Capitolo 2

Simulazione dell'acquisizione di segnali e filtraggio

2.1 Introduzione – function MatLab

L'uso della simulazione per analizzare un problema di acquisizione di segnali dà il vantaggio di poter conoscere tutte le componenti in gioco, consentendo il confronto tra i risultati dell'elaborazione ed il segnale originario, "pulito". La simulazione, d'altra parte, va costruita in modo da presentare una situazione di impiego realistica.

I segnali considerati in questa guida sono volutamente semplici per facilitare la comprensione del problema, ma il modo in cui si riproducono le condizioni di prova rimane valido anche in situazioni più complesse.

L'esercitazione si svolge in quattro fasi:

1. generazione del **segnale campionato** simulato, composto di:
   - una componente utile (sinusoide);
   - un disturbo (sinusoide a frequenza differente);
   - rumore a larga banda;
2. realizzazione di una *function* che riproduca il comportamento di un quantizzatore;
3. sintesi di un filtro che, applicato al segnale, permetta di migliorarne il rapporto segnale-rumore e di eliminare la componente di disturbo;
4. visualizzazione del segnale filtrato e confronto con la componente utile del segnale simulato.

L'obiettivo della prima parte dell'esercitazione è costruire la simulazione dell'acquisizione di un segnale tramite un modulo DAQ. Nella parte successiva, si analizza l'impiego di un filtro per separare la componente utile del segnale

dai disturbi e ridurre l'effetto del rumore a larga banda. A questo scopo ci si avvale di *function* MatLab: alcune di queste sono funzioni predefinite, altre invece vanno realizzate. L'intero lavoro viene svolto in forma di simulazione e richiede il solo ambiente MatLab.

La creazione di una funzione in MatLab segue una sintassi molto semplice:

```
function y = function_name(x1, x2, ...  , xN)
⋮
end
```

dove `function_name` è il nome che si vuole dare alla funzione, `y` è la variabile di uscita, `x1, x2, ..., xN` sono le variabili di ingresso[1]. Il corpo della funzione è costituito da tutte le istruzioni comprese tra le parole chiave `function` ed `end`.

Una *function* può essere memorizzata in un file `.m` separato, oppure trovarsi entro lo *script* principale. In entrambi i casi, in uno *script* MatLab l'istruzione che richiede l'esecuzione della funzione ha esattamente la stessa forma della riga di apertura sopra riportata di solito terminata con ';'). I nomi utilizzati per le variabili entro una *function* possono essere diversi da quelli utilizzati nello *script* principale.

ATTENZIONE
Le variabili utilizzate all'interno di una funzione non sono considerate parte del *workspace* dello *script* principale, quindi i loro valori non sono visualizzati dall'ambiente MatLab, cosa che può ostacolare la messa a punto degli *script* più complessi.

Si suggerisce quindi di utilizzare con accortezza la struttura `function` e di inserirvi righe di codice già verificato e strutturato adeguatamente.

2.2 Generazione del segnale simulato

1. stabilire le caratteristiche delle tre componenti del segnale, ossia **segnale utile**, **disturbo** e **rumore**. In particolare:
   - **frequenza** della sinusoide che rappresenta il **segnale utile**: a scelta, compresa **tra 200 Hz e 1 kHz**;
   - **frequenza** della sinusoide che rappresenta il **disturbo**: a scelta, compresa **tra 4 kHz e 7 kHz**;
   - ampiezza della sinusoide che rappresenta il **disturbo**: a scelta, compresa **tra 10% e 100%** dell'ampiezza del segnale utile;

[1] Se ci sono più variabili di uscita, queste vanno raccolte in un vettore: `function [y1 y2, ..., yM] = function_name(x1, x2, ...  , xN)`.

2. scegliere la **frequenza di campionamento** `Fs`, tenendo presente che il valore massimo è **250 kHz**;

3. stabilire la **durata** del segnale: il valore, a scelta, deve essere compreso **tra 0.1 s e 1 s**;

4. generare **due** vettori di campioni della stessa lunghezza, ciascuno contenente una sinusoide (segnale utile e disturbo). A questo scopo si può realizzare una *function* sfruttando il codice creato nella precedente esercitazione MatLab, o realizzarne una simile;

5. generare un vettore di campioni di **rumore**, della stessa lunghezza degli altri due, utilizzando la funzione MatLab `randn`. La *function* MatLab `length(x)` restituisce la lunghezza del vettore `x`.

