

Introduzione a Matlab



FONDAMENTI DI AUTOMATICA

A cosa serve questa presentazione



- **Scopi di questo materiale:**
 - fornire le informazioni necessarie per l'uso di Matlab e Simulink in relazione ai Laboratori di Fondamenti di Automatica;
 - dare una panoramica generale (tutt'altro che esauriente) delle potenzialita' di Matlab per la formulazione e la soluzione di problemi numerici nell'Ingegneria.

Dove trovare altre informazioni?



- Sito web di Mathworks:

www.mathworks.com

seguendo i link alla voce "support" e' possibile trovare i manuali di Matlab in formato pdf.

(<http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml>)

- Un testo in italiano di introduzione a matlab e Simulink:

Guida Operativa a MATLAB, SIMULINK e Control Toolbox

Alberto Cavallo, Roberto Setola, & Francesco Vasca

Liguori Editore, 1994

Indice del materiale



- Descrizione generale di Matlab (v. 5.3)
- Quadro delle funzioni predefinite
- Definizione di matrici e vettori
- Definizione di polinomi
- Rappresentazione di sistemi dinamici lineari
- Analisi di sistemi di controllo
- Rappresentazione grafica dei dati
- L'ambiente di simulazione Simulink

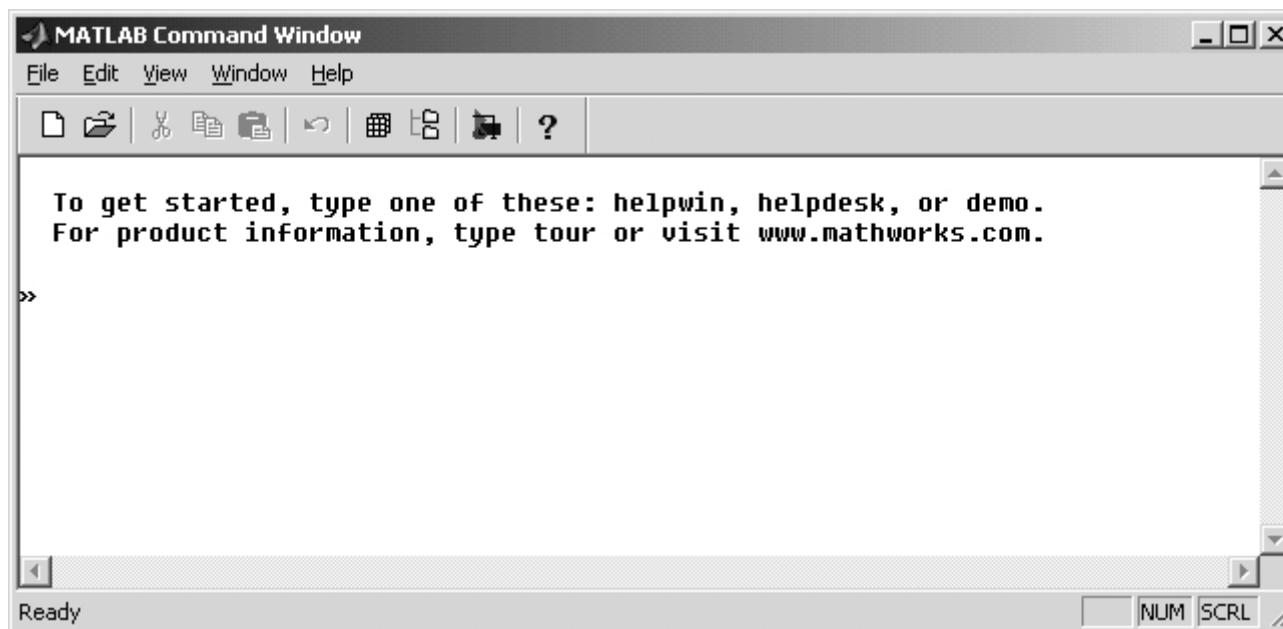
Descrizione generale di Matlab



- MATLAB (= MATrix LABoratory):
 - un linguaggio di programmazione per applicazioni scientifiche e numeriche
 - vasto set di funzioni predefinite
 - interprete di comandi
 - possibilita' di scrivere nuove funzioni
 - libreria di TOOLBOX per svariate applicazioni; ad es. (Signal Processing, Analisi e sintesi di controllori,...).

L'interfaccia di Matlab

- Interfaccia utente: la Command Window da' accesso diretto all'interprete (scrittura diretta di comandi).



Matlab come calcolatrice...

- La modalita' di impiego piu' "semplice": per valutare espressioni numeriche.
- Esempio: per calcolare $4 + \sqrt{2} - \sin(0.2\pi)^2 + e^2$
e' sufficiente digitare al prompt »
» *4 + sqrt(2) - sin(0.2*pi)^2 + exp(2)*
ans =
12.4578
- Il risultato viene scritto nella variabile *ans*.

Definizione di variabili



- E' possibile definire variabili e espressioni non numeriche piu' complesse.

- Esempio:

 - » a=4; b=2;

 - » a*b

 - ans =

 - 8

- Per cancellare una variabile (es. a):

 - » clear a

II Workspace

- Ogni variabile definita in questo modo viene conservata in memoria, nel Workspace.
- Il comando *whos* mostra una lista delle variabili definite:

» whos

Name	Size	Bytes	Class
a	1x1	8	double array
ans	1x1	8	double array
b	1x1	8	double array

Grand total is 3 elements using 24 bytes

Letture e scrittura su file



Mediante i comandi *load* e *save* e' possibile salvare su file le variabili del workspace.

- *load* nomefile variabile1 variabile2 ...
carica dal file nomefile.mat le variabili elencate.
- *save* nomefile variabile1 variabile2 ...
scrive nel file nomefile.mat le variabili elencate.
- *load* nomefile carica tutte le variabili in nomefile.
- *Save* nomefile salva tutto il workspace in nomefile.

Quindi...



- Esiste un insieme (molto vasto) di funzioni predefinite (come *sin* e *sqrt* nell'esempio precedente).
- A differenza dei normali linguaggi (C, Pascal...) non occorre *dichiarare* le variabili. L'assegnazione coincide con la dichiarazione.

Esempi di funzioni predefinite (di uso piu' comune)

- Funzioni trigonometriche (*sin, cos, tan, acos, asin, atan...*);
- Esponenziale e logaritmo (*exp, log, log10, sqrt...*);
- Numeri complessi (*abs* -> modulo, *angle* -> fase, *real* -> parte reale, *imag* -> parte immaginaria...);

Alcuni esempi semplici

■ Calcolare il modulo di $2+3i$:

» $\text{abs}(2+3*i)$

ans =

3.6056

■ Calcolare $20 \log_{10} \left(\left| \frac{2+3i}{4+6i} \right| \right)$

» $20*\log_{10}(\text{abs}((2+3*i)/(4+6*i)))$

ans =

-6.0206

Inf e NaN



- Alcune operazioni numeriche possono dare luogo a problemi, che vengono segnalati da Matlab scrivendo come risultato le variabili Inf e NaN.
- Esempi:

» 5/0

Warning: Divide by zero.

ans =
Inf

» 0/0

Warning: Divide by zero.

ans =
NaN

Una funzione fondamentale!



help

- `help` seguito dal nome di una funzione restituisce una descrizione e la sintassi d'uso della medesima;
- `help "da solo"` restituisce l'elenco di TUTTE le funzioni di Matlab, ordinate per categorie.

