Scuola Universitaria Professionale
della Svizzera Italiana

Dipartimento
di informatica
ed elettronica

Introduzione a SIMULINK

Ing. Roberto Bucher

7 aprile 2003

Indice

1 Introduzione **9**

2 Costruzione di un modello **11**
2.1 Immissione dei blocchi 11
2.2 Vettorizzazione e espansione vettoriale di blocchi 11
2.3 Callback Function 12

3 Simulazione **13**
3.1 Simulazione dalla finestra di SIMULINK 13
3.2 Simulazione dalla linea di comando 14
3.2.1 Scelta dell'algoritmo di integrazione 15
3.2.2 Scelta dei parametri di integrazione 16
3.3 Simulazione di sistemi discreti 17

4 Linearizzazione e equilibrio **19**
4.1 Linearizzazione 19
4.2 Punti di equilibrio 20

5 Mascheramento di blocchi **23**
5.1 Costruzione del blocco 23
5.2 Mascheramento del blocco 24
5.3 Creazione della maschera di immissione dati 24
5.3.1 Pagina di documentazione 24
5.3.2 Editor e inizializzazione 25
5.3.3 Pagina delle icone 25

6 Esecuzione condizionata **29**

7 S-Function **31**

A Comandi principali di Simulink **35**
A.1 SIMULINK Model analysis and construction functions. 35
A.1.1 Simulation. 35
A.1.2 Linearization and trimming. 35
A.1.3 Model Construction. 35
A.1.4 Masking. 36
A.2 Demo di Simulink 36

A.2.1 SIMULINK demonstrations and samples. 36
A.2.2 Demonstration models. 36
A.2.3 Demonstration models for case studies 1,2,3 in the user's guide. . . 36
A.2.4 Demonstration scripts. 36
A.3 Blocks 37
A.3.1 SIMULINK block library. 37
A.3.2 Example S-function models and blocks. 37

B Librerie principali di Simulink **39**
B.1 Librerie di base 39
B.2 Librerie aggiuntive 40

Elenco delle figure

1.1 Biblioteche di SIMULINK 9

2.1 Semplice schema 11

3.1 Risultato della simulazione 14

4.1 Sistema da linearizzare 19

5.1 Sistema driver-motore 24
5.2 Blocco mascherato 24
5.3 Sistema mascherato 24
5.4 Icona del nuovo blocco 26

7.1 Principio di funzionamento di una S-Function 31
7.2 Schema a blocchi del funzionamento di una S-Function 32

B.1 Sources 39
B.2 Sinks 40
B.3 Discrete 41
B.4 Linear 41
B.5 Nonlinear 42
B.6 Connections 43
B.7 LTI Block 43
B.8 Additional Library 44
B.9 Additional Sinks 44
B.10 Additional Discrete 44
B.11 Additional Linear 45
B.12 Conversion 45
B.13 Flip Flop 46
B.14 Linearization 46

Elenco delle tabelle

2.1 Alcuni trucchi per disegnare 12

3.1 Parametri di simulazione 15
3.2 Metodi di integrazione a passo variabile 16
3.3 Metodi di integrazione a passo fisso 16

5.1 Funzioni di trasferimento dei blocchi 23
5.2 Informazioni di help del nuovo blocco 25
5.3 Comandi per il disegno dell'icona 26
5.4 Icon frame 27
5.5 Icon transparency 27
5.6 Icon rotation 27
5.7 Drawing coordinates 28

7.1 Chiamate durante la simulazione 31

Capitolo 1

Introduzione

SIMULINK è l'ambiente di programmazione grafico associato a MATLAB. È particolarmente indicato per costruire schemi a blocchi di sistemi dinamici lineari e non lineari e per eseguire la loro simulazione. Mette a disposizione una grande serie di moduli predefiniti, con la possibilità per l'utente di crearne di nuovi, eventualmente anche programmati in *"C"* o *"Fortran"*.

Per far partire SIMULINK occorre essere nello shell di MATLAB e dare il comando

```
simulink
```

Subito viene mostrata una nuova finestra grafica contenente le biblioteche di simboli di SIMULINK (vedi figura 1.1).

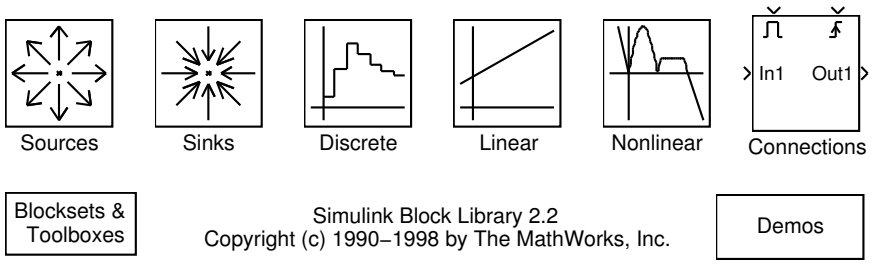


Figura 1.1: Biblioteche di SIMULINK

Tutte le biblioteche possono essere aperte con un doppio click del mouse, per mostrare il loro contenuto. L'allegato B mostra tutti i simboli a disposizione nella varie biblioteche, ben suddivise in categorie. Le prime 6 biblioteche contengono i simboli più importanti e utili, mentre *"Blocksets & Toolbox"* contiene ulteriori librerie di simboli particolari, come pure l'accesso ad eventuali toolbox supplementari.

Capitolo 2

Costruzione di un modello

2.1 Immissione dei blocchi

Come esempio costruiamo un modello di sistema con una funzione di trasferimento di 1. ordine, con entrata di tipo gradino unitario. Prendiamo dalla biblioteca *"Source"* il modulo *"Step Input"*, da *"Linear"* il modulo *"Transfer Fcn"* e da *"Sinks"* il modulo *"Scope"*. Con un doppio *click* del mouse su un blocco possiamo settarne i parametri, direttamente per mezzo di finestre di dialogo.

Diamo per lo step un tempo di partenza a 0 e un valore finale a 1, per la funzione di trasferimento diamo come polinomio al numeratore *1* ([1]) e come polinomio al denominatore *s+1* ([1 1]). In seguito uniamo i blocchi tra di loro utilizzando il mouse con il bottone sinistro premuto. Possiamo vedere come il cursore del mouse si trasformi in una doppia croce avvicinandosi all'entrata del blocco successivo, per poi diventare la punta di una freccia al momento che il collegamento è effettivo.

Si ottiene cosìla figura 2.1:

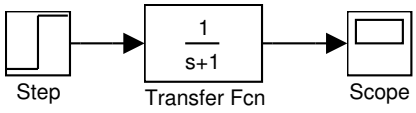


Figura 2.1: Semplice schema

La tabella 2.1 riporta alcuni trucchi che permettono di disegnare più velocemente.

