

Utilizzo di MATLAB per l'analisi di sistemi dinamici

**Sistemi dinamici a tempo continuo
ed a tempo discreto**

Indice del materiale

- Breve **introduzione** a MATLAB
 - Descrizione generale di MATLAB
 - Quadro delle funzioni predefinite
 - Definizione di matrici, vettori e polinomi
 - Rappresentazione grafica dei dati
- **Analisi e simulazione di sistemi dinamici LTI** in ambiente MATLAB
 - Rappresentazione di sistemi dinamici lineari tempo-invarianti nell'ambiente MATLAB
 - Analisi e simulazione di sistemi dinamici lineari tempo-invarianti

Indice del materiale (2)

- **Sistemi lineari interconnessi**
 - Sistemi dinamici lineari interconnessi in ambiente MATLAB
 - Utilizzo dell'istruzione MATLAB *feedback*
- Determinazione della **risposta in frequenza** di sistemi dinamici lineari tempo-invarianti nell' ambiente MATLAB
 - Esempi di utilizzo
- Conversione da **tempo continuo** a **segnali campionati/tempo discreto** e viceversa.

Breve introduzione a MATLAB

**L' ambiente di lavoro,
le istruzioni fondamentali**

A cosa serve questa presentazione

- Scopi di questo materiale:
 - fornire le informazioni necessarie per l'uso di **MATLAB** in relazione alle esercitazioni del corso;
 - dare una panoramica generale (**tutt' altro che esauriente**) delle potenzialità 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 è possibile trovare la documentazione dei vari toolbox
(<https://it.mathworks.com/help/matlab/index.html>)

- 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  in biblioteca

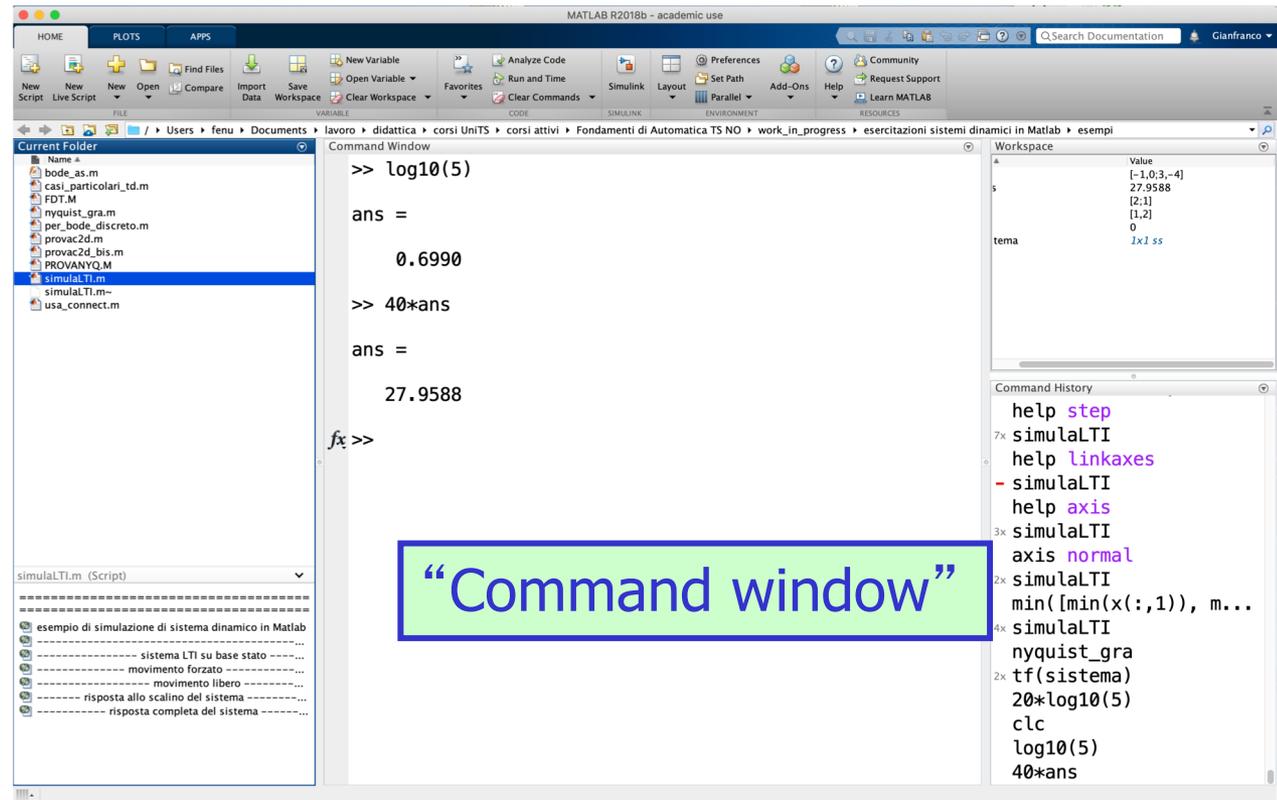
Descrizione generale di MATLAB

- MATLAB (= MATrix LABoratory):
 - un **linguaggio di programmazione** per applicazioni scientifiche e numeriche
 - vasto set di **funzioni predefinite**
 - **interprete** di comandi
 - possibilità di scrivere nuove funzioni
 - libreria di **TOOLBOX** per svariate applicazioni; ad es. Signal Processing, Identificazione di modelli, ...

L' interfaccia di MATLAB

• Interfaccia utente:

- la **Command Window** dà accesso diretto all'interprete
- scrittura diretta di comandi.