Definizione di matrici

- Come si definisce una matrice in Matlab?

Esempio: definire la matrice 2x2 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

```
» A=[1,2;3,4]
```

```
A =
```

```
1 2
```

```
3 4
```

- Come si accede agli elementi di una matrice:

```
» A(1,2)
```

```
ans =
```

```
2
```

**Indici (riga e colonna)
dell'elemento di interesse**

La wildcard :

■ Per accedere a intere righe o colonne di una matrice, si usa la wildcard :

■ Es.: selezionare la prima riga di A

» $A(1,:)$

ans =

1 2

■ Es.: selezionare la seconda colonna di A

» $A(:,2)$

ans =

2

4

Selezionare sottomatrici

■ Se definiamo

» $B = [1,2,3;4,5,6]$

B =

1	2	3
4	5	6

■ Abbiamo che

» $B(1:2, 2:3)$

ans =

2	3
5	6

Indici della sottomatrice di interesse

Operazioni (elementari) sulle matrici

■ Sono definiti gli operatori $+$, $-$, $*$ e $^{\wedge}$.

■ Matrice trasposta:

» A'

ans =

1 3

2 4

■ Matrice inversa:

» $\text{inv}(A)$

ans =

-2.0000 1.0000

1.5000 -0.5000

Operazioni (elementari) sulle matrici (2)

■ Determinante:

» $\det(A)$

ans =

-2

■ Autovalori:

» $\text{eig}(A)$

ans =

-0.3723

5.3723

Altre operazioni



- Osservazione importante: NON occorre definire le dimensioni in modo esplicito!
Per conoscere le dimensioni di una matrice: *size*.
- Altre operazioni:
 - *rank* -> calcolo del rango di una matrice
 - *trace* -> calcolo della traccia di una matrice
 - *norm* -> calcolo della norma di una matrice

Alcune matrici “speciali”

- *eye*(n,n) -> matrice identita' nxn;
- *zeros*(n,m) -> matrice di zeri nxm;
- *ones*(n,m) -> matrice di uni nxm;
- *rand*(n,m) -> matrice nxm con elementi distribuiti uniformemente tra 0 e 1.

Vettori



- I vettori hanno due funzioni fondamentali in Matlab:
 - rappresentazione dei polinomi (un polinomio e' descritto dal vettore dei suoi coefficienti);
 - rappresentazione di segnali (un segnale e' rappresentato mediante la sequenza dei valori che assume in un insieme di istanti di tempo, quindi mediante un vettore).

Definizione di vettori (1)

■ » $v=(0:10)$

$v =$

0 1 2 3 4 5 6 7 8 9 10

■ » $v=(1:0.5:3)$

$v =$

1.0000 1.5000 2.0000 2.5000 3.0000

Valore iniziale

Passo

Valore finale

Definizione di vettori (2)

■ Come matrici riga o colonna:

$$\gg v = [3 \ 6 \ 1 \ 7]$$

v =

$$\begin{matrix} 3 & 6 & 1 & 7 \end{matrix}$$

■ Polinomi: sono rappresentati come vettori.

$$\text{Es.: } 3s^2 + 2s + 1$$

$$\gg \text{pol} = [3 \ 2 \ 1]$$

pol =

$$\begin{matrix} 3 & 2 & 1 \end{matrix}$$

Operazioni sui polinomi

■ Calcolo delle radici -> *roots*

» `roots(pol)`

ans =

-0.3333 + 0.4714i

-0.3333 - 0.4714i

■ Valutazione in un punto -> *polyval*

» `polyval(pol,0)`

ans =

1

Operazioni sui polinomi (2)

■ Prodotto di polinomi \rightarrow *conv*

Esempio: $(s + 1)(s + 1) = s^2 + 2s + 1$

» `pol1=[1 1];pol2=[1 1];`

» `polprod=conv(pol1,pol2)`

`polprod =`

`1 2 1`

Sistemi dinamici lineari



- Un sistema dinamico lineare invariante puo' essere descritto:
 - In forma di variabili di stato mediante quattro matrici A, B, C, D ;
 - In forma di funzione di trasferimento, mediante i due polinomi $N(s)$ e $D(s)$.
- Quindi in Matlab e' possibile definire i sistemi lineari come *oggetti* a partire da entrambe le descrizioni.

Definizione di sistemi lineari (a tempo continuo)

- Dalla forma di stato
 - Definire le matrici A, B, C, D nel workspace;
 - Definire il sistema mediante il comando *ss*.

- Dalla funzione di trasferimento
 - Definire i polinomi num e den (numeratore e denominatore della f. di t.) nel workspace;
 - Definire il sistema mediante il comando *tf*.

Esempi (1)

■ Definizione del sistema:

$$\dot{x} = -x + 3u$$

$$y = 4x + 2u$$

» A=-1;B=3;C=4;D=2;

» sistema=ss(A,B,C,D)

a =

	x1
x1	-1

b =

	u1
x1	3

c =

	x1
y1	4

d =

	u1
y1	2

Continuous-time model.

Esempi (2)

■ Definizione del sistema $G(s) = \frac{s + 1}{s^2 + 3s + 16}$

```
» num=[1 1]; den=[1 3 16];  
» sistema=tf(num,den)
```

Transfer function:

$s + 1$

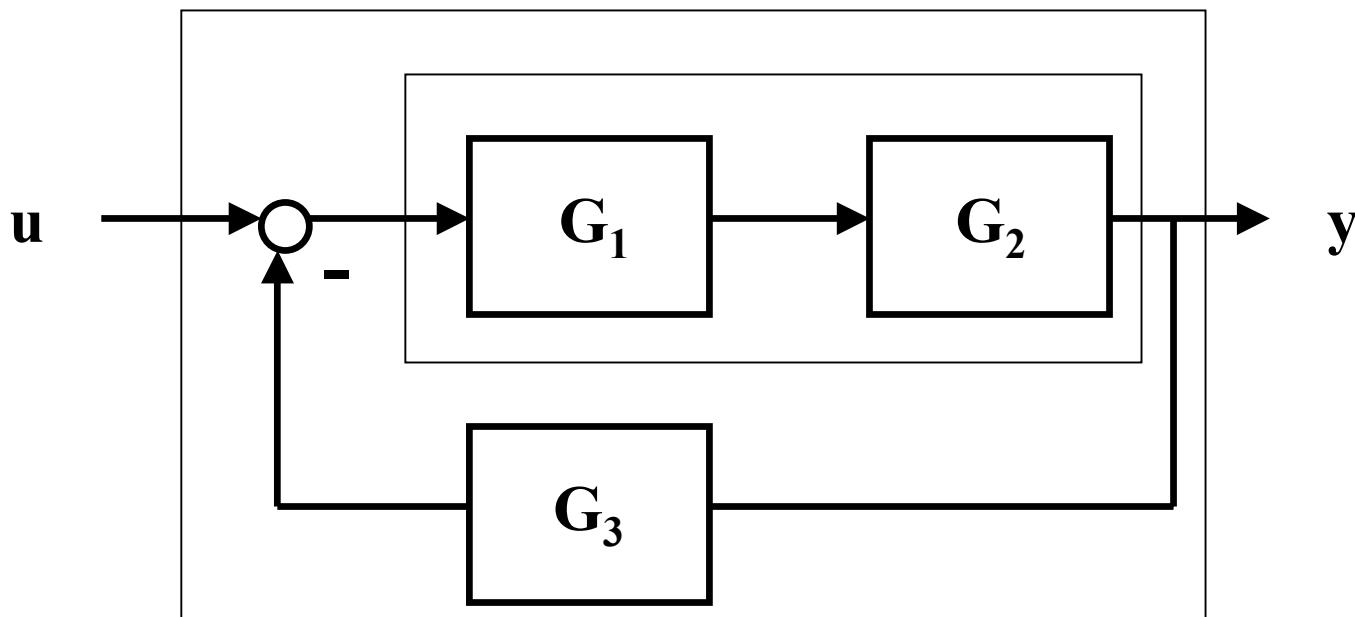
$s^2 + 3s + 16$

Interconnessione di sistemi



- Agli *oggetti* sistemi lineari si applicano i normali operatori $+$, $*$, $/$ con il seguente significato:
 - $+$ connessione in parallelo;
 - $*$ connessione in serie;
 - $/$ usato per definire l'interconnessione in retroazione.