2.2 Vettorizzazione e espansione vettoriale di blocchi

Alcuni blocchi applicano la loro funzionalità ad entrate singole, ma anche ad entrate vettoriali. Il modulo *"Gain"* della libreria *"Linear"*, ad esempio, applica l'amplificazione impostata anche a tutti gli elementi di un vettore, se alla sua entrata viene applicato un segnale con più componenti; altri blocchi, come ad esempio il blocco *"Zero-Order-Hold"* della libreria *"Discrete"*, hanno il medesimo comportamento. Nel manuale di SIMULINK, per ogni blocco, è riportato se esiste o meno questa caratteristica.

copiare blocchi	Trascinare un blocco con il bottone destro del mouse ne crea una copia; questa operazione è equivalente a trascinare un blocco tenendo premuto il tasto *ctrl*
inserire blocchi su un collegamento preesistente	Trascinare e posizionare un blocco su un collegamento (solo dalla versione 2.2)
disconnettere un blocco	Trascinare un blocco tenendo premuto il tasto *shift*
ridimensionare un blocco	Selezionare e trascinare il bordo del blocco
fare una diramazione da un collegamento	Tracciare la linea utilizzando il bottone destro del mouse e partendo dal punto di diramazione o arrivando a questo punto
dividere un collegamento	Per introdurre un angolo in un collegamento, posizionarsi con il mouse dove si vuole deviare la linea, premere il tasto *shift* e trascinare l'angolo cosìottenuto
introdurre annotazioni	Fare Un doppio click con il mouse dove si vuole introdurre un testo. Appena appare il cursore si può introdurre un testo. La formattazione del testo è possibile con il comando *"Format-Font"*

Tabella 2.1: Alcuni trucchi per disegnare

2.3 Callback Function

Si possono definire espressioni di MATLAB da eseguire nel caso che un blocco o un modello di SIMULINK venga attivato in un certo modo. Se ad esempio nel modello *"Mydemo"* ci fosse un blocco chiamato *"Myblock"*, si può definire una funzione da chiamare dopo un certo evento mediante il comando "set_param".

Il comando

```
set_param('mydemo/myblock','OpenFcn',myfunc)
```

associa al blocco *"myblock"* la funzione di callback *"myfunc"*, da eseguire quando questo viene aperto con un doppio click. Il manuale di SIMULINK riporta tutta la lista di attività di blocchi o modelli che possono essere associate a funzioni esterne.

Capitolo 3

Simulazione

Una volta costruito il modello del nostro processo, applicato un segnale in entrata e un dispositivo d'uscita, è possibile vedere come si comporta il nostro sistema. Possiamo far partire la simulazione dalla finestra di SIMULINK, oppure direttamente dallo shell di MATLAB.

3.1 Simulazione dalla finestra di Simulink

Chiamando il menu *"Simulation-Parameters..."* possiamo settare i parametri di simulazione o accettare quelli proposti nella finestra di dialogo.

"Workspace I/O" permette di ritornare o prendere delle variabili dal workspace di MATLAB. *"Relative Tolerance"*, *"Absolute tolerance"*, *"Initial Step Size"* e *"Max Step Size"* controllano gli errori locali relativi e il passo di integrazione minimo e massimo per la simulazione. In questa finestra è anche possibile cambiare l'algoritmo di integrazione.

Impostiamo il valore di *"Stop Time"* a 10, per simulare fino a 10 secondi, e quindi chiudiamo la finestra.

Utilizzando il comando *"Simulation-Start"* possiamo far partire la simulazione. Per vedere il risultato si deve aprire, con un doppio click del mouse, la finestra dello *"Scope"*. Durante una simulazione da menu è possibile:

- Cambiare i parametri di un blocco, a condizione di non aumentare il numero degli stati, delle entrate o delle uscite.
- Modificare i parametri di simulazione, tranne le variabili d'uscita e il tempo di partenza.
- Cambiare l'algoritmo di integrazione.
- Cambiare il tempo di sampling di blocchi discreti.
- Simulare un altro blocco nel medesimo tempo.
- Selezionare una linea nel modello, che viene automaticamente visualizzata su uno *"Scope"* non legato a niente, chiamato *"Floating Scope"* (da settare tramite i parametri dello *"Scope"*).

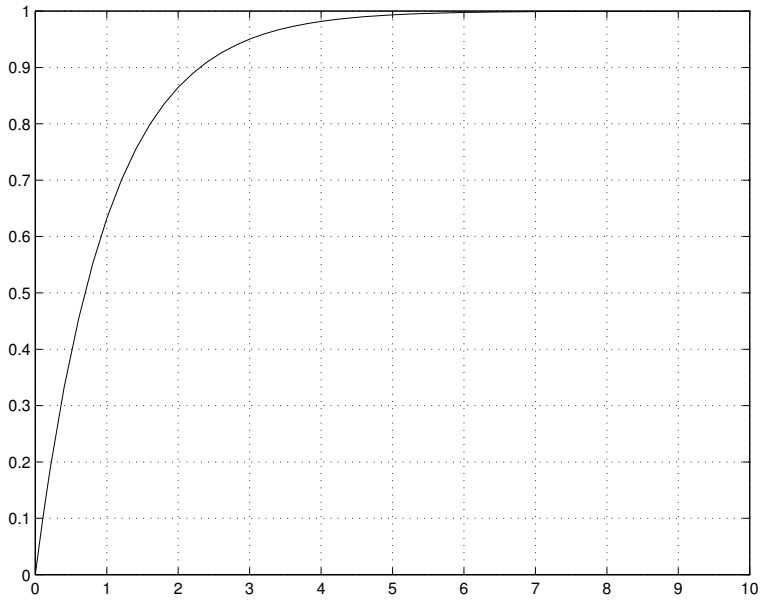


Figura 3.1: Risultato della simulazione

Il risultato della simulazione è visualizzato nella figura 3.1.

Tramite il bottone con il cannocchiale è possibile fare un *"autoscaling"* del grafico, mentre con il bottone accanto ad esso si possono bloccare le scale per ulteriori simulazioni. I 3 bottoni inziali permettono di effettuare degli zoom su parti del grafico.

3.2 Simulazione dalla linea di comando

Salvando il modello di SIMULINK con un nome, è possibile simulare il processo direttamente dando un comando nello shell di MATLAB. Il comando per far partire la simulazione è del tipo

```
[t,x,y]=sim(modello,intervallo-tempo, opzioni, input_esterno, p1,...,pn)) ;
```

I parametri sono descritti nella tabella 3.1.

