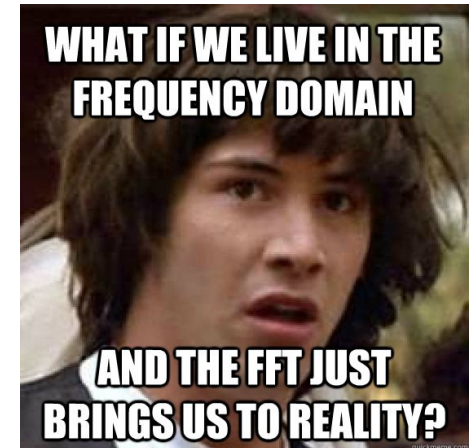
Esercitazione 3

Analisi in frequenza dei segnali fisiologici



Perché?

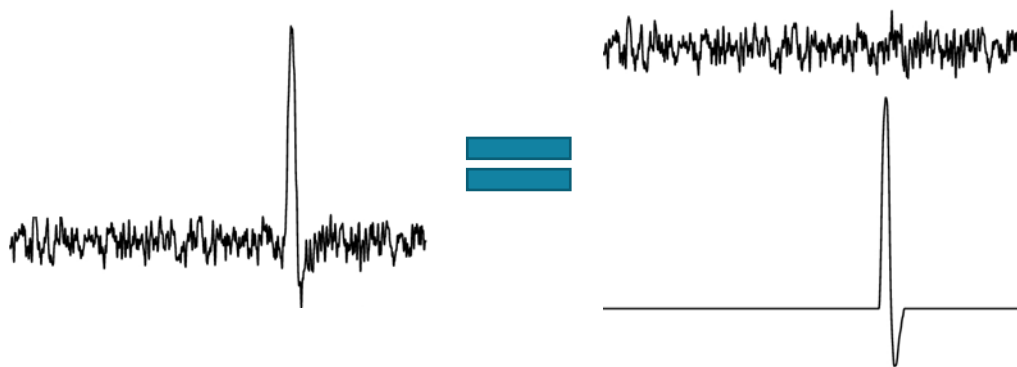
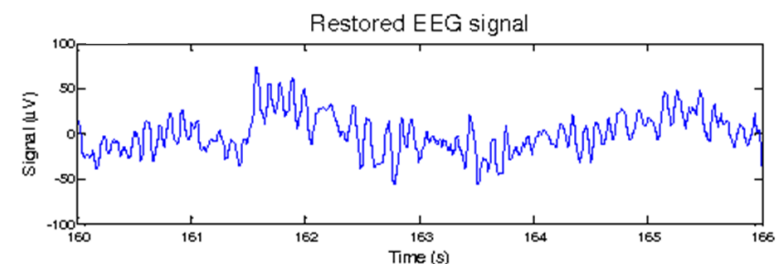
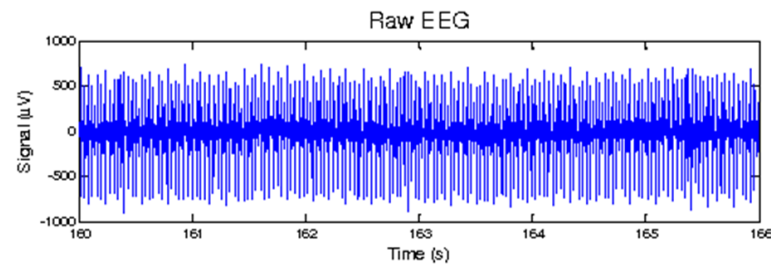
Perché alcuni sistemi fisiologici lavorano di per sé in frequenza (es. sistema uditivo); perché si possono estrarre numerose informazioni funzionali e fisio/patologiche nel dominio frequenziale (es. HRV analysis, EEG analysis, speech, ecc.)



α		8-16 Hz
β		16-30 Hz
θ		4-8 Hz
δ		< 4 Hz
γ		>30 Hz (< 100 Hz?)
μ		8-12 Hz

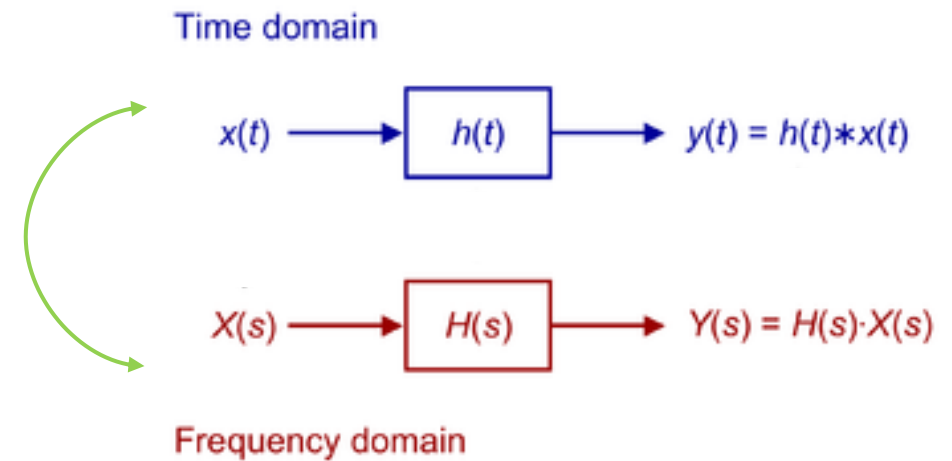
Analisi in frequenza dei segnali fisiologici

Ma anche per necessità di analisi: rimozione degli artefatti che non si sovrappongono (almeno in parte) in frequenza con lo spettro del segnale di interesse (es. frequenza di rete elettrica)



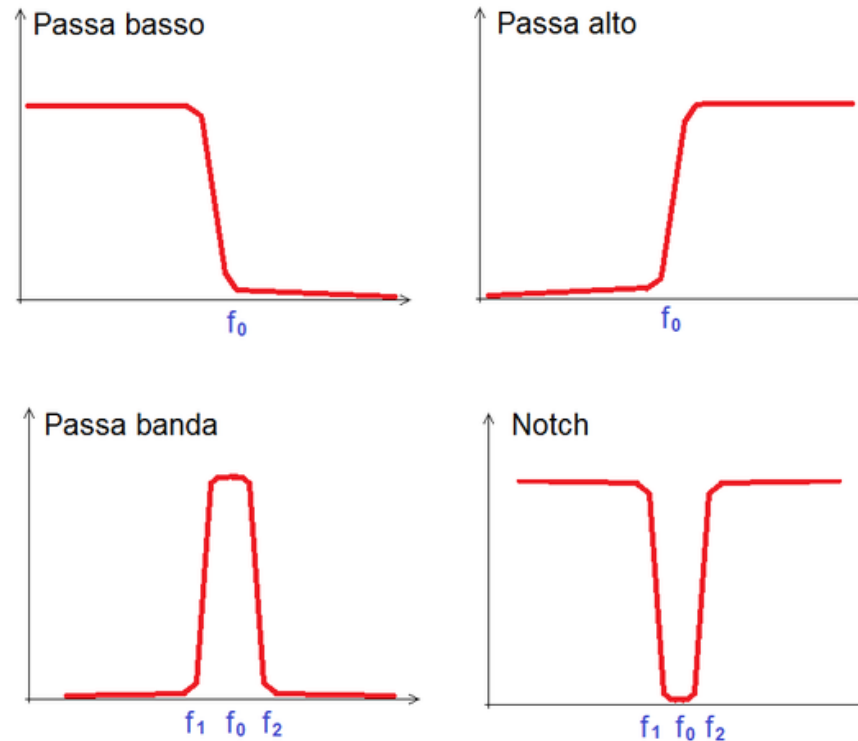
Analisi in frequenza dei segnali fisiologici

Inoltre, si sa che la sequenza di uscita (y) di un sistema lineare tempo invariante è ottenibile dalla convoluzione tra la risposta impulsiva del sistema stesso (h) per il suo ingresso (x), ma questa (y) è pari, nel dominio di Fourier, all'antitrasformata del prodotto tra le trasformate di h e x .

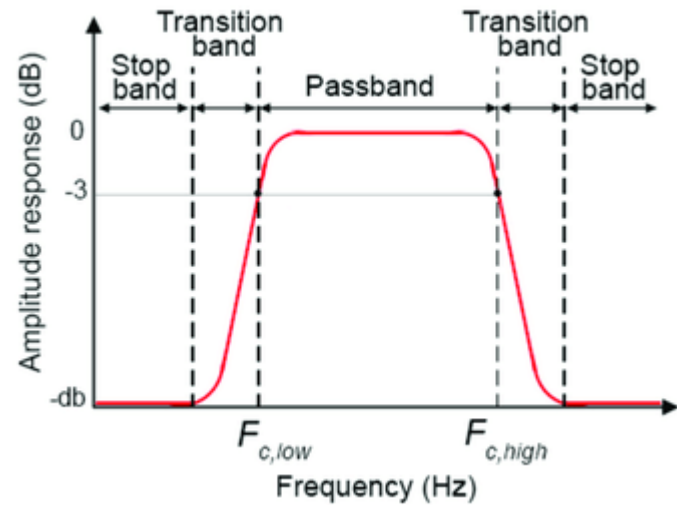


Filtering

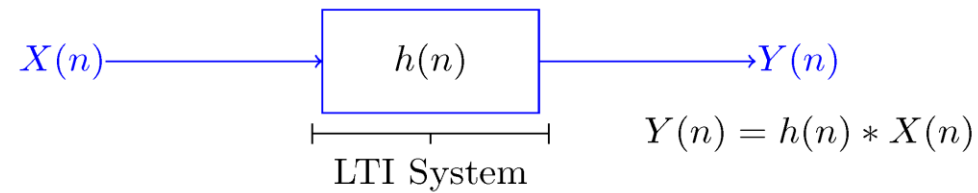
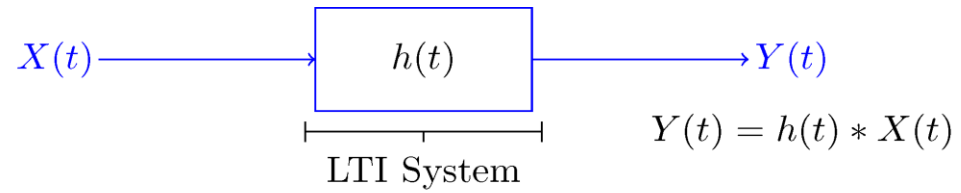
Ciò che si vuole è ottenere un segnale che abbia determinate caratteristiche (es. il cui contenuto comprenda solo le frequenze di interesse o non sia sporcato da componenti frequenziali spurie). Quindi si cerca di creare un sistema h tale da fornire il segnale richiesto y a partire da quello di partenza x . Tale operazione viene definita *filtering* e il sistema lineare stazionario è detto *filtro*.



