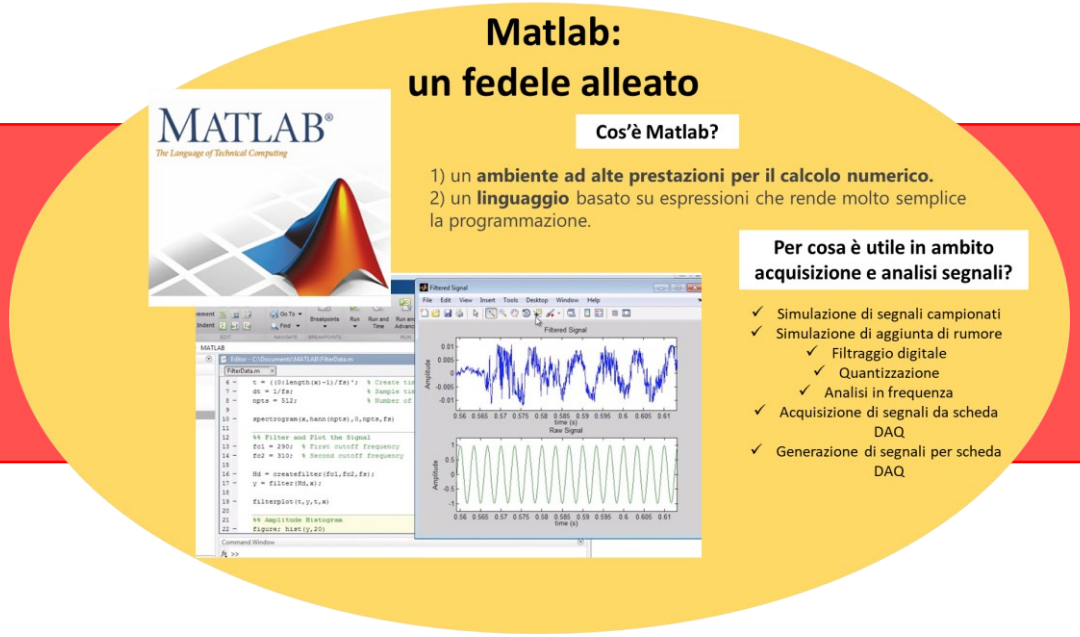


LAB. PRATICO MATLAB #2:

Acquisizione simulata di segnali e filtraggio

Misura e Acquisizione di Dati Biomedici
Sarah Tonello, PhD
Dipartimento di ingegneria dell'informazione
Universita di Padova

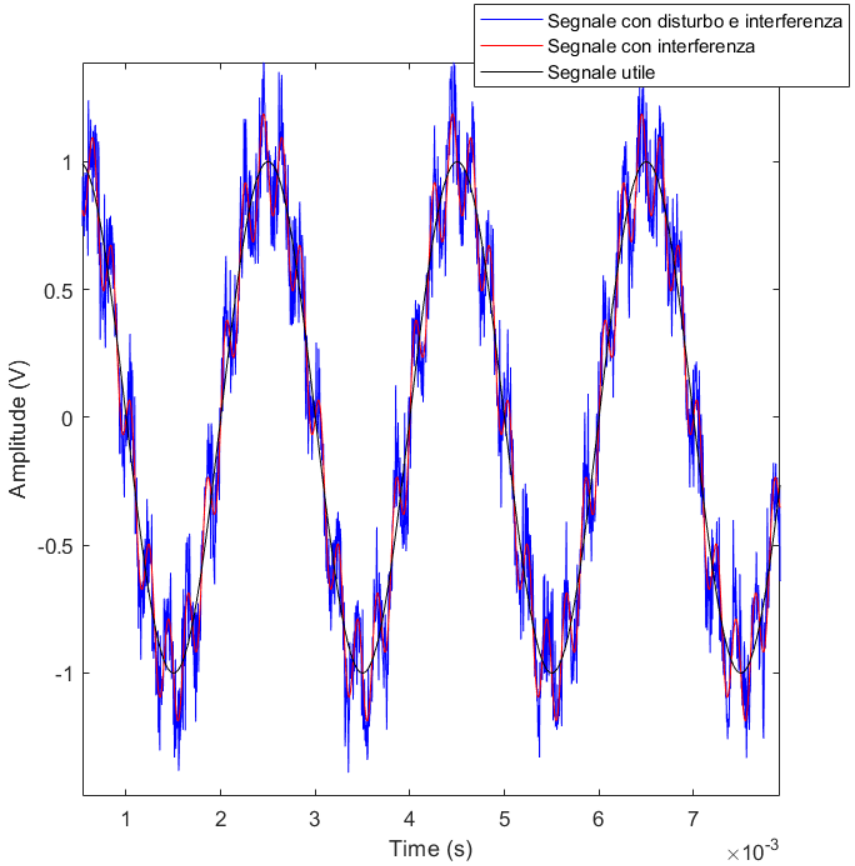
Obiettivi dell'esercitazione...