Il comando *"simset"* permette di impostare tutti i parametri di simulazione:

```
      Solver: [ 'VariableStepDiscrete' |
                'ode45' | 'ode23' | 'ode113' |
                'ode15s' | 'ode23s' |
                'FixedStepDiscrete' |
                'ode5' | 'ode4' | 'ode3' | 'ode2' |
                'ode1' ]
      RelTol: [ positive scalar {1e-3} ]
      AbsTol: [ positive scalar {1e-6} ]
      Refine: [ positive integer {1} ]
    MaxStep: [ positive scalar {auto} ]
InitialStep: [ positive scalar {auto} ]
```

t	vettore tempo ritornato
x	matrice degli stati (prima quelli continui, poi quelli discreti)
y	matrice delle uscite
modello	nome del modello di Simulink
intervallo-tempo	tempo iniziale e finale della simulazione ([t0,tf]) oppure tempo iniziale, tempi da calcolare e tempo finale ([t0, toutput, tf]) se non viene dato vengono utilizzati quelli impostati con "Simulation-parameters"
opzioni	opzioni di simulazione, create con il comando *"simset"*
ut	input esterno opzionale; può essere usata una stringa contenente una funzione (p.es. *sin5t*)
p1,. . .,pn	parametri opzionali per le S-function (vedi più avanti)

Tabella 3.1: Parametri di simulazione

```
      MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
     FixedStep: [ positive scalar ]
  OutputPoints: [ {'specified'} | 'all' ]
OutputVariables: [ {'txy'} | 'tx' | 'ty' | 'xy' | 't' |
                   'x' | 'y' ]
       MaxRows: [ non-negative integer {0} ]
    Decimation: [ positive integer {1} ]
  InitialState: [ vector {[]} ]
 FinalStateName: [ string {''} ]
         Trace: [ comma separated list of 'minstep',
                  'siminfo', 'compile' {''}]
  SrcWorkspace: [ 'base' | {'current'} | 'parent' ]
  DstWorkspace: [ 'base' | {'current'} | 'parent' ]
     ZeroCross: [ {'on'} | 'off' ]
```

Si tratta di parametri che possono anche essere scelti dalla finestra delle opzioni di simulazione in SIMULINK. Per quel che riguarda il significato di ogni singolo parametro si consiglia di consultare il manuale. Importante in ogni caso è la scelta dell'algoritmo di integrazione (*"Solver"*).

3.2.1 Scelta dell'algoritmo di integrazione

La simulazione di un modello di SIMULINK presuppone l'integrazione di un set di equazioni differenziali. La scelta del metodo e dei parametri di integrazione risulta quindi decisiva per un buon risultato finale.

3.2.1.1 Metodo di integrazione predefinito

In caso di modelli che contengono blocchi continui, se non viene specificato altrimenti, SIMULINK utilizza il metodo *"ode45"*, nel caso di sistemi solo discreti viene utilizzato il metodo *"discrete"* a passo variabile.

3.2.1.2 Metodi a passo variabile

I metodi a passo variabile sono descritti nella tabella 3.2.

ode45	metodo di Runge-Kutta(4,5)
ode23	metodo di Runge-Kutta(2,3)
ode113	metodo di Adams-Bashforth-Moulton PECE.
ode15s	metodo che utilizza formule di differenziazione, da utilizzare in presenza di sistemi rigidi (stiff problems)
ode23s	metodo modificato di Rosenbrock per sistemi rigidi
discrete	metodo a passo variabile per sistemi puramente discreti

Tabella 3.2: Metodi di integrazione a passo variabile

3.2.1.3 Metodi a passo fisso

I metodi di integrazione a passo fisso sono descritti nella tabella 3.3.

ode5	versione a passo fisso del metodo ode45
ode4	metodo di Runge-Kutta di 4. ordine
ode3	versione a passo fisso di ode23
ode2	metodo di Heun
ode1	metodo di Eulero
discrete	versione a passo fisso per sistemi puramente discreti

Tabella 3.3: Metodi di integrazione a passo fisso

3.2.2 Scelta dei parametri di integrazione

3.2.2.1 Tolleranza relativa

Rappresenta l'errore percentuale di ogni stato.

3.2.2.2 Tolleranza assoluta

Rappresenta l'errore assoluto del valore di uno stato quando questo si avvicina allo zero.

3.2.2.3 Passo di integrazione massimo

Controlla il valore massimo del passo di integrazione del metodo di integrazione scelto. Il valore di default è impostato dalla formula

$$h_{max} = \frac{t_{stop} - t_{start}}{50}$$

3.2.2.4 Passo di integrazione iniziale

Viene impostato dal sistema analizzando le derivate al primo passo di integrazione. Il valore impostato è solo un valore consigliato che viene ridotto in caso di necessità dal sistema se l'errore non è soddisfacente.

3.3 Simulazione di sistemi discreti

I blocchi discreti hanno un parametro supplementare che è il tempo di campionamento. L'uscita di questi blocchi viene attualizzata unicamente a multipli di questo tempo. È possibile specificare un tempo di campionamento in modo vettoriale nella forma

```
[Ts,offset]
```

In questo caso l'uscita del blocco viene attualizzata ogni

$$t = n * Ts + offset$$

Questo permette di avere blocchi con tempi di offset differenti all'interno di un modello.

Capitolo 4

Linearizzazione e equilibrio

4.1 Linearizzazione

Un modello di SIMULINK può essere linearizzato attorno ad un certo punto di lavoro mediante il comando *"linmod"*. L'equivalente per ottenere un modello discreto è il comando *"dlinmod"*, cui occorre passare anche il tempo di sampling. Un ulteriore metodo di linearizzazione è dato dal comando linmod2, un'evoluzione più accurata ma anche più lenta di linmod. Su un modello lineare possono poi essere fatte le varie analisi come *"bode"*, *"rlocus"*, *"nyquist"*, *"nychols"* ecc. Occorre specificare le entrate e le uscite del modello mediante i blocchi *"Inport"* e *"Outport"*. Il risultato dell'operazione è dato nella forma del piano degli stati.

