Capitolo 3

Introduzione a Simulink

Simulink, prodotto dalla *Mathworks Inc.* è un programma per la simulazione di sistemi dinamici [6]. Estende le potenzialità di *Matlab*, aggiungendo molte funzioni specifiche e mantenendo le caratteristiche generali.

Simulink viene utilizzato attraverso due fasi: quella di definizione del modello da simulare e quella di analisi del sistema stesso. Spesso questi due passi vengono eseguiti sequenzialmente modificando i parametri del sistema al fine di ottenere il comportamento desiderato.

Affinché la definizione del modello possa essere immediata, *Simulink* utilizza un ambiente a finestre, chiamate *Block diagram windows* attraverso cui creare i modelli semplicemente impiegando il mouse.

L'analisi del modello avviene sia scegliendo le opzioni dai menu di *Simulink* che riutilizzando i comandi *Matlab* attraverso la *Matlab Command Windows*. I risultati della simulazione sono disponibili durante la fase di simulazione stessa e l'esito finale disponibile nello spazio di lavoro di *Matlab*.

3.1 Istruzioni base di Simulink

Per aprire *Simulink* si deve digitare all'interno della *Matlab Command Window* il comando

```
>>simulink
```

che provoca la visualizzazione della finestra (Library: Simulink) contenente le icone delle librerie standard di *Simulink* (vedi Figura 3.1) ed una seconda finestra in cui costruire il modello del sistema da simulare.
I blocchi possono essere copiati dalla prima finestra alla seconda trascinandoli col mouse nella posizione desiderata. Tali blocchi possono essere connessi da linee disegnate sempre col mouse: tenendo premuto il tasto sinistro, partendo dall'uscita di un blocco, col puntatore si crea una nuova connessione all'ingresso ad un altro blocco, mentre premendo il tasto destro posizionati su una connessione preesistente, si genera una diramazione per collegare un altro blocco.

Lo schema viene salvato utilizzando le istruzioni *Save* e *Save as* della tendina *File*. L'istruzione *New* apre un nuovo file *Simulink*, mentre *Open* carica un file *Simulink* salvato precedentemente.

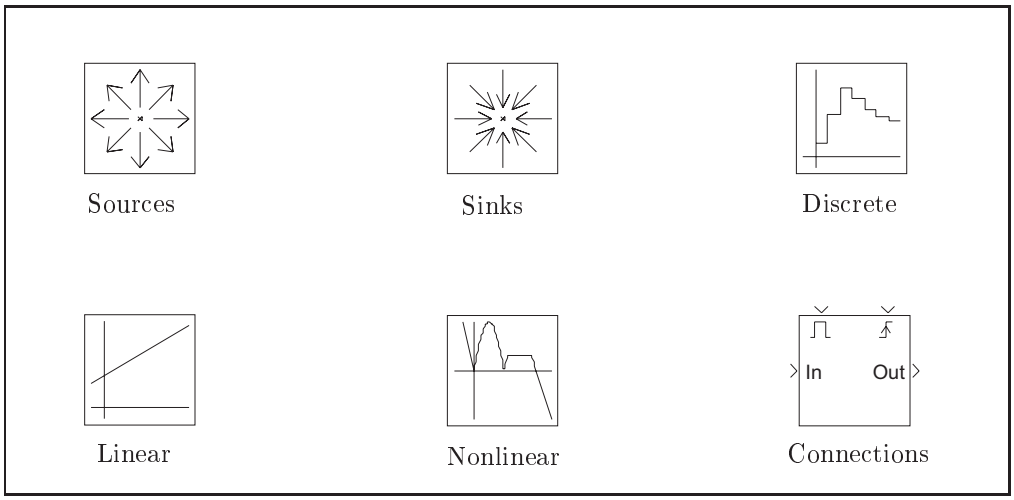


Figura 3.1: *Simulink* block library.

Ciascuna icona della Figura 3.1 contiene i blocchi relativi alla libreria a cui si riferisce. In seguito verranno descritti brevemente i blocchi contenuti in ciascuna libreria

1. **Sources**: (Library: simulink/Sources) contiene alcuni generatori di segnale e vengono visualizzati nella Figura 3.2

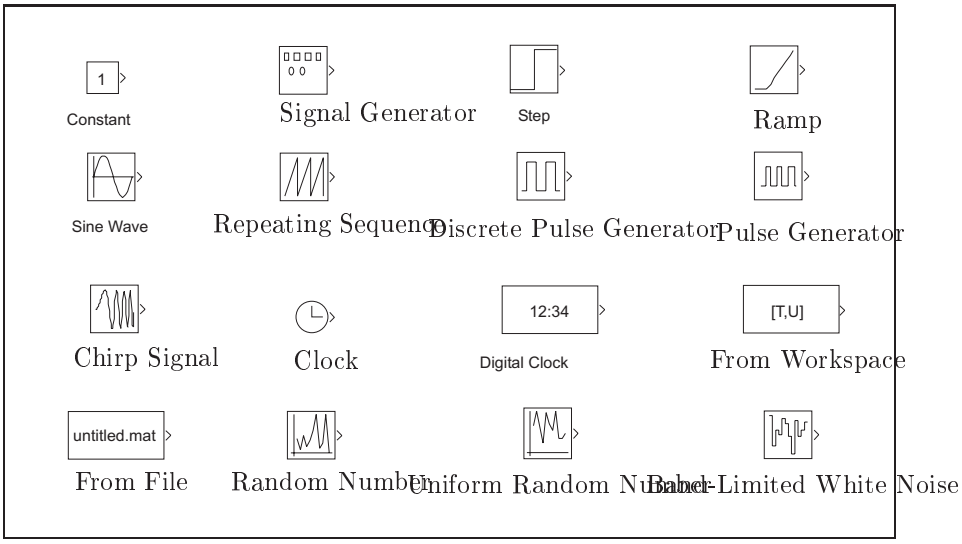


Figura 3.2: *Simulink* Signal Source library.

- **Constant**: genera un valore costante programmabile.
- **Signal Generator**: generatore di segnali sinusoidali, onde quadre,

denti di sega e segnali casuali. Si possono impostare ampiezza e frequenza.