generazione del **segnale campionato** simulato, composto di:

- una componente utile (sinusoide);
- un disturbo (sinusoide a frequenza differente);
- rumore a larga banda;

IN PUNTI:

1. creare un segnale sinusoidale con sovrapposto un disturbo sinusoidale e rumore a larga banda;
2. creare la function `adc_ideal()`;
3. determinare i parametri del filtro;
4. calcolare i coefficienti del filtro;
5. verificare la risposta in frequenza del filtro;
6. filtrare il segnale;
7. confrontare i segnali nel dominio del tempo pre- e pos



Obiettivi dell'esercitazione...

generazione del **segnale campionato** simulato, composto di:

- una componente utile (sinusoide);
- un disturbo (sinusoide a frequenza differente);
- rumore a larga banda;

IN PUNTI:

1. creare un segnale sinusoidale con sovrapposto un disturbo sinusoidale e rumore a larga banda;
2. creare la function `adc_ideal()`;
3. determinare i parametri del filtro;
4. calcolare i coefficienti del filtro;
5. verificare la risposta in frequenza del filtro;
6. filtrare il segnale;
7. confrontare i segnali nel dominio del tempo pre- e post-filtraggio.

Attenzione a:

1) **SEGUIRE SCRUPOLOSAMENTE LE INDICAZIONI FORNITE IN 2.2.** Simulando di avere a che fare con condizioni non modificabili del segnale da trattare e del sistema di acquisizione a disposizione
2) Potete **sfruttare il codice generato nel LAB 3: segnale utile e disturbo saranno due sinusoidi, rumore a larga banda generato come visto con randn.**

Obiettivi dell'esercitazione...

IN PUNTI:

1. creare un segnale sinusoidale con sovrapposto un disturbo sinusoidale e rumore a larga banda;
2. creare la function `adc_ideal();`
3. determinare i parametri del filtro;
4. calcolare i coefficienti del filtro;
5. verificare la risposta in frequenza del filtro;
6. filtrare il segnale;
7. confrontare i segnali nel dominio del tempo pr

adc_real() simula il comportamento di un quantizzatore ideale:

1) Riutilizzare il codice generato durante la scorsa esercitazione, per creare una **funzione** avente come:

Input → valore di fondo scala (V_FS), numero di bit (N_bit) e segnale da quantizzare (V);

Output → segnale quantizzato (Vq) ed eventualmente il passo di quantizzazione (Delta)

(vedi file ***quantizzatore.m*** nelle soluzioni LAB3)

2) Salvare la funzione a parte come visto a lezione in un file avente nome «**nomefunzione.m**»(adc_ideal è indicato solo come esempio).

Attenzione!! Il nome del fine DEVE ESSERE UGUALE al nome della function!