   Si ricorda che la funzione MatLab `randn(i,j)`, genera una matrice con `i` righe e `j` colonne, i cui elementi hanno valori tratti da una densità di probabilità Gaussiana standard (cioè, con media 0 e varianza 1). Se `i` $=$ 1, viene generato un vettore riga.

   Noto il segnale utile e stabilito il valore di SNR, i valori generati vanno moltiplicati per la deviazione standard ottenuta dal calcolo della corrispondente varianza del rumore. Il valore del rapporto segnale-rumore (SNR) potrà essere variato nel corso dell'esercitazione.

6. utilizzando i comandi `figure`, `plot` e `hold on`, creare un grafico in cui siano sovrapposti, **nell'ordine**:

   - la somma di segnale utile, disturbo e rumore;
   - la somma di segnale utile e disturbo;
   - il solo segnale utile.

2.3 Quantizzatore

Questa parte dell'esercitazione prevede la creazione di una *function* che riproduce il comportamento di un quantizzatore ideale. Nel seguito essa viene indicata come:

```
x_q = adc_ideal(Nbit, V_FS, x);
```

La funzione può essere costruita semplicemente riutilizzando le istruzioni di cui si è fatto uso nell'esercitazione precedente. I parametri di ingresso della *function* sono il valore di fondo scala del quantizzatore, `VFS`, specificando il quale si intende che il campo di ingresso sia $\pm$`VFS`, e la variabile `Nbit` che indica il numero di bit. Si deve decidere se la variabile `x` è un singolo campione di

ingresso, oppure il vettore di ingresso contenente tutti i campioni del segnale. La variabile `xq` è la corrispondente uscita in cui i campioni sono stati quantizzati.

In base alla scelta fatta, cambia il modo in cui la *function* viene impiegata nello *script*.

La *function* `adc_ideal( )` realizzata va utilizzata per quantizzare il segnale simulato come somma di componente utile, disturbo e rumore.

2.4 Filtraggio

Filtrare un segnale consiste nell'eseguire la convoluzione nel dominio del tempo tra i campioni del segnale di ingresso $x(n)$ e i campioni della **risposta all'impulso** del filtro, $h(n)$, che è la risposta di un sistema a tempo discreto al campione unitario $\delta(n)$ e ne descrive completamente le caratteristiche. Nel dominio della frequenza l'operazione corrisponde al prodotto tra lo spettro del segnale d'ingresso e la risposta in frequenza del filtro.

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

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{2.1}$$

La prima parte dell'equazione riguarda i campioni precedenti dell'uscita ed è chiamata *autoregressiva* (AR) mentre la seconda parte, detta *a media mobile* (*moving average*, MA) è una combinazione lineare dei valori dei precedenti campioni di ingresso.

I filtri costituiti dalla sola parte MA, hanno risposta impulsiva di durata finita (*finite impulse response*, FIR) e l'uscita dipende solo dagli ingressi precedenti. Nei filtri che hanno sia parte MA sia la componente ricorsiva data dalla parte AR l'uscita del filtro dipende non solo dagli ingressi precedenti ma anche dalle uscite precedenti. In genere, un filtro di questo tipo ha una risposta all'impulso di durata infinita (*infinite impulse response, IIR*).

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.). Una volta noti i valori di a_k e i b_k il comportamento del filtro è descritto completamente. L'operazione di filtraggio consiste nel realizzare la (2.1)in forma di algoritmo di calcolo, ma può essere facilitata sfruttando l'esecuzione della *function* MatLab:

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

dove `x` è il vettore contenente i campioni del segnale da filtrare (ingresso del filtro) e `y` è il vettore che restituisce i campioni del segnale filtrato (uscita del filtro), dati i vettori dei coefficienti, `a` e `b`.

2.4.1 Specifiche del filtro

La definizione delle caratteristiche del filtro, necessaria per determinarne i coefficienti, richiede di stabilire le **specifiche**, cioè indicare il comportamento che si vuole ottenere. In questa esercitazione si realizzerà un **filtro passa basso**, dato che nel segnale simulato le componenti indesiderate sono a frequenza più alta del segnale utile.