Esempio di connessione



$andata = g1 * g2;$

$retroazione = andata / (1 + andata * g3)$

Simulazione di sistemi lineari

- Funzioni disponibili per la simulazione:
 - *impulse* -> simulazione risposta all'impulso;
 - *step* -> simulazione risposta a scalino;
 - *initial* -> simulazione movimento libero;
 - *lsim* -> simulazione con ingresso qualsiasi e stato iniziale qualsiasi.

■ Sintassi:

» $[y,t]=\text{step}(\text{система});$

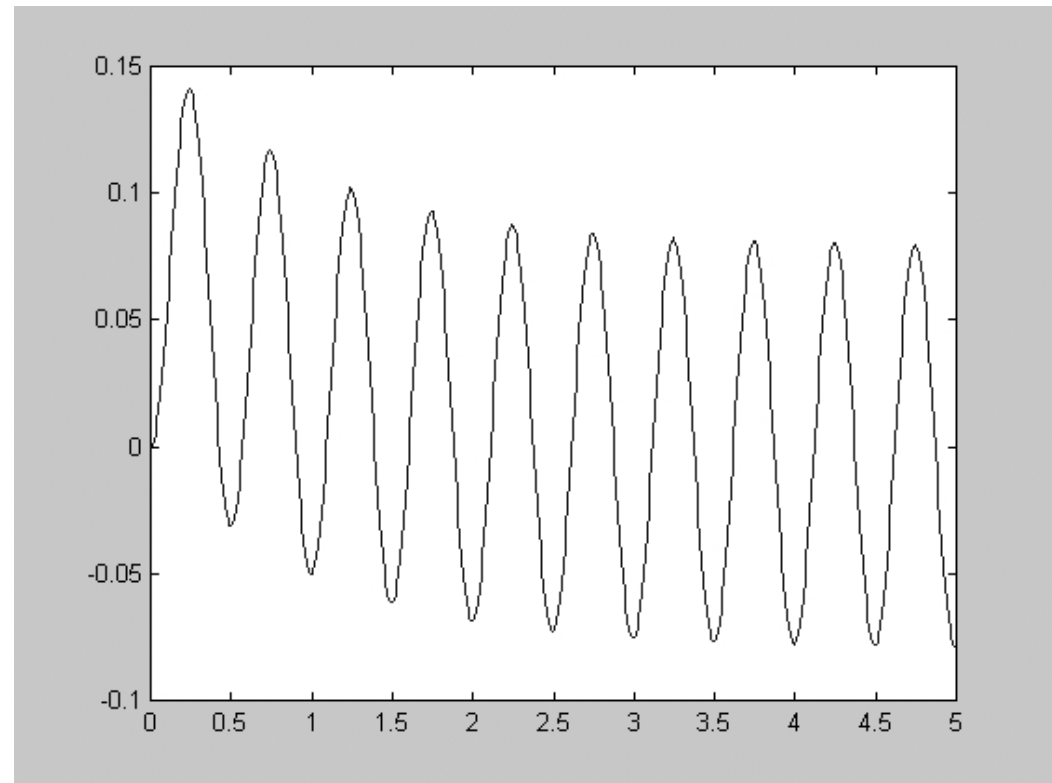
» $[y,x]=\text{lsim}(\text{система}, u, t);$

Vettore sequenza ingresso

Vettore dei tempi

Esempio

```
» sistema=tf(1,[1 1]);  
» t=(0:0.01:5);  
» u=sin(2*pi*2*t);  
» y=lsim(sistema,u,t);  
» plot(t,y)
```

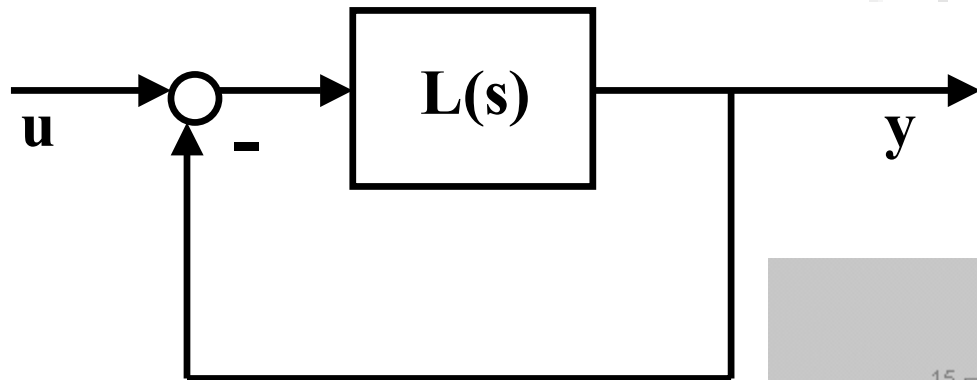


- Chiamando le funzioni senza output si ottiene direttamente il plot.