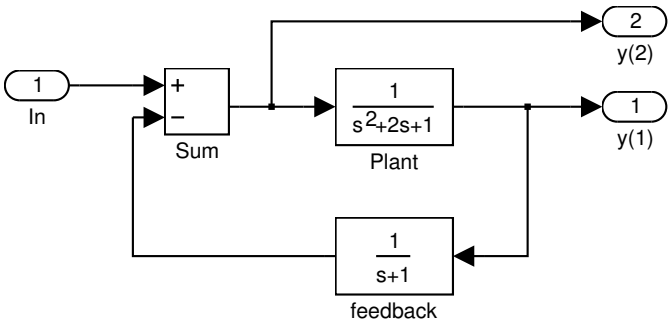


Figura 4.1: Sistema da linearizzare

Linearizziamo ora il processo della figura 4.1

```
 [a,b,c,d]=linmod('mod_lin')
a =
    -2    -1    -1
     1     0     0
     0     1    -1
b =
     1
     0
     0
```

```
c =
     0     1     0
     0     0    -1
d =
     0
     1
```

Possiamo trovare ora altre caratteristiche del sistema, come ad esempio la sua funzione di trasferimento

```
plant=ss(a,b,c,d);
g=tf(plant)

Transfer function from input to output...
              s + 1
 #1:  ---------------------
      s^3 + 3 s^2 + 3 s + 2




      s^3 + 3 s^2 + 3 s + 1
 #2:  ---------------------
      s^3 + 3 s^2 + 3 s + 2
```

Con il comando *"linmod"* si possono anche specificare il punto di lavoro e di disturbo, ottenibili da un'analisi del punto di equilibrio.

4.2 Punti di equilibrio

Con il comando *"trim"* si può trovare il punto di equilibrio di un modello. Riprendendo l'esempio precedente vogliamo trovare i valori dell'entrata e degli stati che portano le uscite a 1.

Diamo dei valori iniziali all'entrata e agli stati

```
x=[0;0;0];
u=0;
```

quindi settiamo il valore desiderato per le due uscite

```
y=[1;1];
```

Usiamo ora degli indici per fissare quali variabili possono essere fatte variare e quali no

```
ix=[];        %stati possono variare
iu=[];        %entrata pu\'o variare
iy=[1;2];     %fisso il valore dell'uscita 1 e 2
```

Ora possiamo utilizzare la funzione *"trim"* e trovare il risultato cercato

```
 [x,u,y,dx]=trim('mod_lin',x,u,y,ix,iu,iy)
x =
    0.0000
    1.0000
    1.0000
u =
     2
y =
    1.0000
    1.0000
dx =
  1.0e-015 *
   -0.2220
   -0.0227
    0.3331
```

Nel caso di sistemi non lineari, le informazioni trovate possono poi essere utilizzate per linearizzare il processo attorno a questo punto di equilibrio.

Capitolo 5

Mascheramento di blocchi

5.1 Costruzione del blocco

Vediamo con un esempio concreto come sia possibile creare dei nuovi blocchi e come si possa interfacciarli verso l'esterno.

Prendiamo come esempio un motore con il suo driver di potenza. Come funzione di trasferimento del driver di potenza possiamo utilizzare quella di un filtro passabasso con frequenza limite ω_{lim}, mentre la funzione di trasferimento del motore può essere trovata analizzando il sistema elettromeccanico e costruendo le equazioni differenziali che lo descrivono. Nel *"Corso di Regolazione"* è descritto questo procedimento.

Le funzioni di trasferimento trovate sono rappresentate nella tabella 5.1

driver	$G_{driver}(s) = \dfrac{\omega_{lim}}{s + \omega_{lim}}$
motore	$G_{tot}(s) = \dfrac{\dfrac{K_t}{R_a J_m}}{s + \dfrac{1}{J_m}\left(D_m + \dfrac{K_t K_b}{R_a}\right)}$

Tabella 5.1: Funzioni di trasferimento dei blocchi

In SIMULINK inseriamo dapprima le due funzioni di trasferimento prendendole dalla biblioteca *"linear"*. In seguito, con i dialoghi dei parametri, inseriamo i valori come da tabella 5.1 e 5.1.

Numerator	`[w_lim]`
Denominator	`[1 w_lim]`

Numerator	`[Kt/(Ra*Jm)]`
Denominator	`[1 1/Jm*(Dm+Kt*Kb/Ra)]`

Unendo i due blocchi si ottiene il diagramma della figura 5.1.

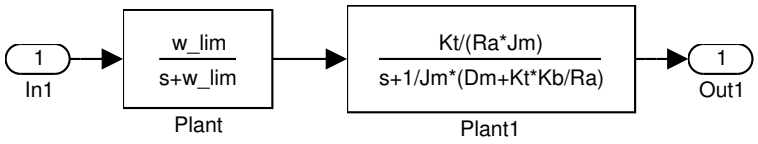


Figura 5.1: Sistema driver-motore

5.2 Mascheramento del blocco

Si tratta ora di creare un nuovo blocco da chiamare motore, utilizzabile in seguito semplicemente impostando i parametri necessari w_lim, Kt, Kb, Ra, Jm e Dm. Selezioniamo con il mouse tutti i blocchi e chiamiamo il comando *"Edit-Create Subsystem"*. Ridimensionando il blocco ottenuto possiamo trovare la figura 5.2.

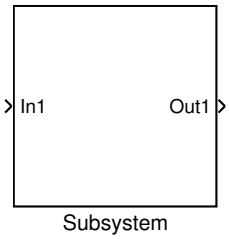


Figura 5.2: Blocco mascherato

Facendo un doppio click con il mouse possiamo rivedere il contenuto del blocco mascherato, e modificare i nomi dell'entrata e dell'uscita, che sono la tensione u(t) in entrata e la velocità w(t) in uscita. Richiudendo il modulo e modificandone il nome come "Motore & Driver" si ottiene la figura 5.3.

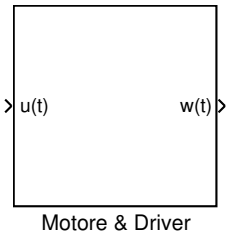


Figura 5.3: Sistema mascherato

5.3 Creazione della maschera di immissione dati

5.3.1 Pagina di documentazione

Ora, utilizzando il comando *"Edit-Create Mask"* possiamo personalizzare questo blocco e impostarne alcuni parametri.

Scegliendo l'opzione *"Documentation"* possiamo impostare le informazioni relative al nuovo blocco come descritto nella tabella 5.2

Mask type	Motore & Driver
Block Description	Questo blocco modellizza un motore e il suo driver di potenza
Block help	L'input di questo blocco è la tensione u(t) applicata all'entrata dell'amplificatore di potenza, mentre l'output è la velocità in uscita. All'entrata occorre specificare i parametri seguenti: Kt : costante di momento Kb : costante di indizione (back emf) Ra : resistenza dell'armatura Jm : momento di inerzia del motore Dm : attrito di rotazione del motore f_lim : frequenza limite del driver

Tabella 5.2: Informazioni di help del nuovo blocco

5.3.2 Editor e inizializzazione

In questa pagina possono essere impostate tutte le variabili che compaiono all'interno del blocco e che non sono ancora state inizializzate. Inoltre è possibile determinare dei comandi da eseguire durante l'inizializzazione del blocco, per creare ad esempio altre variabili interne. Definiamo una maschera di tipo editor per tutte le variabili Kt, Kb, Ra, Jm, Dm e f_lim, con il prompt di descrizione. L'inizializzazione svolge le operazioni seguenti:

```
w_lim=2*pi*f_lim;
fi=[0:0.2:2*pi 0];
```

Quando ora si farà un doppio click sul nuovo blocco di SIMULINK si otterrà una finestra di dialogo che richiede tutti i parametri necessari.