Il punto, quindi, è progettare dei sistemi tali da avere le migliori caratteristiche possibili



Linear Time Invariant System



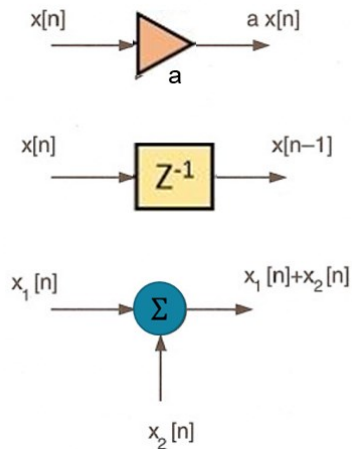
Finite Impulse Response filter

$$y[n] = \sum_{k=0}^N x[k]h[n-k]$$

Infinite Impulse Response filter

$$y[n] = \sum_{k=0}^{\infty} x[k]h[n-k]$$

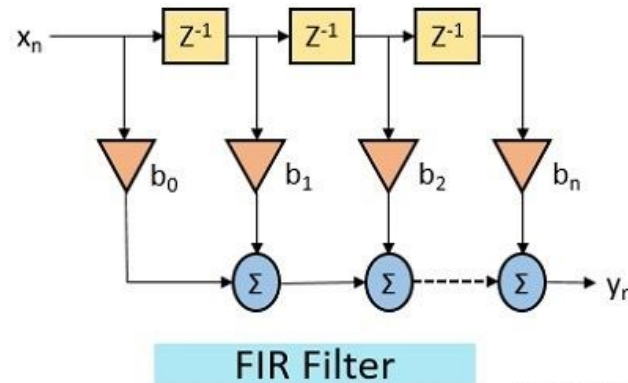
FIR vs IIR



Finite Impulse Response filter

$$y[n] = \sum_{k=0}^N x[k]h[n-k]$$

$$y[n] = \sum_{k=0}^N b_k x[n-k]$$



FIR Filter

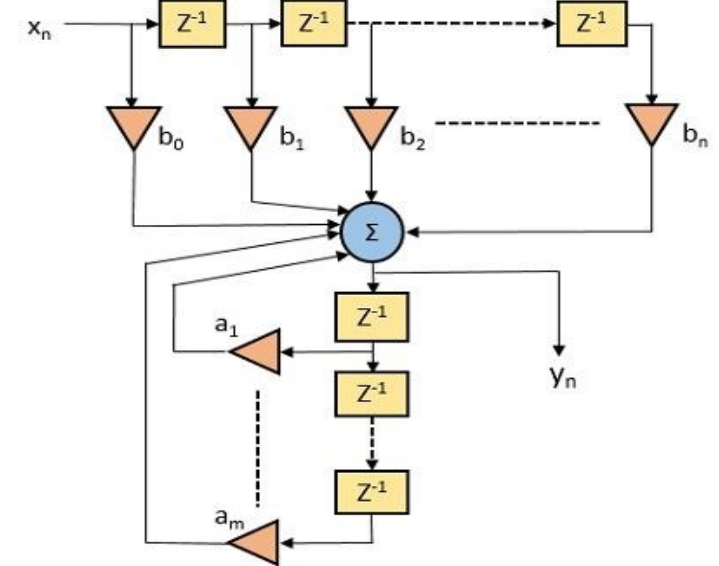
BIBO stability

A linear time invariant (LTI) system (such as a digital filter) is said to be **bounded input, bounded output stable**, or BIBO stable, if every bounded input gives rise to a bounded output, as $\sum_{k=0}^{\infty} |h[k]| < \infty$

Infinite Impulse Response filter

$$y[n] = \sum_{k=0}^{\infty} x[k]h[n-k]$$

$$y[n] = \sum_{m=1}^M a_m y[n-m] + \sum_{k=0}^N b_k x[n-k]$$



IIR Filter

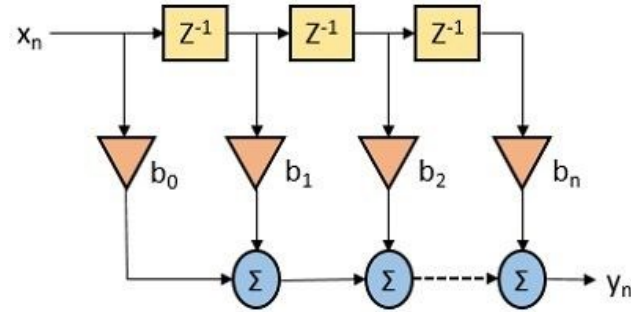
Es. $M = 1, N = 0$:

$$y[n] = a_1 y[n-1] + b_0 x[n] =$$

$$= a_1 (a_1 y[n-2] + b_0 x[n-1]) + b_0 x[n] = \dots$$

FIR vs IIR

Finite Impulse Response filter



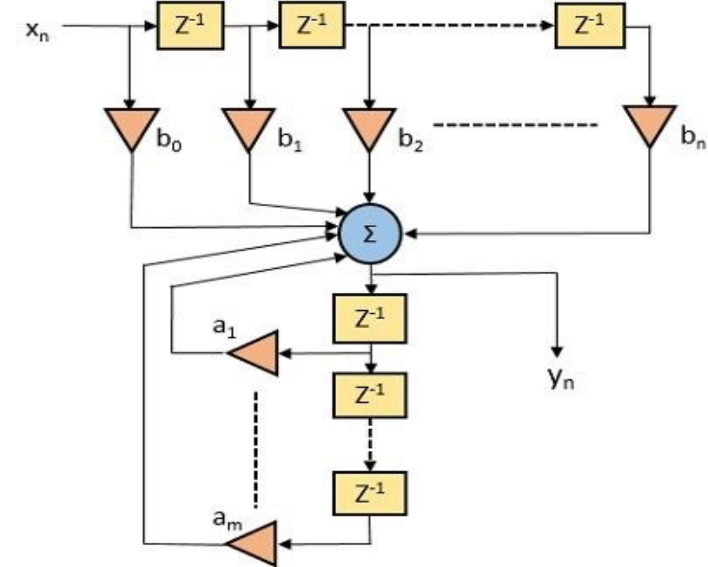
FIR Filter

$$y[n] = \sum_{k=0}^N b_k x[n-k] \rightarrow Y(z) = H(z)X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^N b_k z^{-k}$$

a_m and b_k are the filter's denominator and numerator polynomial coefficients, whose roots are equal to the filter's poles and zeros respectively. Thus, we can write a relationship between the equation and the z -transform. As seen, the transfer function is a frequency domain representation of the filter. The poles act on the output data, and the zeros on the input data. Since the poles act on the output data, and affect stability, it is essential that their radii remain inside the unit circle (i.e., <1) for BIBO stability. The radii of the zeros are less critical, as they do not affect filter stability. That's why all-zero FIR filters are always stable.

Infinite Impulse Response filter



IIR Filter

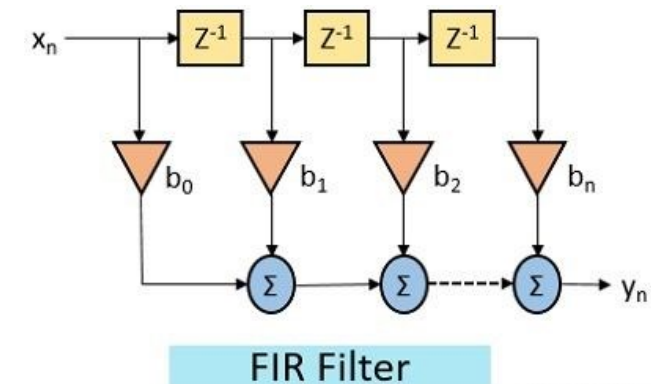
$$y[n] = \sum_{m=1}^M a_m y[n-m] + \sum_{k=0}^N b_k x[n-k]$$

$$Y(z) \left[1 - \sum_{m=1}^M a_m z^{-m} \right] = X(z) \sum_{k=0}^N b_k z^{-k}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^N b_k z^{-k}}{1 - \sum_{m=1}^M a_m z^{-m}}$$

FIR vs IIR

FIR (finite impulse response) filters are generally chosen for applications where linear phase is important and a decent amount of memory and computational performance are available. They are widely employed in audio and biomedical signal enhancement applications. Their all-zero structure ensures that they never become unstable for any type of input signal, which gives them a distinct advantage over the IIR.