Analisi di sistemi di controllo

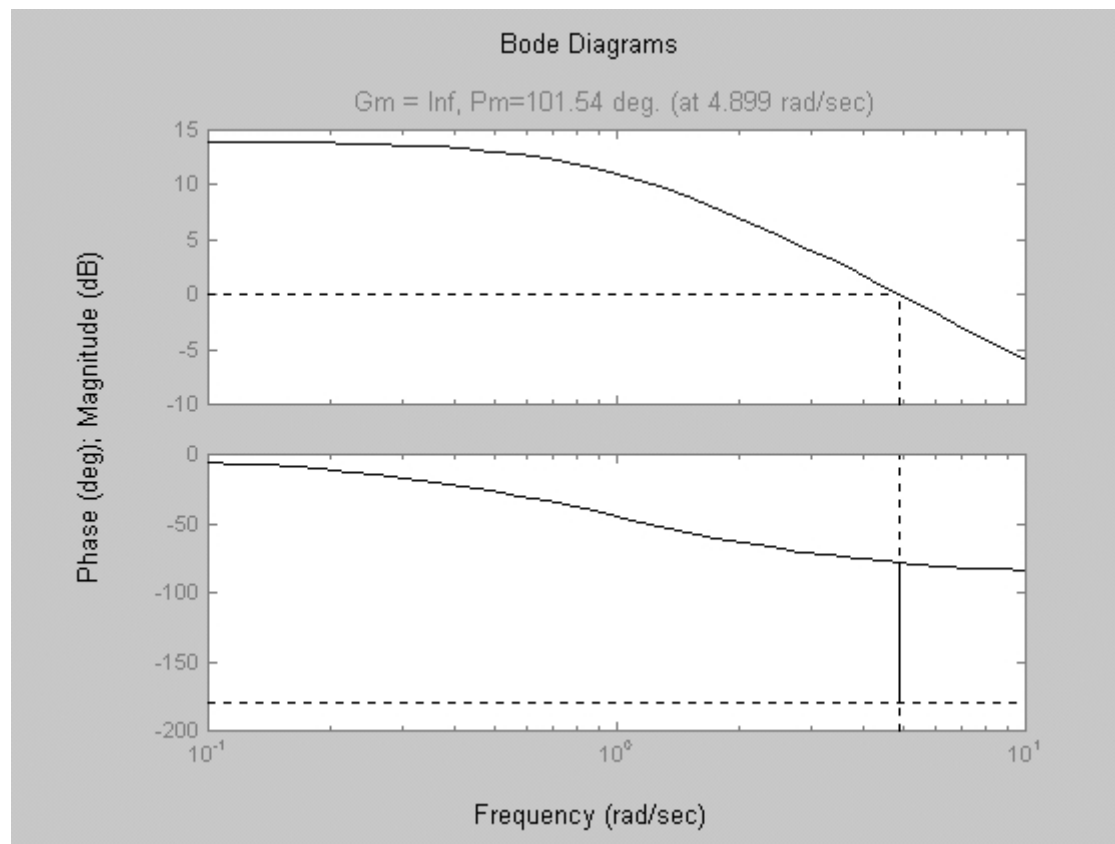
- Per i problemi di controllo lineari invarianti SISO esistono le seguenti funzioni:
 - $bode(sistema)$ -> tracciamento diagrammi di Bode della risposta in frequenza;
 - $margin(sistema)$ -> come $bode$ ma in piu' calcola pulsazione critica, margine di fase e margine di guadagno;
 - $nyquist(sistema)$ -> tracciamento diagramma di Nyquist della risposta in frequenza;
 - $rlocus(sistema)$ -> tracciamento luogo delle radici;

Esempi (1)

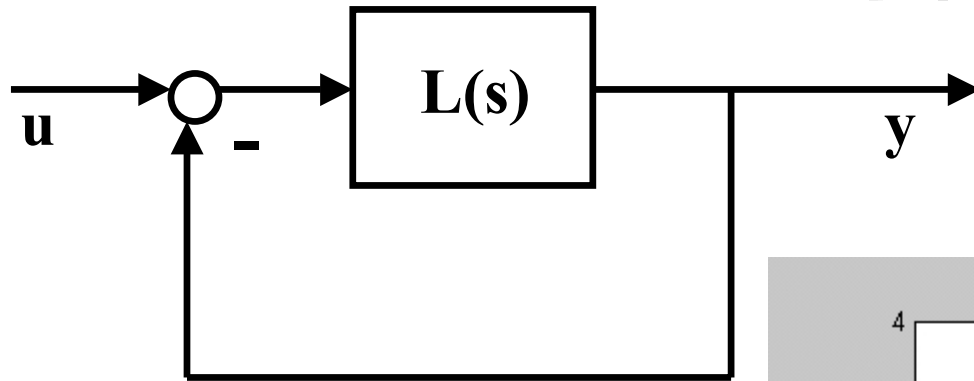


» `L=tf(5,[1 1]);`

» `margin(L)`

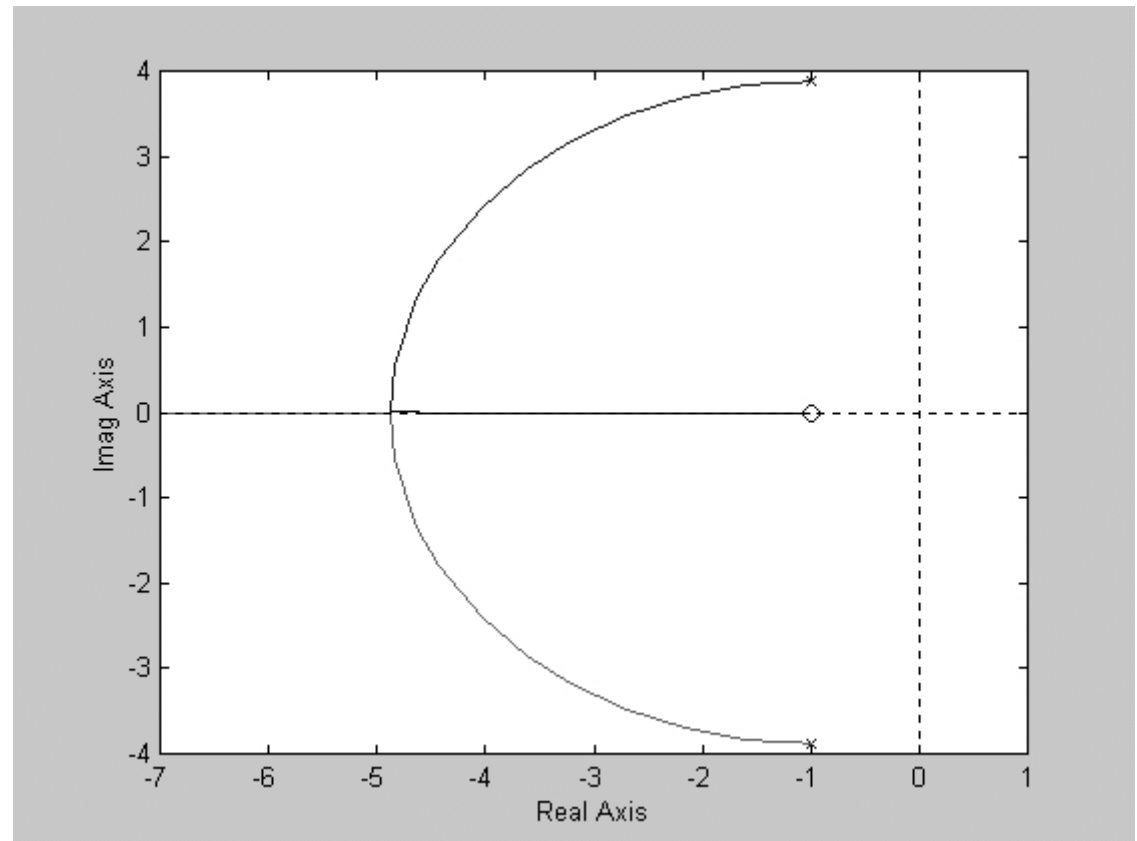


Esempi (2)



» `L=tf([1 1],[1 2 16]);`

» `rlocus(L)`



Rappresentazione grafica

■ Grafici 2D:

- In scala lineare -> *plot*

`plot(x,y)` traccia il grafico dei punti che hanno come ascisse (ordinate) gli elementi del vettore x (y).

- In scala semilogaritmica o logaritmica ->

semilogx, *semilogy*, *loglog*

stessa sintassi di *plot*

- Diagrammi polari -> *polar*

Rappresentazione grafica (2)

■ Altre funzioni utili:

- cambiamenti di scala -> `axis([xmin,xmax,ymin,ymax])`
- sovrapposizione di piu' plot -> `hold`
- aggiunta di grigliatura al plot -> `grid`
- titolo e etichette agli assi -> `title('..')`,
`xlabel('..')`, `ylabel('..')`
- piu' grafici in una finestra -> `subplot`
- inserimento testo in una figura -> `gtext`

Rappresentazione grafica



- Grafici 3D, animazioni, rendering:
vedere i manuali di Matlab!

