Fondamenti di Automatica

Risposta di Funzioni di Trasferimento e Diagrammi a Blocchi (Matlab/Simulink + Control Systems + Symbolic)

Prof. Marcello Bonfè

Dipartimento di Ingegneria - Università di Ferrara

Tel. +39 0532 974839

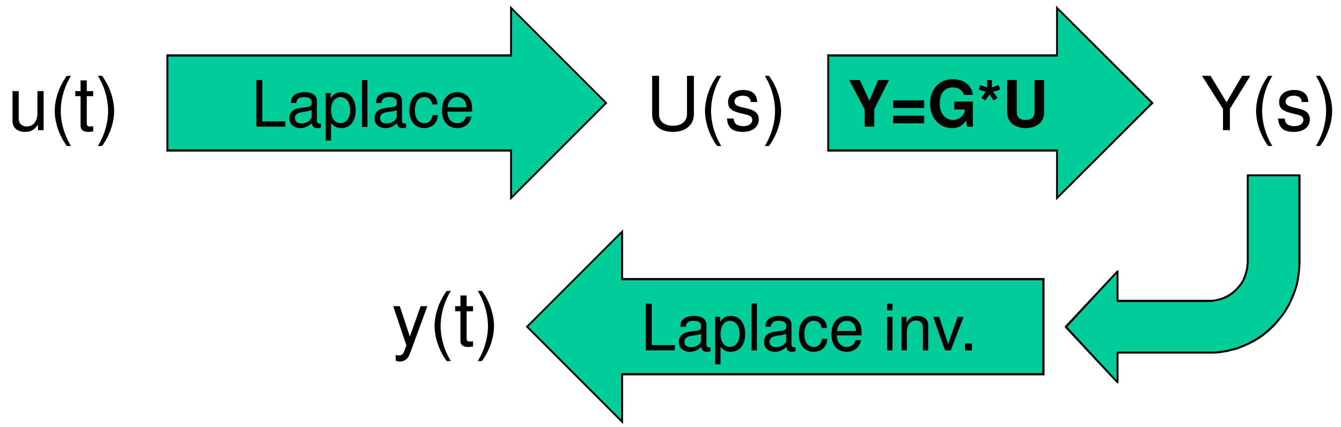
E-mail: marcello.bonfe@unife.it

Department of Engineering Ferrara

Università degli Studi di Ferrara

Matlab: Laplace e risposta del sistema

- Come visto in precedenza, l’utilizzo di trasformata e anti-trasformata di Laplace permette di calcolare **l’espressione analitica della risposta** di un sistema rispetto a qualunque segnale, senza svolgere integrali di convoluzione (necessari invece nel dominio del tempo):



Matlab: risposta di Funzioni di Trasferimento

- Lo svolgimento manuale dei calcoli necessari per ottenere l'antitrasformata di Laplace di una funzione razionale (caso tipico delle Funzioni di Trasferimento, FdT, di un Sistema LTI) richiede la **scomposizione in fratti semplici** della funzione:

$$F(s) = \frac{N(s)}{D(s)} = \frac{N(s)}{(s-p_1)(s-p_2)\dots(s-p_n)} = \sum_{i=1}^{n} \frac{k_i}{s-p_i}$$

$$\boxed{k_i = \left[(s-p_i)\frac{N(s)}{D(s)}\right]_{s\to p_i}}$$

Residuo di F(s) nel polo p^i

$$f(t) = \sum_{i=1}^{n} k_i e^{p_i t}$$

Matlab: risposta di Funzioni di Trasferimento

- La scomposizione in fratti semplici è supportata da Matlab sia per operazioni numeriche che simboliche
- **Matlab** (standard):
  - `[R,P,k]=residue(Num,Den)` restituisce il vettore dei residui R, quello dei poli P e l'eventuale termine costante k
- **Symbolic Toolbox**:
  - `partfrac(F,s)` restituisce la funzione F(s) scomposta nelle sue frazioni parziali (i.e. fratti semplici)

Matlab: scomposizione in fratti semplici

- Esempio (**Matlab standard**):

$$F(s) = \frac{6s+26}{s^2+8s+15}$$

```
>> [R,P,k]=residue([6 26],[1 8 15])
R =
     2
     4
P =
    -5
    -3
k = []
```

Matlab: scomposizione in fratti semplici

- Esempio (**Symbolic toolbox**):

$$F(s) = \frac{6s+26}{s^2+8s+15}$$

```
>> syms s
>> G=(6*s + 26)/(s^2 + 8*s + 15)
G =
(6*s + 26)/(s^2 + 8*s + 15)
>> partfrac(G)
ans =
4/(s + 3) + 2/(s + 5)
```

Matlab: trasformate e antitrasformate di Laplace

- **NOTA**: il risultato ottenuto è ovviamente propedeutico all'espressione dell'antitrasformata della F(s), che si ottiene immediatamente ricordando che:

$$\mathcal{L}^{-1}\left[\frac{k_i}{s-p_i}\right] = k_i e^{p_i t}$$

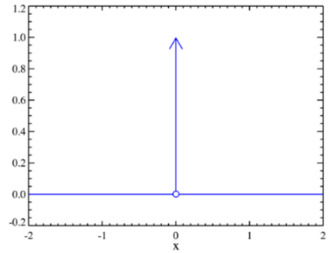
e quindi: $f(t) = 4e^{-3t} + 2e^{-5t}$

- Chiaramente, il risultato finale corrisponde a quello calcolabile direttamente con **`ilaplace(F)`** (per la funzione <u>simbolica</u>)

Matlab: risposta della FdT con Control Systems Tlbx

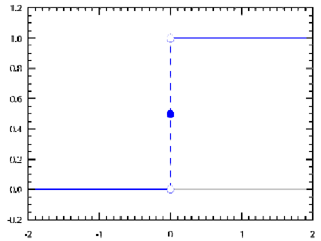
- La F(s) precedente può essere una Funzione di Trasferimento (la sua antitrasformata corrisponde alla risposta della FdT all’impulso di Dirac) oppure una FdT moltiplicata per un ingresso (es. gradino ➔ 1/s)
- Come già visto (FdA-E.2-Matlab-Laplace) esistono funzioni specifiche del Control Systems Toolbox:

```
>> impulse(G)
```



G(s)

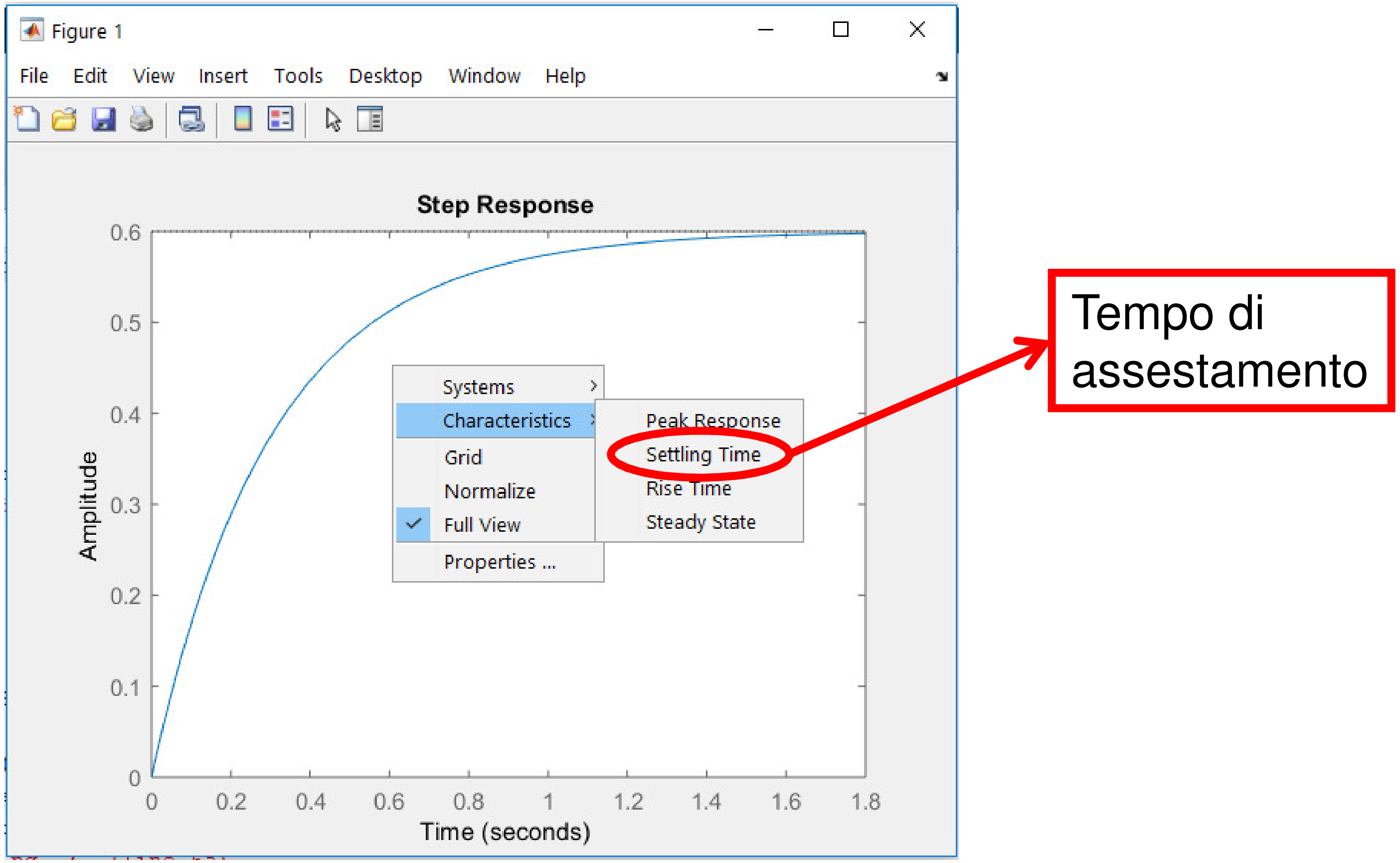
```
>> step(G)
```



G(s)