Interfaccia standard:

The screenshot displays the MATLAB R2018b - academic use interface. The main window is the Editor, showing a script named `simulaLTI.m` with the following code:

```

9  %-----
10
11 %-----
12 % un po' di pulizia
13 close all
14 clear all
15
16 %----- sistema LTI su base stato -----
17 % definizione del sistema dinamico su base stato
18 a = [-1 , 0 ; 3 , -4];

```

The interface includes several key components:

- Current Folder:** Located on the left, it shows a list of files in the current directory, including `bode_as.m`, `casi_particolari_td.m`, `FDT.M`, `nyquist_gra.m`, `per_bode_discreto.m`, `provac2d.m`, `provac2d_bis.m`, `PROVANYQ.M`, `simulaLTI.m`, `simulaLTI.m~`, and `usa_connect.m`.
- Workspace:** Located on the right, it displays the current workspace variables and their values:

Name	Value
a	[-1,0;3,-4]
A	[1,2;3,4]
ans	[5,-6]
b	1x20 double
c	1x1 Line
d	1x20 double
F	1x20 double
Hp	1x1 Line
i	20
sistema	1x1 ss
t	1x501 double
- Command Window:** Located at the bottom, it shows the execution of the command `b = [1;2;3;4;5;6]`, resulting in the output:


```

>> b = [1;2;3;4;5;6]

b =

     1
     2
     3
     4
     5
     6

```
- Command History:** Located on the right, it shows the sequence of commands entered in the Command Window, including `F(1) = 0; F(2)...`, `% inizializzaz...`, `for i = 3:20`, `F(i) = F(i-1) ...`, `e`, `%`, `%`, `Hp = plot(1:15)...`, `hold on, xla...`, `plot([0 20], (...`, `legend([Hp], '...`, `clc`, and `b = [1;2;3;4;5;6]`.

- **Command Window:** è la finestra dell'interprete di comandi di MATLAB. Vi si possono scrivere direttamente comandi o lanciare programmi scritti nel linguaggio di programmazione del MATLAB.
- **Command History:** è la finestra che contiene l'elenco dei comandi inseriti nella Command Window.
- **Current Directory:** visualizza in forma grafica il contenuto della cartella di lavoro.
- **Workspace:** è la finestra che visualizza il contenuto dell'area di memoria utilizzata dal MATLAB (nomi, tipi e dimensioni delle variabili).
- **Editor:** permette di creare file di testo (M-File: estensione .m) contenenti script e/o function realizzati utilizzando il linguaggio di programmazione di MATLAB.

MATLAB come calcolatrice...

- La modalità di impiego più “semplice”: per valutare espressioni numeriche.
- Esempio: per calcolare $4 + \sqrt{2} - \sin(0.2\pi)^2 + e^2$ è sufficiente digitare al prompt »

» $4 + \text{sqrt}(2) - \sin(0.2*\text{pi})^2 + \text{exp}(2)$

ans =

12.4578

- Il risultato viene salvato nella variabile *ans*.

Definizione di variabili

- È possibile definire variabili e espressioni non numeriche più complesse.
- Esempio:
 - » **a=4; b=2;**
 - » **a*b**
 - ans =**
 - 8**
- Per cancellare una variabile (es. a):
 - » **clear a**

Il 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*** è 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.mat*

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.

Operazioni matematiche elementari

- Moltiplicazione $ab \iff a * b$
- Divisione (divisione “a destra”) $\frac{a}{b} \iff a / b$
- Divisione “a sinistra” $\frac{b}{a} \iff a \setminus b$
- Elevamento a potenza $a^b \iff a ^ b$
- Addizione e sottrazione $\begin{matrix} a + b \\ a - b \end{matrix} \iff \begin{cases} a + b \\ a - b \end{cases}$

Operazioni elementari: precedenza

- La precedenza nello svolgere le operazioni matematiche elementari in MATLAB è quella standard, facendo però attenzione alle operazioni di “divisione a destra” e “divisione a sinistra”.
- Infatti valgono le regole:
 - L’ **elevamento a potenza** viene valutato prima delle operazioni di **moltiplicazione** e **divisione**, che hanno la medesima priorità,
 - L’operazione di “**divisione a destra**” ha priorità rispetto alla “**divisione a sinistra**”;
 - Le operazioni di **addizione** e **sottrazione** hanno la priorità più bassa.
 - Per assegnare una diversa precedenza nella sequenza dei calcoli è necessario **utilizzare le parentesi**.

L'operatore di assegnazione “=”

- L'operatore “=” è utilizzato in MATLAB per **assegnare** un valore (numerico, come il risultato di un'operazione di calcolo, oppure testuale o simbolico [ne parliamo più avanti]) ad una variabile.
- Quindi è appropriato pensare all'operatore “=” come ad un'istruzione con la quale si assegna un valore ad una variabile, in un qualsiasi linguaggio di programmazione (C, Fortran ecc.).
- Se l'assegnazione termina con un “;” allora il risultato dell'operazione di assegnazione non viene visualizzato, altrimenti si.

```
» x = 2; y = 4; z = x*y  
z = 8
```

L'operatore di assegnazione "=" (2)

- Nel caso di assegnazioni lunghe si può utilizzare il simbolo "..." per poi proseguire l'assegnazione nella riga seguente

» **FirstClassHolders = 72;**

» **Coach = 121;**

» **Crew = 8;**

» **TotalPeopleOnPlane = FirstClassHolders + Coach...
+ Crew**

**TotalPeopleOnPlane =
201**

Il formato di visualizzazione

- Il risultato di un'operazione numerica viene visualizzato di solito tramite un numero reale, con sole 4 cifre decimali.
- Questo è il formato di visualizzazione standard.
- Esistono altre possibilità e si scelgono tramite il comando **format**:
 - **format short** 4 cifre decimali
 - **short e** 4 cifre decimali, notazione esponenziale
 - **format long** 15 cifre decimali
 - **long e** 15 cifre decimali, notazione esponenziale
 - ...

Il formato di visualizzazione

- **format rat** risultato approssimato tramite la più vicina frazione
- **format bank** formato finanziario, con sole due cifre decimali
- ...