L'ambiente Simulink



- Simulink: un ambiente grafico per la simulazione di sistemi complessi.
- Perché non basta Matlab?
 - E' spesso necessario simulare sistemi complessi, composti da numerosi blocchi interconnessi tra loro;
 - Spesso i singoli blocchi sono nonlineari o tempo-varianti;
 - Può essere necessario integrare blocchi continui e discreti.

Principio di funzionamento

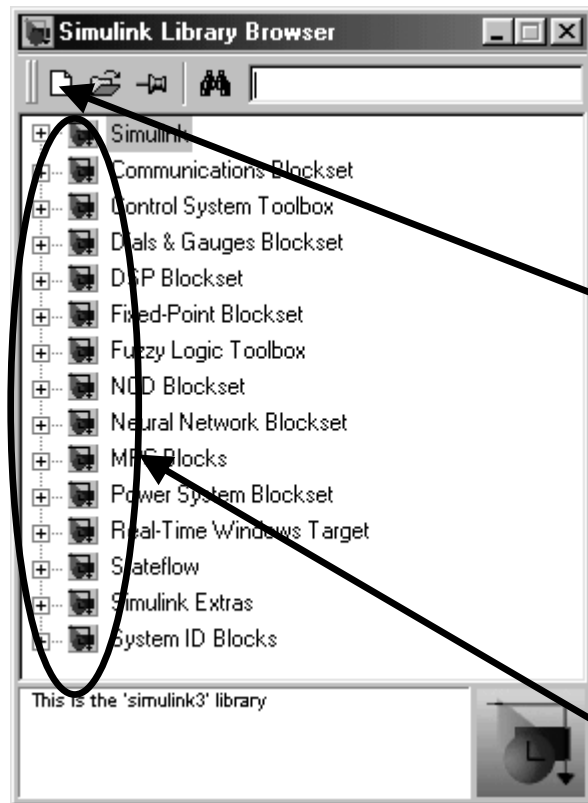


- Simulink contiene una *libreria di blocchi* che descrivono elementi statici e dinamici elementari;
- L'utente compone lo schema a blocchi del sistema da simulare mediante l'interconnessione dei blocchetti elementari;
- Simulink genera *automaticamente* le equazioni e risolve il problema numerico di simulazione desiderato.

Principio di funzionamento (2)

- Simulink interagisce con Matlab attraverso il Workspace \Rightarrow i modelli Simulink possono contenere variabili del Workspace;
- Allo stesso modo il risultato delle simulazioni puo' essere esportato nel Workspace e analizzato con Matlab.

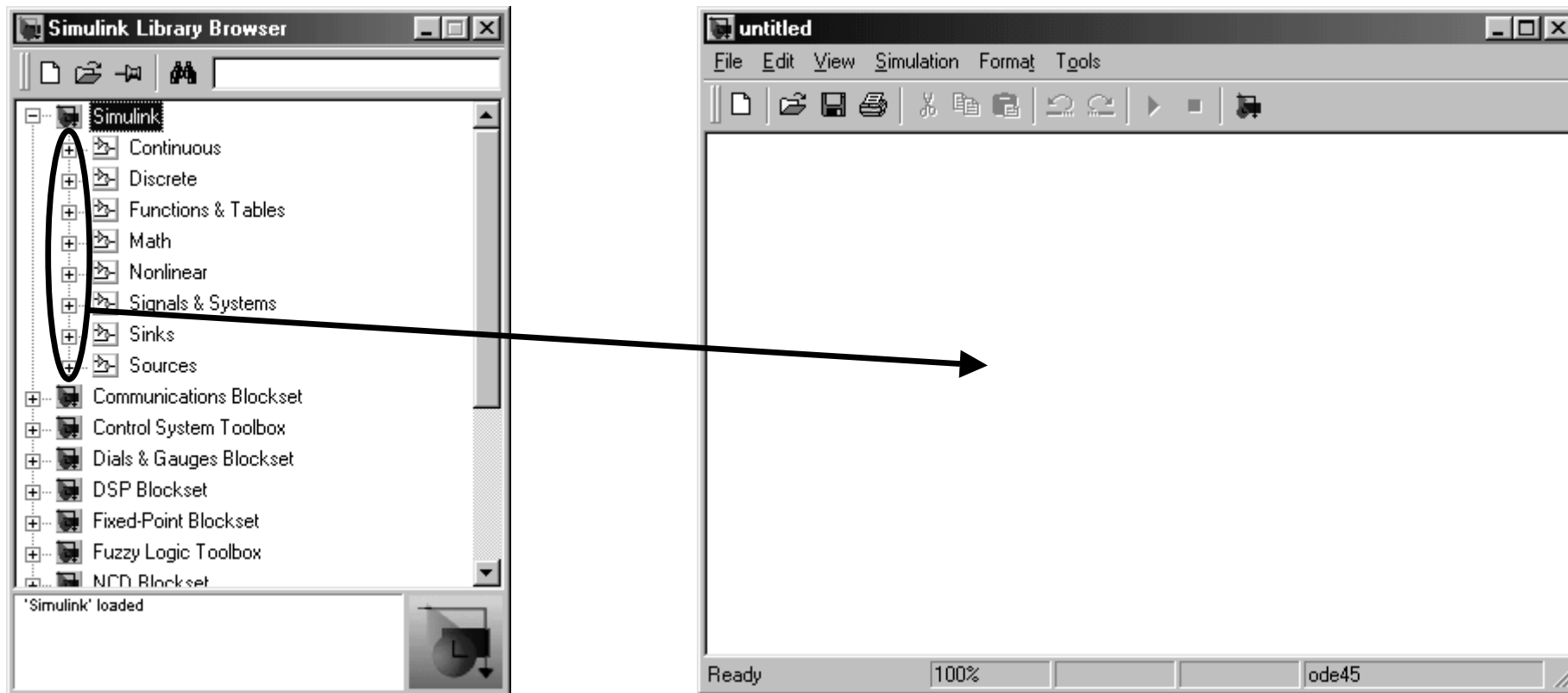
L'interfaccia grafica



- Digitando 'simulink' al Matlab prompt si apre la libreria dei modelli.

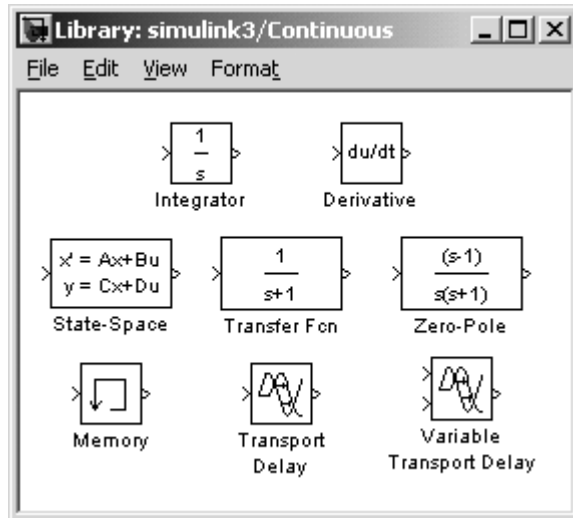
- Da qui e' possibile creare un nuovo modello (foglio bianco) e comporre il sistema da simulare mediante i diversi blocchi.

