# Capitolo 1

# Creazione di uno script MatLab per la generazione di un segnale semplice

## 1.1 Script MatLab – Premessa

L'ambiente MatLab permette sia di eseguire singole istruzioni, impartite attraverso la *Command Window*, sia di porre in esecuzione sequenze di istruzioni raccolte in uno *script*. Gli *script* sono file di testo, che è possibile creare utilizzando l'*Editor* MatLab ed il cui *filename* è contraddistinto dall'estensione `.m`.

Le istruzioni in uno script sono eseguite in sequenza, con la possibilità di introdurre i comuni costrutti della programmazione, come ad esempio cicli `for` o istruzioni condizionate di tipo `if ...then ...else`.

È buona cosa iniziare sempre uno *script* Matlab con i comandi:

- `clc` – ripulisce la *command window*;
- `clear all` – elimina tutte le variabili presenti nel *workspace*;
- `close all` – cancella tutte le figure precedenti.

Prima di generare il segnale desiderato è opportuno stabilire i valori dei parametri attraverso istruzioni di assegnazione. Per creare un segnale sinusoidale, descritto dall'equazione $v(t) = V_0 \sin(2\pi F_0 t + \phi_0)$ bisogna quindi assegnare valori ai parametri:

- `V_0` – ampiezza del segnale;
- `F_0` – frequenza del segnale;
- `Phi_0` – fase iniziale del segnale.

La sintassi dell'ambiente MatLab non è particolarmente vincolante e si deve ricordare che le variabili sono implicitamente definite dalla prima istruzione che ne fa uso. Tutte le variabili utilizzate da uno *script* saranno messe in evidenza nel *Workspace*.

Si deve tenere presente che MatLab non è a conoscenza delle unità di misura associate ai valori numerici assegnati. È consigliabile avvalersi di commenti (preceduti dal carattere '%') per ricordare in modo esplicito che, ad esempio, l'assegnazione: `F_0 = 20;` significa che la frequenza del segnale è di 20 Hz.

Raccogliere tutte le istruzioni di assegnazione in una parte dedicata del codice ne facilita la lettura e l'interpretazione. Un esempio è riportato qui di seguito:

```
% **********************
% Parametri del segnale
% **********************

 V_0 = 1;            % ampiezza [V]
 F_0 = 20;           % frequenza [Hz]
 Phi_0 = pi/6;       % fase iniziale [rad]
```

Si osservi che tutte le istruzioni sono state terminate con il carattere ';', per evitare che il loro risultato sia riportato sulla *Command Window*.

## 1.2 Creazione dei campioni

Il segnale simulato viene generato come **vettore** di valori numerici. Va definita la sua **durata**, indicata nel seguito dalla variabile `T_W` (espressa in secondi), tenendo conto del fatto che esso dovrà essere visualizzato sul *monitor* creando una `figure`, sulla quale saranno tracciati uno o più `plot`. Si suggerisce di scegliere il valore di `T_W` in modo il numero, non necessariamente intero, di periodi visualizzati sia limitato. In caso contrario, diventerebbe impossibile vedere in dettaglio il segnale.

In simulazione, la grandezza **tempo** (indicata dalla variabile continua $t$ nella formula sopra) è costituita da un **vettore** di valori di tempo. Ovviamente è impossibile realizzare un segnale realmente **continuo nel tempo** ma, data la durata `T_W`, prendendo un numero molto elevato di punti essi saranno così vicini da approssimare molto bene l'andamento continuo del segnale.

L'asse dei tempi del segnale continuo può essere creato tramite l'istruzione:

```
t = linspace(0, T_W, N);
```

che pone `N` punti equispaziati nell'intervallo tra $0$ e $T_W$; in questo modo la distanza tra punti consecutivi è $T_W/(N-1)$.
Il segnale sinusoidale continuo si ottiene quindi con il comando:

```
V = V_0 * sin(2*pi*F_0*t + Phi_0);
```

**ATTENZIONE**
I parametri dalla funzione `linspace` non danno l'indicazione diretta dell'intervallo di campionamento. Si presti attenzione al valore del rapporto $(T_W/N)$ per evitare scelte inappropriate.