» **format long**

» **x = 3 + 11/16 + 2^1.2**
x = 5.98489670999407

» **format short**

» **x = 3 + 11/16 + 2^1.2**
x = 5.9849

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* \Rightarrow modulo, *angle* \Rightarrow fase, *real* \Rightarrow parte reale, *imag* \Rightarrow parte immaginaria...);

Alcuni esempi semplici

- Calcolare il modulo di $(2+3i)$:

» **abs(2+3*i)**
ans =
3.6056

Unità immaginaria

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

» **20*log10(abs((2+3*i) / (4+6*i)))**
ans =
-6.0206

Inf e NaN

- Alcune operazioni numeriche possono dare luogo a risultati non corretti, esprimibili soltanto facendo uso di “forme indeterminate”, che vengono definite da MATLAB tramite le grandezze **Inf** e **NaN**.
- Esempi:

» **5/0**

Warning: Divide by zero.

**ans =
Inf**

» **0/0**

Warning: Divide by zero.

**ans =
NaN**

Help! Una funzione fondamentale!

- **help** per vedere la lista dei toolbox installati
- **help *nome_toolbox*** per vedere la lista dei comandi installati in un *toolbox*
- **help *nome_comando*** guida 'on-line' di MATLAB sullo specifico comando
- **ver** info sulla versione di MATLAB
- **helpwin** finestra di help di MATLAB

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
```

Sottomatrice di interesse

- scrivendo

```
» B(1:2, 2:3)
```

```
ans =
```

```
  2   3
  5   6
```

Indici della sottomatrice di interesse

Operazioni (elementari) sulle matrici

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

```
» A = [ 1 2; 3 4 ]; B = [ 5 6; 7 8 ]
```

```
» A+B
```

```
ans =
```

```
6 8
```

```
10 12
```

```
» B-A
```

```
ans =
```

```
4 4
```

```
4 4
```

```
» A*B
```

```
ans =
```

```
19 22
```

```
43 50
```

```
» A^2
```

```
ans =
```

```
7 10
```

```
15 22
```

Operazioni (elementari) sulle matrici (2)

- Sono definiti gli operatori `.*`, `./` e `.^`, che si applicano elemento per elemento:

```
» A = [ 1 2; 3 4 ]; B = [ 5 6; 7 8 ];
```

```
» A.*B
```

```
ans =
```

```
    5    12
```

```
   21    32
```

```
» A.^B
```

```
ans =
```

```
    1    64
```

```
  2187  65536
```

Operazioni (elementari) sulle matrici (3)

- Determinante:

```
» det(A)
ans =
    -2
```

- Autovalori:

```
» eig(A)
ans =
   -0.3723
    5.3723
```

- Matrice inversa:

```
» inv(A)
ans =
   -2.0000    1.0000
    1.5000   -0.5000
```

- Matrice trasposta:

```
» A'
ans =
    1    3
    2    4
```

Operazioni (elementari)

- Funzioni matematiche di base **help elfun**
- Funzioni matematiche specifiche **help specfun**
- Funzioni di manipolazione delle matrici **help elmat**

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)** \Rightarrow matrice identità $n \times n$
- **zeros(n,m)** \Rightarrow matrice di “0” $n \times m$
- **ones(n,m)** \Rightarrow matrice di “1” $n \times m$
- **rand(n,m)** \Rightarrow matrice $n \times m$ con elementi pseudocasuali con distribuzione di probabilità uniforme nell'intervallo $[0 \ 1]$.

Vettori

- I vettori in MATLAB possono:
 - rappresentare **polinomi** (un polinomio è descritto dal vettore dei suoi coefficienti);
 - rappresentare un **segnale di durata finita** (mediante la sequenza dei valori che assume in un insieme finito di istanti di tempo, quindi mediante un vettore);
 - rappresentare (ovviamente) **vettori** in **spazi vettoriali**

Definizione di sequenza di valori

- `>> t=(0:10)`

t =

0 1 2 3 4 5 6 7 8 9 10

- `>> t=(1 0.5:3)`

t =

1.0000 1.5000 2.0000 2.5000 3.0000

Valore finale

Passo

Valore iniziale

$$y = \sin(2*t - \pi/3)$$

Definizione di vettori

- Come matrici riga o colonna:

» **v = [3 6 1 7]**

v =

3 6 1 7

» **b = [1;2;3;4;5;6]**

b =

**1
2
3
4
5
6**

Risoluzione di un problema matriciale

- Si vuole risolvere il problema $Ax = b$
 - con A matrice quadrata, invertibile $x = A^{-1}b$
 - e se A non invertibile? Soluzione «ai minimi quadrati»
 - In MATLAB

$$\mathbf{A} = [1 \ 2; \ 3 \ 4]$$

$$\mathbf{b} = [5; 6]$$

$$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$$

Definizione di polinomi

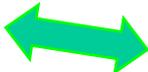
- Polinomi: sono rappresentati come vettori

Es.:

» **pol = [3 2 1]**

pol =

3 2 1

 $p(z) = 3z^2 + 2z + 1$

Operazioni sui polinomi

- Calcolo delle radici \Rightarrow *roots*

» `roots(pol)`

`ans =`

`-0.3333 + 0.4714i`

`-0.3333 - 0.4714i`

- Valutazione in un punto \Rightarrow *polyval*

» `polyval(pol, 0)`

`ans =`

`1`

Operazioni sui polinomi (2)

- Prodotto di polinomi \Rightarrow ***conv***

Esempio:

$$(z + 1)(z + 1) = z^2 + 2z + 1$$

» **pol1 = [1 1]; pol2 = [1 1];**

» **polprod = conv(pol1, pol2)**

polprod =

1 2 1

Rappresentazione grafica

- Grafici 2D:
 - In **scala lineare** \Rightarrow ***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** \Rightarrow ***semilogx, semilogy, loglog***
stessa sintassi di *plot*
 - Diagrammi **polari** \Rightarrow ***polar***

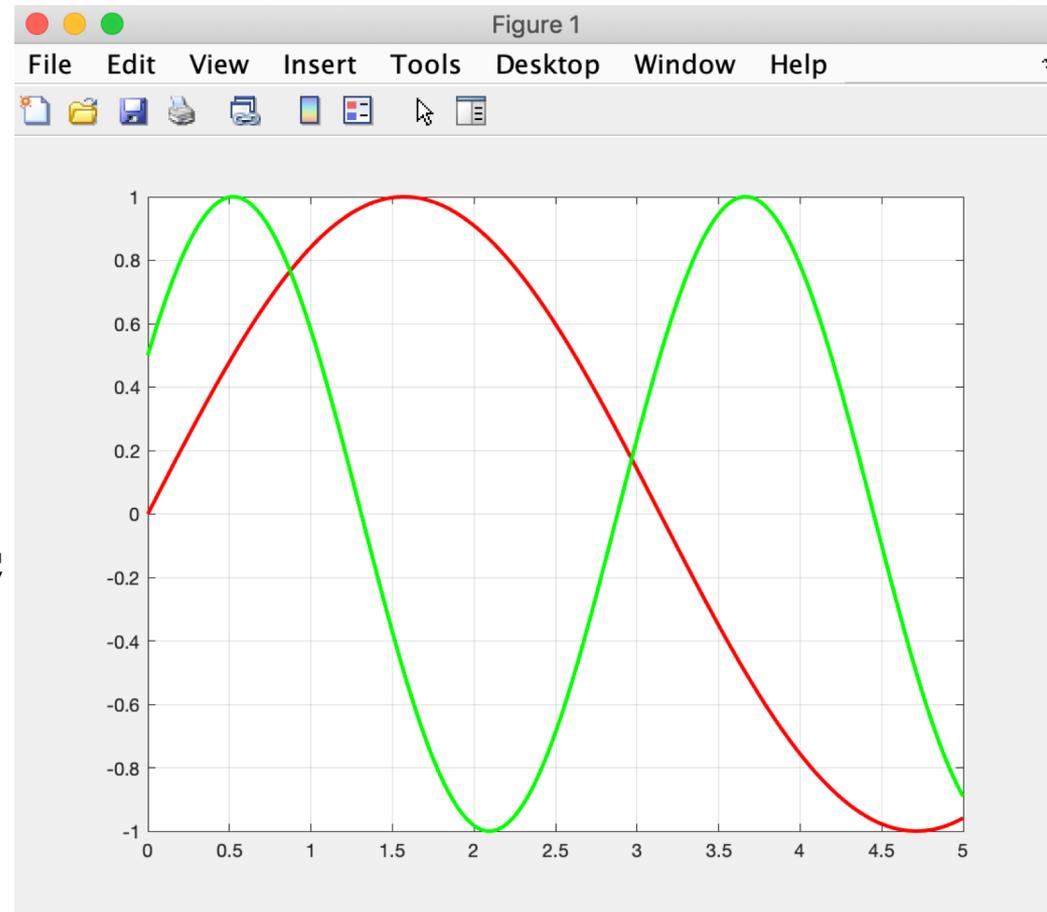
Rappresentazione grafica (2)

- Altre funzioni utili:
 - cambiamenti di scala \Rightarrow ***axis***([xmin,xmax,ymin,ymax])
 - sovrapposizione di più grafici \Rightarrow ***hold***
 - aggiunta di griglia al grafico \Rightarrow ***grid***
 - titolo al grafico ed etichette agli assi \Rightarrow ***title***('..'), ***xlabel***('..'), ***ylabel***('..')
 - più grafici in una finestra \Rightarrow ***subplot***
 - inserimento di testo in una figura \Rightarrow ***gtext***

Esempio

- Grafici sovrapposti

```
» t=(0:0.01:5);  
» y1=sin(t);  
» y2=cos(2*t-pi/3);  
» figure;  
» plot(t,y1,'r',t,y2,'g');  
» grid on;
```



Un tipo di dato “speciale”: *cell array*

- Il tipo di dato “cell array” sta ad indicare che si vuole creare un oggetto di tipo **array** i cui **elementi** possono essere **di tipo diverso** (valori numerici, testo, altri tipi di dati).
- Operativamente, un dato di tipo “cell array” si crea in modo simile agli array “classici” in MATLAB, semplicemente sostituendo ai delimitatori classici [] i delimitatori { }.
- Esempio: per creare un cell array 2-x-2

A = {[1 4 3; 0 5 8; 7 2 9], 'Paolino Paperino';3+7i, -pi:pi/4:pi};

Cenni alla programmazione in ambiente MATLAB

Operazioni con vettori, caratteri e testo

Cicli, operazioni e condizioni logiche

Funzioni e struttura di un programma in MATLAB

Operazioni sui vettori e matrici

- I vettori (e le matrici) possono essere manipolati in MATLAB non solo per estrarre parti di essi, ma anche per
 - Concatenare più vettori (matrici) assieme
 - » **t1 = [1 :10];**
 - » **t2 = [20:-1:11];**
 - » **t3=[t1, t2];**
 - Modificare le dimensioni di un vettore (matrice)
 - » **help reshape**
 - » **t3bis = reshape(t3,2,10);**

- Cancellare/sostituire elementi in un vettore (matrice)

```
» disp(t3bis);  
» t3bis(:,4:6)=[];  
» disp(t3bis);  
» whos
```

```
» disp(t3bis)  
» t3bis(1,4:6)=-3;  
» whos
```

Operazioni su caratteri e testo

- Le stringhe di caratteri sono viste come particolari array in MATLAB, quindi su di esse possono essere applicate le operazioni di manipolazione sui vettori già descritte:

```
» t1 = 'A'
```

```
» t2 = 'BCDE'
```

```
» t3 = [t1,t2]
```

```
» t4 = [t3,' are the first 5 '];...
```

```
'characters in the alphabet.']
```

- A volte e' necessario convertire un numero in testo e viceversa: vedere le istruzioni **str2num**, **num2str**, **int2str** e simili.

Cicli, operazioni e condizioni logiche

- Operazioni cicliche:
 - for** indice = inizio:passo:fine
 - fai_qualcosa;
 - end**

- Operazioni logiche, condizioni logiche elementari:
 - “vero” e “falso”: **true = 1 false = 0**
 - Confronti logici:
 - **x == 2** x e' uguale a 2?
 - **x ~= 2** x NON e' uguale a 2?
 - **x > 2** x e' maggiore di 2?
 - **x < 2** x e' minore di 2?
 - **x >= 2** x e' maggiore oppure uguale a 2?
 - **x <= 2** x e' minore oppure uguale a 2?

- Attenzione a questo esempio:

```
» x = pi
```

```
x =
```

```
3.1416
```

Questo non e' un test,
ma una assegnazione!