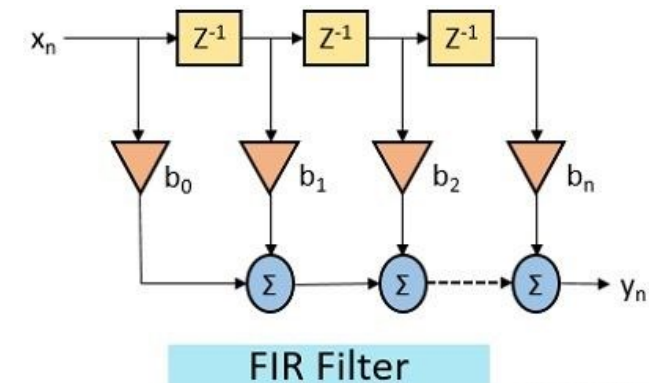
Advantages

- **Linear phase:** FIRs can be easily designed to have linear phase. This means that no phase distortion is introduced into the signal to be filtered, as all frequencies are shifted in time by the same amount – thus maintaining their relative harmonic relationships (i.e., constant group and phase delay). This is certainly not the case with IIR filters, that have a non-linear phase characteristic. (some connectivity measures are based on phase lag).
- **Stability:** As FIRs do not use previous output values to compute their present output (i.e., they have no feedback) they can never become unstable for any type of input signal, which gives them a distinct advantage over IIR filters.
- **Arbitrary frequency response:** a FIR can be customized more easily than an IIR.
- **Fixed point performance:** the effects of quantization are less severe than that of an IIR.

FIR vs IIR

Disadvantages

- **High computational and memory requirement:** FIRs usually require many more coefficients for achieving a sharp cut-off than their IIR counterparts. The consequence of this is that they require much more memory and significantly a higher amount of MAC (multiple and accumulate) operations. However, modern microcontroller architectures expedite the filtering operation significantly.
- **Higher latency:** the higher number of coefficients means that in general a linear phase FIR is less suitable than an IIR for fast applications. This becomes problematic for real-time closed-loop control applications, where a linear phase FIR filter may have too much group delay to achieve loop stability.
- **No analog equivalent:** using the Bilinear, matched z-transform (s-z mapping), an analog filter can be easily transformed into an equivalent IIR filter. However, this is not possible for a FIR as it has no analog equivalent.



FIR vs IIR

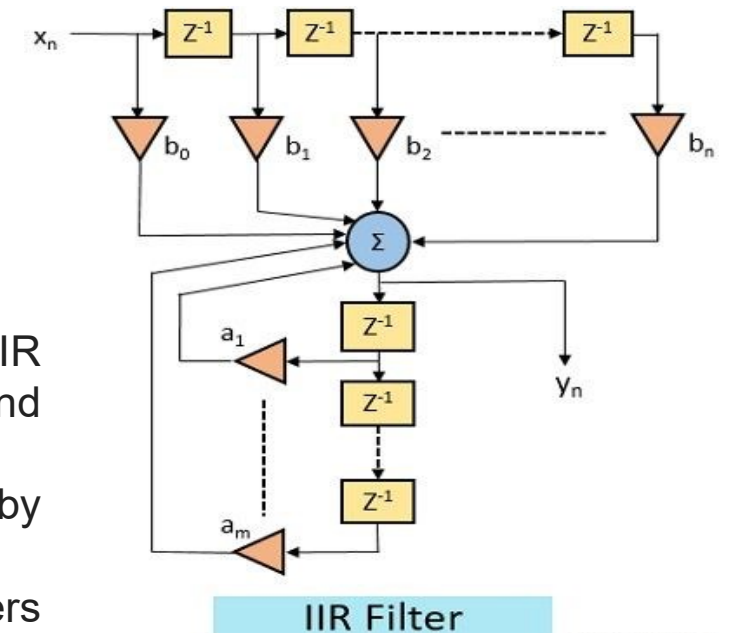
IIR (infinite impulse response) filters are generally chosen for applications where linear phase is not too important and memory is limited. They have been widely deployed in audio equalization, biomedical sensor signal processing, IoT/IIoT smart sensors and high-speed real-time applications.

Advantages

- **Low implementation cost:** they require less coefficients and memory than FIR filters in order to satisfy a similar set of specifications, i.e., cut-off frequency and stopband attenuation.
- **Low latency:** suitable for real-time control and very high-speed applications by virtue of the low number of coefficients.
- **Analog equivalent:** May be used for mimicking the characteristics of analog filters using s-z (Laplace-Zed) plane mapping transforms.

Disadvantages

- **Non-linear phase characteristics:** The phase characteristics of an IIR filter are generally nonlinear, especially near the cut-off frequencies. All-pass equalization filters can be used in order to improve the passband phase characteristics.
- **More detailed analysis:** Requires more scaling and numeric overflow analysis when implemented in fixed point. IIR filters are especially sensitive to the effects of quantization and require special care during the design phase.
- **Numerical stability:** Less numerically stable than their FIR counterparts, due to the feedback paths.

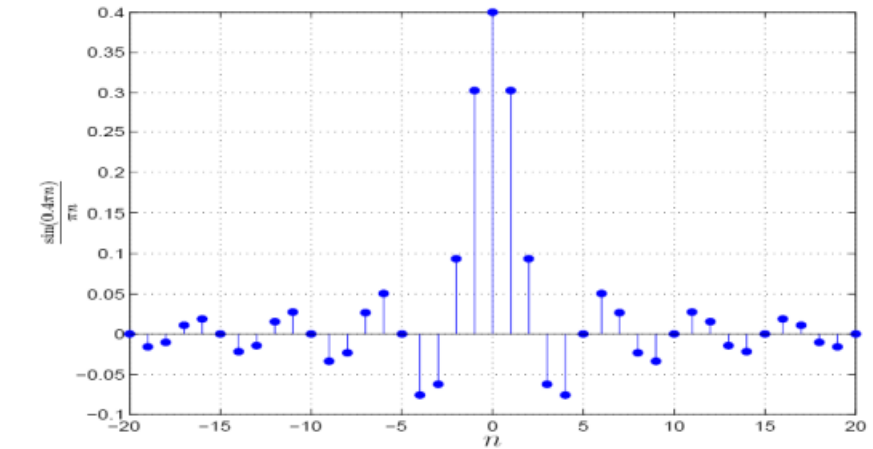


FIR design

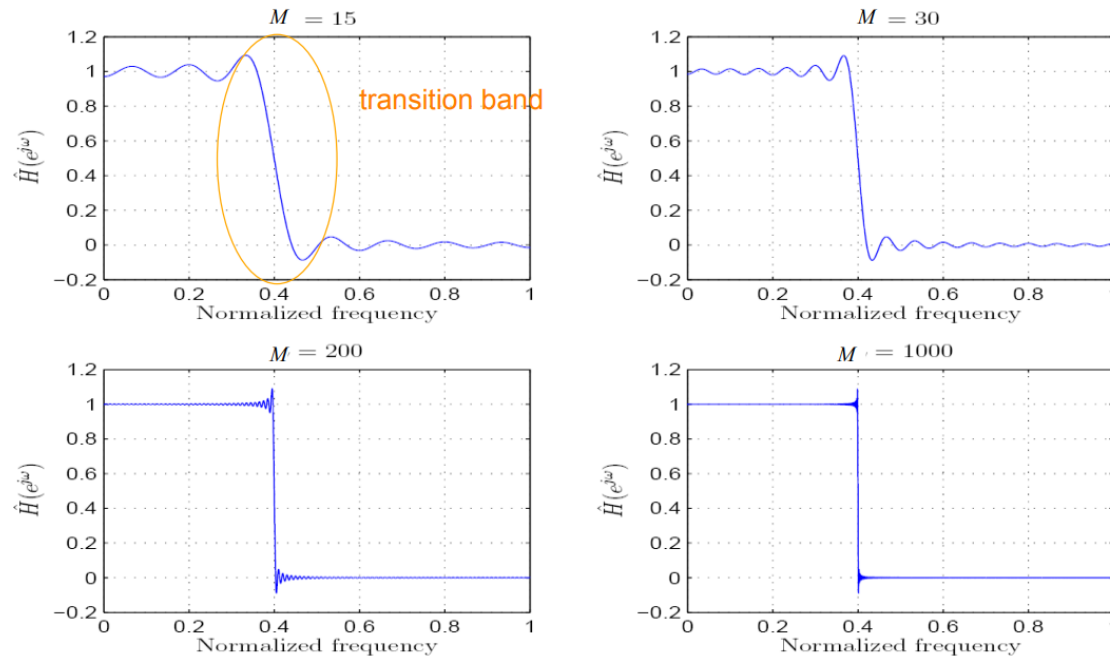
The ideal filter is impossible to be used in real world scenario, since it has a *sinc()* shape which is non-causal and infinite in duration

$$H_d(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq \omega_c \\ 0 & \text{if } \omega_c < |\omega| < \pi \end{cases}$$

$\xleftarrow{\text{freq}} \quad \longleftrightarrow \quad \xrightarrow{\text{time}}$