Per rappresentare invece un **segnale campionato** è necessario definire una frequenza di campionamento `F_s` e costruire un vettore di valori di tempo distanziati tra loro dell'intervallo `T_s = 1/F_s`. In questo caso, se si vuole determinare un numero di campioni che corrisponda approssimativamente all'intervallo $T_W$ va calcolato il valore:

```
N_sample = round(T_W / T_s);
```

(il comando `round()` arrotonda all'intero più vicino) ed il vettore dei campioni temporali si può generare direttamente con l'istruzione:

```
t_sample = [0:1:(N_sample - 1)] * T_s;
```

Il corrispondente vettore di valori campionati (indicato con `V_sample` nel seguito) si ottiene semplicemente sostituendo, nell'istruzione precedente, il vettore `t` con il vettore `t_sample`:

```
V_sample = V_0 * sin(2*pi*F_0*t_sample + Phi_0);
```

Il segnale continuo e la sua versione campionata si possono confrontare nello stesso grafico, generato usando la sequenza di comandi:

```
figure
plot (t, V)
hold on
plot (t_sample, V_sample, '*r')
```

dove il comando `hold on` permette di sovrapporre più grafici nella stessa figura. I caratteri tra apici permettono invece di scegliere le caratteristiche del grafico (colore, linea continua, tratteggiata – leggere l'help della funzione `plot()`).

La figura generata dallo *script* permette di confrontare l'andamento nel tempo dei due segnali. È possibile ripetere l'esecuzione per diversi valori della frequenza di campionamento e verificare che (in base alle condizioni del teorema del campionamento) il valore assegnato a `F_s` sia adeguato.

In un nuovo grafico, ottenuto con le istruzioni:

```
figure
plot (t_sample, V_sample)
```

si può passare a considerare l'andamento del solo segnale campionato. Si noti che, in assenza di indicazioni specifiche, nel grafico i campioni del segnale sono collegati da segmenti di retta. Questo si può interpretare come una **ricostruzione** del segnale continuo nella quale è utilizzata una interpolazione **lineare** dei campioni.

Si può osservare che l'aspetto del segnale può variare sia con la frequenza di campionamento scelta, sia con differenti valori della fase iniziale `Phi_0`.

**Qual è la frequenza di campionamento che permette di visualizzare una forma d'onda simile a quella continua? Confrontare il valore con quello minimo richiesto dal teorema del campionamento.**

## 1.3 Quantizzatore uniforme

I parametri che definiscono il quantizzatore sono:

- il valore di fondo scala, qui indicato con `V_FS`, specificando il quale si intende che il campo di ingresso sia $\pm V_{FS}$;
- la variabile `Nbit` che indica il numero di bit $b$. A questo corrispondono $2^b$ livelli di quantizzazione, i cui valori numerici appartengono all'insieme:

$$\{-2^{\texttt{Nbit}-1}, \ldots, 0, \ldots 2^{\texttt{Nbit}-1} - 1\}.$$

Le operazioni necessarie sono il calcolo del passo di quantizzazione ed il calcolo del valore quantizzato. Applicando la **quantizzazione con arrotondamento** si può fare uso della funzione MatLab `round( )` per esprimere l'ampiezza del segnale in "numero di passi di quantizzazione". Un successivo test è necessario per verificare se il valore intero così ottenuto sia entro il range di valori consentito dal numero di bit assegnato al quantizzatore.