```
» x ~= 3, x ~= pi
```

```
ans =
```

```
1
```

```
ans =
```

```
0
```

Questi sono test logici.

Composizione di operazioni logiche

- Le operazioni logiche elementari si possono comporre con gli operatori logici “and”, “or”, “xor”, “not”:
 - And **&**
 - Or **|**
 - Xor **xor**
 - Not **~**
- Per una descrizione completa: **help relopt**

Cicli “while”

- La struttura del ciclo e' quella classica

while test logico

Comandi_da_eseguire mentre la condizione logica
ha valore “true”

end

```
» S = 1; n = 1;  
» while S + (n+1)^2 < 100  
    n = n+1; S = S + n^2;  
end  
» [n, S]
```

“if” ... “then” ... “else” ...

- La struttura è quella classica

if test_logico_1

Comandi da eseguire se test1 e' "true"

elseif test_logico_2

Comandi da eseguire quando test2 e' "true" ma test1 e' "false"

...

end

Un esempio

- Si vuole risolvere l'equazione $x = \cos(x)$
- La trasformiamo in una successione e cerchiamo di trovar a quale valore converge la successione

$$x_n = \cos(x_{n-1}) \quad n \in \mathbb{N}$$

- Il tutto deve essere uno script MATLAB.

```
% risolvi x= cos(x)

% metodo 1
% spreca memoria, memorizza anche passi intermedi
tic % <-- a che serve? provare "help tic"
x = zeros(1,20); x(1) = pi/4;
n = 1; d = 1;
while d > 0.001
    n = n+1; x(n) = cos(x(n-1));
    d = abs( x(n) - x(n-1) );
end
[n,x(n)]
toc % <-- a che serve? provare "help toc"
```

```
ans =
```

```
14.0000 0.7388
```

```
Elapsed time is 0.007740 seconds.
```

```
% risolvi x= cos(x)

% metodo 2
% migliore memorizza solo l'ultimo dato
tic
xold = pi/4; n = 1; d = 1;
while d > 0.001 & n < 20
    n = n+1; xnew = cos(xold);
    d = abs( xnew - xold );
    xold = xnew;
end
[n, xnew, d]
toc
```

```
ans =
```

```
14.0000 0.7388 0.0007
```

```
Elapsed time is 0.005321 seconds.
```

MATLAB function

- Un m-file è una MATLAB function **se e solo se** la prima riga del file contiene il seguente testo
 - **function** [elenco variabili in uscita] =
nome_della_funzione (elenco variabili in ingresso)
- Non è necessario terminare il file con una parola-chiave: normalmente una funzione termina quando viene raggiunta l'ultima riga di istruzioni nel M-file che la descrive (che può essere una dichiarazione **end**)
- Si può forzare la terminazione in qualunque punto utilizzando l'istruzione “**return**” (vedere **help return**)

Alcuni esempi

- Calcolo dell'area di un triangolo qualsiasi:

- Vale la formula
$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a + b + c}{2}$$

- con a, b, c , lunghezze dei lati del triangolo.

- La funzione allora (nella sua versione più semplice) avrà 3 parametri in ingresso ed 1 solo parametro in uscita