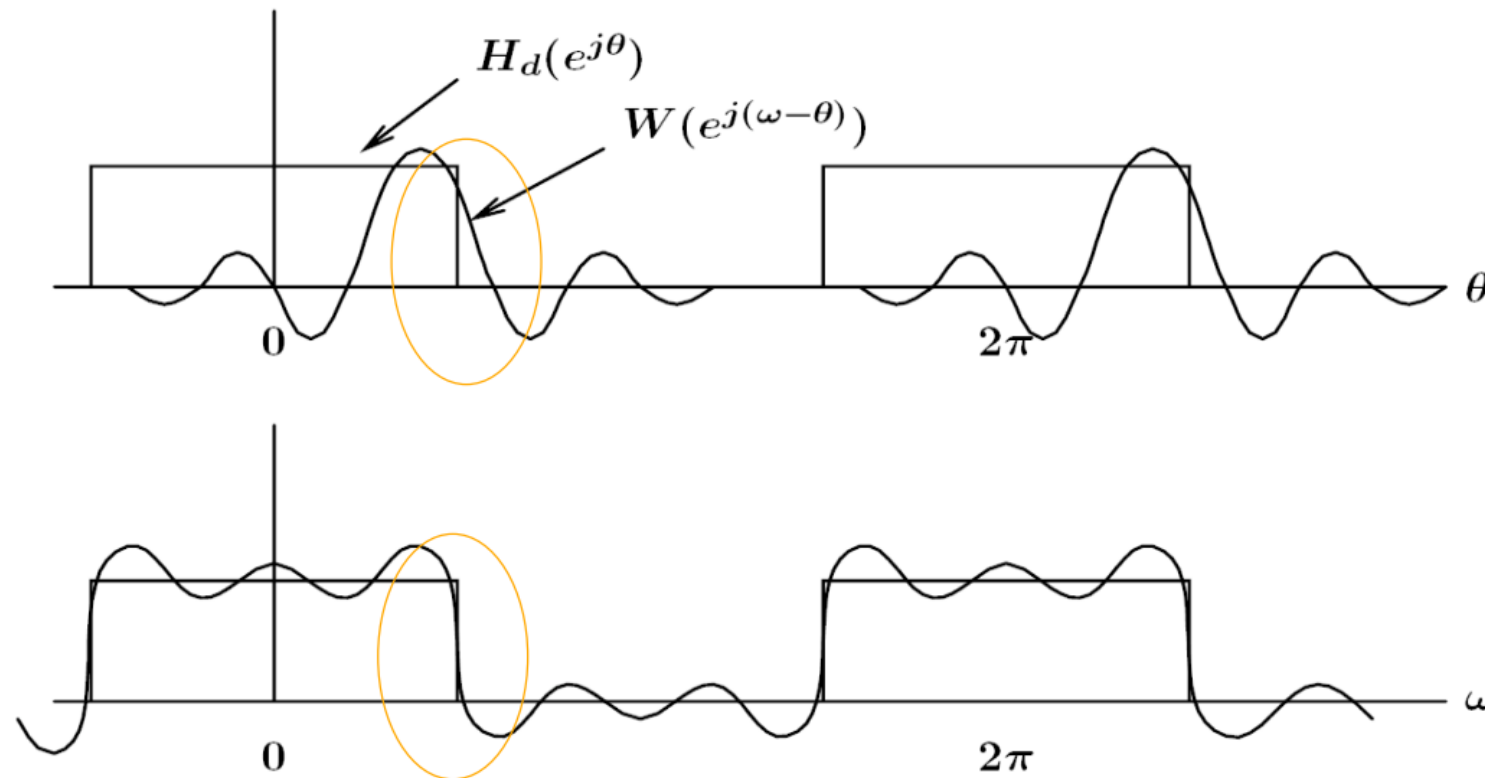
Do you remember what we did to obtain a periodic and limited TDF?
TRUNCATION (in freq it means convolution with a sinc)



Though the transition band gets narrower as $M \rightarrow \infty$, the ripple remains!

FIR design

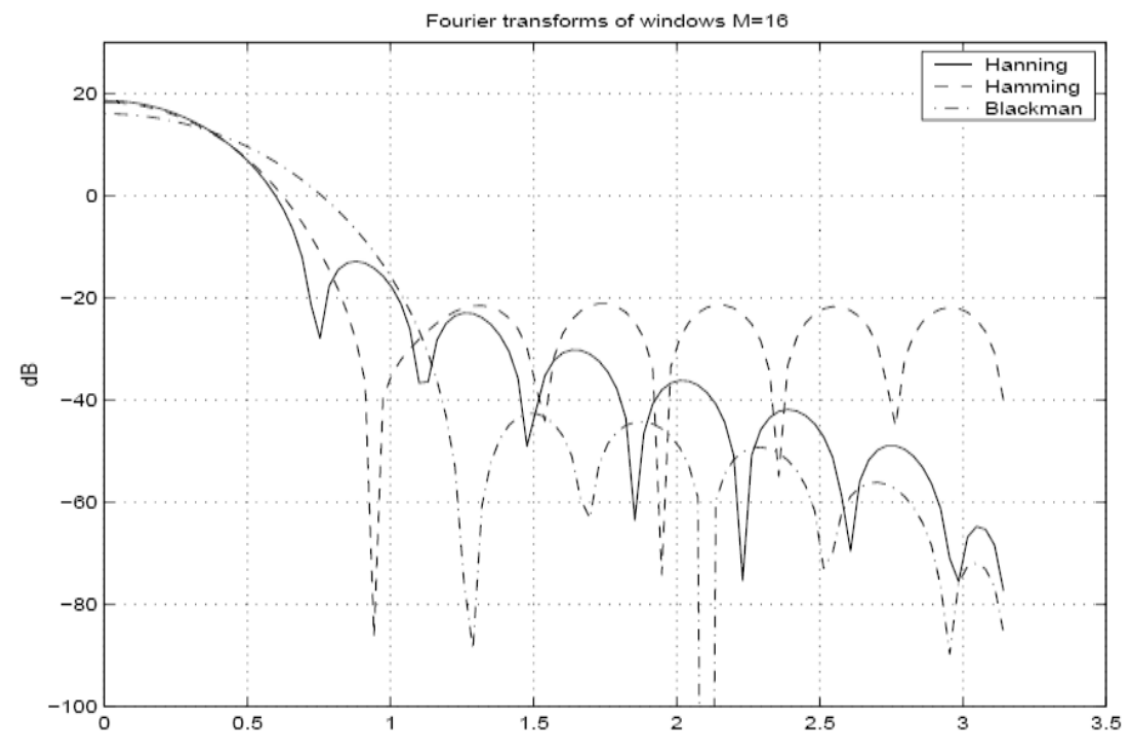
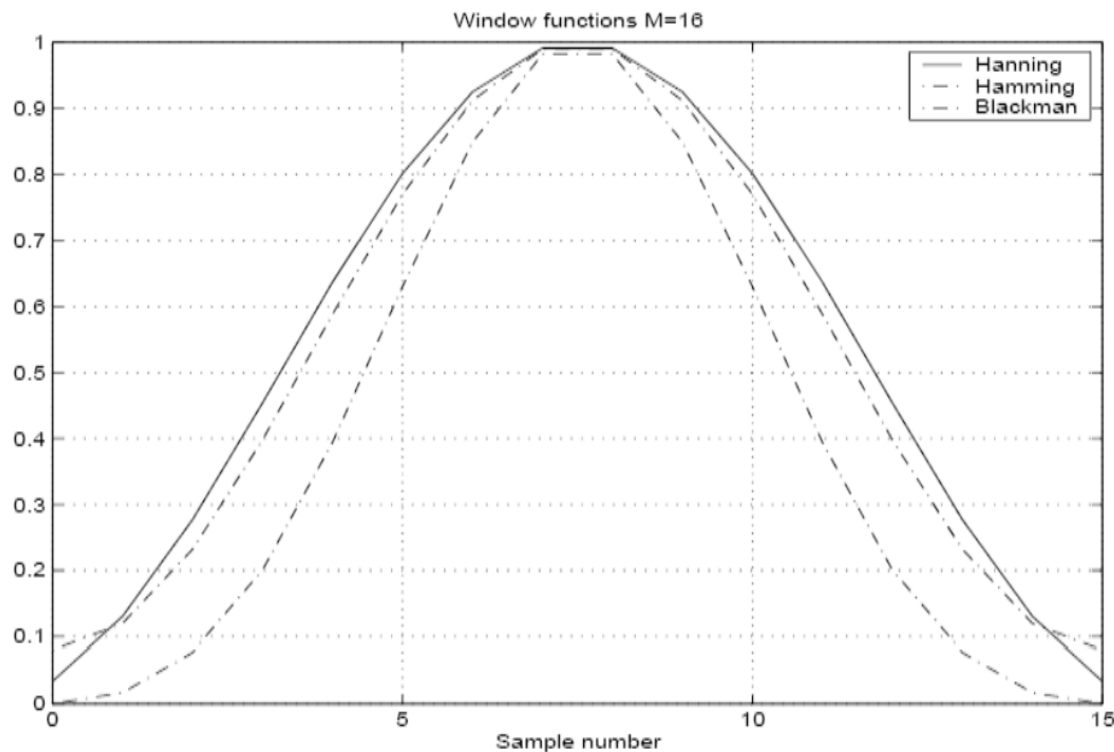
Ideally, we would like to have M small (few computations), and $W(\omega)$ close to a Dirac' delta, but these requirements are conflicting between each other.