Nella zona di funzionamento lineare, indicato con `Delta_q` il passo di quantizzazione, il numero intero che indica il livello di quantizzazione si può ottenere quindi come:

```
q(i) = round(x(i)/Delta_q);
```

È importante limitare i valori degli interi così ottenuti, in modo che corrispondano a valori compresi nel campo di ingresso. Quindi, si dovranno anche verificare le condizioni:

| | | | |
|---|---|---|---|
| se | $\texttt{q(i)} > 2^{\texttt{Nbit}-1} - 1$ | allora | $\texttt{q(i)} = 2^{\texttt{Nbit}-1} - 1$ |
| se | $\texttt{q(i)} < -2^{\texttt{Nbit}-1}$ | allora | $\texttt{q(i)} = -2^{\texttt{Nbit}-1}$ |

che introducono l'effetto di **saturazione** del quantizzatore. Il valore quantizzato del campione `x` si ottiene infine come:

```
x_q(i) = q(i) * Delta_q;
```

Può essere calcolato anche l'errore di quantizzazione, che è definito come la differenza tra il segnale quantizzato e quello reale: `eq(i) = xq(i) - x(i)`.

### Nota

Per verificare il comportamento del quantizzatore realizzato si può utilizzare come vettore di ingresso `x_test` una rampa lineare da $-$`VFS` a $+$`VFS` (questo può essere creado, ad esempio, usando nuovamente il comando `linspace()`), che deve avere un numero di campioni $K$ molto maggiore del numero di livelli di quantizzazione (ossia, $K \gg 2^b$). Il grafico del corrispondente vettore di uscita dovrà riprodurre l'andamento della caratteristica di quantizzazione; la differenza tra i due dà l'errore di quantizzazione.

## 1.4 Stima dei parametri del segnale

In questa parte dell'esercitazione si utilizzano i campioni simulati del segnale per determinare parametri del corrispondente segnale "analogico", di cui si conosce l'espressione matematica. Si suppone che il **numero** di campioni sia prefissato (ad esempio `N_sample = 250`), condizione che corrisponde, ad esempio, al caso in cui si fa uso di una memoria di dimensione finita per registrare i valori dei campioni acquisiti.

Il valore efficace del segnale sinusoidale si può calcolare matematicamente, noto il valore `V_0` della sua ampiezza, utilizzando la formula $V_{RMS} = \frac{V_0}{\sqrt{2}}$. Questo è il risultato del calcolo del valore efficace secondo la relazione teorica:

$$V_{RMS} = \sqrt{\frac{1}{T_0}\int_{t_0}^{t_0+T_0} v^2(t)dt}$$

dove $T_0 = 1/F_0$ è il periodo della sinusoide.

Quando si conoscono soltanto campioni del segnale in numero finito $N$, si può cercare di ottenere una stima approssimata del valore efficace applicando la formula:

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} v^2(nT_s)}$$

Per calcolare il valore di $V_{RMS}$ si può, per prima cosa, calcolare il vettore $v^2(nT_s)$, che si ottiene elevando al quadrato ciascun elemento di $v(nT_s)$. L'istruzione MatLab è:

```
v2_sample = v_sample .^ 2;
```

dove il punto anteposto all'operatore di elevamento a potenza sta ad indicare che l'operazione va eseguita, individualmente, su ciascun elemento del vettore. Il risultato è lo stesso che si otterrebbe con l'istruzione:

```
v2_sample = v_sample .* v_sample;
```

dove, ancora, l'operatore '`.*`' indica la moltiplicazione tra i corrispondenti elementi di ciascun vettore.

Per completare il calcolo del valore efficace, ci si può avvalere delle funzioni predefinite di MatLab:

- `sum()`, che calcola la somma degli elementi di un vettore;
- `sqrt()`, che calcola la radice quadrata di un numero.

In generale la stima così ottenuta non corrisponde esattamente al valore teorico, dato che il numero di campioni $N$ non coincide esattamente con un numero intero di periodi della sinusoide. Fissata la frequenza `F_0` della sinusoide e la sua ampiezza `V_0`, si provi a variare la fase `Phi_0` e/o la frequenza di campionamento `F_s`.