5.3.3 Pagina delle icone

In questa pagina si possono impostare i parametri relativi al disegno del nuovo blocco.

5.3.3.1 Disegno

Nella parte di inzializzazione possono essere create delle variabili necessarie per il disegno, come ad esempio nel nostro caso particolare l'angolo *fi*. Questo ci serve per poter disegnare il cerchio che descrive il motore. I comandi di disegno possibili sono rappresentati nella tabella 5.3.

Nel nostro caso impostiamo i comandi di disegno seguenti:

disp(string)	visualizza una stringa al centro dell'icona
text(x,y,string)	visualizza un testo alle coordinate x,y
fprintf(string,list)	visualizza il risultato di fprintf al centro dell'icona
plot(vett_x,vett_y)	disegna un grafico nell'icona
dpoly(num,den)	funzione di trasferimento in s al centro dell'icona
dpoly(num,den,'z') dpoly(num,den,'z-')	funzione di trasferimento in z al centro dell'icona
droots(zeros,poles,gain)	funzione di trasferimento nella forma zeri-poli-guadagno

Tabella 5.3: Comandi per il disegno dell'icona

```
plot(0.5+0.2*cos(fi),0.5+0.2*sin(fi)));
text(0.5,0.5,'M');
plot([0.4 0.4 0.6 0.6],[0.7 0.8 0.8 0.7]);
plot([0.4 0.4 0.6 0.6],[0.31 0.2 0.2 0.31]);
plot([0.2 0.5 0.5],[0.9 0.9 0.8]);
plot([0.2 0.5 0.5],[0.1 0.1 0.2]);
```

Si ottiene l'icona della figura 5.4.

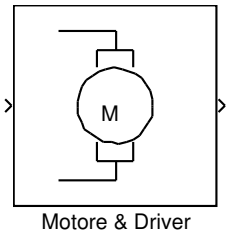


Figura 5.4: Icona del nuovo blocco

5.3.3.2 Icon frame

Serve per visualizzare la cornice dell'icona (vedi tabella 5.4).

5.3.3.3 Icon transparency

Serve per modificare la trasparenza dell'icona (vedi tabella 5.5).

5.3.3.4 Icon rotation

Imposta la rotazione dell'icona (vedi tabella 5.6).

5.3.3.5 Drawing coordinates

Imposta il sistema di riferimento per le coordinate (vedi tabella 5.7).

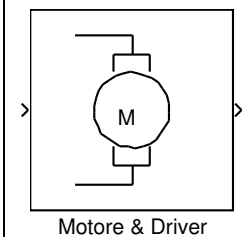
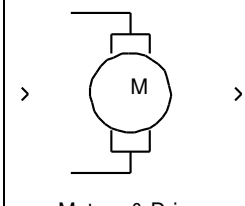
Visible	
Invisible	

Tabella 5.4: Icon frame

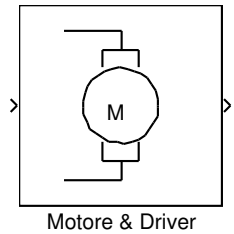
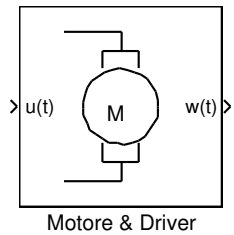
Opaque	
Transparent	

Tabella 5.5: Icon transparency

Fixed	icona mostrata normale
Rotates	icona mostrata capovolta

Tabella 5.6: Icon rotation

Pixel	Coordinate assolute con angolo in basso a sinistra pari a (0,0)
Autoscale	Scale automatiche in modo da adattare il comando di display all'icona
Normalized	Normalizzazione dell'icona tra le coordinate (0,0) in basso a sinistra e le coordinate (1,1) in alto a destra.

Tabella 5.7: Drawing coordinates

Capitolo 6

Esecuzione condizionata

Dalla versione 2.0 di SIMULINK sono a disposizione nella libreria *"Connections"* due nuovi elementi che permettono di condizionare l'esecuzione di blocchi a segnali esterni. Si tratta dei moduli *"Enable"* e *"Trigger"*. Questi due moduli vanno usati all'interno di un sottosistema (*"Subsystem"*) e gestiscono l'esecuzione condizionata di queste parti del modello. La logica per gestire questi segnali può essere generata in altri moduli, utilizzando anche funzioni logiche presenti in SIMULINK.

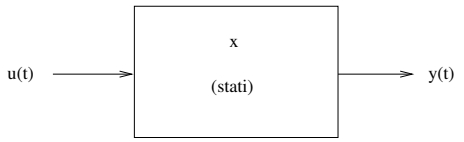
Con questi moduli è quindi possibile inserire diversi regolatori (blocchi condizionati) in un sistema e cambiare durante la simulazione il tipo di regolatore a seconda delle condizioni di lavoro (segnale *"Enable"*).

Capitolo 7

S-Function

Tramite l'utilizzo di S-Function (System-Function) si possono aumentare le possibilità di SIMULINK

Queste funzioni possono essere implementate con il linguaggio di programmazione di MATLAB oppure in un linguaggio evoluto come C o Fortran, tramite le cosiddette MEX-Function. L'uso di queste S-Function permette ad esempio di integrare le animazioni dei risultati in SIMULINK, oppure di programmare delle nonlinearità particolari, oppure ancora di integrare programmi in C o Fortran all'interno di MATLAB e SIMULINK. Una S-Function lavora secondo il principio rappresentato nella figura 7.1.



$y = f_0(t, x, u)$	(output)
$\dot{x}_c = f_d(t, x, u)$	(derivate)
$x_{d+1} = f_u(t, x, u)$	(aggiornamento)

Figura 7.1: Principio di funzionamento di una S-Function

La S-Function viene costantemente chiamata dal sistema per eseguire tutte le operazioni necessarie alla soluzione del problema secondo lo schema della figura 7.2.