(vedi file ***quantizzatore.m*** su moodle)

Ese2_matlab.m | quantizzatore.m

```
%% Function quantizzatore
%function che realizza un quantizzatore ideale. In output restituisce il segnale quantizzato e il passo di quantizzazione,
% utile poi per il calcolo dell'SNRQ.Volendo è possibile aumentare il
% numero di output forniti, per esempio se si vuole che la funzione
% restituisca anche l'errore allora si modificherà la prima riga come
% [V_q,Delta,error]=.... e lo stesso si farà nel momento in cui si chiama
% la funzione

function [V_q,Delta] = quantizzatore(N_bit, V_FS, V)
N_liv=2^N_bit;                %numero livelli disponibili
Delta=2*V_FS/(N_liv-1);       %passo di quantizzazione
lev=round(V/Delta);           %vettore di livelli
for i=1:length(lev)           %ciclo for che descrive saturazione al di fuori del campo di ingresso
    if lev(i)> 2^(N_bit-1)-1
       lev(i)=2^(N_bit-1)-1;
    end
    if lev(i)< -2^(N_bit-1)
       lev(i)=-2^(N_bit-1);
    end
end
V_q=lev*Delta;
error = V_q - V;
end

```

La funzione **quantizzatore.m**, salvata come file separato, può essere richiamata nel codice utilizzando semplicemente la linea di codice: [Vq,delta] = quantizzatore(N_bit,V_FS,V) dove N_bit, V_FS, V saranno variabili definite prima di invocare la function.

Attenzione!! I nomi delle variabili definite nel file principale possono essere diversi rispetto a quelli dati all'interno della function. L'importante è che vengano forniti alla function nell'ordine corretto!!

Vq sarà il nome della variabile in cui troveremo salvato il segnale quantizzato e delta conterrà il passo di quantizzazione (anche per questi vale la regola dell'ordine e non è importante il nome).

Obiettivi dell'esercitazione...

In punti:

1. creare un segnale sinusoidale con sovrapposto un disturbo sinusoidale e rumore a larga banda;
2. creare la function `adc_ideal()`;
3. determinare i parametri del filtro;
4. calcolare i coefficienti del filtro;
5. verificare la risposta in frequenza del filtro;
6. filtrare il segnale;
7. confrontare i segnali nel dominio del tempo pre- e post-filtraggio.

Cosa vuol dire filtrare? Eliminare alcune componenti del segnale sulla base del suo contenuto in frequenza.

Come si può implementare un filtro in Matlab?

Obiettivi dell'esercitazione: filtraggio

Un filtro può anche essere definito tramite una **relazione ingresso-uscita** nel ***dominio del tempo*** rappresentata da una equazione alla differenze, nella forma:

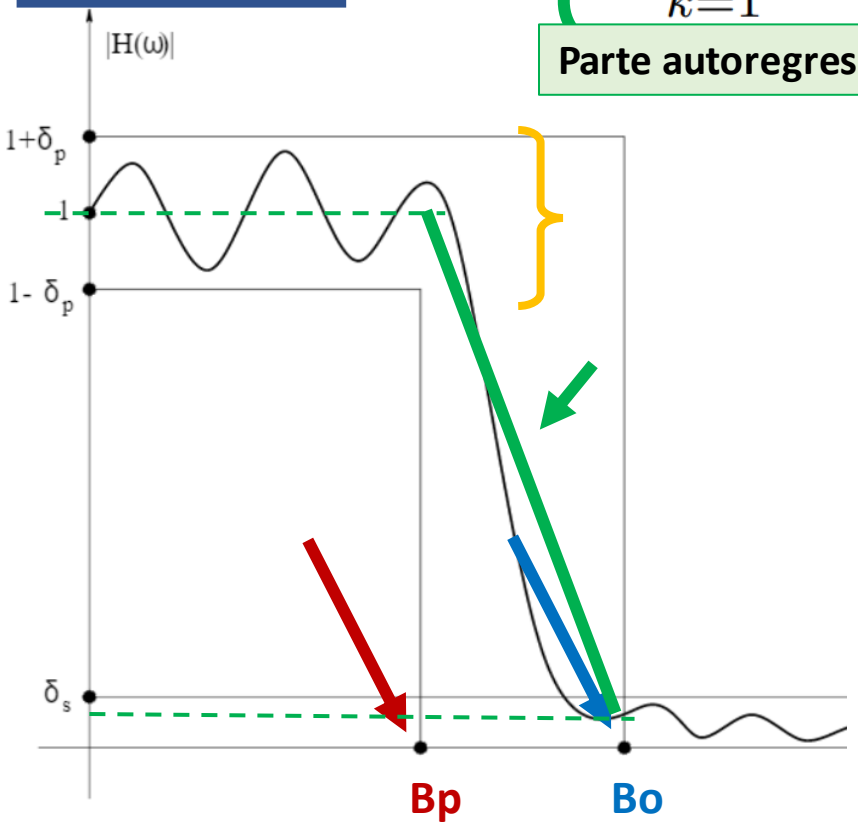
Esempio passa – basso (PB)