- **Step**: genera un gradino di ampiezza prefissata, specificando il valore iniziale e quello finale.
- **Sine Wave**: genera un'onda sinusoidale di ampiezza, frequenza e fase determinate.
- **Repeating Sequence**: ripete una sequenza di valori e ad istanti predeterminati.
- **Discrete Pulse Generator**: genera impulsi ad intervalli regolari, specificando l'ampiezza, il periodo e ritardo di fase come interi multipli del tempo di campionamento.
- **Pulse Generator**: genera impulsi, specificando il periodo in secondi, il duty cicle (percentuale del periodo), l'ampiezza e l'istante di partenza.
- **Chirp Signal**: genera un segnale sinusoidale con frequenza crescente. Si devono specificare la frequenza iniziale e dopo quanti secondi deve essere raggiunta una certa frequenza predeterminata.
- **Clock**: generatore della base dei tempi.
- **Digital Clock**: genera il tempo di simulazione secondo il tempo di campionamento impostato. Durante il periodo di campionamento vengono mantenuti i valori della simulazione fino al successivo istante di campionamento.
- **From File**: legge il contenuto di una matrice specificata dal `<file>.mat`. La prima riga della matrice deve contenere i valori degli istanti di campionamento e in quelle successive sono memorizzati i corrispondenti valori delle uscite.
- **From Workspace**: legge i valori specificati in una matrice presente nel *WorkSpace* di *Matlab*. La matrice deve contenere nella prima colonna i valori corrispondenti agli istanti di campionamento. Le successive colonne rappresentano i valori delle uscite.
- **Random Number**: genera valori con distribuzione normale gaussiana, dati il valore medio, la varianza e un valore iniziale per il seme.
- **Uniform Random Number**: genera numeri aventi distribuzione uniforme tra due valori prefissati. Si deve specificare anche il seme.
- **Band-Limited White Noise**: genera rumore bianco per sistemi continui. Si specifica la potenza del rumore, istante di campionamento e il seme.

2. **Sinks**: (Library: simulink/Sinks) contiene alcuni rivelatori di segnale, come si può vedere nella Figura 3.3

   - **Scope**: visualizza in funzione del tempo il segnale di ingresso applicato.
   - **XY Graph**: visualizza un grafico (x, y) utilizzando la finestra grafica di *Matlab*. Il primo ingresso corrisponde all'ascissa del grafico e generalmente coincide con la base dei tempi. Si possono introdurre i valori del *range* del grafico.

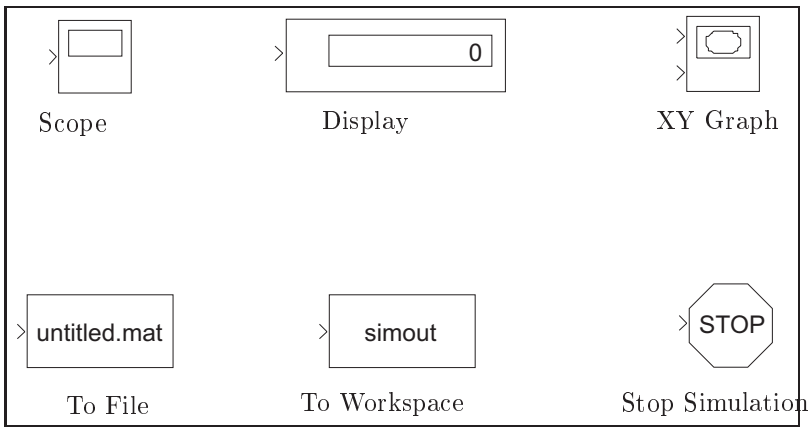


Figura 3.3: *Simulink* Signal Sinks library.

- **Display**: display numerico dei valori dell'ingresso. Si specifica il formato del parametro da visualizzare.
- **To File**: salva gli ingressi applicati all'interno di una matrice in un file `<untitled>.mat`. Si specifica il nome del file e il nome della variabile. I valori vengono salvati per righe. La prima riga della matrice contiene la base dei tempi.
- **To Workspace**: vengono scritti gli ingressi applicato nel *WorkSpace* di *Matlab*. La matrice ha una colonna per ciascun ingresso ed una riga per ogni istante della simulazione. Il dato si perde se la simulazione viene interrotta o messa in pausa. Si specifica il nome della variabile di ingresso e il massimo numero di righe.
- **Stop**: arresta la simulazione quando l'ingresso applicato è diverso da zero.

3. **Discrete**: (Library: simulink/Discrete) sono contenuti i blocchi necessari all'analisi dei sistemi lineari tempo-discreti e vengono raccolti nella Figura 3.4

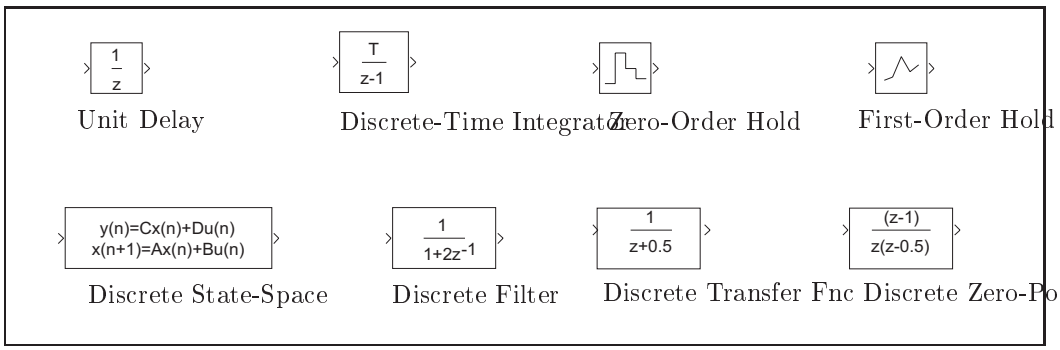


Figura 3.4: *Simulink* Discrete-Time library.

- **Unit Delay**: campiona e mantiene il valore all'ingresso per un periodo di campionamento. Ha come parametri le condizioni iniziali e il tempo di campionamento.
- **Discrete-Time Integrator**: integrazione a tempo-discreto del segnale di ingresso. Utilizza diversi metodi di integrazione, fornite le condizioni iniziali.
- **Zero-Order Hold**: dispositivo di tenuta di ordine zero. Mantiene costante in uscita il valore all'ingresso nell'intervallo di campionamento. Deve essere fornito il tempo di campionamento.
- **First Order Hold**: dispositivo di tenuta di ordine uno. L'uscita cresce linearmente rispetto il valore dell'ingresso nell'intervallo di campionamento. Deve essere fornito il tempo di campionamento.
- **Discrete State-Space**: modello discreto nello spazio degli stati. Vengono fornite le matrici del modello `(A,B,C,D)`, le condizioni iniziali e il tempo di campionamento.
- **Discrete Zero-Pole**: rappresentazione un modello FIR (Finite Impulse Response) o IIR (Infinite Impulse Response) secondo guadagno, poli e zeri, forniti come matrici e vettori. Il numero delle uscite coincide con il numero di colonne della matrice degli zeri. L'uscita è uno scalare se gli zeri sono in un vettore.
- **Discrete Filter**: implementa un filtro tempo-discreto FIR o IIR. Il numeratore e il denominatore sono vettori, i cui elementi sono i coefficienti del polinomio secondo potenze crescenti di z^{-1}.
- **Discrete Transfer Fnc**: implementa una funzione di trasferimento discreta. Il numeratore è una matrice, mentre il denominatore un vettore. Il numero delle uscite coincide con il numero delle righe del numeratore, i cui elementi sono i coefficienti del polinomio secondo potenze decrescenti di z. Anche il vettore a denominatore contiene i coefficienti del relativo polinomio secondo potenze decrescenti di z.