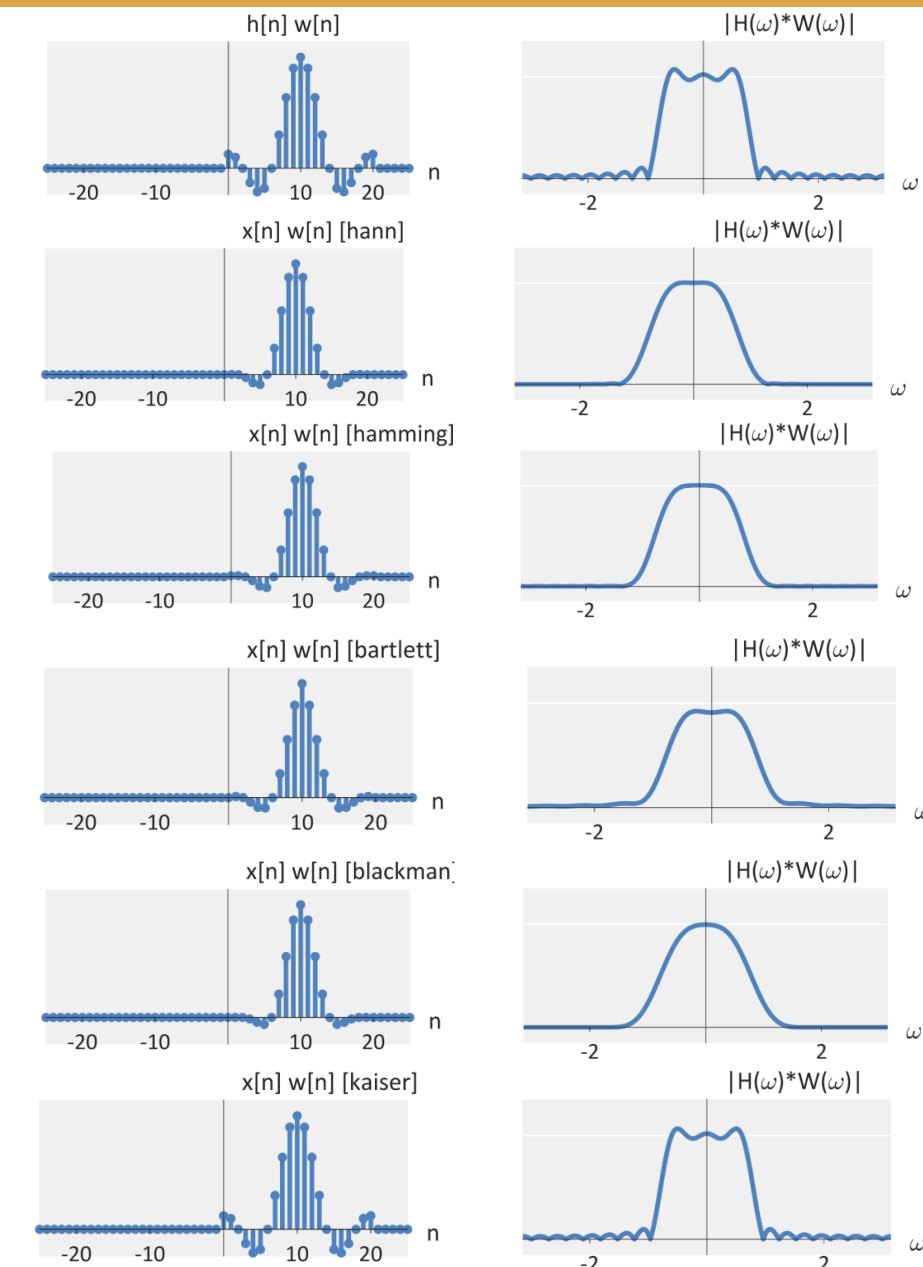
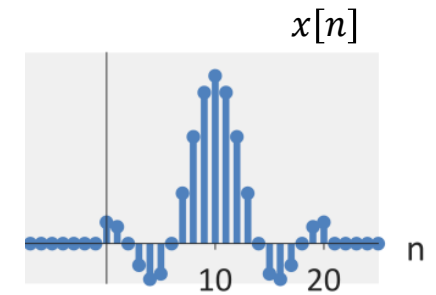
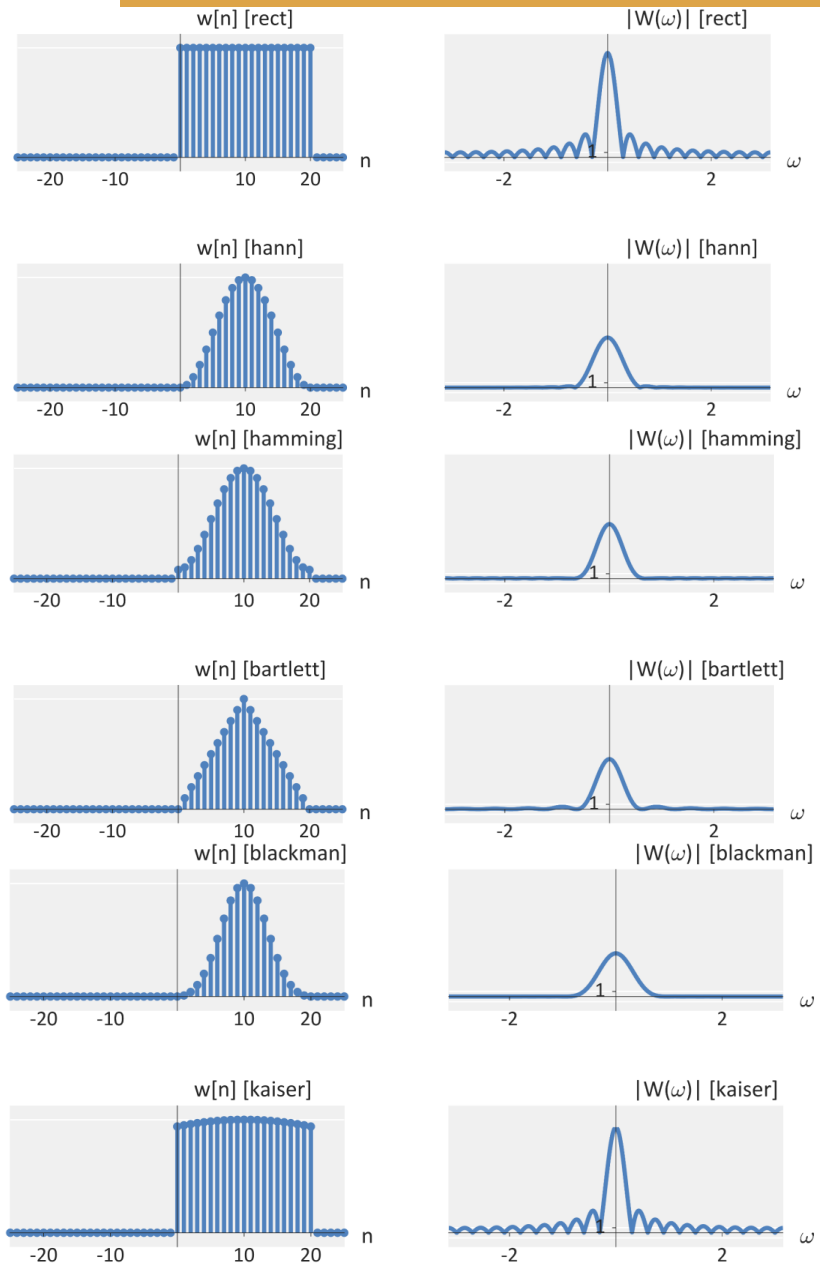
FIR design by window method

A partial solution can be obtained applying a different window substituting the rectangular window

Hanning
Hamming
Blackman
Kaiser
Bartlett



FIR design by window method



FIR design

There are, of course, several other methods:

- EQUIRIPPLE: Find the optimal b_k that satisfies passband/stopband ripple constraints.
- WEIGHTED LEAST SQUARE: it exploits the least square optimization method to find the coefficient of FIR filter enforcing even symmetry in the impulse response.

IIR design

IIR classical design methods:

Butterworth

Chebyshev Type I

Chebyshev Type II

Elliptic

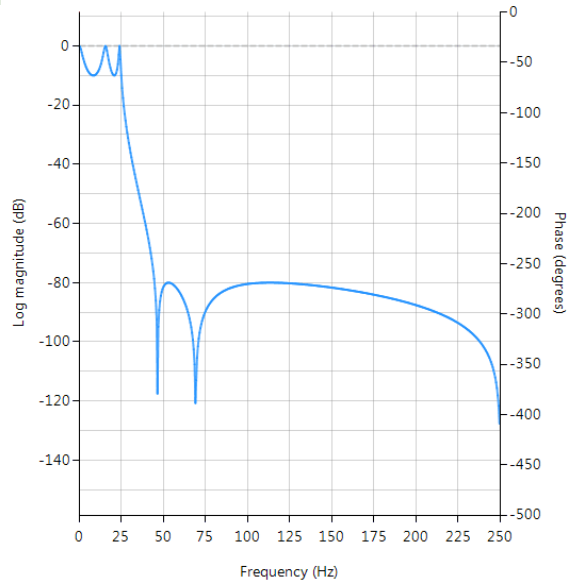
Each method has its pros and cons, but in general the Elliptic method is chosen as it meets the design specifications with the lowest order of any of the methods. However, this desirable property comes at the expense of ripple in both the passband and stopband, and very non-linear passband phase characteristics. Therefore, the Elliptic filter should only be used in applications where memory is limited, and passband phase linearity is less important.

The Butterworth and Chebyshev Type II methods have flat passbands (no ripple), making them a good choice for low frequency measurement applications. However, this desirable property comes at the expense of wider transition bands, resulting in low passband to stopband transition (slow roll-off). The Chebyshev Type I and Elliptic methods roll-off faster but have passband ripple and very non-linear passband phase characteristics.