**Confrontare il valore efficace (RMS, Root Mean Square) del segnale ottenuto a partire dai campioni discreti con quello derivante dall'espressione teorica; ci sono variazioni del valore RMS al variare della frequenza di campionamento o della fase?**

## 1.5 Rumore additivo

Qualunque insieme di dati sperimentali ottenuti tramite un sistema di acquisizione è affetto da rumore, che può presentarsi in forma di fluttuazioni casuali nel segnale acquisito, oppure in forma di componenti spurie del segnale stesso. In questa parte dell'esercitazione si riproduce questa condizione tramite l'aggiunta di valori casuali a ciascuno dei campioni del segnale. Si tratta quindi di generare un vettore, della stessa dimensione del vettore dei campioni del segnale, che contenga i valori del rumore e sommare poi i due vettori.

Per questo scopo MatLab mette a disposizione le due funzioni:

`rand(n1, n2)` e `randn(n1, n2)`.

La prima genera numeri casuali, **uniformemente distribuiti** tra 0 e 1, con i quali forma una matrice di `n1` righe ed `n2` colonne. La funzione `randn()` procede allo stesso modo, ma genera numeri causali con **densità di probabilità**

**Gaussiana**, media 0 e varianza 1. Molto spesso il rumore è caratterizzato da una distribuzione Gaussiana dei valori, quindi la funzione utilizzata sarà la seconda.

Come prima cosa bisogna stabilire il livello di rumore che si desidera generare. Di solito si fa riferimento al **rapporto segnale/rumore** (SNR, *signal-to-noise ratio*), ossia il rapporto tra la potenza del segnale e quella del rumore. Con un segnale simulato, si può considerare il quadrato del valore efficace, $V_{RMS}^2$ ed allo stesso modo la potenza del rumore $P_{noise}$, che si può intendere in senso statistico come **varianza**, ha dimensione di [*ampiezza*$^2$].

Una volta calcolato il valore $V_{RMS}^2$, si può quindi assegnare un valore al rapporto $SNR = V_{RMS}^2/P_{noise}$. Ad esempio, se si vuole che la potenza del rumore sia $\frac{1}{10}$ di quella del segnale, si avrà $SNR = 10$ e, di conseguenza, $P_{noise} = V_{RMS}^2/10$. Per ottenere un vettore di campioni di rumore con questo valore di potenza occorre quindi moltiplicare i campioni generati dalla funzione `randn()` per il corrispondente valore di deviazione standard, ossia $\sqrt{P_{noise}}$. L'istruzione con cui si genera il vettore desiderato è:

```
noise = sqrt(P_noise) * randn(1, N_sample);
```

Il segnale totale è la somma di quello sinusoidale e del rumore: `v_noise = v_sample + noise`.

Generare una figura in cui sono tracciati i grafici di entrambi i segnali, quello rumoroso e quello non, ed inserire le etichette per gli assi e la legenda: usare i comandi `hold on`, `xlabel`, `ylabel` e `legend`. Utilizzare colori diversi per le due tracce, specificandoli attraverso stringhe nel comando `plot()`, ad esempio: `'r'` = rosso, `'k'` = nero, `'g'` = verde.

Confrontare l'SNR teorico con quello del segnale simulato, ottenibile attraverso un comando del tipo: `x = snr(y)`, dove `y` è il vettore contenente i campioni di un segnale rumoroso di cui si vuole stimare il valore di SNR e `snr()` è una funzione MatLab.

In punti:

1. definire i parametri della sinusoide;
2. definire l'intervallo di osservazione $T_w$;
3. definire l'asse dei tempi $t$;
4. creare il segnale sinusoidale *continuo*;
5. definire la frequenza di campionamento;
6. calcolare l'asse dei tempi discretizzato;
7. confrontare i due segnali sinusoidali;
8. costruire il segnale quantizzato;
9. determinare l'RMS;
10. osservare le variazioni dell'RMS al variare della frequenza di campionamento;
11. aggiungere rumore al segnale;
12. confrontare il segnale rumoroso con quello non;
13. calcolare SNR.