4. **Linear**: (Library: simulink/Linear) contiene i blocchi necessari all'analisi dei sistemi lineari tempo-continui evidenziati nella Figura 3.5

   - **Gain**: guadagno scalare o vettoriale. Si imposta il guadagno k e il blocco calcola l'uscita y dato l'ingresso u secondo l'espressione $y = k. * u$.
   - **Sum**: effettua la somma o la differenza degli ingressi. Si deve inserire la lista dei segni con cui ogni ingresso entra nel blocco.
   - **Integrator**: calcola l'integrazione tempo continua del segnale di ingresso, stabilite le condizioni iniziali ed eventuali limiti superiore ed inferiore di saturazione.
   - **Transfer Fnc**: espressione per la funzione di trasferimento, in cui il numeratore viene rappresentato da una matrice e il denominatore da un vettore. Il numero delle uscite eguaglia il numero dellle righe della matrice al numeratore, i cui elementi sono i coefficienti del polinomio secondo potenze decrescenti di s. Anche il vettore al denominatore rappresenta i coefficienti del polinomio secondo potenze decrescenti di s.

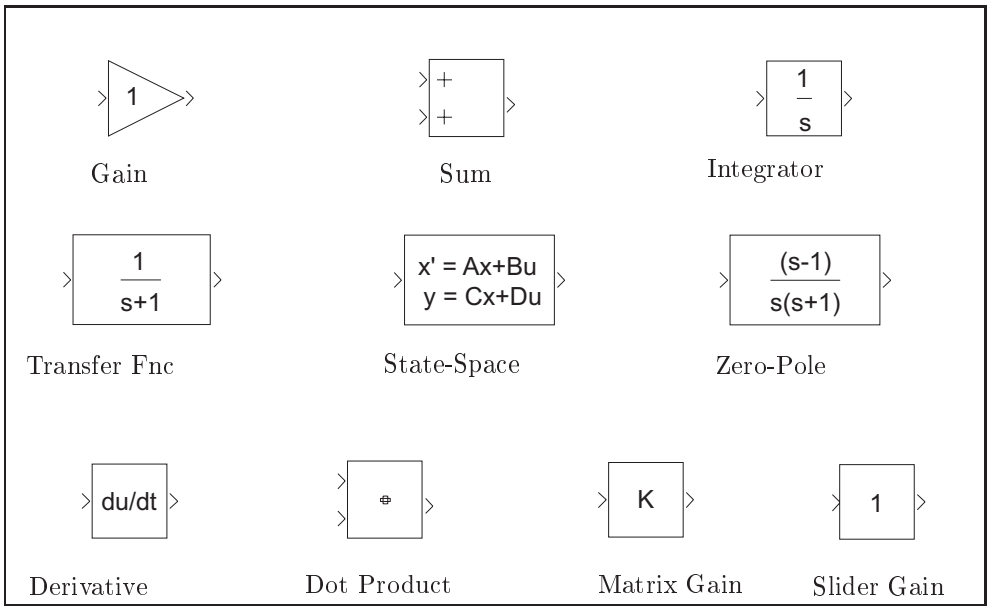


Figura 3.5: *Simulink* Linear library.

- **State-Space**: modello nello spazio degli stati. Occorre inserire le matrici del modello `(A,B,C,D)` e le relative condizioni iniziali.
- **Zero-Pole**: funzione Guadagno, Zeri e Poli. Gli zeri vengono rappresentati da una matrice, mentre i poli da un vettore. Il numero delle uscite coincide con il numero delle colonne della matrice degli zeri.
- **Derivative**: effettua la derivata numerica dell'ingresso.
- **Dot Product**: effettua il prodotto (prodotto scalare) elemento per elemento degli ingressi u_1 e u_2 secondo l'espressione `y = sum(`u_1 `.*` u_2`)`.
- **Matrix Gain**: restituisce in uscita l'ingresso moltiplicato per una matrice predefinita.
- **Slider Gain**: guadagno regolabile tra un valore superiore ed uno inferiore.

5. **Nonlinear**: (Library: simulink/Nonlinear) contiene i blocchi che svolgono funzioni non lineari e sono riportati nella Figura 3.6

   - **Abs**: dato l'ingresso u, calcola l'uscita $y = |u|$.
   - **Trigonometric Function**: implementa diverse funzioni trigonometriche ed iperboliche: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `sinh`, `cosh` e `tanh`.
   - **Math Function**: implementa funzioni matematiche come quelle logaritmiche, esponenziali, potenze e modulo: `exp`, `log`, 10^u, `log10`, `square`, `sqrt`, `pow`, `reciprocal`, `hypot`, `rem` e `mod`.