Matlab: risposta della FdT con Control Systems Tlbx

- Il grafico ottenuto con il Control Systems Toolbox è interattivo e ricco di funzionalità, supportate tramite il click con tasto destro del mouse…



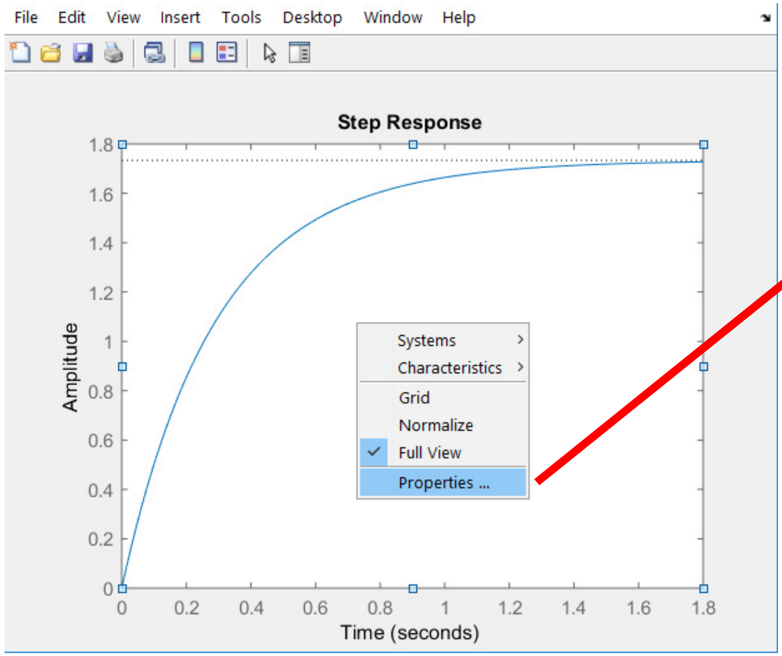
Matlab: risposta al gradino e tempo di assestamento

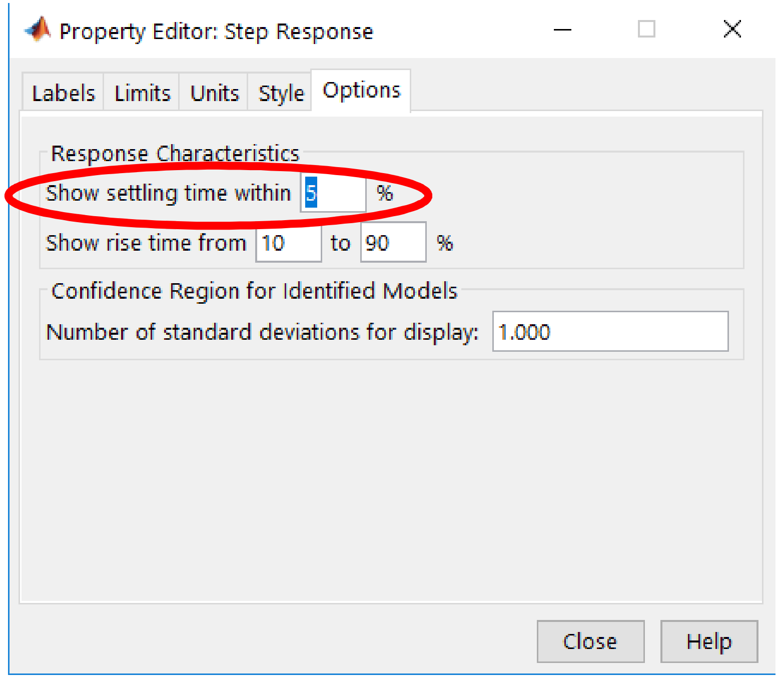
- **NOTA**: è importante osservare che il tempo di assestamento (*settling time*) calcolato da Matlab corrisponde al raggiungimento della risposta di una fascia del +/- 2% (**di default**) rispetto al valore a regime (steady state), cioè per t→∞
- Nelle dispense di questo corso e negli esercizi d'esame si considerano invece **formule per il tempo di assestamento** valide rispetto ad una **fascia del +/- 5%**
- Le impostazioni del grafico ottenuto con **`step()`** si possono (devono!) modificare coerentemente…

Matlab: risposta al gradino e tempo di assestamento

- Modifica delle impostazioni sulla soglia per il tempo di assestamento:

1. Tramite **configurazione del plot specifico**:





Matlab: risposta al gradino e tempo di assestamento

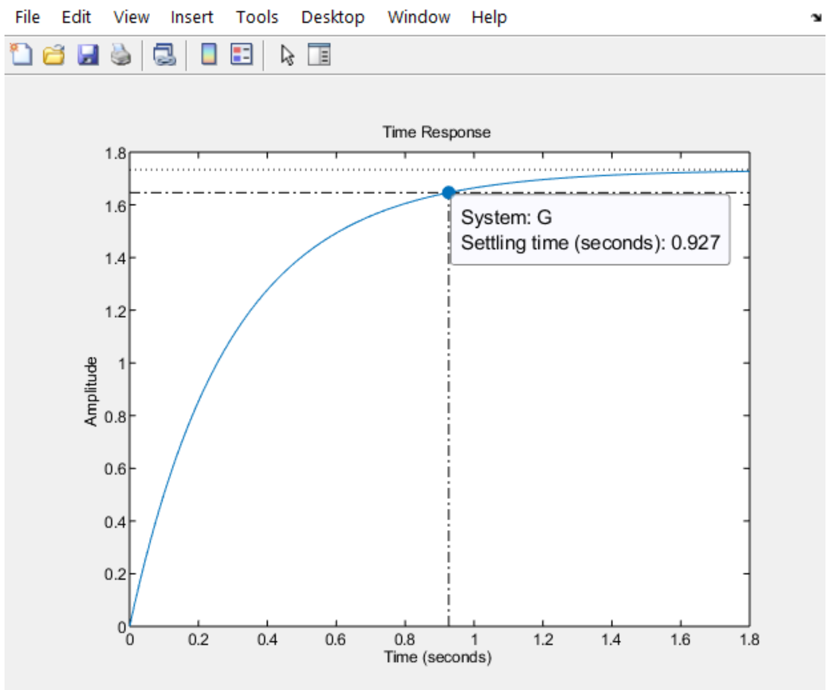
- Modifica delle impostazioni sulla soglia per il tempo di assestamento:

2. Tramite parametro con struttura **`timeoptions`** (riutilizzabile per ogni chiamata successiva):

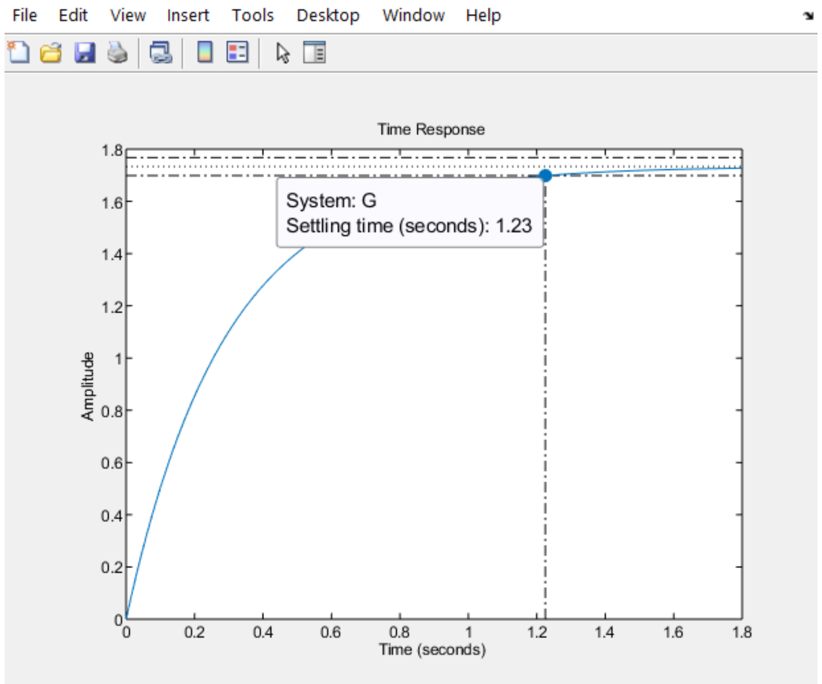
```
>> Popts = timeoptions
>> Popts.SettleTimeThreshold = 0.05
>> step(G,Popts)
```