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Parte autoregressiva (AR)

Parte a media mobile (MA)

*I termini a_k e b_k sono detti **coefficienti del filtro** e vanno scelti in modo da realizzare **le specifiche volute** (cioè le caratteristiche del filtro che ne determinano la tipologia, i.e.,passa-basso, passa-alto, ecc.).*



Banda Passante (Bp): l'intervallo di frequenze che vogliamo lasciare inalterate (non filtrate), ciò lo si ottiene avendo una risposta in frequenza pari al valore unitario. Nel caso PB: Bp indica la massima frequenza alla quale il guadagno del filtro si mantiene entro la specifica di ondulazione data.

Banda Oscura (Bo): l'intervallo di frequenze che vogliamo eliminare, ciò lo si ottiene mettendo il modulo della risposta in frequenza del filtro pari a zero. Nel caso PB: Bo indica la minima frequenza alla quale il guadagno del filtro è ridotto del fattore di attenuazione dato.

Attenuazione A: indica di quanto vogliamo ridurre l'ampiezza delle frequenze in banda oscura. L'indicazione viene data come rapporto (A= guadagno in Bp/ guadagno in B0).

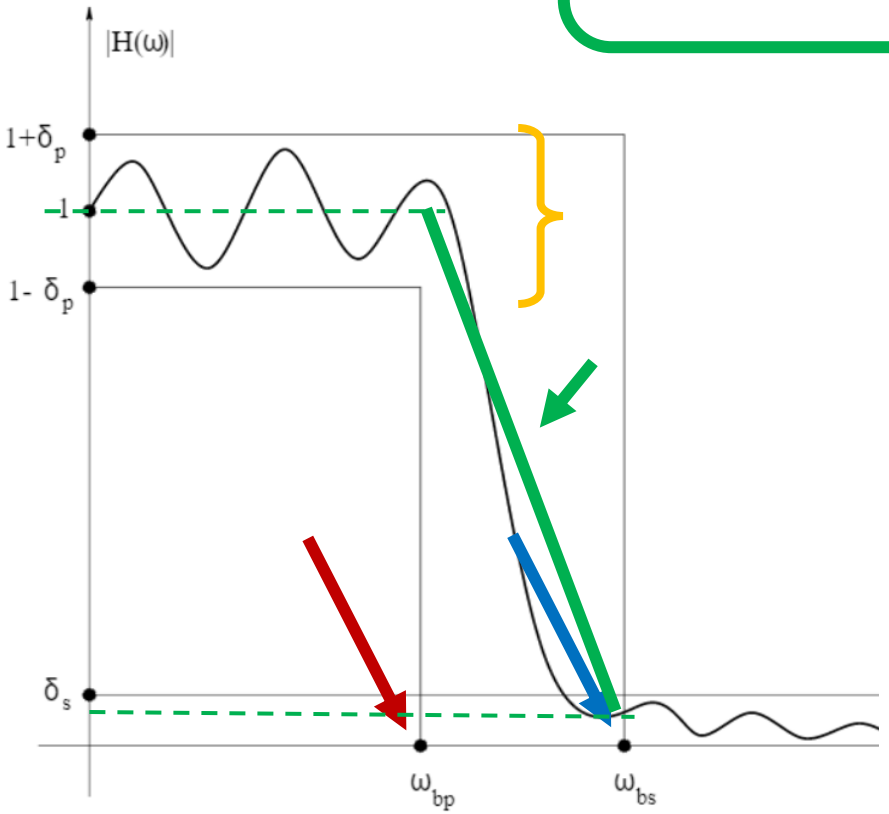
Ondulazione (ripple) in banda passante (R): la massima variazione relativa di guadagno tollerabile in banda passante rispetto al guadagno unitario.