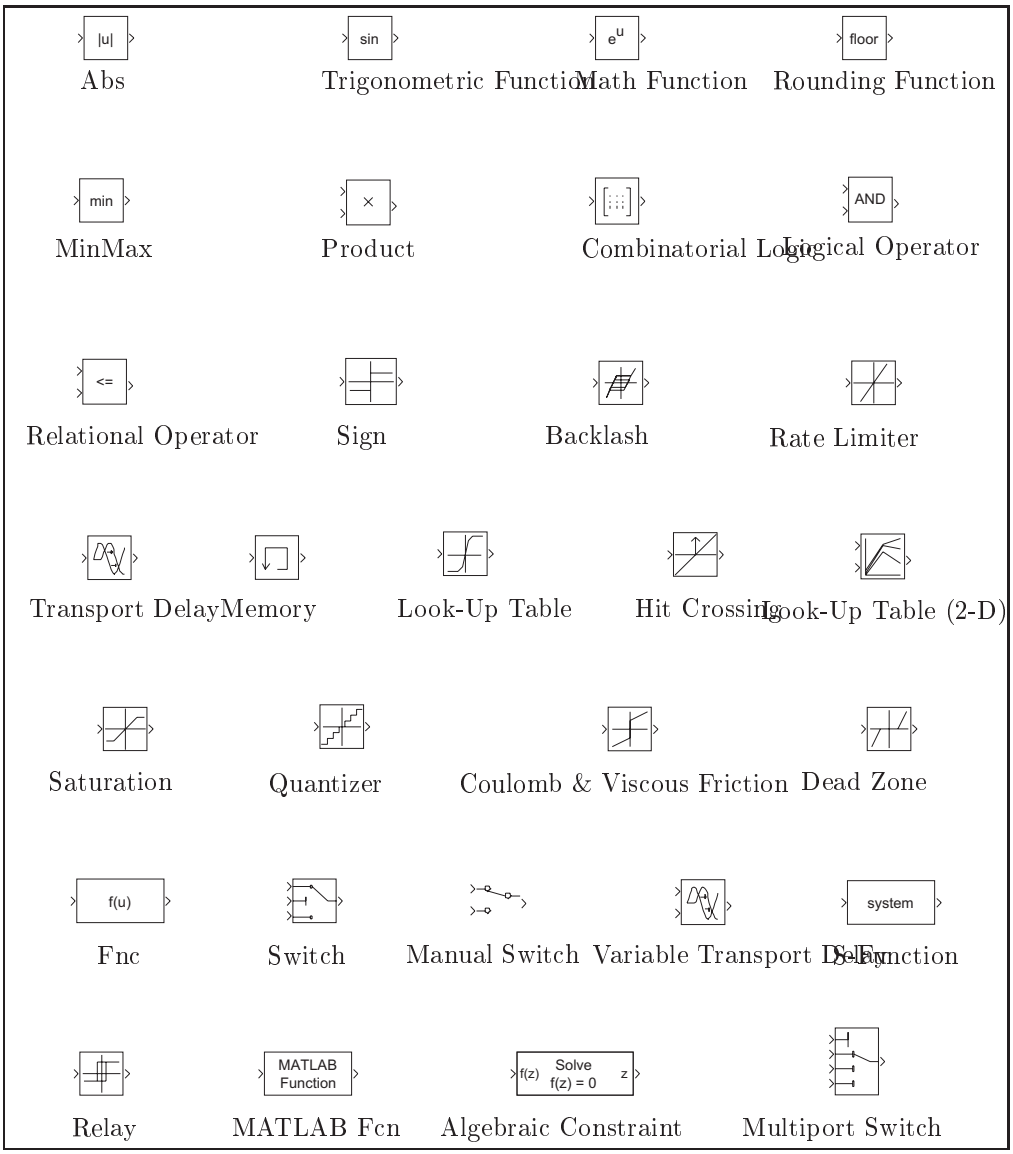


Figura 3.6: *Simulink* Nonlinear library.

- **Rounding Function**: contiene le operazioni di arrotondamento: `floor`, `ceil`, `round` e `fix`.
- **MinMax**: restituisce il minimo od il massimo dell'ingresso. Prevede la scelta del numero degli ingressi e quale operazione deve essere svolta su ogni ingresso.
- **Product**: Moltiplica o divide gli ingressi. Occorre specificare il numero degli ingressi.
- **Combinatorial Logic**: ricerca gli elementi specificati nel vettore

d'ingresso (trattati come valori booleani) nella tabella della verità impostata e restituisce le righe della tabella della verità stessa.

- **Logical Operator**: effettua una operazione logica per un prefissato numero di ingressi: AND, OR, NAND, NOR, XOR, NOT. Per un singolo ingresso, l'operazione viene effettuata tra tutti i valori dell'ingresso memorizzati in un vettore. Per ingressi multipli, l'operazione logica viene eseguita sugli elementi dei diversi vettori di ingresso che occupano la stessa posizione.
- **Relational Operator**: effettua confronti tra gli ingressi: `==`, `=`, `>`, `>=`, `<` e `<=`.
- **Sign**: signum. Restituisce il valore 1 se l'ingresso è positivo, -1, per ingresso negativo e 0 per ingresso nullo.
- **Rate limiter**: limita lo slew-rate (velocità di variazione) del segnale di ingresso. Si imposta lo slew-rate positivo e negativo.
- **Saturation**: limita superiormente ed inferiormente il segnale di ingresso secondo due limiti prefissati.
- **Quantizer**: quantizza l'ingresso all'interno di un intervallo prefissato.
- **Coulomb & Viscous Friction**: funzione di attrito viscoso e forza di Coulomb. La forza coulombiana è modellata da una discontinuità nello zero (`y=sign(x)`) mentre l'attrito viscoso è rappresentato da una relazione lineare (`Gain*abs(x)+Offset`), Complessivamente l'uscita risulta `y=sign(x)*(Gain*abs(x)+Offset)`. `Gain` e `Offset` sono parametri del blocco.
- **Backlash**: simula una zona d'isteresi o un certo "gioco" di ampiezza prefissata. Ad esempio, due ruote dentate i cui denti sono abbastanza spaziati.
- **Dead Zone**: l'uscita rimane a zero per valori interni alla "deadzone". Si specifica l'inizio e la fine dell'intervallo.
- **Look-Up Table**: effettua una interpolazione monodimensionale dei valori dell'ingresso usando quelli nella tabella specificata. I valori esterni a quelli della tabella vengono estrapolati.
- **Look-Up Table (2D)**: effettua una interpolazione bidimensionale dei valori dell'ingresso usando quelli nella tabella specificata. I valori esterni a quelli della tabella vengono estrapolati.
- **Memory**: rappresenta un ritardo di durata unitaria. L'uscita coincide con il valore assunto precedentemente dall'ingresso. Occorre specificare le condizioni iniziali.
- **Transport Delay**: ritarda di una quantita specificata il segnale di ingresso. Il ritardo deve essere più grande del passo utilizzato nella simulazione.
- **Variable Transport Delay**: ritarda il primo segnale di ingresso di una quantita specificata dal secondo ingresso. Il ritardo deve essere più grande del passo utilizzato nella simulazione.

- **Hit Crossing**: segnala quando il segnale di ingresso attraversa lo zero secondo un certo margine prefissato. Si può specificare la direzione di attraversamento dello zero.
- **Fnc**: permette di specificare una funzione f arbitraria dell'ingresso u, `y = f(u)`.
- **MATLAB function**: passa i valori dell'ingresso ad una funzione *Matlab* affinché possa essere valutata. La funzione `Matlab` deve restituire un vettore la cui lunghezza deve essere definita.
- **S-Function**: blocco che può essere progettato dall'utente in *Matlab*, C, Fortran o usando le funzioni di *Simulink* standard. I parametri `t`, `x`, `u` e `flag` sono passati automaticamente alla funzione di *Simulink*. Possono essere specificati anche altri parametri.
- **Switch**: l'uscita coincide con il primo ingresso quando il secondo ingresso è maggiore od uguale ad una certa soglia, altrimenti assume i valori del terzo ingresso.
- **Manual Switch**: commutatore regolabile col mouse senza parametri.
- **Multiport Switch**: coincide con gli ingressi secondo i valori arrotondati assunti dal primo di questi.
- **Relay**: l'uscita assume due valori impostati se l'ingresso è maggiore dell'estremo superiore o minore dell'estremo inferiore di un certo intervallo specificato attraverso due parametri. Lo stato del Relay non dipende dall'ingresso quando questo assume un valore interno dell'intervallo.
- **Algebraic Contraint**: vincola il segnale d'ingresso `f(z)` a zero e restituisce il corrispondente valore algebrico `z`. Quindi il blocco fornisce il valore `z` tale per cui `f(z) = 0`. L'uscita deve influenzare l'ingresso attraverso una certa retroazione. Occorre fornire un valore di tentativo per `z`.