Nuovo modello

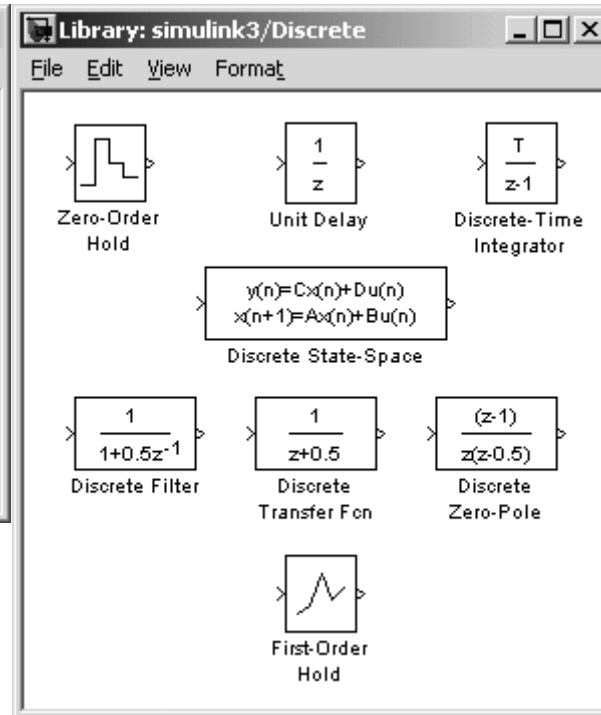


Il menu' 'Simulink' contiene la maggior parte dei blocchi che useremo.

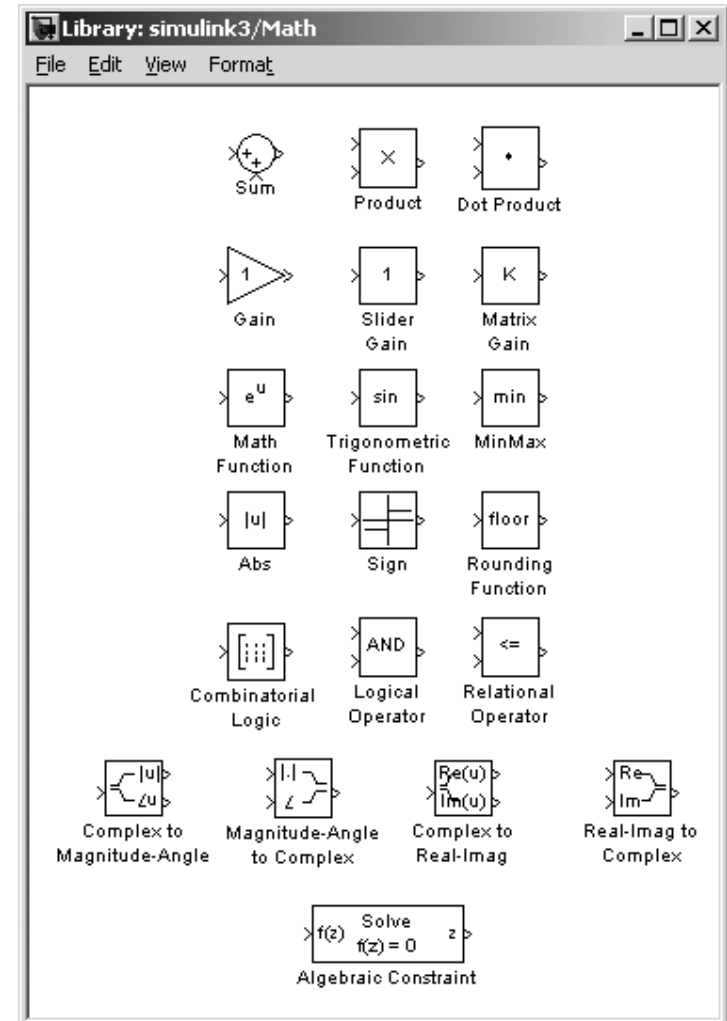
Principali librerie Simulink



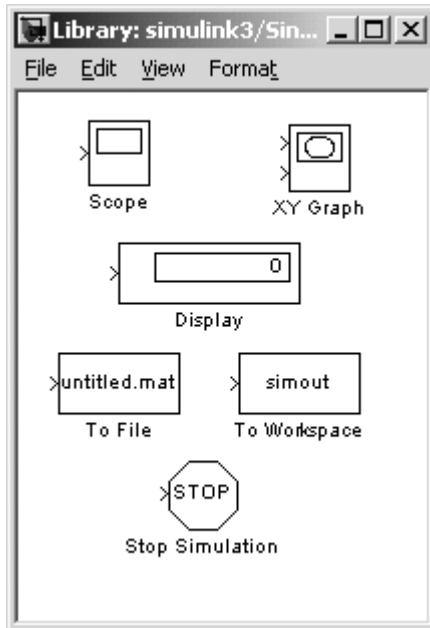
**Blocchi dinamici
a tempo continuo
(Continuous)**



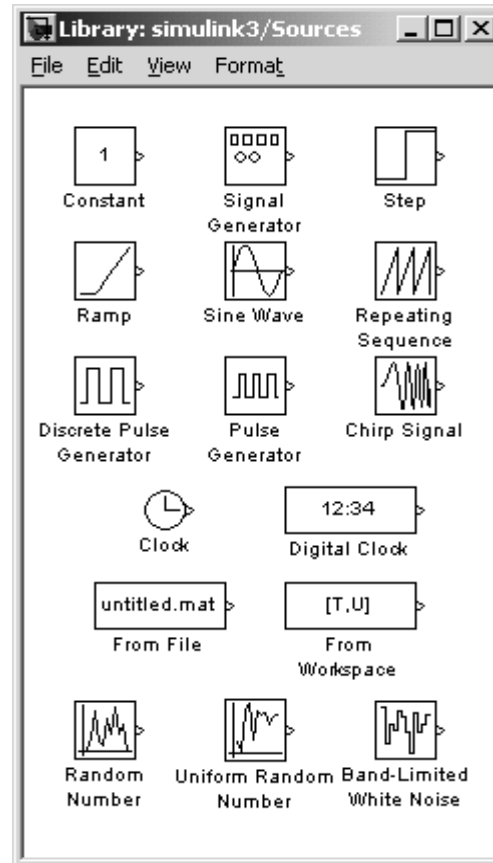
**Blocchi dinamici
a tempo discreto
(Discrete)**



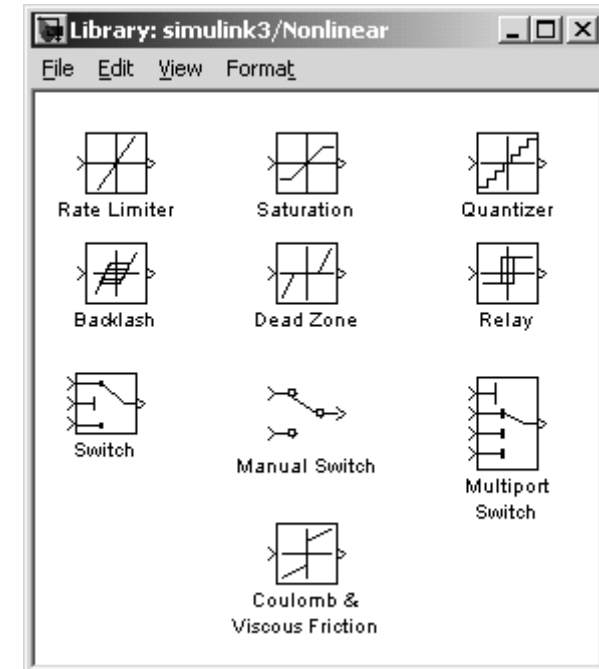
Principali librerie Simulink (2)