IIR filters

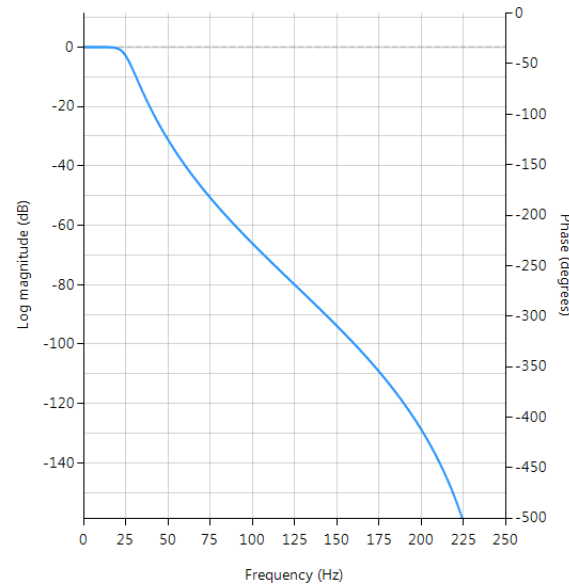
The frequency response below show the differences between the various design methods for a 5th order lowpass filter with the same specifications.

Elliptic



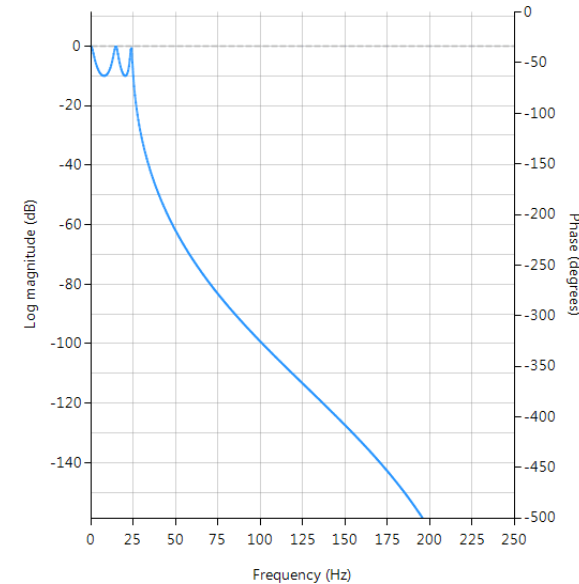
- Fastest roll-off of all supported prototypes
- Equiripple in both the passband and stopband
- Lowest order filter of all supported prototypes
- Non-linear passband phase characteristics
- Good choice for real-time control and high-throughput applications

Butterworth



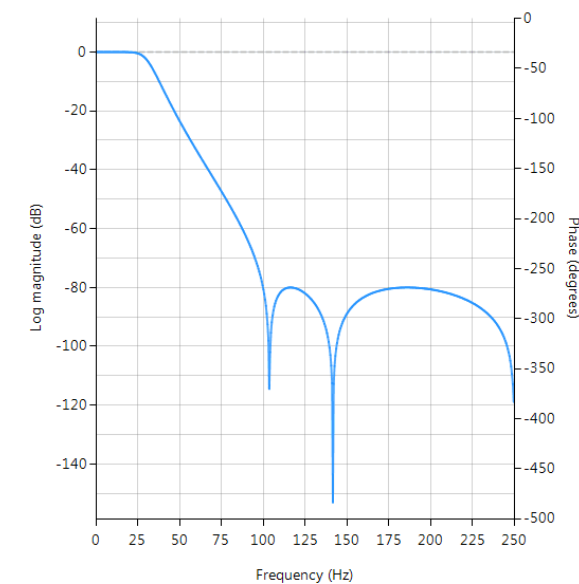
- Smooth monotonic response (no ripple)
- Slowest roll-off for equivalent order
- Highest order of all supported prototypes
- More linear passband phase response than all other methods
- Good choice for very low freq and audio application

Chebyshev Type I



- Passband ripple
- Maximally flat stopband
- Faster roll-off than Butterworth and Chebyshev Type II
- Good compromise between Elliptic and Butterworth

Chebyshev Type II



- Maximally flat passband
- Faster roll-off than Butterworth
- Slower roll-off than Chebyshev Type I
- Good choice for very low freq applications

IIR biquad filters

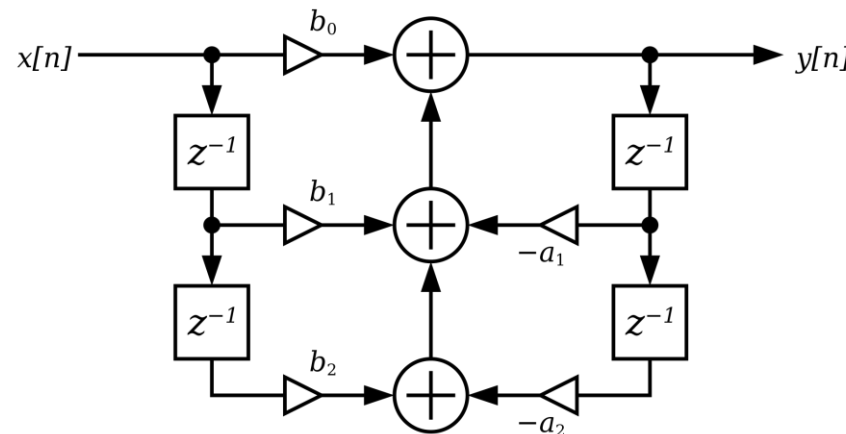
A digital biquad filter is a second order recursive linear filter, containing two poles and two zeros. "Biquad" is an abbreviation of "biquadratic", which refers to the fact that in the Z domain, its transfer function is the ratio of two quadratic functions:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

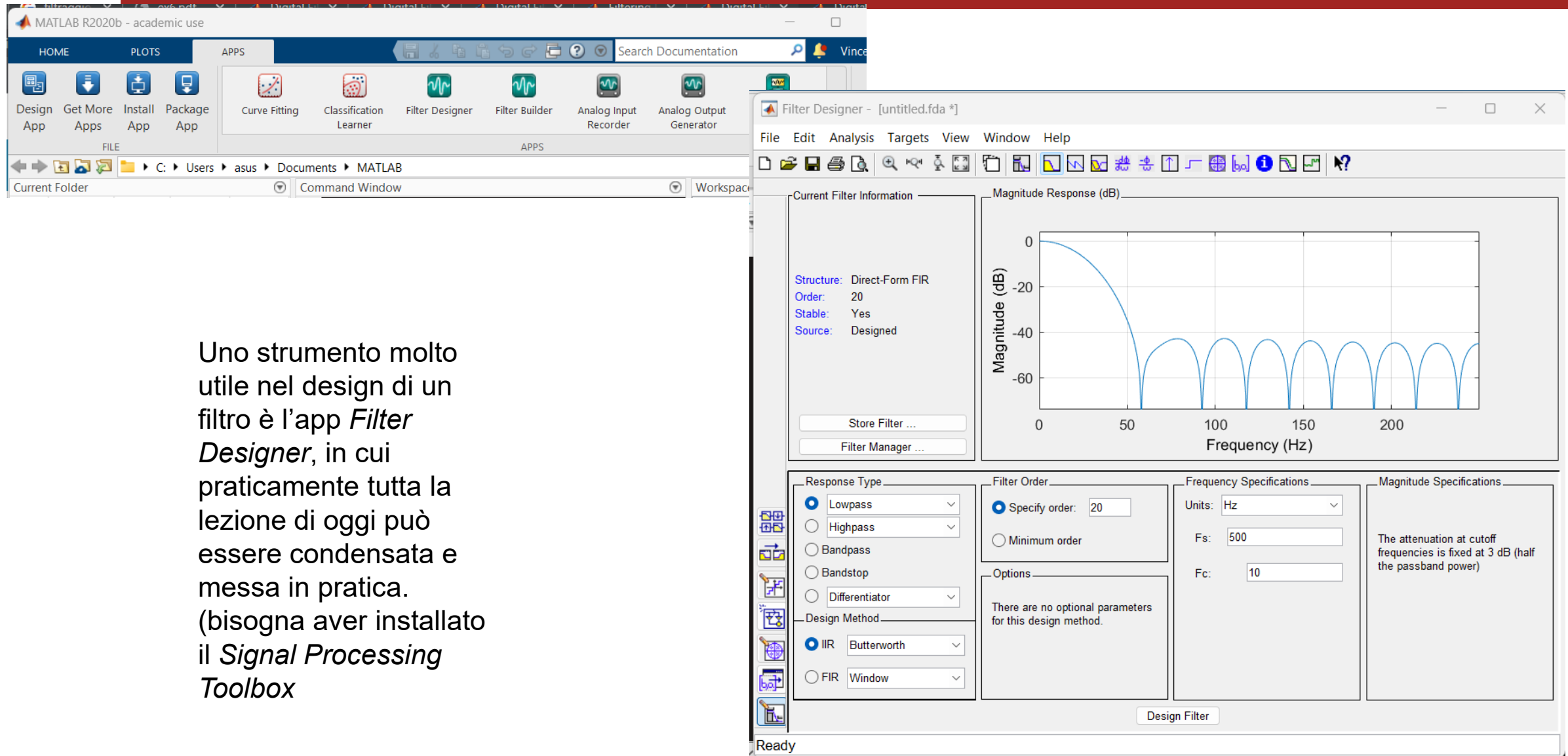
The coefficients are often normalized such that $a_0 = 1$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

High-order infinite impulse response filters can be highly sensitive to quantization of their coefficients and can easily become unstable. This is much less of a problem with first and second-order filters; therefore, higher-order filters are typically implemented as serially-cascaded biquad sections (and a first-order filter if necessary). The two poles of the biquad filter must be inside the unit circle for it to be stable. In general, this is true for all discrete filters, i.e., all poles must be inside the unit circle in the Z-domain for the filter to be stable.