6. **Connections**: (Library: simulink/Connections) contiene i blocchi necessari ad effettuare connessioni come mostra la Figura 3.7

   - **In**: fornisce una porta d'ingresso per un modello. Occorre specificare il tempo di campionamento.
   - **Out**: fornisce una porta d'uscita per un modello. Quando il modello non è disabilitato, occorre fornire il corrispondente valore dell'uscita.
   - **Mux**: raggruppa scalari o vettori in un vettore di dimensioni maggiori.
   - **Demux**: disaggrega i vettori d'ingresso in scalari o vettori di dimensioni inferiori.
   - **From**: riceve i segnali dal blocco **Goto** secondo l'etichetta (*tag*) specificata.
   - **Goto**: invia i segnali al blocco **From** avente l'etichetta specificata. Permette di definire la visibilità dell'etichetta.
   - **Goto Tag Visibility**: viene usato con i blocchi **From** e **Goto** e permette di specificare la visibilità di una etichetta.

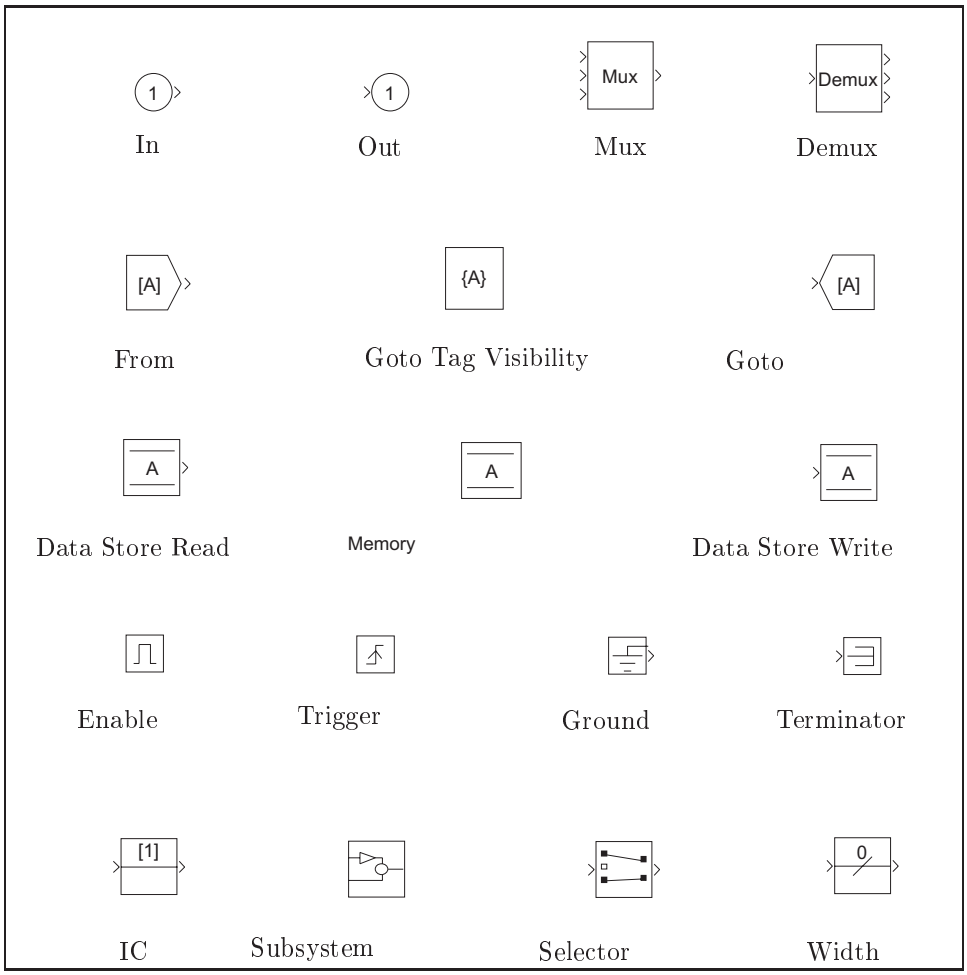


Figura 3.7: *Simulink* Connection library.

- **Data Store Read**: legge i dati memorizzati in una certa regione definita dal blocco **Data Store Memory** secondo un nome prefissato. Occorre definire il nome della zona di memoria e il tempo di campionamento.
- **Data Store Memory**: permette di definire nome e valore iniziale di una regione di memoria utilizzata dai blocchi **Data Store Read** e **Data Store Write**.
- **Data Store Write**: scrive la zona di memoria specificata dal nome. Viene definito anche il tempo di campionamento.
- **Enable**: il blocco viene posto all'interno di un modello affinché sia abilitato.
- **Trigger**: il blocco fornisce una porta di trigger predefinito
- **Ground**: viene utilizzato per mettere a zero i segnali di ingresso.

Si evitano i problemi dovuti agli ingressi non collegati. Fornisce una uscita nulla.

- **Terminator**: usato per isolare un segnale di uscita e per prevenire così i problemi provocati dalle uscite non connesse.
- **IC**: permette di specificare le condizioni iniziali per un segnale.
- **Subsystem**: fornisce una finestra in cui costruire un modello di subsystem.
- **Selector**: seleziona e riordina gli elementi specificati del vettore d'ingresso.
- **Width**: fornisce in uscita l'ampiezza del segnale d'ingresso.

È possibile assegnare ad ogni variabile o intero blocco un nome che verrà evidenziato sia nello schema a blocchi, sia nei grafici che riportano gli andamenti delle variabili. Fino ad ora si sono mantenuti i nomi di default per i blocchi predefiniti da *Simulink*.

Una volta costruito uno schema a blocchi, utilizzando l'apposita finestra fornita da *Simulink* e i blocchi necessari, si passa alla fase di simulazione, ovvero all'integrazione delle equazioni differenziali che descrivono il sistema costruito.