I parametri da definire sono:

Banda Passante B_P: l'intervallo di frequenze in cui il modulo della risposta in frequenza del filtro ha circa valore unitario, cioè le frequenze che vogliamo lasciare inalterate, è $(0, B_P)$. B_P indica la massima frequenza alla quale il guadagno del filtro si mantiene entro la specifica di **ondulazione** data (vedere al quarto punto);

Banda Oscura B_O: l'intervallo di frequenze in cui il modulo della risposta in frequenza del filtro tende a zero, cioè le frequenza che vogliamo vengano eliminate, è (B_O, `Fs/2`). B_O indica la minima frequenza alla quale il guadagno del filtro è ridotto del fattore di **attenuazione** dato (vedere al terzo punto);

Attenuazione A: tale valore indica il fattore di riduzione di ampiezza che si vuole ottenere per le componenti di segnale nella banda oscura, rispetto alla banda passante. L'indicazione va intesa come **rapporto**: supponendo che nella banda passante il guadagno del filtro sia pari ad 1, a guadagno nullo in banda oscura corrisponde attenuazione infinita (totale eliminazione delle componenti di segnale in banda oscura, praticamente irrealizzabile). Si possono considerare accettabili per questa esercitazione valori compresi tra 10 e 1000, ricordando che più alto è il valore di attenuazione, migliore è la riduzione di disturbi e rumore;

Ondulazione (ripple) in banda passante R: rappresenta la **massima variazione relativa di guadagno** tollerabile in banda passante rispetto al guadagno unitario. Si possono considerare accettabili per questa esercitazione valori compresi tra 1% e 10%, ricordando che più basso è il valore di ondulazione, più accurata è la riproduzione della componente utile del segnale.

Una volta determinate le specifiche, si procede con il calcolo dei coefficienti del filtro.

ATTENZIONE

Le specifiche del filtro vanno tradotte in valori da assegnare ad altrettante variabili MatLab (indicate come `Wp`, `Ws`, `Rs` e `Rp`) che vengono passate alle funzioni per la sintesi del filtro. Porre attenzione al fatto che:

- le frequenze devono essere **normalizzate rispetto alla frequenza di Nyquist,** `Fs/2`. I valori di `Wp` $= B_P/($`Fs/2`$)$ e `Ws` $= B_O/($`Fs/2`$)$ sono valori compresi tra 0 e 1, con `Wp` $<$ `Ws`;

- i valori di attenuazione ed ondulazione vanno espressi in **decibel** (dB), pertanto:

  - per l'attenuazione si ha: `Rs` $= 20 \log_{10} A$;
  - per il parametro `Rp`, invece, si devono prima valutare i valori massimi e minimi del guadagno normalizzato, ossia:

$$\begin{aligned} R_1 &= 20 \log_{10}(1 + R) \\ R_2 &= 20 \log_{10}(1 - R) \end{aligned}$$

    quindi si può porre: `Rp` $= (R_1 - R_2)/2$

2.4.2 Realizzazione del filtro

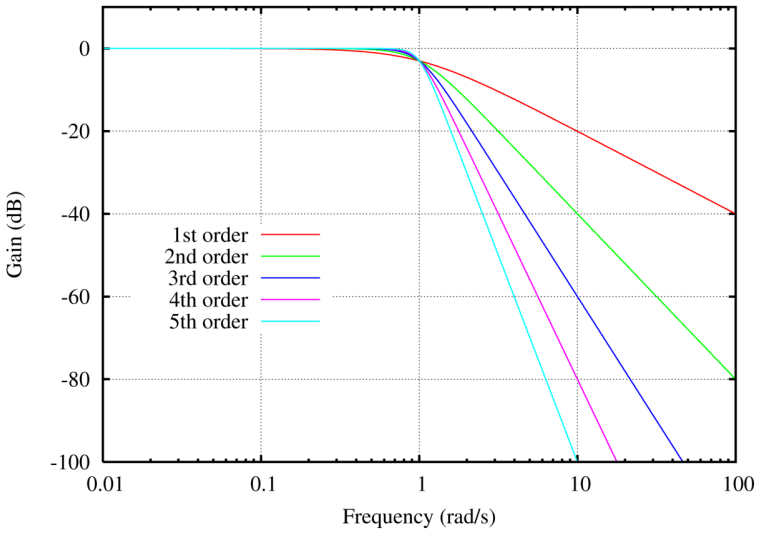


Figura 2.1: Risposta in frequenza di filtri di Butterworth di diverso ordine. L'asse delle frequenze è normalizzato rispetto alla banda passante B_P. (da *Wikimedia Commons*)

La progettazione di un filtro numerico 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. I filtri numerici possono essere approssimazioni di espressioni matematiche ottenute tramite funzioni interpolatrici (filtri di Chebyshev, filtri ellittici), oppure **filtri di Butterworth**. Il modulo della risposta in frequenza di un filtro di

Butterworth di ordine n è:

$$|G(f)| = \frac{1}{\sqrt{1 + \left(\frac{f}{B_n}\right)^{2n}}}, \tag{2.2}$$

dove B_n è la banda passante a -3 dB. L'andamento di questa funzione per diversi valori di n è illustrato in Fig. 2.1.