Obiettivi dell'esercitazione: filtraggio

Un filtro può anche essere definito tramite una **relazione ingresso-uscita** nel ***dominio del tempo*** rappresentata da una equazione alla differenze, nella forma:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

*I termini a_k e b_k sono detti **coefficienti del filtro** e vanno scelti in modo da realizzare **le specifiche volute** (a seconda dei valori si possono cioè realizzare filtri passa-passo, passa-alto, ecc.).*



Banda Passante (Bp):

ωp=Bp/(Fs/2) – *normalizzazione per f. di Nyquist*

Banda Oscura (Bo):

ωo=Bo/(Fs/2) - *normalizzazione per f. di Nyquist*

Una volta definiti i parametri legati alle specifiche richieste al filtro, dobbiamo calcolare i coefficienti del filtro (i.e., ωp, ωo, Rs e Rp).

Attenuazione A:

Rs = 20 log10 (A); ***[accettabili valori di A tra 10 e 1000]***

Ondulazione (ripple) in banda passante (R): Attenzione!! Bisogna prima calcolare i valori massimi e minimi del guadagno normalizzato.

[accettabili valori di R tra 1% e 10%]

$R_1 = 20 \log_{10}(1 + R)$

$R_2 = 20 \log_{10}(1 - R)$

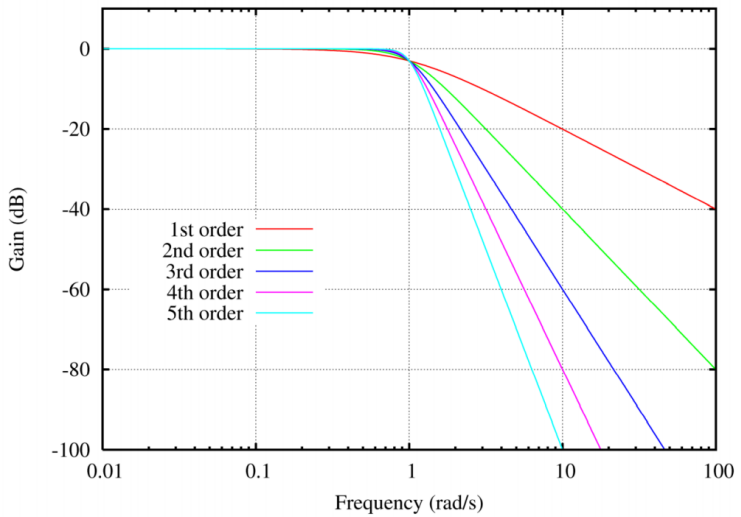
$Rp = (R_1 - R_2)/2;$

Obiettivi dell'esercitazione: filtraggio

La progettazione a partire dalle specifiche è un problema complesso, che in questa esercitazione viene risolto sfruttando **funzioni di elaborazione predefinite nel Signal Processing Toolbox dell'ambiente MatLab.**
Tra le varie opzioni offerte, selezionato **filtro di Butterworth, ottima piattezza e attenuazione accettabile**

$$|G(f)| = \frac{1}{\sqrt{1+\left(\frac{f}{B_n}\right)^{2n}}},$$

Dove n: ordine
Bn: banda passante a -3dB



Come otteniamo un filtro di Butterworth?c

1) DEFINITI I PARAMETRI DEL FILTRO IN TERMINI DI ω_p, ω_s, R_p, R_s

2) SFRUTTATA LA FUNZIONE «buttord» CHE DETERMINA L'ORDINE DEL FILTRO E LA BANDA PASSANTE A -3dB (necessari poi per determinarne la risposta all'impulso)

```
[n Wn] = buttord(Wp, Ws, Rp, Rs);
```

3) SFRUTTATA LA FUNZIONE «butter» CHE DETERMINA I DUE VETTORI b E a CHE CONTERRANNO I COEFFICIENTI DELLA PARTE MOVING AVERAGE E RICORSIVA DELLA RISPOSTA ALL'IMPULSO.

```
[b, a] = butter(n, Wn, type);
```

tipologia di fitro:
'low' per passa-basso
'high' per passa-alto
'bandpass' per passa-banda