Matlab: risposta al gradino e tempo di assestamento

T_a al +/-5%



T_a al +/-2%



Matlab: risposta al gradino e tempo di assestamento

- **NOTA**: le caratteristiche importanti della risposta al gradino si possono anche ottenere da prompt, senza passare dal grafico, e con impostazione diretta della soglia per il tempo di assestamento:

```
>> stepinfo(G,'SettleTimeThreshold',0.05)
      RiseTime: 0.6732
      SettlingTime: 0.9265
      SettlingMin: 1.5613
      SettlingMax: 1.7319
      ...
```

DIAGRAMMI A BLOCCHI IN SIMULINK

Simulink: introduzione

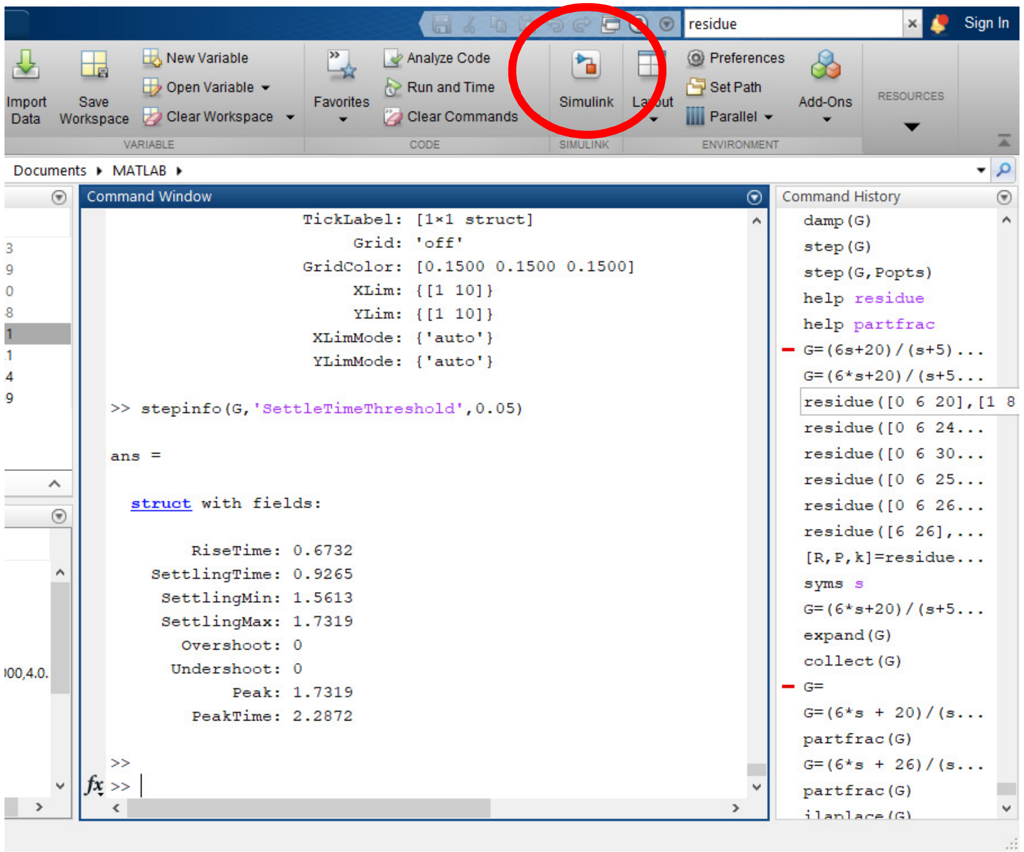
- Il software Simulink è l'estensione di Matlab per la simulazione numerica di sistemi descritti da diagrammi a blocchi
- Simulink supporta la descrizione di qualunque genere (i.e. sistemi lineari e nonlineari, tempo-continui e tempo-discreti, stazionari e non stazionari), svolgendone la simulazione tramite la scelta (<u>automatica</u>) dell'algoritmo più opportuno per la **soluzione numerica della corrispondente equazione differenziale**

Simulink: introduzione

- Da Matlab:

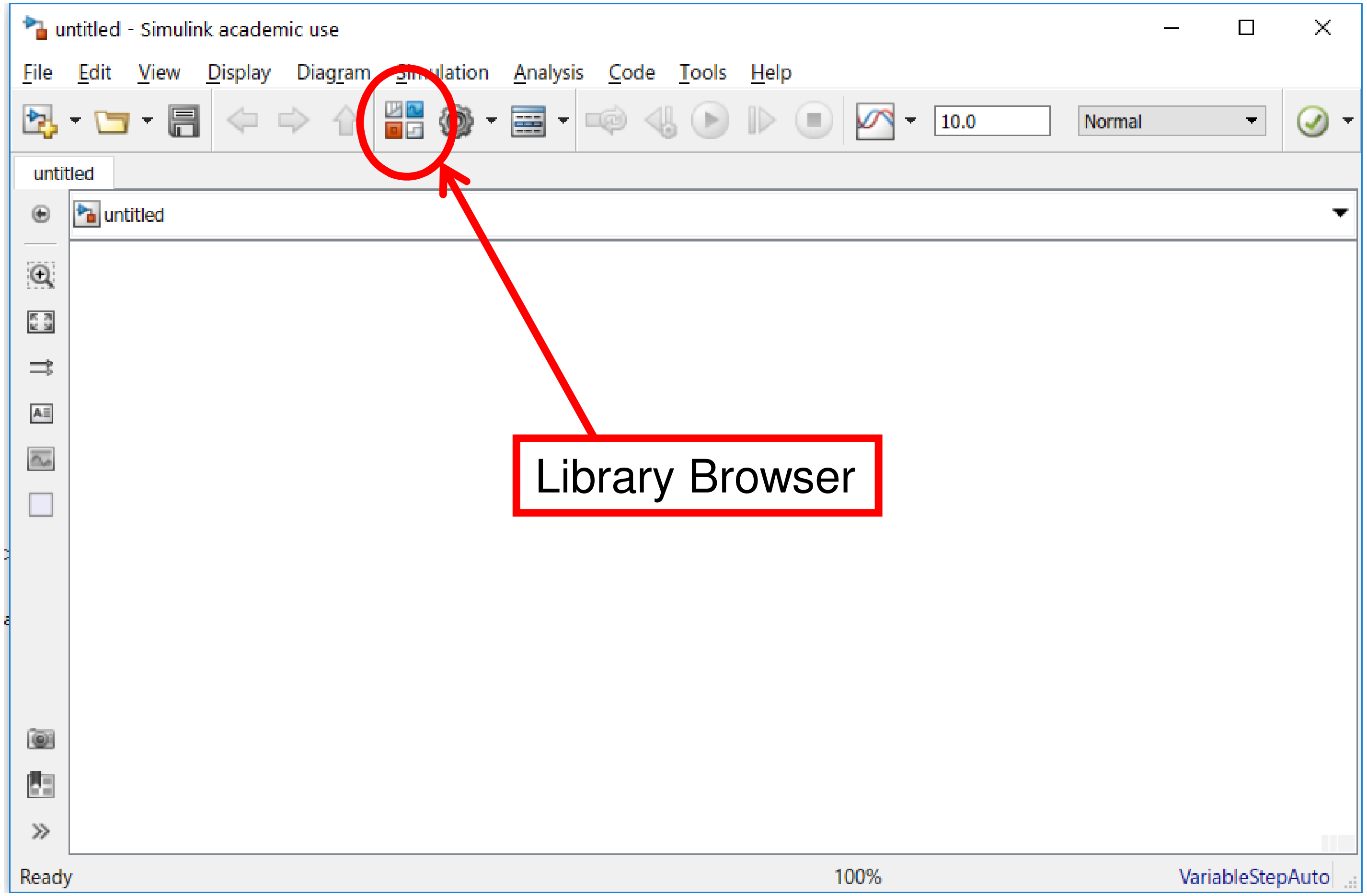
```
>> simulink
```

Oppure:



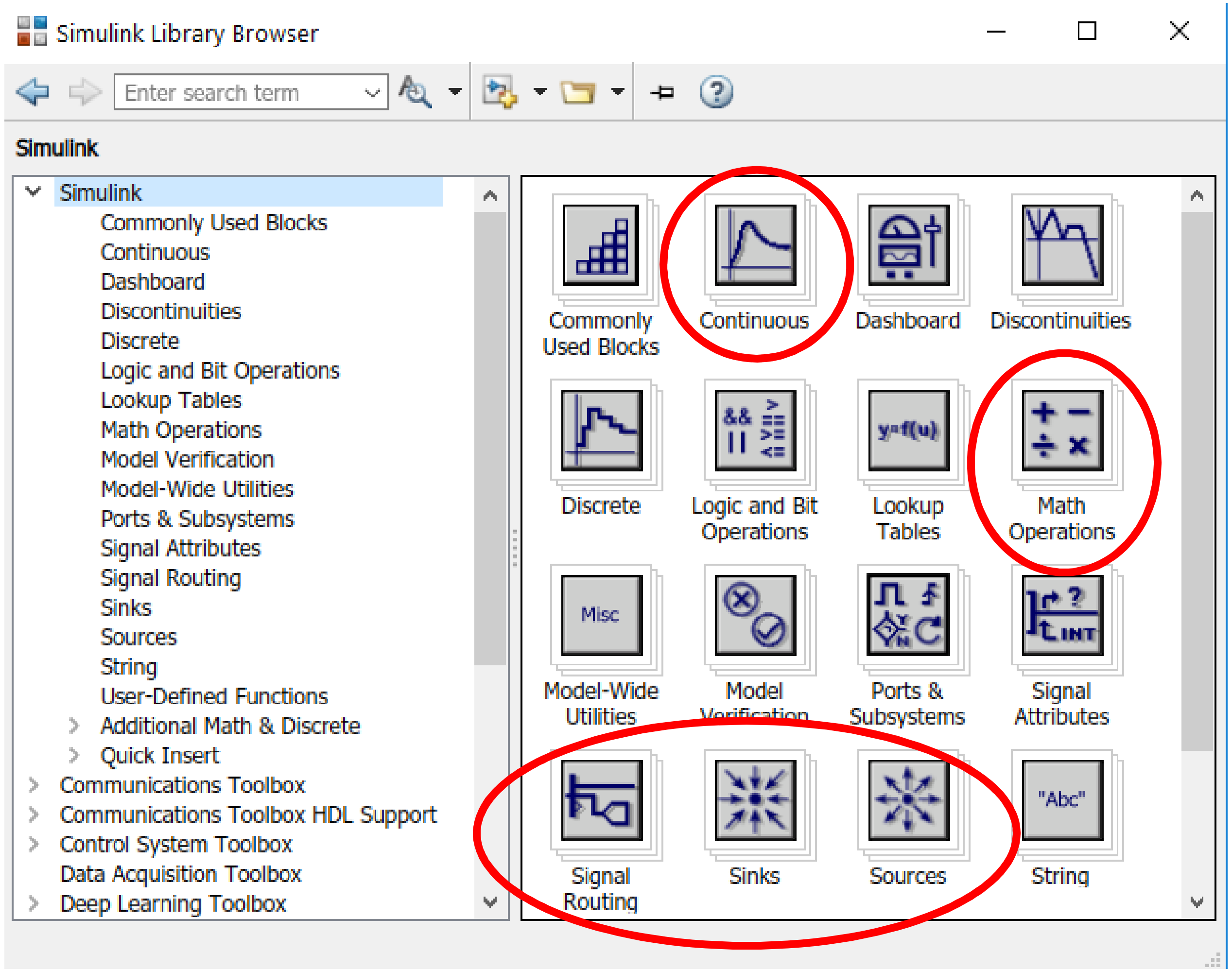
Simulink: introduzione

- New ➔ Blank Model:



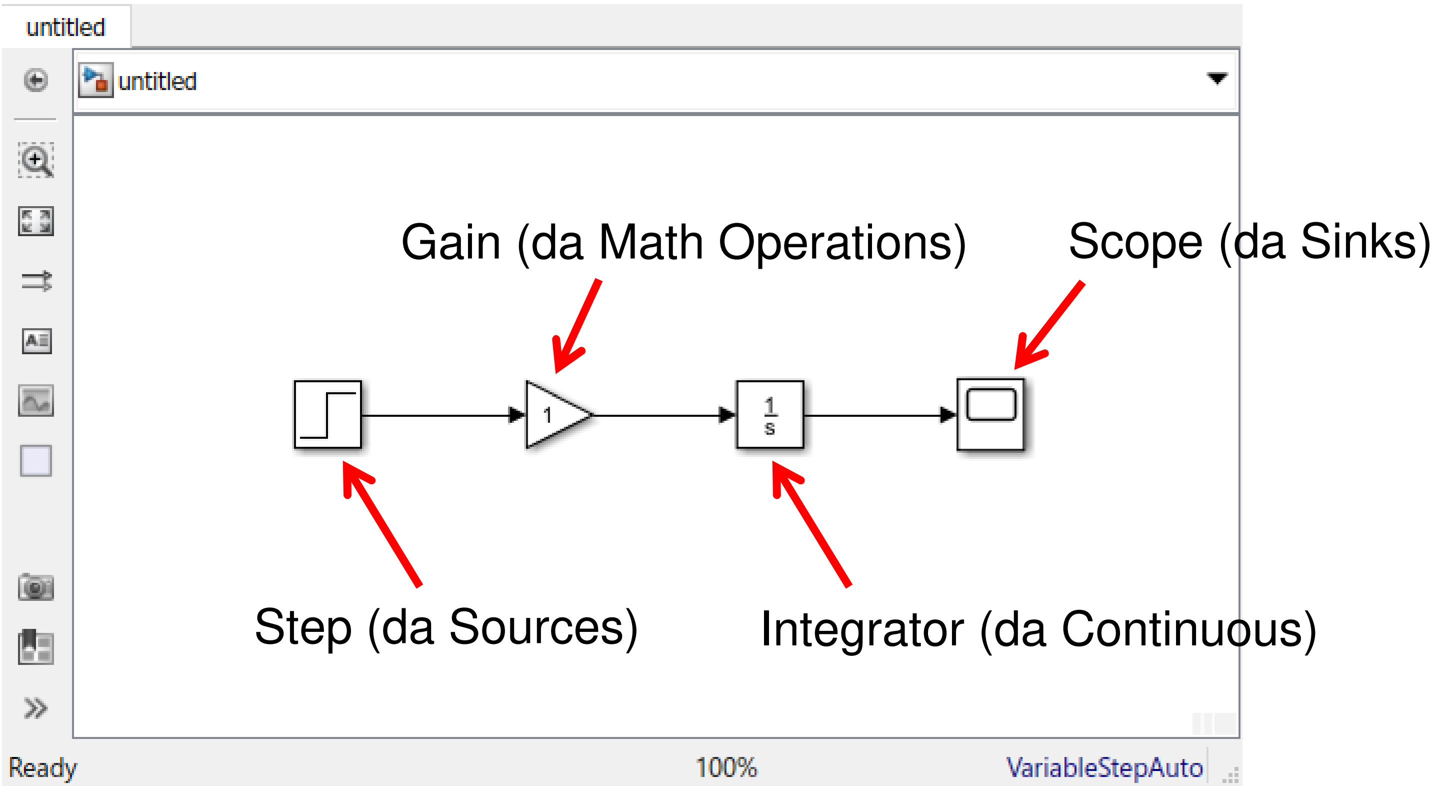
Simulink: introduzione

- Library Browser, sezioni di interesse per sis. LTI



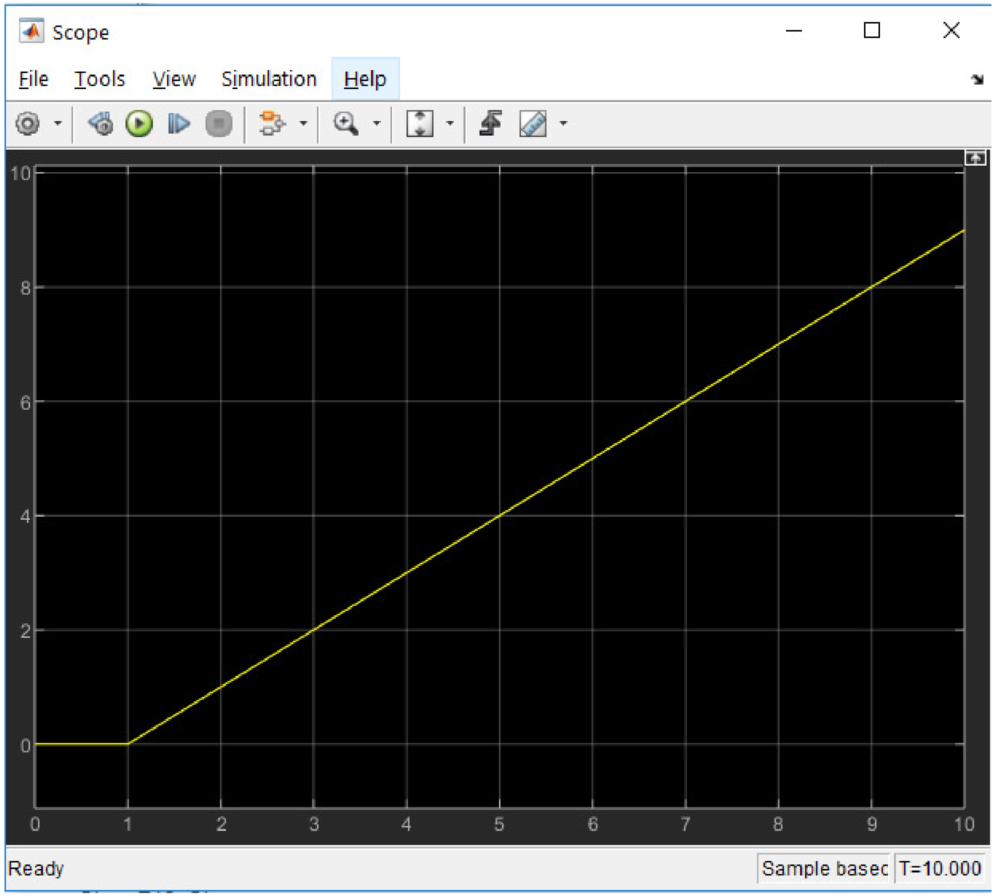
Simulink: introduzione

- Drag/Drop dal Library Brower + trascinamento del mouse dai punti di ingresso (o uscita) verso altri punti di uscita (o ingresso):