```
function [A] = evalArea(a,b,c)
% Compute the area of a triangle whose
% sides have length a, b and c.
% Inputs:
%     a,b,c: Lengths of sides
% Output:
%     A: area of triangle
% Usage:
%     Area = evalArea(2,3,4);
% Written by XXX, MM/DD/YY.
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
end % function
%%%%%%%%%% end of area %%%%%%%%%%%
```

- Provare a creare il file “evalArea.m” e poi a scrivere il comando “**help evalArea**” nella Command Window di MATLAB.
- Utilizzo della funzione
 - Assegnando una variabile in uscita
 - » Area = evalArea(10,15,20)
Area =
72.6184
 - non assegnando la variabile d’uscita
 - » evalArea(10,15,20)
ans =
72.6184

Un esempio: i numeri di Fibonacci

- Calcolare la sequenza:

$$f_n = f_{n-1} + f_{n-2} \quad n = 3, 4, 5 \dots$$

- Calcolare il valore a cui tende il rapporto tra due termini successivi al crescere dell'indice

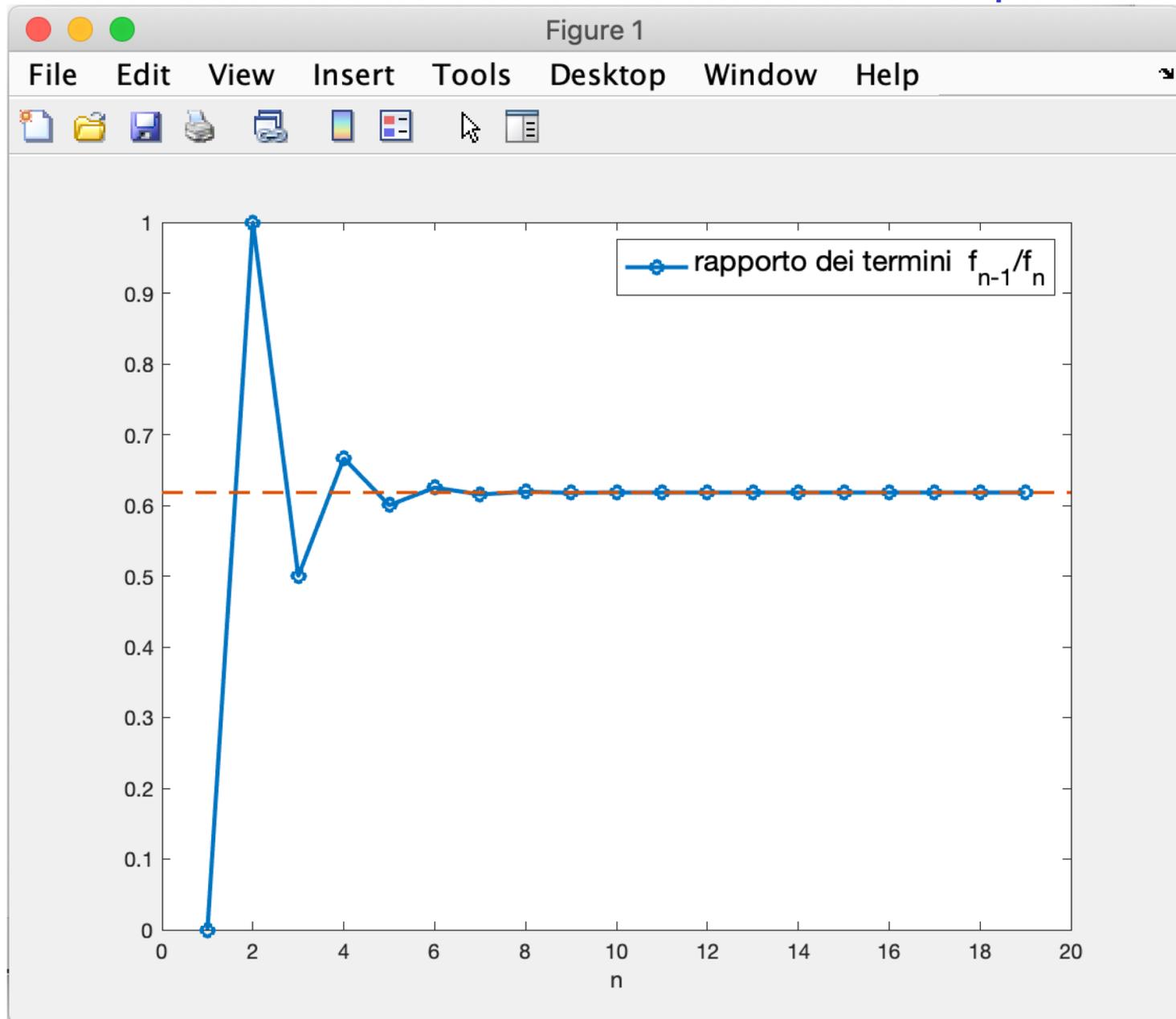
$$\frac{f_{n-1}}{f_n} = ?$$

- Condizioni iniziali

$$f(1) = 0 \qquad f(2) = 1$$

```
F(1) = 0; F(2) = 1;
% inizializzazione
for i = 3:20
    F(i) = F(i-1) + F(i-2);
end
% calcolo della sequenza

% grafici
Hp = plot(1:19, F(1:19)./F(2:20), '-o' );
hold on, xlabel('n')
plot([0 20], (sqrt(5)-1)/2*[1,1], '--')
legend([Hp], 'rapporto dei termini f_{n-1}/f_n');
```



Programmazione in MATLAB: suggerimenti

Well-written programs are better than badly-written ones -- they have fewer errors and are easier to debug and to modify -- so it is important to think about style from the beginning.

Brian W. Kernighan and Rob Pike,
The Practice of Programming
Addison-Wesley, Inc., 1999.

- Alcuni suggerimenti e linee guida per ottenere codice comprensibile, facile da mantenere ed efficiente:
 - [MATLAB Style Guidelines 2.0](#) forniscono indicazioni a proposito di nomi di variabili, funzioni e sullo stile di programmazione
 - [Guidelines for writing clean and fast code in MATLAB](#) consigliano come ottenere codice comprensibile ed efficiente
 - [Programming style guidelines](#) buone pratiche di programmazione (di ambito generale) portate in ambiente MATLAB

Analisi e simulazione di sistemi dinamici LTI in ambiente MATLAB

**Definizioni
Proprietà**

Sistemi dinamici lineari

- Un sistema dinamico **lineare tempo-invariante (LTI system)** può essere descritto:
 - in forma di **equazioni di stato**, assegnando le **matrici A,B,C,D**

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

Sistemi dinamici lineari (...)

- Un sistema dinamico **lineare tempo-invariante (LTI system)** può essere descritto:
 - in forma di **matrice di funzioni di trasferimento**
 - mediante i **polinomi a numeratore e denominatore** delle funzioni di trasferimento

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0}$$

$$H(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + \dots + b_0}{a_n z^n + a_{n-1} z^{n-1} + \dots + a_0}$$

Sistemi dinamici lineari (...)

- Un sistema dinamico **lineare tempo-invariante (LTI system)** può essere descritto:
 - in forma di **matrice di funzioni di trasferimento**
 - assegnando **zeri, poli e guadagno** delle funzioni di trasferimento

$$H(s) = K \frac{\prod_q (s + z_q)}{\prod_r (s + p_r)}$$

$$H(z) = K \frac{\prod_q (z + z_q)}{\prod_r (z + p_r)}$$

Sistemi dinamici lineari (...)

- Nell'ambiente MATLAB è possibile definire un sistema dinamico lineare tempo-invariante come **oggetto di tipo LTI model** a partire da qualsiasi di queste descrizioni (ne esistono anche altre, ma non le analizziamo [si veda il comando **ltimodels**]).

Sistemi dinamici lineari (...)

- Utilizzando questi oggetti di tipo **LTI model** è possibile **analizzare le proprietà** del sistema dinamico corrispondente (stabilità ecc.) ed è possibile **simulare l'evoluzione nel tempo** del sistema dinamico, con condizioni iniziali ed ingressi assegnati.

Sistemi lineari LTI SISO

- Nella descrizione dei comandi MATLAB e negli esempi consideriamo soltanto **sistemi SISO**.

Sistemi lineari SISO: dalle equazioni di stato

- partendo dalle **equazioni di stato**
 - definire le matrici **A,B,C,D** nel **workspace**;
 - definire il sistema mediante il comando **ss**
 - **sintassi** del comando **ss**
 - **$SYS = ss(A,B,C,D)$** crea un sistema dinamico a tempo continuo
 - **$SYS = ss(A,B,C,D,Ts)$** crea un sistema dinamico a tempo discreto, con intervallo di campionamento specificato da T_s [s].
 - Ponendo **T_s** pari a **-1**, si lascia non specificato l'intervallo di campionamento associato al sistema.

Esempi (1)

- Definizione del sistema dinamico a tempo continuo:

$$\begin{cases} \dot{x}_1 = -x_2 + 3u \\ \dot{x}_2 = -3x_1 + 2x_2 \\ y = 4x_1 + 2u \end{cases}$$