Nel seguito si utilizza un filtro di Butterworth, che viene sintetizzato tramite due *function* MatLab. La prima:

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

determina l'ordine del filtro `n` e la sua banda passante a -3 dB normalizzata `Wn`. Dovrà risultare `Wp` $<$ `Wn` $<$ `Ws`. Questi parametri vengono passati alla successiva *function*:

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

che restituisce i due vettori `b` ed `a` contenenti, rispettivamente, i coefficienti della parte MA e della parte AR del filtro secondo la (2.1), necessari alla *function* `filter` per realizzare il filtro di Butterworth.

Il parametro `type` indica la tipologia di fitro desiderata: `'low'` realizza un filtro passa-basso, `'high'` un passa-alto e `'bandpass'` un filtro passa-banda. **Per questa esercitazione è necessario un filtro passa-basso, pertanto va specificata la stringa** `'low'`.

Inserendo di seguito nello *script* l'istruzione

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

descritta in precedenza, si ottiene il vettore dei campioni filtrati.

Per verificare se la risposta in frequenza del filtro che è appena stato progettato corrisponde a ciò che si voleva ottenere si può utilizzare la *function* Matlab:

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

che, dati i coefficienti del filtro contenuti nei vettori `a` e `b` e la frequenza di campionamento `Fs`, calcola la risposta in frequenza su `K` campioni (dove `K` **deve** essere una potenza di 2, p. es., 1024). In uscita sono forniti i vettori `f` ed `H`, cioè l'asse delle frequenze e la risposta in frequenza calcolata. Per creare il grafico della risposta in frequenza si utilizzano le istruzioni:

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

dove va notato che l'espressione `20*log10(abs(H)` serve a presentare il guadagno del filtro in scala logaritmica.

Va tenuto presente che più le richieste sono stringenti, cioè più ci si avvicina ad un filtro ideale, più l'ordine del filtro `n` risulterà essere elevato. Dato che l'ordine del filtro indica il numero di campioni dell'ingresso che servono per determinare l'uscita, va ricordato che si avrà un transitorio di risposta ed il filtro inizierà a funzionare correttamente solo dopo almeno `n` campioni.

2.5 Valutazione dei risultati

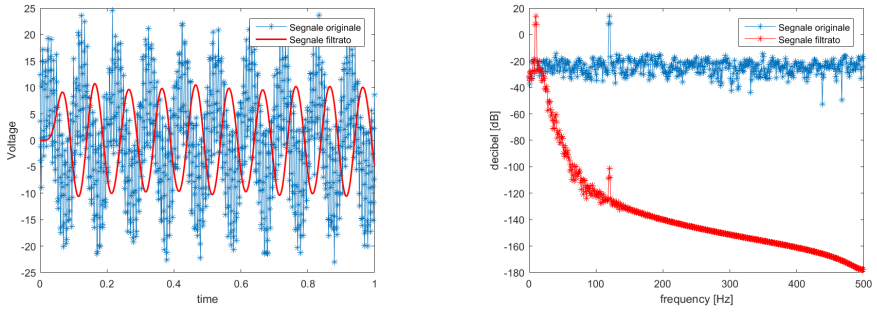
Per completare l'esercitazione, utilizzando i comandi MatLab `figure`, `plot` e `hold on`, creare un grafico in cui siano sovrapposti:

- il segnale ottenuto all'uscita del quantizzatore;
- il solo segnale utile, creato all'inizio della simulazione;
- il segnale filtrato,

in modo da poter verificare l'effetto del filtro e la qualità del segnale ottenuto.

In figura 2.2(a) è mostrato un esempio di segnale da filtrare, formato da un segnale sinusoidale a 10 Hz, un disturbo a 120 Hz e rumore bianco gaussiano. Il segnale in rosso è invece il segnale ottenuto in uscita dal filtro applicato: la componente sinusoidale a 10 Hz è mantenuta invariata mentre tutto il resto del segnale è stato eliminato.

Nella figura 2.2(b) sono riportati gli spettri dei due segnali nel dominio della frequenza, per metter in evidenza l'effetto del filtro.



(a) Traccia nel tempo.

(b) Spettro del segnale.

Figura 2.2: Confronto segnale originale e segnale filtrato.

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.