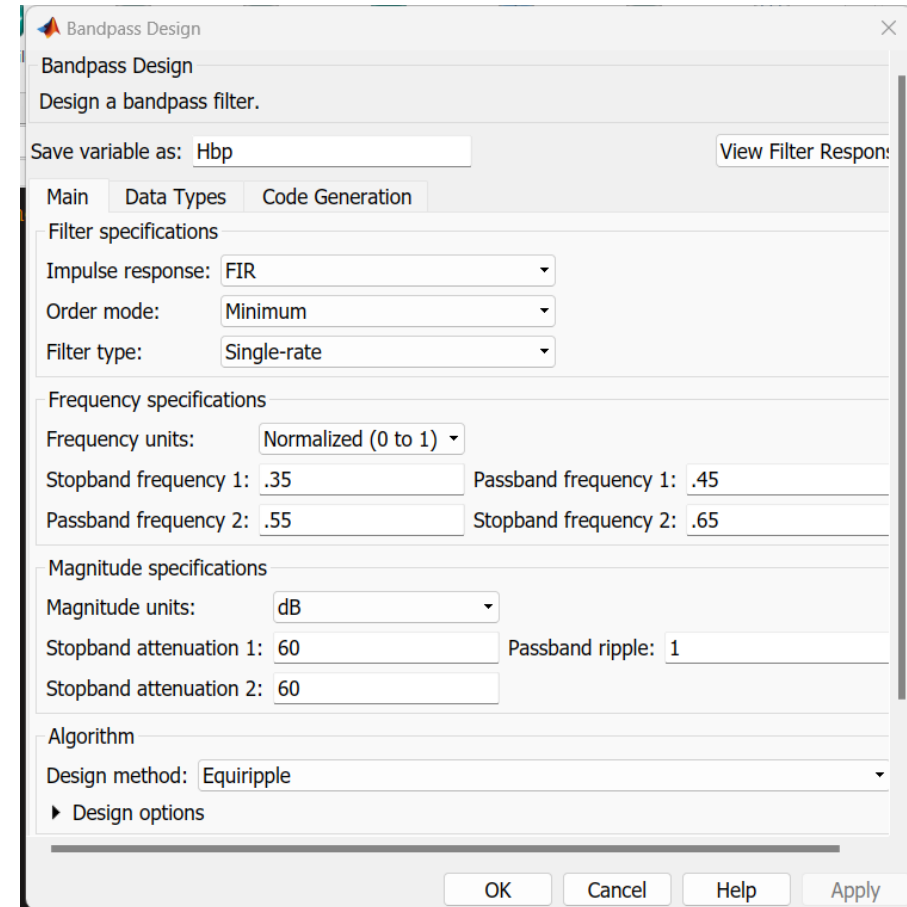
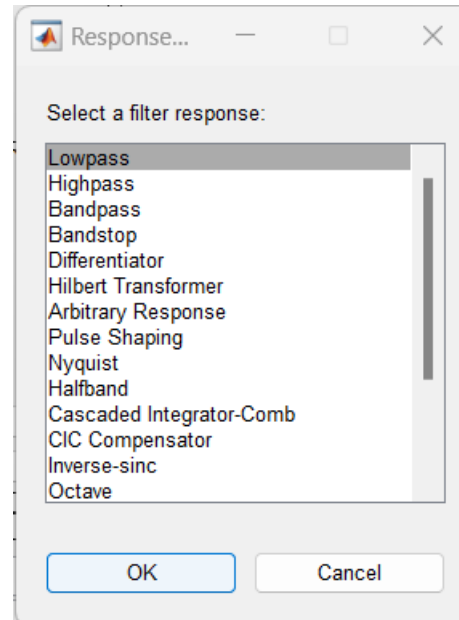
Filtering in MATLAB



Uno strumento molto utile nel design di un filtro è l'app *Filter Designer*, in cui praticamente tutta la lezione di oggi può essere condensata e messa in pratica. (bisogna aver installato il *Signal Processing Toolbox*)

Filtering in MATLAB

Un'altra app simile è *Filter Builder*



Filtering in MATLAB

Queste app sono degli utili strumenti grafici, sono delle interfacce, per la progettazione dei filtri. Essi, comunque, si basano su altre funzioni Matlab che possono tranquillamente essere usate via codice

▼ IIR Filters

<code>butter</code>	Butterworth filter design
<code>buttord</code>	Butterworth filter order and cutoff frequency
<code>cheby1</code>	Chebyshev Type I filter design
<code>cheb1ord</code>	Chebyshev Type I filter order
<code>cheby2</code>	Chebyshev Type II filter design
<code>cheb2ord</code>	Chebyshev Type II filter order
<code>designfilt</code>	Design digital filters
<code>ellip</code>	Elliptic filter design
<code>ellipord</code>	Minimum order for elliptic filters
<code>yulewalk</code>	Recursive digital filter design

▼ FIR Filters

<code>cfirpm</code>	Complex and nonlinear-phase equiripple FIR filter design
<code>designfilt</code>	Design digital filters
<code>fir1</code>	Window-based FIR filter design
<code>fir2</code>	Frequency sampling-based FIR filter design
<code>fircls</code>	Constrained-least-squares FIR multiband filter design
<code>fircls1</code>	Constrained-least-squares linear-phase FIR lowpass and highpass filter design
<code>firls</code>	Least-squares linear-phase FIR filter design
<code>firpm</code>	Parks-McClellan optimal FIR filter design
<code>firpmord</code>	Parks-McClellan optimal FIR filter order estimation
<code>gaussdesign</code>	Gaussian FIR pulse-shaping filter design
<code>intfilt</code>	Interpolation FIR filter design
<code>kaiserord</code>	Kaiser window FIR filter design estimation parameters
<code>maxflat</code>	Generalized digital Butterworth filter design
<code>rcosdesign</code>	Raised cosine FIR pulse-shaping filter design
<code>sgolay</code>	Savitzky-Golay filter design

Filtering in MATLAB

Una volta progettati, i filtri vanno verificati

Frequency-Domain Responses

<code>abs</code>	Absolute value and complex magnitude
<code>angle</code>	Phase angle
<code>freqz</code>	Frequency response of digital filter
<code>grpdelay</code>	Average filter delay (group delay)
<code>phasedelay</code>	Phase delay of digital filter
<code>phasez</code>	Phase response of digital filter
<code>unwrap</code>	Shift phase angles
<code>zerophase</code>	Zero-phase response of digital filter
<code>zplane</code>	Zero-pole plot for discrete-time systems

Filtering Functions

<code>bandpass</code>	Bandpass-filter signals
<code>bandstop</code>	Bandstop-filter signals
<code>highpass</code>	Highpass-filter signals
<code>lowpass</code>	Lowpass-filter signals
<code>fftfilt</code>	FFT-based FIR filtering using overlap-add method
<code>filtfilt</code>	Zero-phase digital filtering
<code>filtic</code>	Initial conditions for transposed direct-form II filter implementation
<code>hampel</code>	Outlier removal using Hampel identifier
<code>latcfilt</code>	Lattice and lattice-ladder filter implementation
<code>medfilt1</code>	1-D median filtering
<code>residuez</code>	Z-transform partial-fraction expansion
<code>sgolayfilt</code>	Savitzky-Golay filtering
<code>sosfilt</code>	Second-order (biquadratic) IIR digital filtering

Time-Domain Responses

<code>impz</code>	Impulse response of digital filter
<code>impzlength</code>	Impulse response length
<code>stepz</code>	Step response of digital filter

Filter Exploration

<code>filtord</code>	Filter order
<code>filternorm</code>	2-norm or infinity-norm of digital filter
<code>firtype</code>	Type of linear phase FIR filter
<code>isallpass</code>	Determine whether filter is allpass
<code>isfir</code>	Determine if digital filter has finite impulse response
<code>islinphase</code>	Determine whether filter has linear phase
<code>ismaxphase</code>	Determine whether filter is maximum phase
<code>isminphase</code>	Determine whether filter is minimum phase
<code>isstable</code>	Determine whether filter is stable

e soprattutto applicati

Filtering in MATLAB

Esempio

```
Fs = 1000;  
t = linspace(0,1,Fs);  
x = cos(2*pi*100*t)+0.5*randn(size(t));  
fc = 150;  
Wn = (2/Fs)*fc;  
b = fir1(20,Wn,'low',kaiser(21,3));
```