Per utilizzare il simulatore offerto da *Simulink* occorre utilizzare l'apposita finestra richiamabile dal *Simulink Control Panel* alla quale si accede dal menu principale selezionando in sequenza le opzioni *Simulation* e successivamente *Parameters*.

Le informazioni principali del menu *Solver* all'interno della finestra *Simulation Parameters* sono analoghe a quelle relative ai parametri definibili nelle funzioni *Matlab* utilizzate per integrare sistemi di equazioni differenziali ordinarie, introdotte nel Capitolo 2 e in particolare nel Paragrafo 2.3:

1. **Simulation time: Start time**: istante iniziale della simulazione.
2. **Simulation time: Stop time**: istante finale della simulazione.
3. **Solver Options: Type**: permette di definire se si vuole utilizzare un passo di integrazione fisso (Fixed-step) o variabile (Variable-step).
4. **Solver Options**: permette di scegliere la funzione di integrazione ottimale. Sono disponibili `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s` e un metodo per sistemi discreti (discrete).
5. **Solver Options**: Si possono inoltre definire ampiezza massima e iniziale (Max step size e Initial step size) del passo di integrazione, nonché le tolleranze relative ed assolute (Relative e Absolute tolerance) legate all'accuratezza della soluzione.

3.2 Analisi di un circuito non lineare.

Si riprenda l'esempio del circuito non lineare utilizzato nel Capitolo 2 Paragrafo 2.2. Utilizzando *Simulink*, il modello non lineare è rappresentato in Figura 3.8

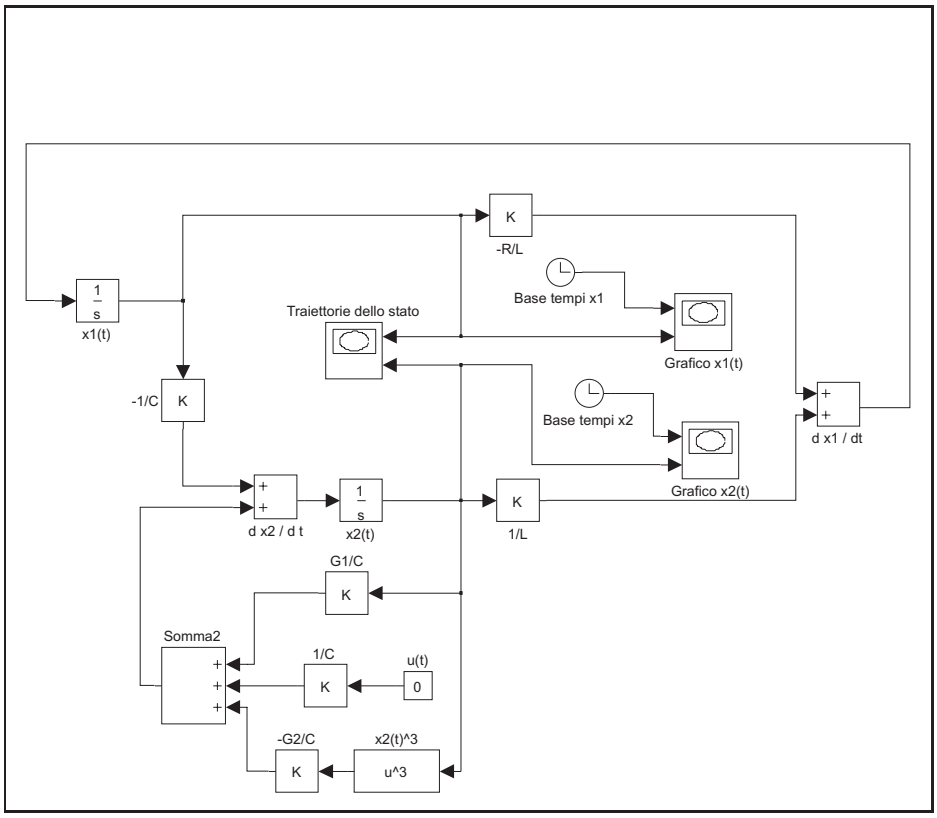


Figura 3.8: Circuito non lineare in *Simulink*.

da cui si ottengono i seguenti risultati delle simulazioni, già presentati nel Capitolo 2, impostando i parametri ai valori $G_1 = 0.8$, $G_2 = 0.05$, $R = 0.5$, $L = 1$ e $C = 1$, con condizioni iniziali $x(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

In particolare, le traiettorie degli stati sono presentate in Figura 3.9
e si confronti con la Figura 2.7.

Analogamente, l'andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo è graficato nelle Figure 3.10
Si confronti l'andamento con quello analogo mostrato nella Figura 2.8 del Capitolo 2.

Si conclude quindi che si può equivalentemente integrare un modello differenziale costruito in *Matlab*, usando le relative funzioni, oppure progettare e simulare lo stesso modello in ambiente *Simulink*.

3.3 Modello di un motore in corrente continua

Si consideri un motore in corrente continua controllato sull'armatura e con l'avvolgimento di eccitazione alimentato a corrente e tensione costante. Sull'asse del motore è presente, oltre al carico inerziale (J), una coppia resistente (f)

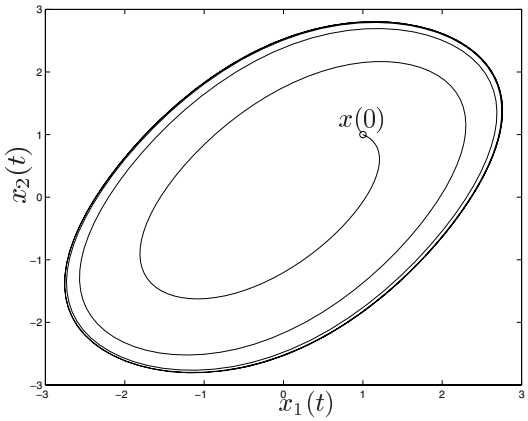


Figura 3.9: Traiettorie dello stato in *Simulink*.

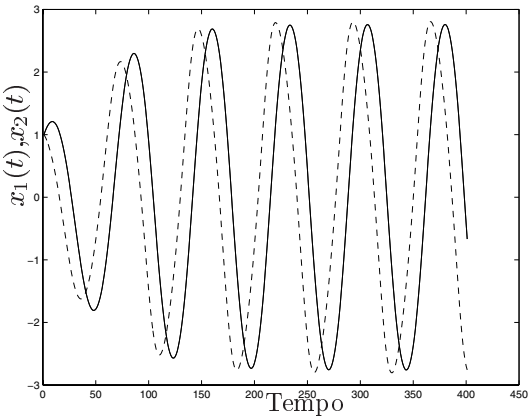


Figura 3.10: Andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo calcolata in *Simulink*.

dovuta all'attrito dei cuscinetti ed alle perdite di ventilazione (proporzionali alla velocità di rotazione) ed una coppia di carico C_c. Lo schema è riportato in Figura 3.11.