Simulink: introduzione

- Simulation ➔ Run (o Ctrl+T o icona "Run" con triangolo nero su tondo verde) + double-click Scope:



Simulink: esempio di sistema di ordine 2

- Da “Simulink for begineers” © Heikki Koivo:

DAMPED OSCILLATOR

Solve the damped oscillator problem

$$\frac{d^2x}{dt^2} + 5\frac{dx}{dt} + 9x = u(t)$$

$$\frac{dx}{dt} = \dot{x}(0) = -2$$

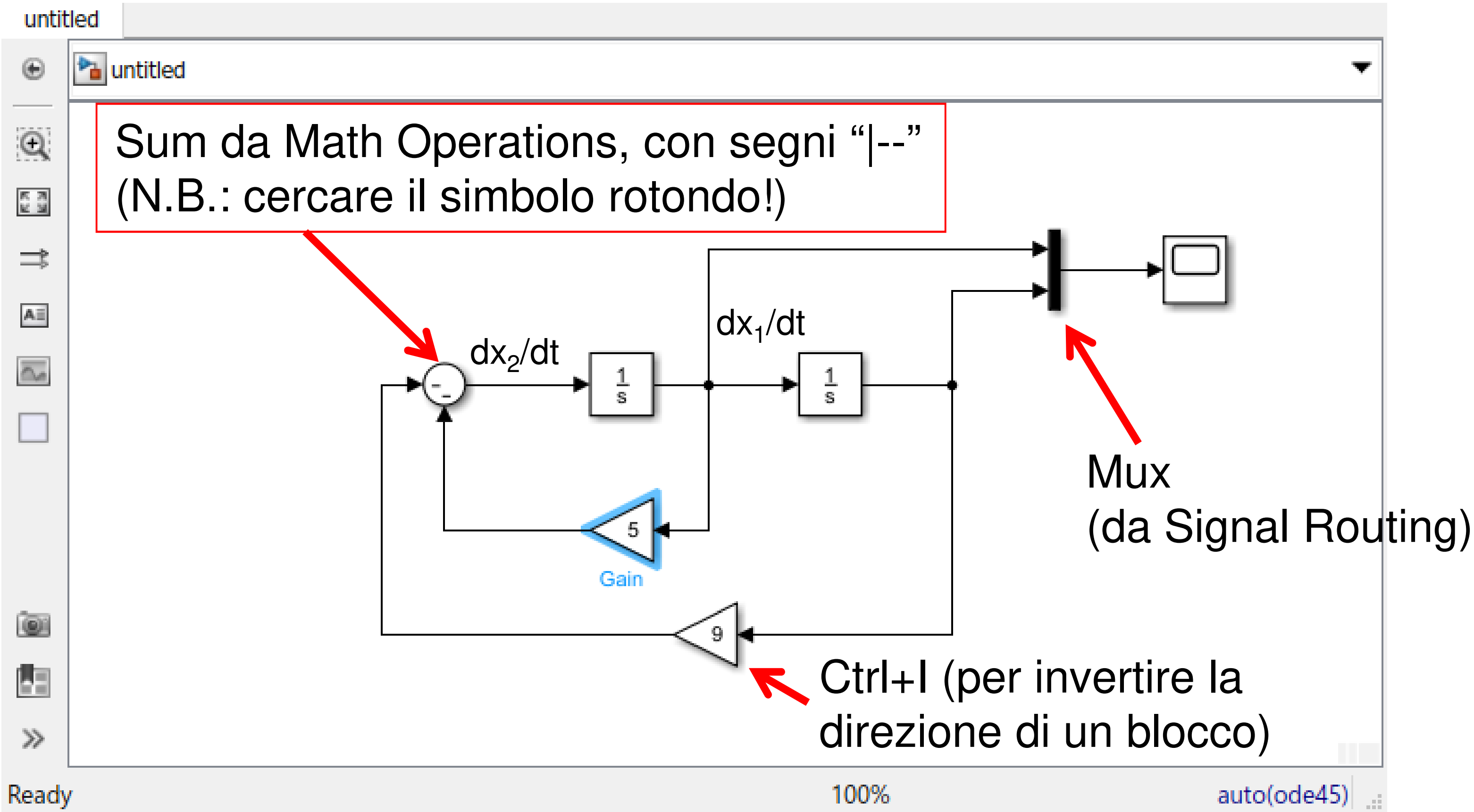
$$x(0) = 2$$

Assume that u(t) = 0, that is, there is no input.

Simulink: esempio di sistema di ordine 2

- Da "Simulink for beginners" © Heikki Koivo:

(**NOTA**: cambiare la Initial Condition degli integratori)



Simulink: esempio di sistema di ordine 2 nonlineare

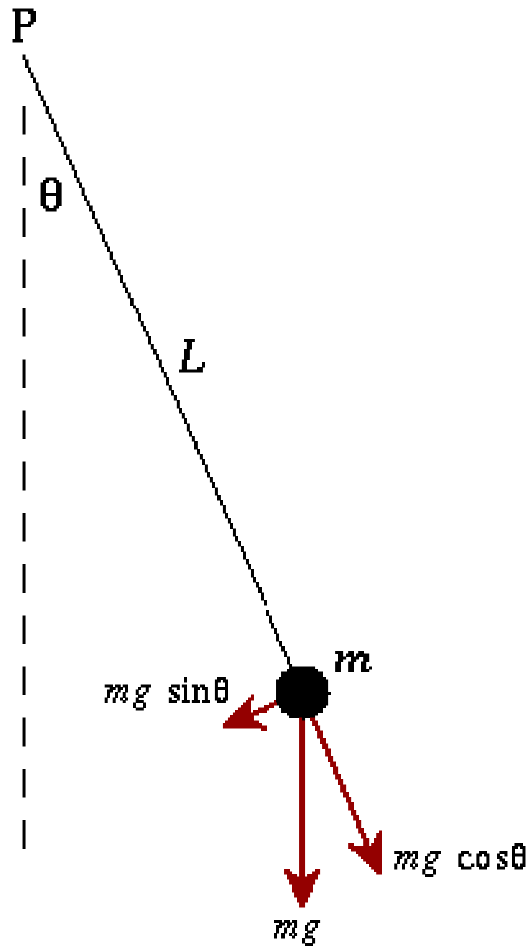
- Il pendolo semplice (con smorzamento…):

$$\tau = I\alpha \quad \Rightarrow \quad -mg\sin\theta\, L = mL^2 \frac{d^2\theta}{dt^2}$$

$$-b\frac{d\theta}{dt}$$

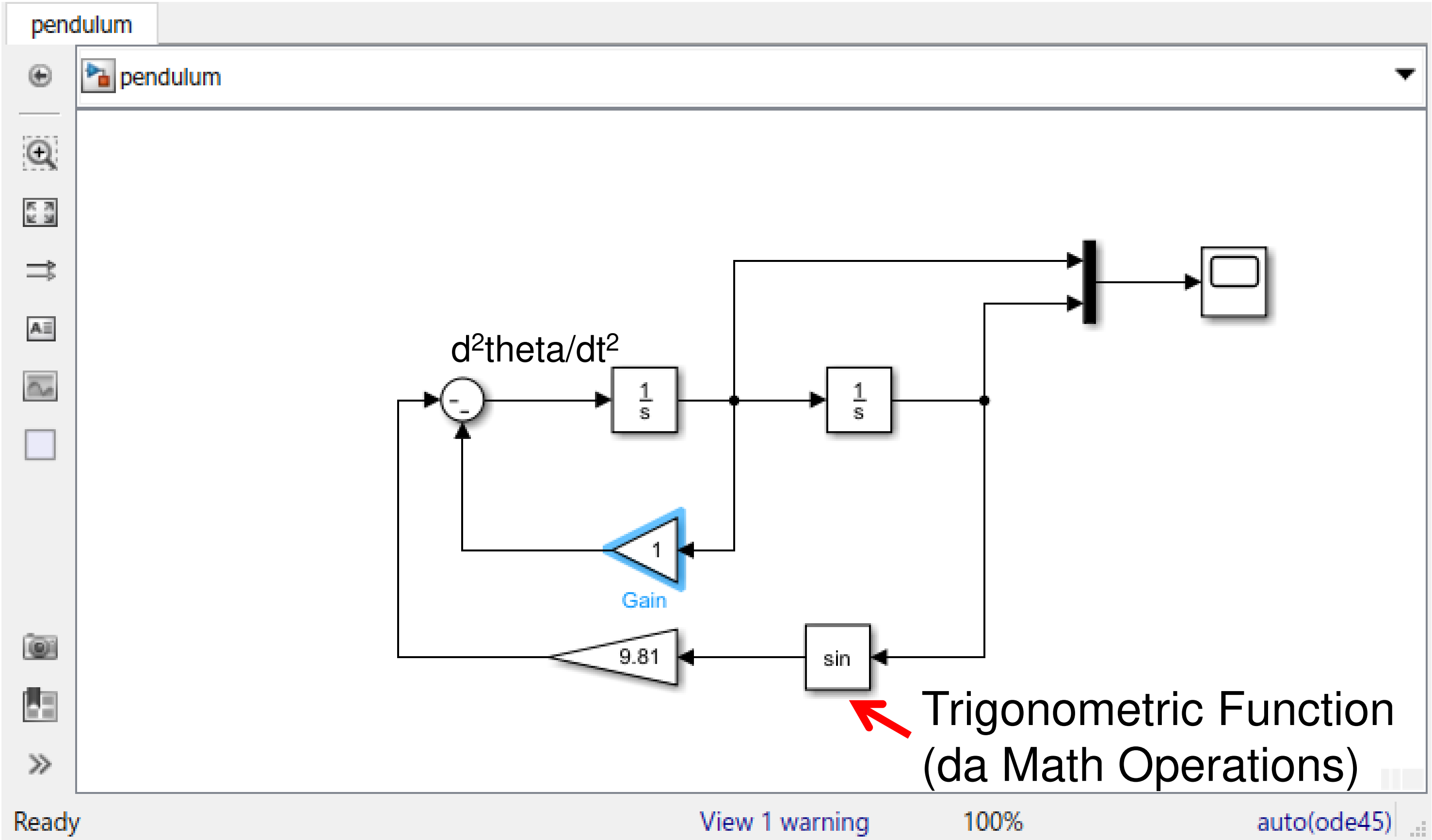
$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0$$