La chiamata di ogni stadio di integrazione viene fatta chiamando le procedure specifiche per ogni azione. Queste procedure hanno lo stesso nome all'interno di un M- file e di un MEX-File scritto in C. La differenza è che nel caso di un M-file, viene passato alla S-Function un flag supplementare che serve a decidere quale operazione effettuare (vedi tabella 7.1.

Stadio di simulazione	Procedura della S-Function	Flag
Inizializzazione	mdlInitializeSizes	flag=0
Calcolo del prossimo tempo di sampling	mdlGetTimeOfNextVarHit	flag=4
Calcolo delle uscite	mdlOutputs	flag=3
Aggiornamento degli stati discreti	mdlUpdate	flag=2
Calcolo delle derivate	mdlDerivatives	flag=1
Fine della simulazione	mdlTerminate	flag=9

Tabella 7.1: Chiamate durante la simulazione

Nel caso di C-MEX-Files, le procedure devono avere l'esatto nome riportato nella

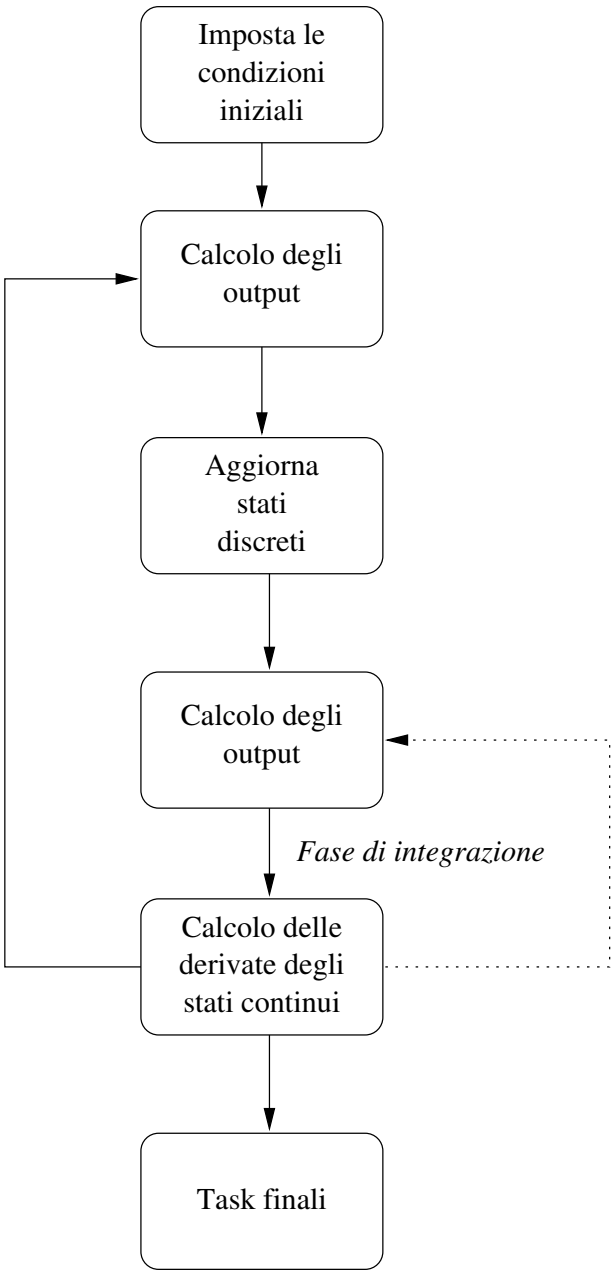


Figura 7.2: Schema a blocchi del funzionamento di una S-Function

2. colonna, mentre nel caso di M-Files occorre implementare del codice per gestire le chiamate a sottoprocedure tramite i flags.

Si consiglia di analizzare gli esempi di S-Function forniti con SIMULINK, esempi che si trovano sotto /toolbox/simulink/blocks (M-Files) e /simulink/src (C-MEX- Files).

Appendice A

Comandi principali di Simulink

A.1 SIMULINK Model analysis and construction functions.

A.1.1 Simulation.

```
sim          - Simulate a SIMULINK model.
simset       - Define options to SIM Options structure.
simget       - Get SIM Options structure
```

A.1.2 Linearization and trimming.

```
linmod       - Extract linear model from continuous-time system.
linmod2      - Extract linear model, advanced method.
dlinmod      - Extract linear model from discrete-time system.
trim         - Find steady-state operating point.
```

A.1.3 Model Construction.

```
close_system  - Close open model or block.
new_system    - Create new empty model window.
open_system   - Open existing model or block.
save_system   - Save an open model.
add_block     - Add new block.
add_line      - Add new line.
delete_block  - Remove block.
delete_line   - Remove line.
replace_block - Replace existing blocks with a new block.
set_param     - Set parameter values for model or block.
get_param     - Get simulation parameter values from model.
bdclose       - Close a SIMULINK window.
bdroot        - Root level model name.
gcb           - Get the name of the current block.
gcs           - Get the name of the current system.
getfullname   - get the full path name of a block
slupdate      - Update older 1.x models to 2.x.
addterms      - Add terminators to unconnected ports.
```

A.1.4 Masking.

```
hasmask       - Check for mask.
hasmaskdlg    - Check for mask dialog.
hasmaskicon   - Check for mask icon.
iconedit      - Design block icons using ginput function.
lookundermask - Look under mask and past the OpenFcn.
```

A.2 Demo di Simulink

A.2.1 SIMULINK demonstrations and samples.

simdemo - Main menu of SIMULINK demonstrations.

A.2.2 Demonstration models.

```
vdp        - Van der Pol system of equations.
f14        - F-14 aircraft model.
thermo     - Thermodynamics of a house.
pops       - Population dynamics.
parmest    - Parameter estimation.
multi      - Multivariate lines.
steps      - Effects of varying step size.
rlsest     - Adaptive control.
penddemo   - Inverted pendulum (on cart) with animation.
filters    - Three different filters.
```

A.2.3 Demonstration models for case studies 1,2,3 in the user's guide.

```
f14c       - F-14 benchmark, closed-loop form.
f14o       - F-14 benchmark, open-loop form.
f14n       - F-14 block diagram, alternate form.
ben2asys   - JHU/APL continuous benchmark, simple 13-state model
ben2bsys   - JHU/APL continuous benchmark, intermediate 42-state model.
ben2csys   - JHU/APL continuous benchmark, difficult 74-state model.
ben3asys   - JHU/APL sampled-data benchmark, simple 10-state model.
ben3bsys   - JHU/APL sampled-data benchmark, complex 29-state model.
```