I parametri del motore sono $R_a = 3$ Ohm, $L_a = 30$ mH, $k_m = 2$ N m/A, $J = 3\ kg\ m^2$ e $f = 5 \times 10^{-3}$ N m s / rad. R_a è la resistenza di armatura, L_a la relativa induttanza e k_m una costante che lega la forza controelettromotrice sviluppata dal motore alla velocità angolare $\omega(t)$. Se si assumono come ingressi la tensione di alimentazione di armatura $V_a(t)$ e la coppia assorbita dal carico $C_c(t)$ e come uscite la corrente di armatura $i_a(t)$ e la velocità angolare $\omega(t)$, si

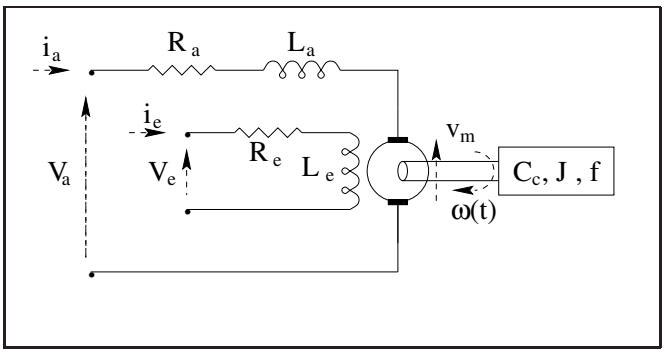


Figura 3.11: Motore in corrente continua.

ottiene il modello nello spazio degli stati (A,B,C)

$$\begin{aligned}
\begin{bmatrix} \dot{i}_a(t) \\ \dot{\omega}(t) \end{bmatrix} &= \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{f}{J} \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix} \\
\omega(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix}
\end{aligned} \tag{3.1}$$

Pertanto, le matrici del sistema risultano

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{f}{J} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

se si pone $y(t) = \omega(t)$ e

$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} \text{ e } u(t) = \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix}.$$

Utilizzando lo schema del motore in corrente continua in ambiente *Simulink* e con i parametri forniti precedentemente, si alimenta il motore come in Figura 3.12 con un gradino di tensione di armatura che si mantiene al valore di 0V da 0 a 50s, per poi portarsi al valore di 5V per altri 50s.
La Figura 3.13 riporta il grafico relativo alla velocità angolare $w(t)$ del motore e quella del segnale di ingresso.
Successivamente, si è alimentato il motore con un impulso di tensione di ampiezza $V_a(t) = 10$V e durata $\tau = 40$s. I grafici della posizione del rotore $\alpha(t)$ in radianti e della relativa velocità angolare sono rappresentati nelle Figure 3.14 e 3.15.
Per ottenere l'ulteriore uscita $\alpha(t)$ a partire dal modello del secondo ordine descritto nell'Equazione 3.1 occorre notare che $\dot{\alpha}(t) = \omega(t)$ e, pertanto, ponendo

$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \\ \alpha(t) \end{bmatrix} \text{ e } y(t) = \begin{bmatrix} \omega(t) \\ \alpha(t) \end{bmatrix}$$

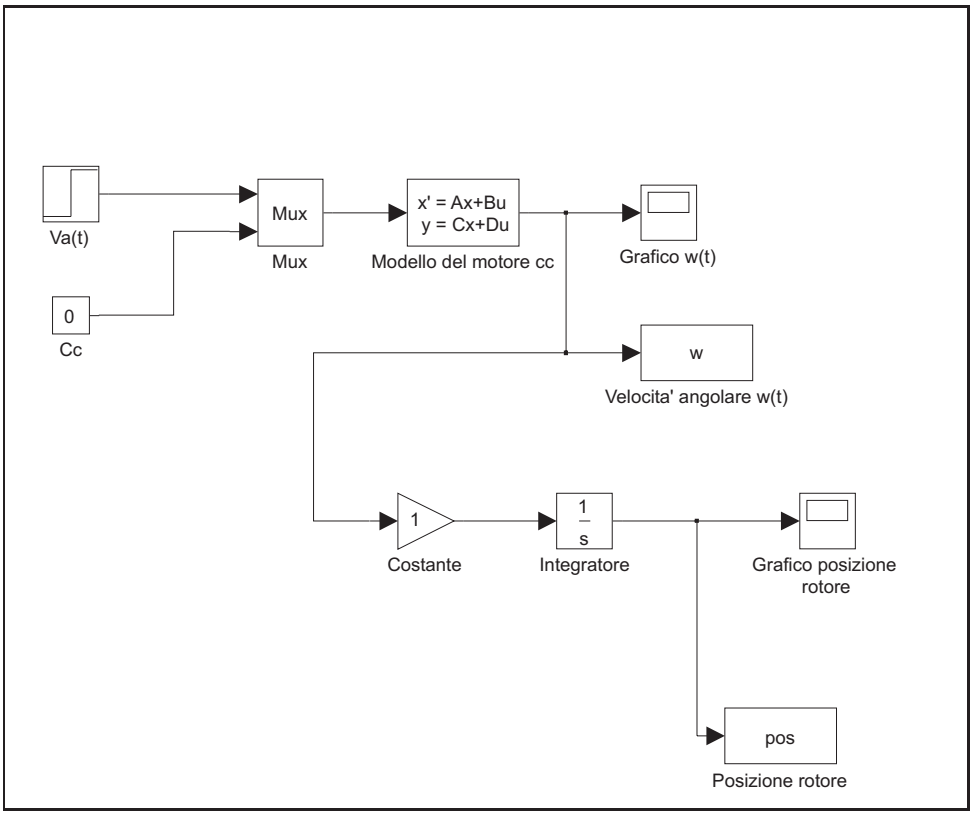
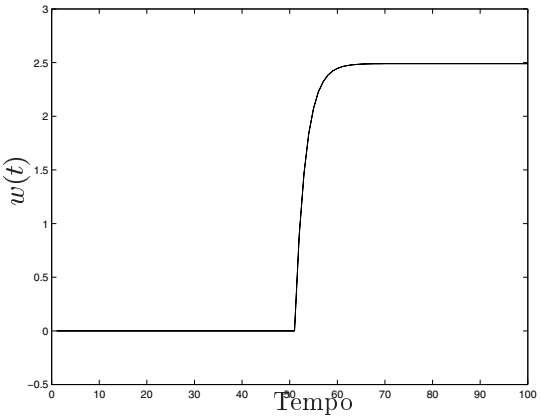