$$+\frac{b}{mL^2}\frac{d\theta}{dt}$$



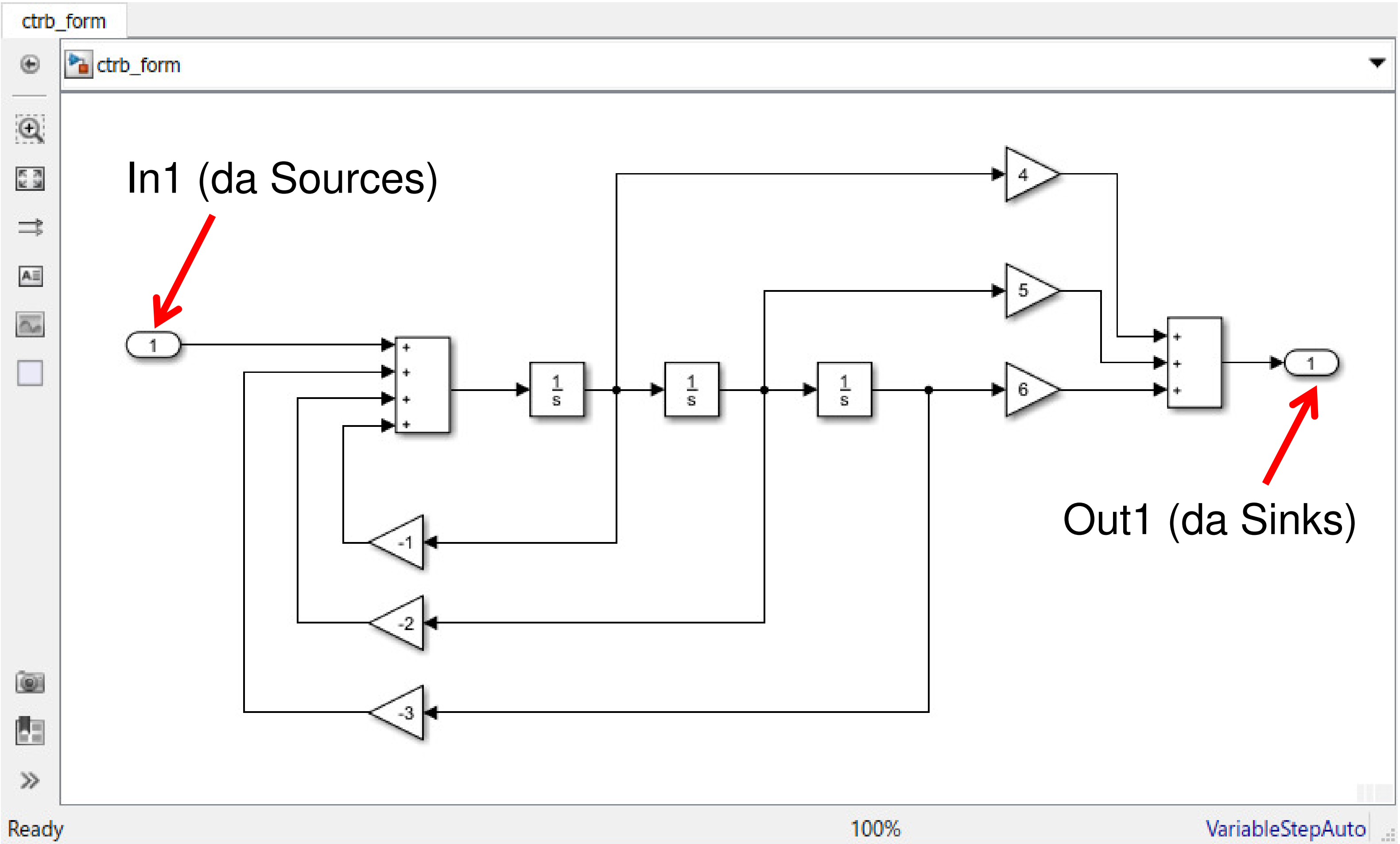
Simulink: esempio di sistema di ordine 2 nonlineare

- Il pendolo semplice:



Simulink: esempio (FdA-2.1-FunzioniTrasferimento)

- Realizzazione con forma **canonica controllabile**:



Simulink: esempio (FdA-2.1-FunzioniTrasferimento)

- Una volta salvato il diagramma (nome *ctrb_form*), è possibile estrarre le matrici del correspondente modello nello spazio degli stati:

```
>> [A,B,C,D] = linmod('ctrb_form')
A =
    -1    -2    -3
     1     0     0
     0     1     0
B =
     1
     0
     0
```

Coefficienti del polinomio caratteristico di A (e del denominatore della corrispondente FdT

Identità 2x2

```
C =
     4     5     6
D =
     0
```

Coefficienti del numeratore della corrispondente FdT

Simulink: esempio (FdA-2.1-FunzioniTrasferimento)

- **NOTA1**: il comando **`linmod()`** è utilizzabile con qualsiasi diagramma Simulink, anche (anzi soprattutto...) se quest'ultimo descrive un sistema nonlineare
- La quaterna di matrici A,B,C,D è infatti ottenuta tramite metodi numerici di linearizzazione approssimata nell'intorno di una condizione iniziale specificata (dal diagramma stesso o dai parametri opzionali del comando **`linmod()`**)

Simulink: esempio (FdA-2.1-FunzioniTrasferimento)

- **NOTA2**: la **forma canonica controllabile** è la stessa che si ottiene convertendo direttamente una Funzione di Trasferimento nell'equivalente modello nello spazio degli stati, tramite il comando **`tf2ss(num,den)`**:

```
>> G=tf([4 5 6],[1 1 2 3])
G =
    4 s^2 + 5 s + 6
   -------------------
   s^3 + s^2 + 2 s + 3
>> [A,B,C,D]=tf2ss([4 5 6],[1 1 2 3])
(vedi risultato in Matlab...)
```