3) SFRUTTATA LA FUNZIONE «filter» CHE SFRUTTANDO I COEFFICIENTI a E b DELLA RISP IN FREQUENZA BUTTERWORTH OTTENUTI IN PRECEDENZA, FILTRA I CAMPIONI DEL SEGNALE

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

Per ottenere la risposta in frequenza del nostro filtro:

```
[H, f] = freqz(b, a, K, Fs);
```

K→ #campioni per cui calcolata risp (deve essere potenza di 2)
vettori f ed H → frequenze e la risposta in frequenza calcolata

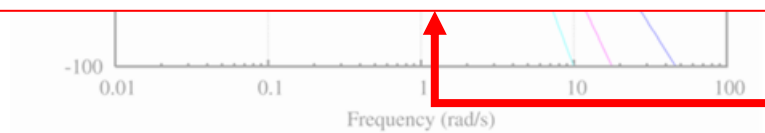
Obiettivi dell'esercitazione: filtraggio

La progettazione a partire dalle specifiche è un problema complesso, che in questa esercitazione viene risolto sfruttando **funzioni di elaborazione predefinite nell Signal Processing Toolbox dell'ambiente MatLab.**

Attenzione a controllare graficamente la risposta in frequenza!

```
figure
plot(f, 20*log10(abs(H)));
```

Per alcuni valori di ordine (> 4) combinati con attenuazione potrebbero verificarsi delle instabilità numeriche a causa degli arrotondamenti e degli step seguiti da Matlab (per più info vedere la sezione Limitation al link https://it.mathworks.com/help/signal/ref/butter.html)



Come otteniamo un filtro di Butterworth?c

1) DEFINITI I PARAMETRI DEL FILTRO IN TERMINI DI ω_p, ω_s, R_p, R_s

2) SFRUTTATA LA FUNZIONE «buttord» CHE DETERMINA L'ORDINE DEL FILTRO E LA BANDA PASSANTE A -3dB (necessari poi per determinarne la risposta all'impulso)

```
[n Wn] = buttord(Wp, Ws, Rp, Rs);
```

3) SFRUTTATA LA FUNZIONE «butter» CHE DETERMINA I DUE VETTORI b E a CHE CONTERRANNO I COEFFICIENTI DELLA PARTE MOVING AVERAGE E RICORSIVA DELLA RISPOSTA ALL'IMPULSO.

```
[b, a] = butter(n, Wn, type);
```

tipologia di fitro:
'low' per passa-basso
'high' per passa-alto
'bandpass' per passa-banda

3) SFRUTTATA LA FUNZIONE «filter» CHE SFRUTTANDO I COEFFICIENTI a E b DELLA RISP IN FREQUENZA BUTTERWORTH OTTENUTI IN PRECEDENZA, FILTRA I CAMPIONI DEL SEGNALE

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

Per visualizzare la risposta in frequenza del filtro:

```
[H, f] = freqz(b, a, K, Fs);
```

K→ #campioni per cui calcolata risp (deve essere potenza di 2)
vettori f ed H → frequenze e la risposta in frequenza calcolata

Obiettivi dell'esercitazione...

IN PUNTI:

1. creare un segnale sinusoidale con sovrappost
   rumore a larga banda;
2. creare la function `adc_ideal()`;
3. determinare i parametri del filtro;
4. calcolare i coefficienti del filtro;
5. verificare la risposta in frequenza del filtro;
6. filtrare il segnale;
7. confrontare i segnali nel dominio del tempo pre- e post-filtraggio.

Per poter apprezzare se il design del filtro ha funzionato:

1) **plottare sovrapposti nel tempo**:

→ Segnale utile (l'obiettivo creato inizialmente, pulito da rumori);

→ Segnale sporco in uscita dal quantizzatore (segnale utile + disturbo + rumore + rumore di quantizzazione) ;

→ Segnale filtrato

2) Volendo osservare quanto fatto anche nel **dominio delle frequenze,** sovrapporre gli spettri in frequenza dei segnali (utilizzare pure periodogram).