```
» A = [ 0 -1;-3 2]; B = [3;0];
      C = [4 0]; D = 2;
```

```
» sistema = ss(A,B,C,D)
```

```
a =
```

```
      x1 x2
```

```
      x1  0 -1
```

```
      x2 -3  2
```

```
b =
```

```
      u1
```

```
      x1  3
```

```
      x2  0
```

```
c =
```

```
      x1 x2
```

```
      y1  4  0
```

```
d =
```

```
      u1
```

```
      y1  2
```

Continuous-time model.

Esempi (2)

- Definizione del sistema dinamico a tempo discreto:

$$\begin{cases} x_1(k+1) = -x_2(k) + 3u(k) \\ x_2(k+1) = -3x_1(k) + 2x_2(k) \\ y(k) = 4x_1(k) + 2u(k) \end{cases}$$

```
>> A = [ 0 -1; -3 2]; B = [3;0];
      C = [4 0]; D = 2;
```

```
>> sistema = ss(A,B,C,D,-1)
```

```
a =
```

```
    x1  x2
```

```
    x1  0 -1
```

```
    x2 -3  2
```

```
b =
```

```
    u1
```

```
    x1  3
```

```
    x2  0
```

```
c =
```

```
    x1  x2
```

```
    y1  4  0
```

```
d =
```

```
    u1
```

```
    y1  2
```

Sampling time: unspecified

Discrete-time model.

Sistemi lineari SISO: dalla FdT

- partendo dalla **funzione di trasferimento**
 - assegnare i coefficienti dei polinomi a numeratore e denominatore della fdt nel workspace (nel seguito vettori **NUM** e **DEN**);
 - definire il sistema mediante il comando ***tf***
 - **Sintassi** del comando ***tf***
 - ***SYS = tf(NUM,DEN)*** crea un sistema a tempo continuo
 - ***SYS = tf(NUM,DEN,Ts)*** crea un sistema a tempo discreto con intervallo di campionamento specificato da T_s [s].
 - Ponendo ***Ts*** pari a **-1**, si lascia non specificato l'intervallo di campionamento associato al sistema.

Esempi (3)

- Definizione del sistema tramite la funzione di trasferimento

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

Transfer function:

s + 1

s² + 3 s + 16

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

Esempi (4)

- Definizione del sistema tramite la funzione di trasferimento

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

Transfer function:

z + 1

z² + 3 z + 16

$$G(z) = \frac{z + 1}{z^2 + 3z + 16}$$

Sistemi lineari SISO: dalla FdT (2)

- partendo dalla **funzione di trasferimento**
 - assegnare i vettori degli zeri, dei poli ed il guadagno del sistema nel workspace (nel seguito vettori **Z**, **P** e **K**);
 - definire il sistema mediante il comando **zpk**.
 - **Sintassi** del comando **zpk**
 - **$SYS = zpk(Z,P,K)$** crea un sistema a tempo continuo
 - **$SYS = zpk(Z,P,K,Ts)$** crea un sistema a tempo discreto con intervallo di campionamento specificato da Ts [s].
 - Ponendo **Ts** pari a **-1**, si lascia non specificato l'intervallo di campionamento associato al sistema.

Esempi (5)

- Definizione del sistema tramite la funzione di trasferimento

$$G(s) = -5 \frac{s-1}{s+1}$$

» **Z = [1]; P = [-1]; K = [-5]**
 » **sistema = zpk(Z,P,K)**
 » **Zero/pole/gain:**
 -5 (s-1)

 (s+1)

Esempi (6)

- Definizione del sistema tramite la funzione di trasferimento

$$G(z) = -5 \frac{z-1}{z+1}$$

» **Z = [1]; P = [-1]; K = [-5]**

» **sistema =zpk(Z, P, K ,-1)**

Zero/pole/gain:

-5 (z-1)

(z+1)

Sampling time: unspecified

Sintassi alternativa del comando tf

- È possibile definire le FdT "elementari" a tempo continuo ed a tempo discreto

$$G(s) = s \qquad \gg s = \text{tf('s')};$$

$$T(z) = z \qquad \gg z = \text{tf('z',-1)};$$

ed utilizzarle come "mattoni elementari" nella scrittura delle espressioni delle effettive FdT

- Nel caso di sistemi LTI SISO a tempo continuo

$$G(s) = e^{-2s} \frac{s + 1}{s^2 + 3s + 2}$$

```
s=tf('s');
```

```
Gs = exp(-2*s) *  
(s+1)/(s^2+3*s+2)
```

Gs =

$$\exp(-2s) * \frac{s + 1}{s^2 + 3s + 2}$$

Continuous-time transfer function.

- Nel caso di sistemi LTI SISO a tempo discreto

$$H(z) = z^{-3} \frac{(z + 1.0)(z + 0.4)}{(z - 1)(z + 0.8)}$$