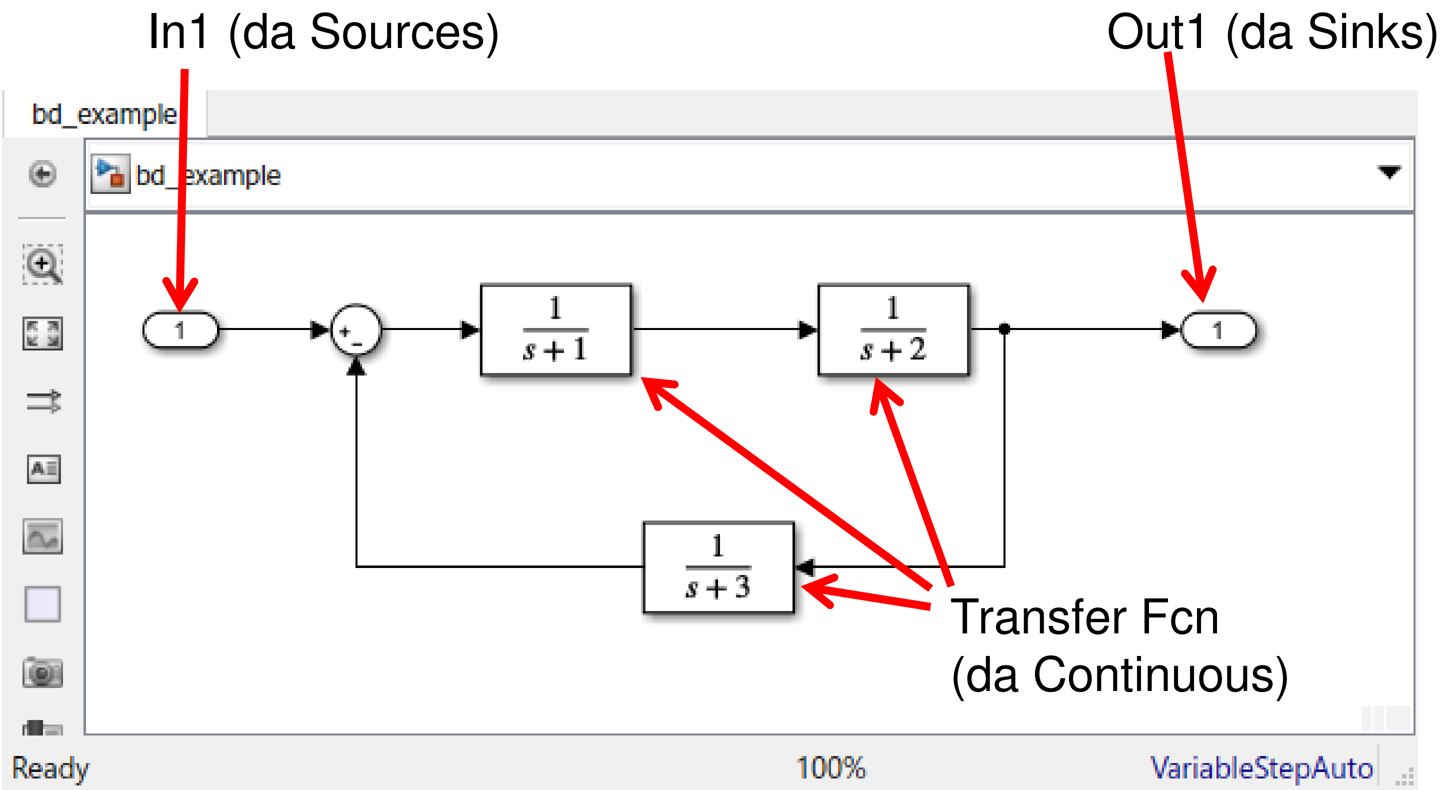
RIDUZIONE DI DIAGRAMMI A BLOCCHI:
- SIMULINK
- MATLAB

Simulink e la riduzione degli schemi a blocchi

- Le funzionalità di Simulink, sebbene molto sofisticate e complete, NON permettono di semplificare i diagrammi a blocchi di modelli LTI con FdT, applicando le regole grafiche mostrate in questo corso
- Al più, è possibile estrarre le matrici A,B,C,D per l'intero diagramma, come visto in precedenza, con **`linmod()`**, per poi costruire l'equivalente FdT con **`ss2tf()`** e **`tf()`** (oppure **`ss()`** + **`tf()`**)

Simulink e la riduzione degli schemi a blocchi

- Esempio:



Simulink e la riduzione degli schemi a blocchi

- Esempio:

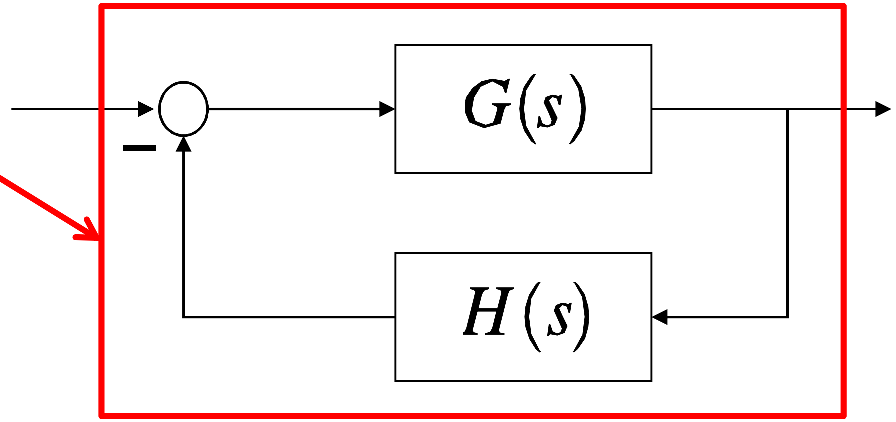
```
>> [A,B,C,D]=linmod('bd_example')
>> [Num,Den]=ss2tf(A,B,C,D)
>> G=tf(Num,Den)
G =
            s + 3
   ----------------------
   s^3 + 6 s^2 + 11 s + 7
```

Riduzione di schemi a blocchi in Matlab

- Il **Control Systems Toolbox** di Matlab permette di effettuare TUTTE le operazioni di connessione tra FdT richieste dai diagrammi a blocchi (LTI)
- Ovviamente, le FdT coinvolte devono essere opportunamente definite nel workspace, con coefficienti numerici e già assegnati (i.e. NO coefficienti simbolici)

Riduzione di schemi a blocchi (Control Sys. Tlbx)

```
>> Gcl = feedback(G,H)
```




```
>> Gp = parallel(G1,G2)
```

oppure (se parallelo di FdT SISO)

```
>> Gp = G1+G2
```

```
>> Gs = series(G1,G2)
```

oppure (se serie di FdT SISO)

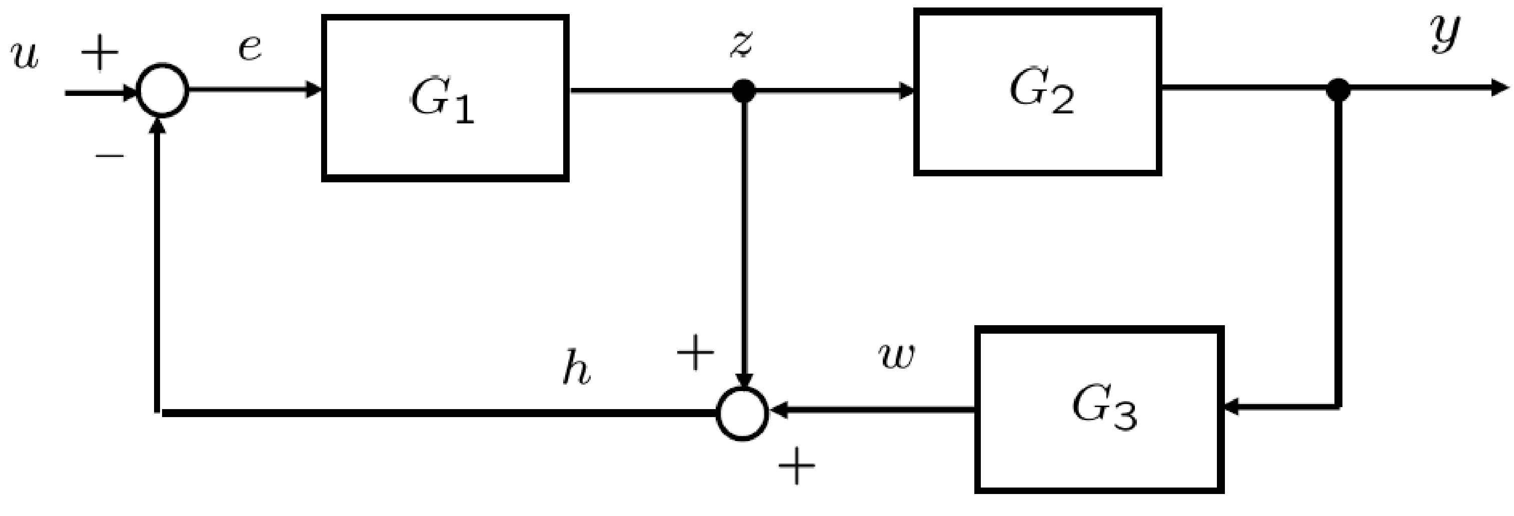
```
>> Gs = G1*G2
```

Riduzione di schemi a blocchi (Control Sys. Tlbx)

```
>> G1 = tf(1,[1 1])
>> G2 = tf(1,[1 2])
>> G3 = tf(1,[1 3])
>> Gcl = feedback(G1*G2,G3)
Gcl =
          s + 3
  ----------------------
  s^3 + 6 s^2 + 11 s + 7
```

Riduzione di schemi a blocchi in Matlab

- Interconnessioni più complesse richiedono l'uso delle funzioni **`connect()`** e **`sumblk()`**, oltre che una definizione più dettagliata di ingressi/uscite delle varie FdT, tramite **opportuna scelta di stringhe identificative**
- **Esempio** (con le G1,G2,G3 definite nella slide prec.):



Riduzione di schemi a blocchi (Control Sys. Tlbx)

```
>> G1.u = 'e'
>> G1.y = 'z'
>> G2.u = 'z'
>> G2.y = 'y'
>> G3.u = 'y'
>> G3.y = 'w'
>> S1 = sumblk('h=z+w')
>> S2 = sumblk('e=u-h')
>> sys=connect(G1,G2,G3,S1,S2,'u','y')
>> Gtot = tf(sys)
```

Riduzione di schemi a blocchi (Control Sys. Tlbx)

Risultato:

```
Gtot =

  From input "u" to output "y":
           s + 3
  -----------------------
  s^3 + 7 s^2 + 16 s + 13
```

NOTA: applicando le regole grafiche come descritto sulle dispense (FdA-2.1-FunzioniTrasferimento)

```
>> Gtot = feedback(G1*G2,G3+1/G2)
```