A.2.4 Demonstration scripts.

```
popdemo    - Population dynamics.
vdpdemo    - Van der Pol equation.

Execute the MATLAB command "simdemo" to see a menu of all demos and
models in this directory (except for case studies #1-3).  The demo menu
is also available from inside the "Extras" block in the main
SIMULINK block library (which is displayed by executing "simulink").
```

A.3 Blocks

A.3.1 SIMULINK block library.

```
Block libraries.
 simulink   - Open main block library.
```

A.3.2 Example S-function models and blocks.

```
 simo       - simo model, block diagram form.
 simom      - simo model, M-file form.
 simom2     - simo model, M-file form #2.
 simosys    - simo model, S-function block form.
 vdp        - Van der Pol model, block diagram form.
 vdpm       - Van der Pol model, M-file form.
 mixed      - Mixed continuous/discrete model, block diagram form.
 mixedm     - Mixed continuous/discrete model, M-file form.
 limintm    - Limited integrator block, M-file form.
 vlimintc   - Vectorized limited integrator, M-file form.
 vdlmintc   - Discrete-time vectorized limited integrator, M-file.
 sfuntmpl   - M-file S-function template.
 csfunc     - Continuous-time model, M-file form.
 dsfunc     - Discrete-time model, M-file form.
 vsfunc     - Variable sample-time model, M-file form.
 sfuncont   - M-file S-function template, continuous-time model.
 sfundsc1   - M-file S-function template, discrete real-time model.
 sfundsc2   - M-file S-function template, discrete sampled model.
 timestwo   - M-file S-function example.

See also the "simulink" directory off of your main MATLAB
installation area. This area contains example C and Fortran
implementations of the above blocks.
```


Appendice B

Librerie principali di Simulink

B.1 Librerie di base

Le figure B.1, B.2, B.3, B.4, B.5 e B.6 mostrano le biblioteche principali con i simboli di SIMULINK, suddivise per tipo.

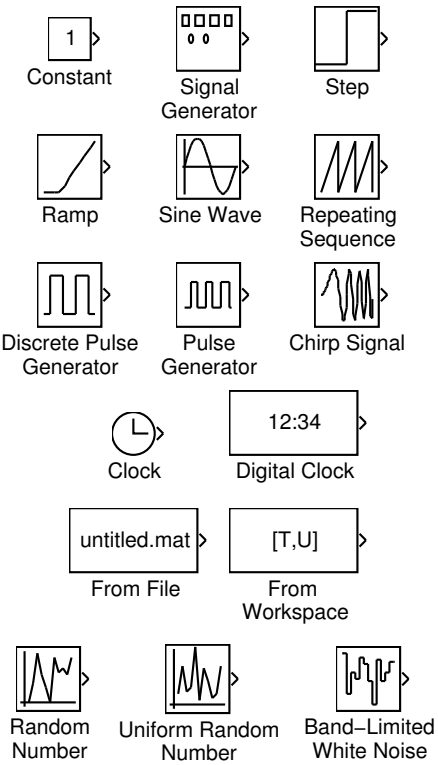


Figura B.1: Sources

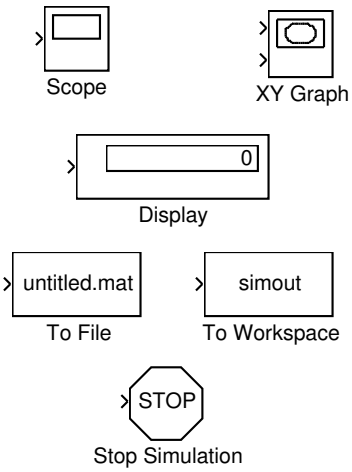


Figura B.2: Sinks

B.2 Librerie aggiuntive

Le figure B.7, B.8, B.9, B.10, B.11,B.12, B.13 e B.14 mostrano le biblioteche aggiuntive presenti in SIMULINK, suddivise per tipo.