**Output dati
(Sinks)**



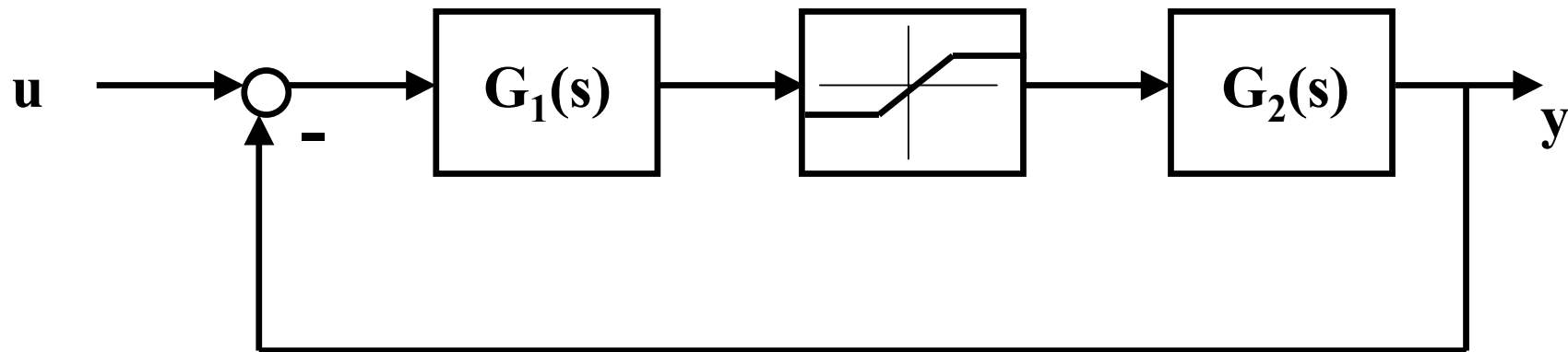
**Segnali di ingresso
(Sources)**



**Blocchi non lineari
(Nonlinear)**

Esempio

- Vogliamo simulare con Simulink il seguente sistema di controllo che contiene una non linearita':



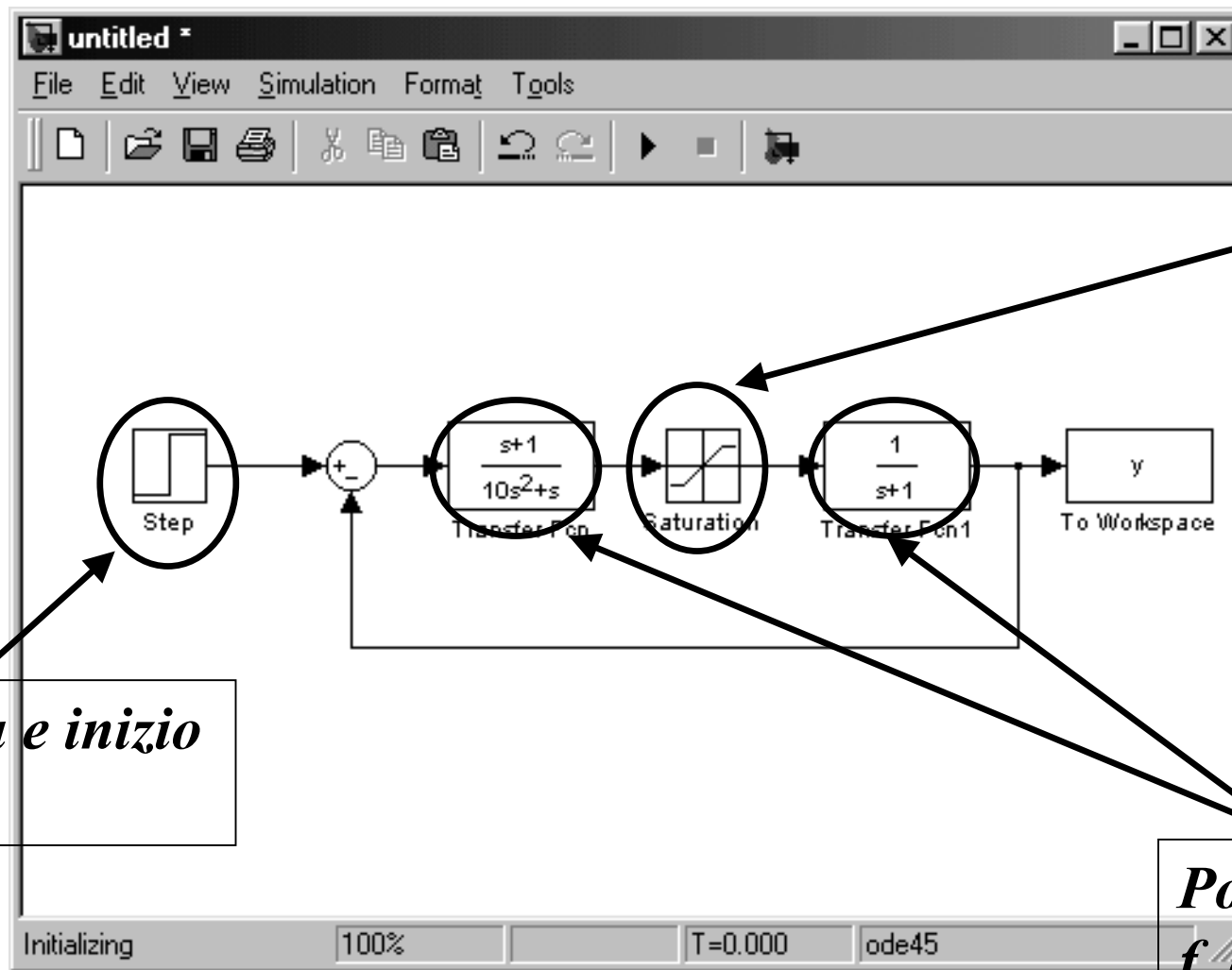
nel caso in cui $u(t) = \text{sca}(t)$.

Blocchi da usare



- Blocco 'Transfer Function', menu' 'Continuous';
- Blocco 'Saturation', menu' 'Nonlinear';
- Blocco 'Sum', menu' 'Continuous';
- Blocco 'Step', menu' 'Sources';
- Blocco 'To Workspace', menu' 'Sinks';
- Le operazioni da eseguire sono:
 - Trascinare ciascuno dei blocchi nella finestra del modello;
 - Connetterli come nello schema a blocchi di partenza;
 - Occorre infine definire i valori dei parametri di ciascun blocco.