Riduzione di schemi a blocchi in Matlab

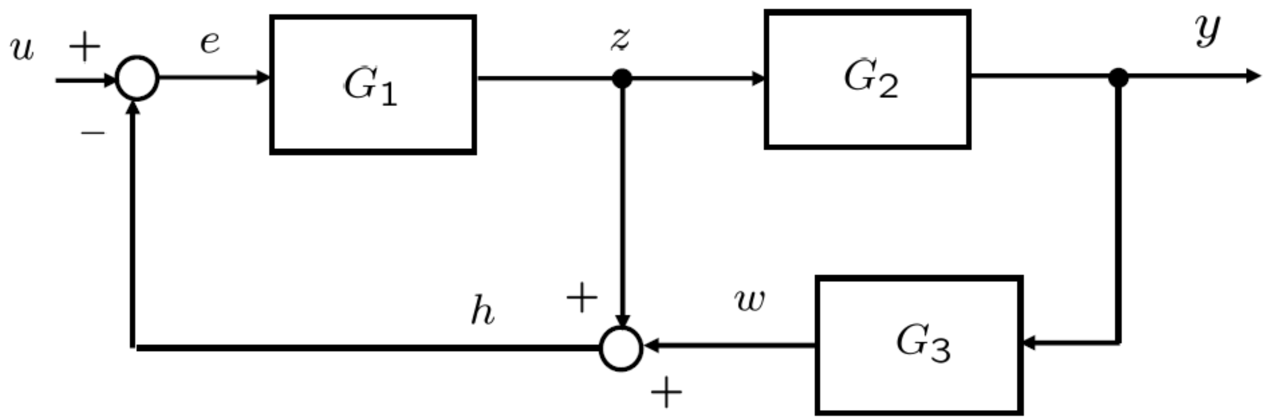
- Il **Symbolic Toolbox** di Matlab non è direttamente orientato alle operazioni di elaborazione dei diagrammi a blocchi
- TUTTAVIA, le operazioni di riduzione del diagramma, una volta “nominati” segnali e FdT coinvolte in modo simbolico, si riconducono a quelle di eliminazione delle variabili in un Sistema di equazione, risolvendolo rispetto a *u* ed *y* (i.e. ingresso e uscita)
- L’operazione può essere fatta sfruttando la funzione **`solve()`**

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

- **Passaggi:**
  1. Definire simbolicamente (**`syms`**) le FdT dei singoli blocchi e gli **N+1 segnali** nominati con lettere opportune (**inclusi *u* e *y***), costruire il vettore N x 1 contenente le equazioni corrispondenti ad ognuno degli N segnali (escluso l'ingresso)
  2. Chiamare **`solve()`** sul vettore di N equazioni, considerando come incognite le N variabili simboliche corrispondenti ai segnali (**escluso l'ingresso *u***)
  3. I campi della struttura risultante sono gli N segnali espressi in funzione dell'ingresso e delle FdT dei singoli blocchi
  4. Chiamare **`coeffs()`** sull'uscita y rispetto all'ingresso *u* per estrarre la FdT complessiva

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

- **Esempio 1** (v. FdA-2.1-FunzioniTrasferimento):



CON: e = u – h

z = G_1 e

y = G_2 z

h = z + w

w = G_3 y

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

```
>> syms G1 G2 G3
>> syms u y e z h w

>> sys = [e == u-h;
          z == G1*e;
          y == G2*z;
          h == z+w;
          w == G3*y]
```

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

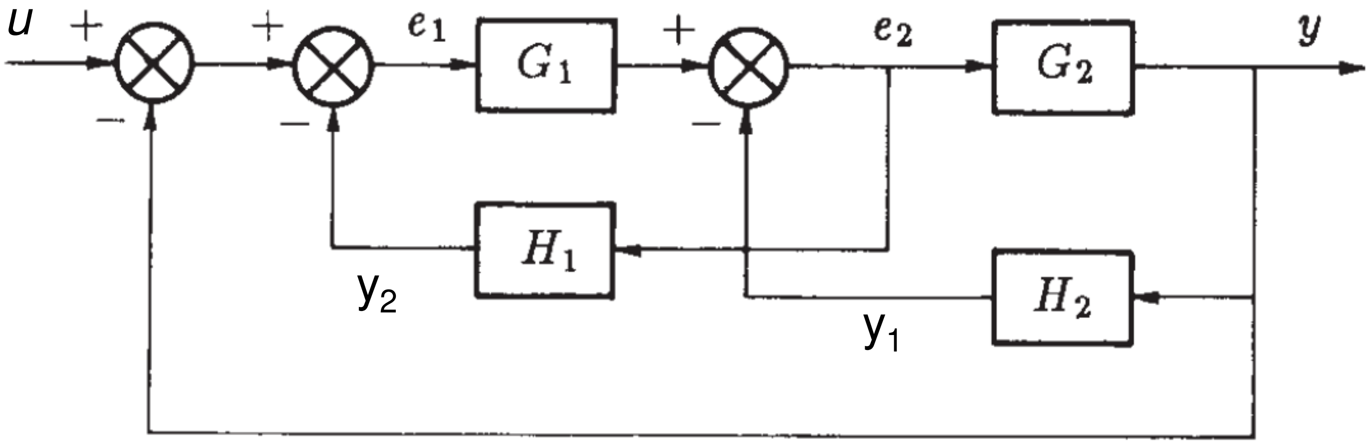
```
>> sol = solve(sys,[e,z,y,h,w]);
>> sol.y
ans =

(G1*G2*u)/(G1 + G1*G2*G3 + 1)

>> Gtot = coeffs(sol.y,u)
Gtot =
(G1*G2)/(G1 + G1*G2*G3 + 1)
```

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

- **Esempio 2** (v. FdA-2.1-FunzioniTrasferimento):



CON: $e_1 = u - y - y_2$

$e_2 = G_1\, e_1 - y_1$

$y_1 = H_2\, y$

$y_2 = H_1\, e_2$

$y = G_2\, e_2$

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

```
>> syms G1 G2 H1 H2
>> syms u y e1 e2 y1 y2

>> sys = [e1 == u-y-y2;
          e2 == G1*e1-y1;
          y1 == H2*y;
          y2 == H1*e2;
          y == G2*e2]
```

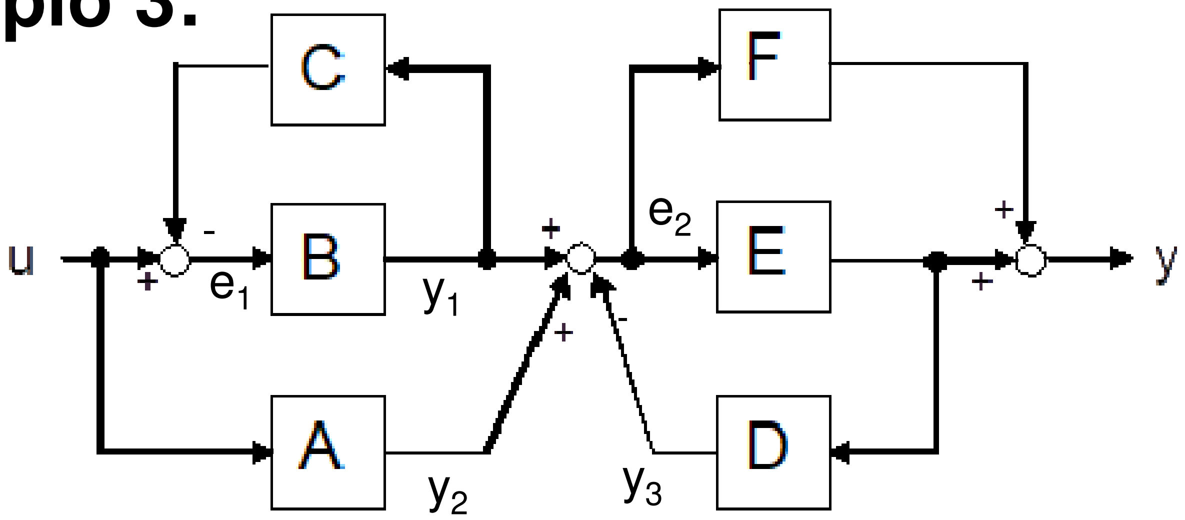
Riduzione di diagrammi a blocchi (Symbolic Tlbx)

```
>> sol = solve(sys,[e1,e2,y1,y2,y]);
>> sol.y
ans =
(G1*G2*u)/(G1*G2 + G1*H1 + G2*H2 + 1)

>> Gtot = coeffs(sol.y,u)
Gtot =
(G1*G2)/(G1*G2 + G1*H1 + G2*H2 + 1)
```

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

- **Esempio 3:**



CON: $e_1 = u - C\,y_1$

$e_2 = y_1 + y_2 - y_3$

$y_1 = B\,e_1$

$y_2 = A\,u$

$y_3 = D\,E\,e_2$

$y = (E + F)\,e_2$

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

```
>> syms A B C D E F
>> syms u y e1 e2 y1 y2 y3

>> sys = [e1 == u-C*y1;
          e2 == y1+y2-y3;
          y1 == B*e1;
          y2 == A*u;
          y3 == D*E*e2;
          y == (E+F)*e2]
```

Riduzione di diagrammi a blocchi (Symbolic Tlbx)

```
>> sol = solve(sys,[e1,e2,y1,y2,y3,y]);
>> sol.y
ans =
(u*(E + F)*(A + B + A*B*C))/((B*C + 1)*(D*E +
1))

>> Gtot = coeffs(sol.y,u)

Gtot =
((E + F)*(A + B + A*B*C))/((B*C + 1)*(D*E + 1))
```

RISPOSTA DI FdT E DIAGRAMMI A BLOCCHI IN MATLAB

FINE