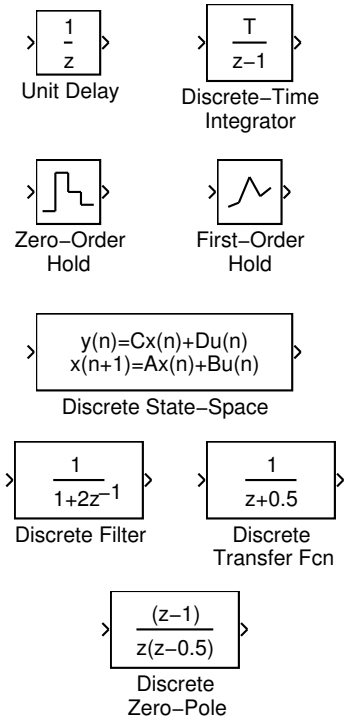


Figura B.3: Discrete

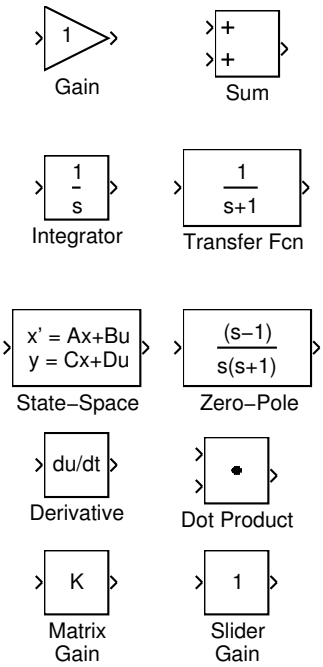


Figura B.4: Linear

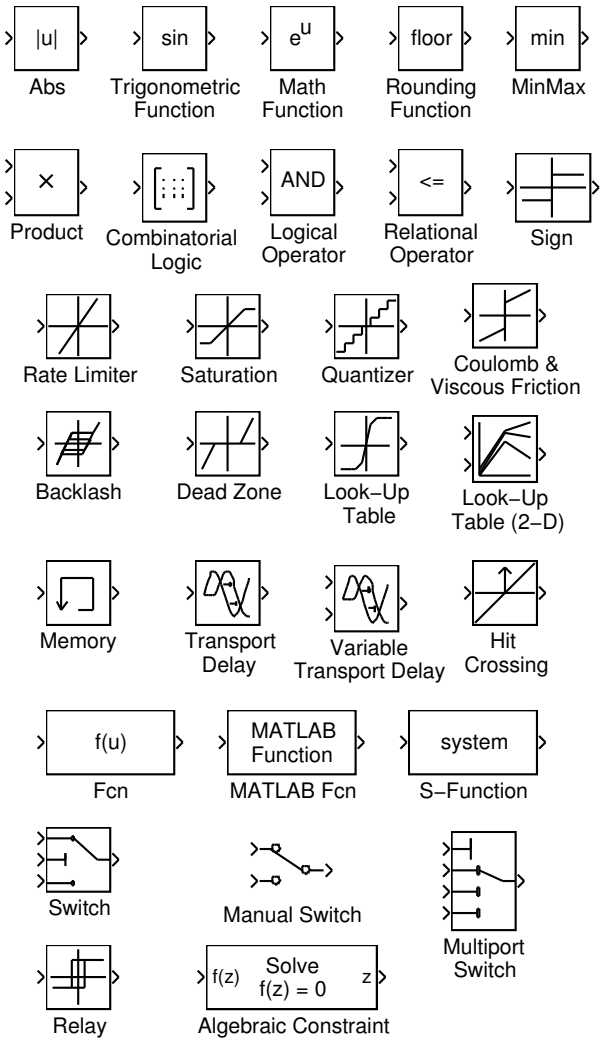


Figura B.5: Nonlinear

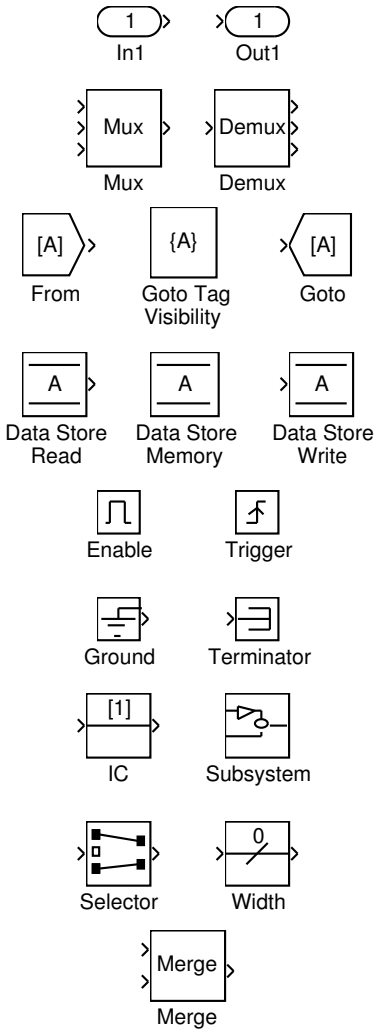


Figura B.6: Connections

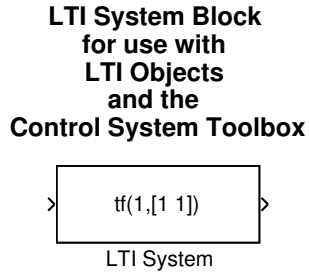


Figura B.7: LTI Block

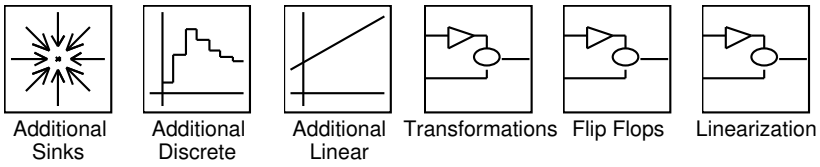


Figura B.8: Additional Library

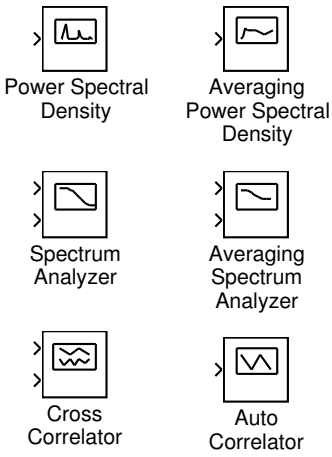


Figura B.9: Additional Sinks

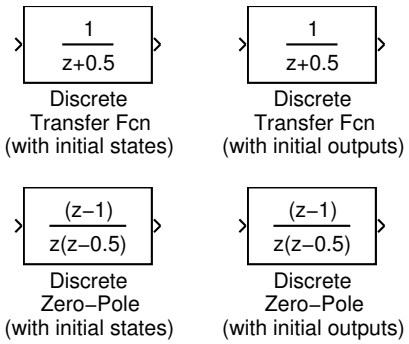


Figura B.10: Additional Discrete

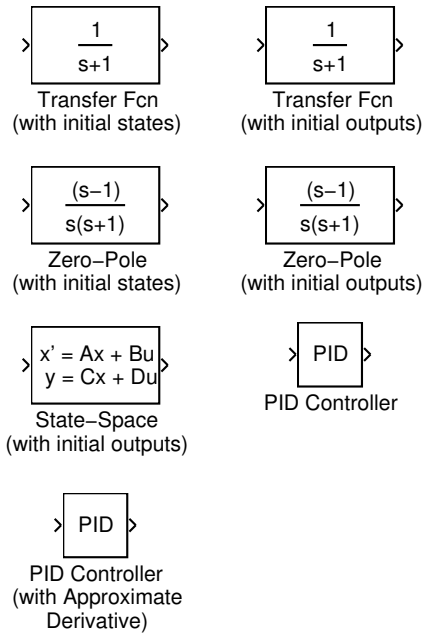


Figura B.11: Additional Linear

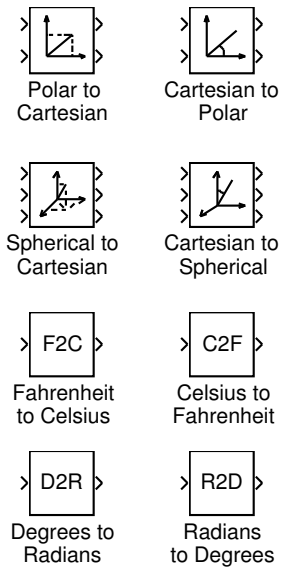


Figura B.12: Conversion

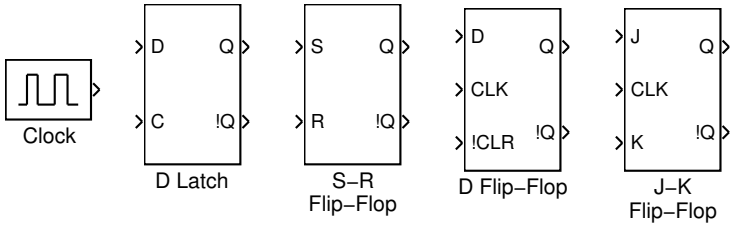


Figura B.13: Flip Flop

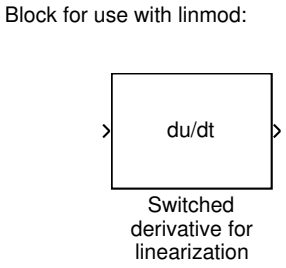


Figura B.14: Linearization