Modello e parametri

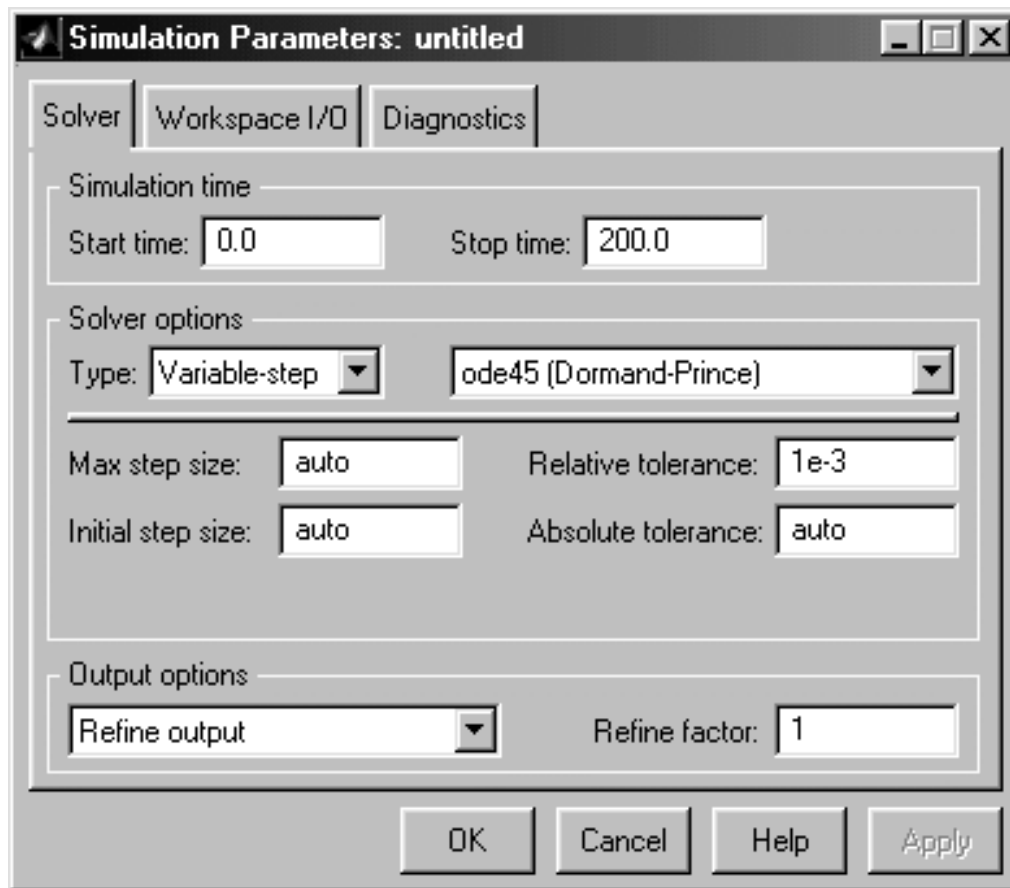


Ampiezza e inizio scalino

Ampiezza saturaz.

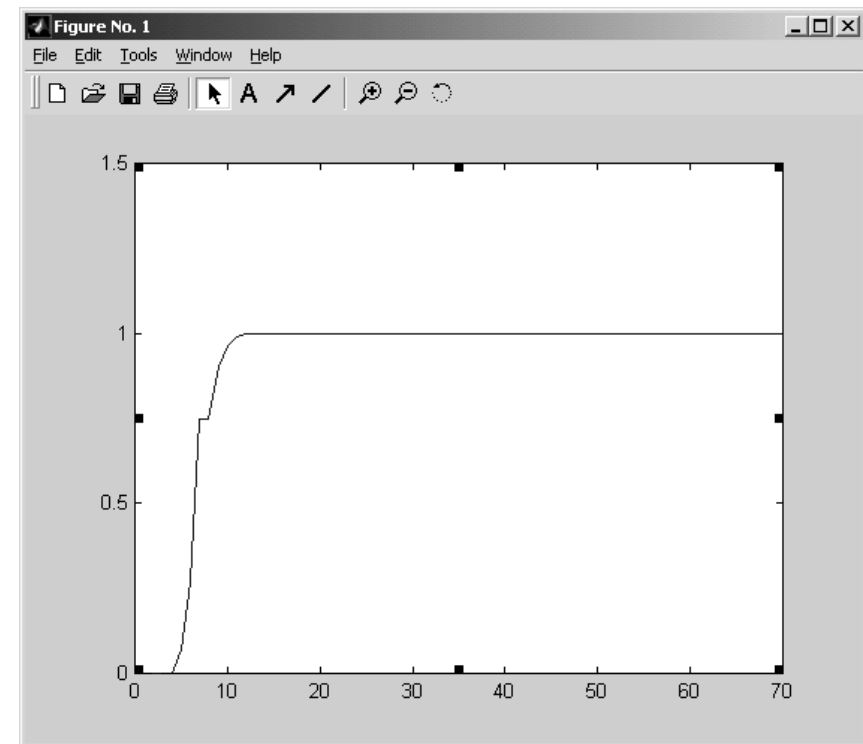
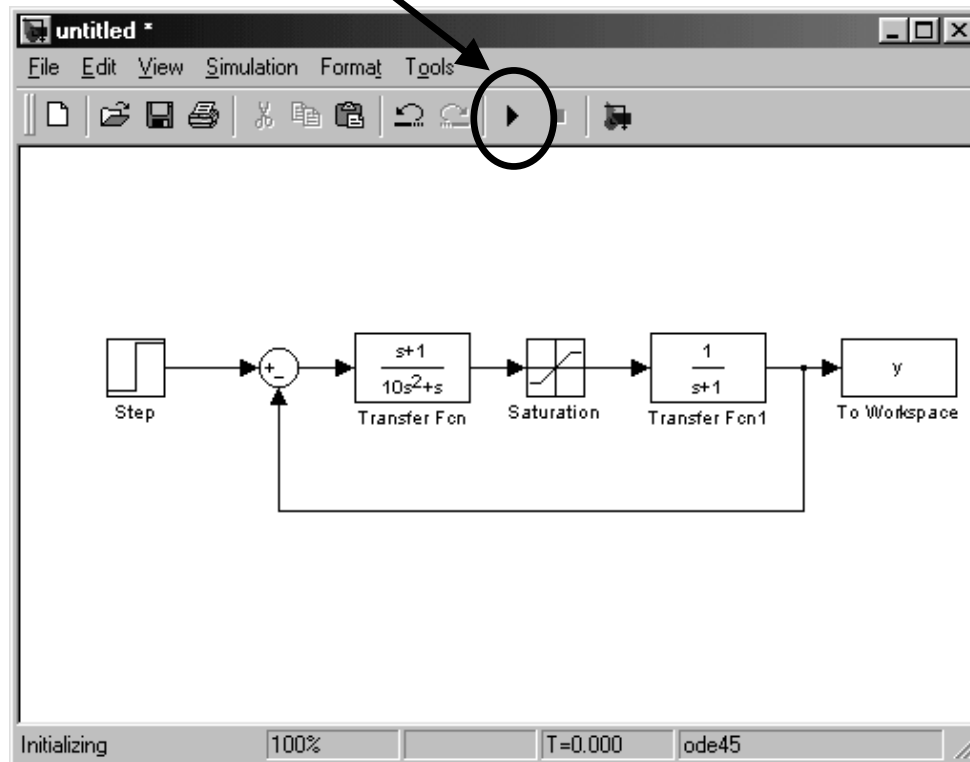
Polinomi f. di t.

Parametri della simulazione



- L'utente deve definire:
 - Istanti di inizio e fine della simulazione;
 - Tipo di solutore numerico (se il problema richiede metodi particolari);
 - Parametri del solutore (in genere i default vanno bene...).

Avvio simulazione e analisi risultati



plot(y) al prompt di Matlab consente di visualizzare il risultato della simulazione.