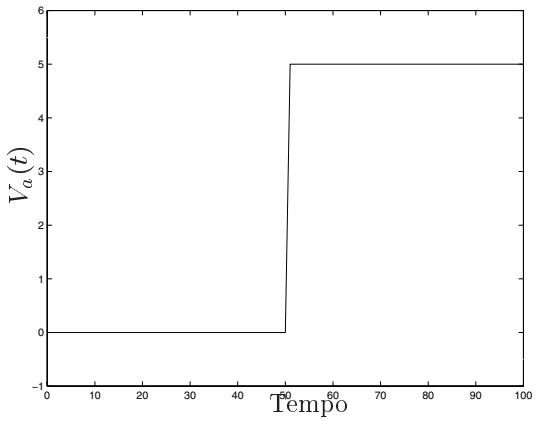
Figura 3.12: Modello *Simulink* del motore in corrente continua.

le matrici del sistema (A,B,C) si modificano in

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} & 0 \\ \frac{k_m}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.2)$$



(a)



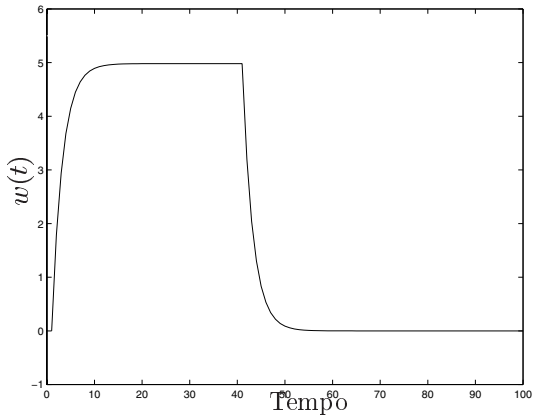
(b)

Figura 3.13: Velocità angolare del motore in cc (a) soggetto ad un gradino di tensione (b).

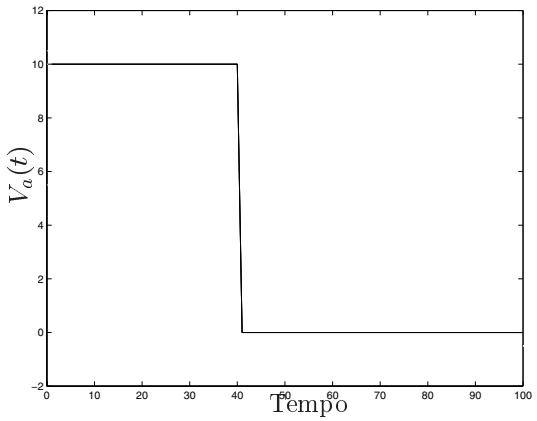
3.4 Esercizi proposti in aula didattica.

1. Si realizzi in ambiente *Simulink* il sistema di equazioni differenziali relative ai modello del motore in corrente continua (3.1) e (3.2). Se ne verifichi successivamente la correttezza confrontandolo con le realizzazioni equivalenti nello spazio degli stati (3.1) e (3.2).

2. Utilizzando gli stessi i valori dei parametri del motore in corrente continua, determinare l'ampiezza del gradino $V_a(t)$ necessaria a raggiungere una velocità angolare *di regime* pari a $\omega(t) = 10\text{rad/s}$, nelle ipotesi di assenza del carico $C_c = 0$ e con il modello del motore del secondo ordine. Si verifichi analiticamente il risultato ottenuto.



(a)

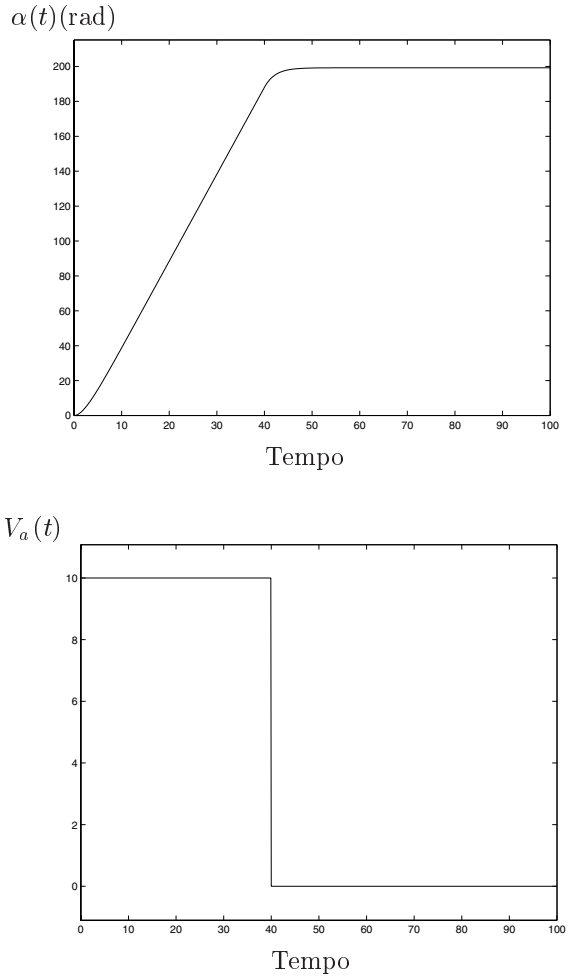


(b)

Figura 3.14: Velocità angolare del motore (a) soggetto ad un gradino di 10V. e durata 5s (b).

3. Fissata l'ampiezza della tensione di armatura a $V_a = 10$V., progettare la durata dell'impulso τ in modo da raggiungere una posizione assegnata $\alpha = 20$rad., sempre nelle ipotesi di assenza di carico.

4. Fissato τ, graficare l'andamento temporale della posizione del rotore per una tensione pari alla metà e al doppio della tensione fissata al punto precedente.



(b)

Figura 3.15: Posizione in radianti del rotore dell motore (a) soggetto ad un impulso di tensione (b).

Bibliografia

[1] K. Sigmon, *MATLAB Primer*. University of Florida, Florida, Second Edition ed., 1992. (Si scarica dalla rete).

[2] The MathWorks, Inc., *Matlab, The Language of Technical Computing. Getting Started with MATLAB.*, version 5.1 ed., May 1997. (In formato pdf su CD Matlab).

[3] The MathWorks Inc., *Matlab User's Guide*, 1993.

[4] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison-Wesley, Third Edition ed., 1998.

[5] L. F. Shampine and M. W. Reichel, "The Matlab Ode Suite," tech. rep., The MathWorks, Inc, 1997. (Disponibile anche come file in formato pdf).

[6] The MathWorks Inc., *Simulink User's Guide*, 1995.