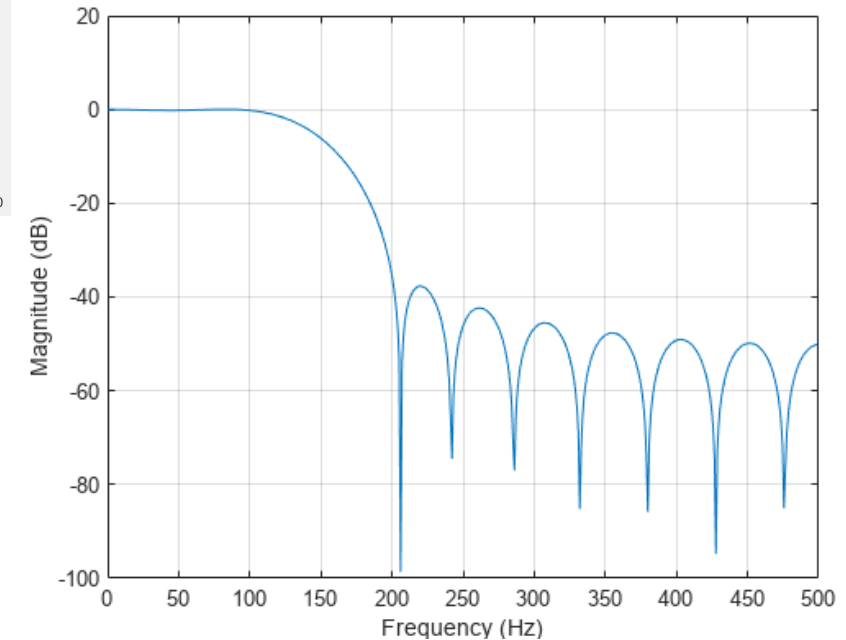
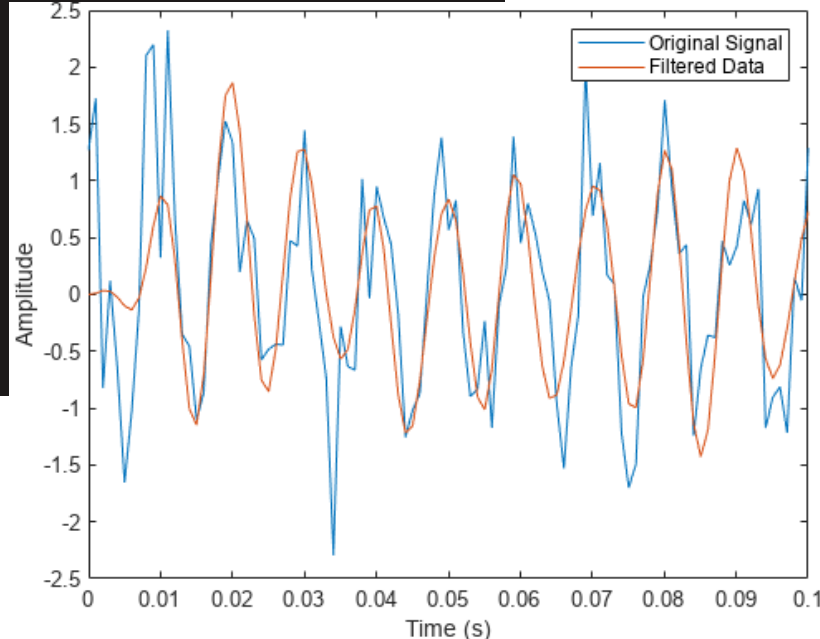
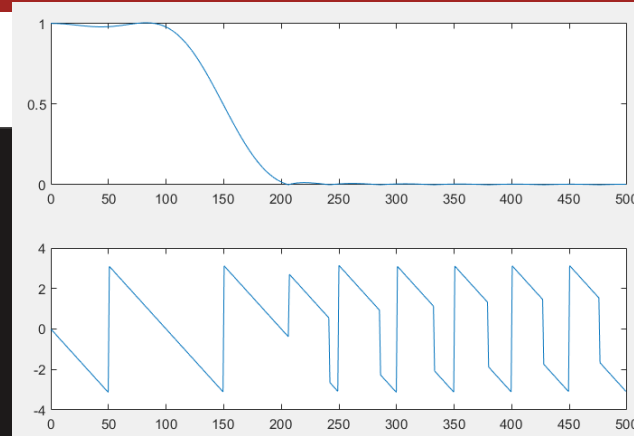
```
[h,f] = freqz(b,1,[],Fs);  
figure;subplot(2,1,1); plot(f,abs(h));subplot(2,1,2);plot(f,angle(h))
```

```
plot(f,mag2db(abs(h))); xlabel('Frequency (Hz)');  
ylabel('Magnitude (dB)'); grid
```

```
y = filter(b,1,x);
```

```
plot(t,x,t,y)  
xlim([0 0.1])
```

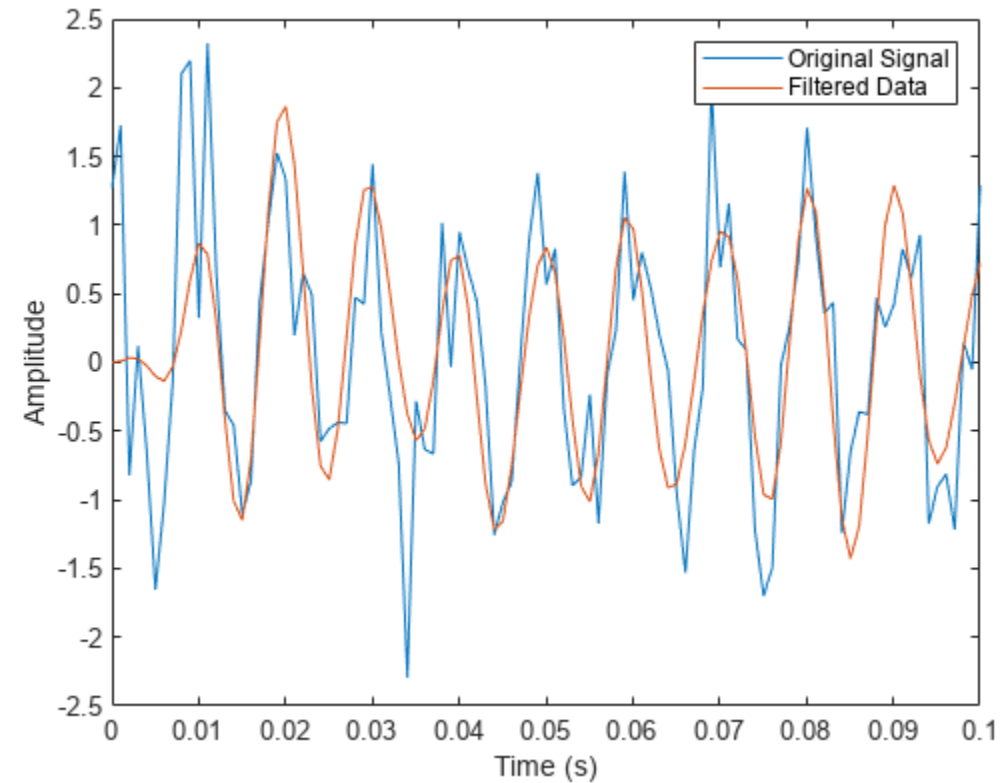
```
xlabel('Time (s)')  
ylabel('Amplitude')  
legend('Original Signal','Filtered Data')
```



Filtering in MATLAB

Esempio equivalente

```
Fs = 1000;  
Hd = designfilt('lowpassfir','FilterOrder',20,'CutoffFrequency',150, ...  
    'DesignMethod','window','Window',{@kaiser,3},'SampleRate',Fs);  
y1 = filter(Hd,x);  
plot(t,x,t,y1); xlim([0 0.1]);  
xlabel('Time (s)'); ylabel('Amplitude');  
legend('Original Signal','Filtered Data')
```



Esercitazione (Lezione 13.2)

Caricare il file SampleEEG, presente nella cartella Teams, già usato in una precedente esercitazione. Il file contiene un segnale EEG *SampleEEG* campionato con una frequenza di campionamento salvata nella variabile *sampling_rate* registrato su 9 canali, i cui nomi sono contenuti nella variabile *Chans_names*.

Per ogni canale trovare:

- La TDF calcolata sui primi 2 secondi del segnale;
- La TDF calcolata sull'intero segnale;
- La media tra le TDF calcolate su finestre non sovrapposte di 2 secondi ciascuna
- Prendere il risultato del punto precedente e normalizzare per il vettore delle frequenze $X'(f) = (X(f)./f)$

Per 3 canali a scelta fare un grafico (scegliere il modo più congeniale per farlo) in cui siano rappresentati i vettori calcolati, fare attenzione alla taratura degli assi.

Infine per gli $X'(f)$ di ogni canale fin qui derivati trovare a quale frequenza si ottiene il picco (il massimo) della $X'(f)$ corrispondente e salvarlo in una variabile f_{picco} .

Esercitazione

Progettare i seguenti filtri (scegliere arbitrariamente gli altri parametri dei filtri e le funzioni più adatte):

- Filtro FIR passabanda con frequenze di taglio [1, 8] Hz
- Filtro FIR passabanda per selezionare la banda alpha di un segnale EEG
- Filtro IIR passabanda con frequenze di taglio [1, 8] Hz
- Filtro IIR passabanda per selezionare la banda alpha di un segnale EEG

Caricare il file SampleEEG, presente nella cartella Teams, già usato in precedenza.

Scegliere un canale EEG e applicare i due filtri FIR sia come convoluzione nel tempo che come prodotto delle trasformate in frequenza, applicare anche i filtri IIR e plottare il segnale originale e quello filtrato con le diverse tecniche (scegliere autonomamente il numero di subplots e/o hold on) nel dominio del tempo. Ripetere lo stesso tipo di plot nel dominio della frequenza, porre attenzione all'opportuna taratura degli assi.

Applicare separatamente i filtri FIR e IIR in banda alpha a tutti i canali EEG e salvare la lista dei nomi dei canali ordinata in ordine decrescente di potenza dei segnale ottenuti, valutare se vi siano differenze tra i due filtri applicati.

All'esame si chiederà di giustificare le scelte dei parametri dei filtri, per cui delle prove ripetute con diversi parametri, tipo ordine del filtro, tipo di metodo usato, parametro specifico della finestra ecc, sono molto ben viste.