```
z = tf('z',-1);
```

```
HHz = z^(-3)*  
(z+1)*(z+0.4)/(z-1)/  
(z+0.8)
```

HHz =

$$\frac{z^2 + 1.4z + 0.4}{z^5 - 0.2z^4 - 0.8z^3}$$

Sample time: unspecified

Discrete-time transfer function.

Simulazione in MATLAB di sistemi lineari

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

Simulazione in MATLAB di sistemi lineari (2)

- Sintassi:

» `[y,t]=step(sistema);`

» `[y,t]=step(sistema,t);`

Vettore d'uscita

Vettore dei tempi

Vettore sequenza d'ingresso

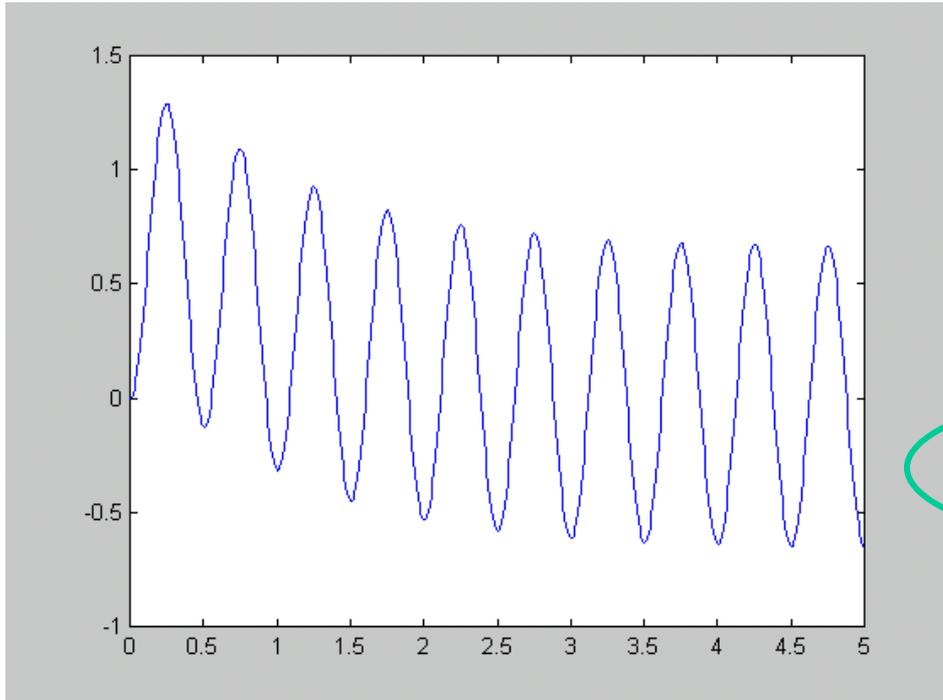
» `[y,x]=lsim(sistema,u,t);`

Vettore d'uscita

Vettore di stato

Vettore dei tempi

Esempio di utilizzo (1)

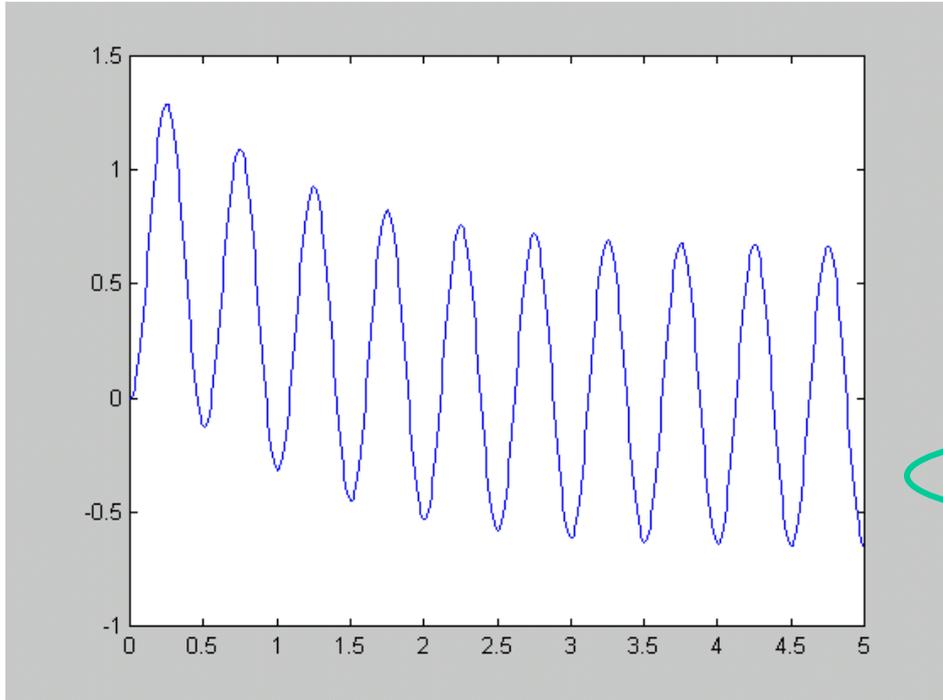


```
» a=[-1 ,0;3,-4];  
» b=[2;1];c=[1,2];d=0;  
» sistema=ss(a,b,c,d);  
» t=(0:0.01:5);  
» u=2*sin(2*pi*2*t);  
» y=lsim(sistema,u,t);  
» plot(t,y);
```



Il risultato della simulazione è stato assegnato ad una variabile e successivamente visualizzato in un grafico.

Esempio di utilizzo (2)



```
» a=[-1 ,0;3,-4];  
» b=[2;1];c=[1,2];d=0;  
» sistema=ss(a,b,c,d);  
» t=(0:0.01:5);  
» u=2*sin(2*pi*2*t);  
» lsim(sistema,u,t);
```



Utilizzando le funzioni senza assegnare il risultato della simulazione a variabili d'uscita si ottiene direttamente il grafico dell'evoluzione temporale.

Esempi di utilizzo (3)

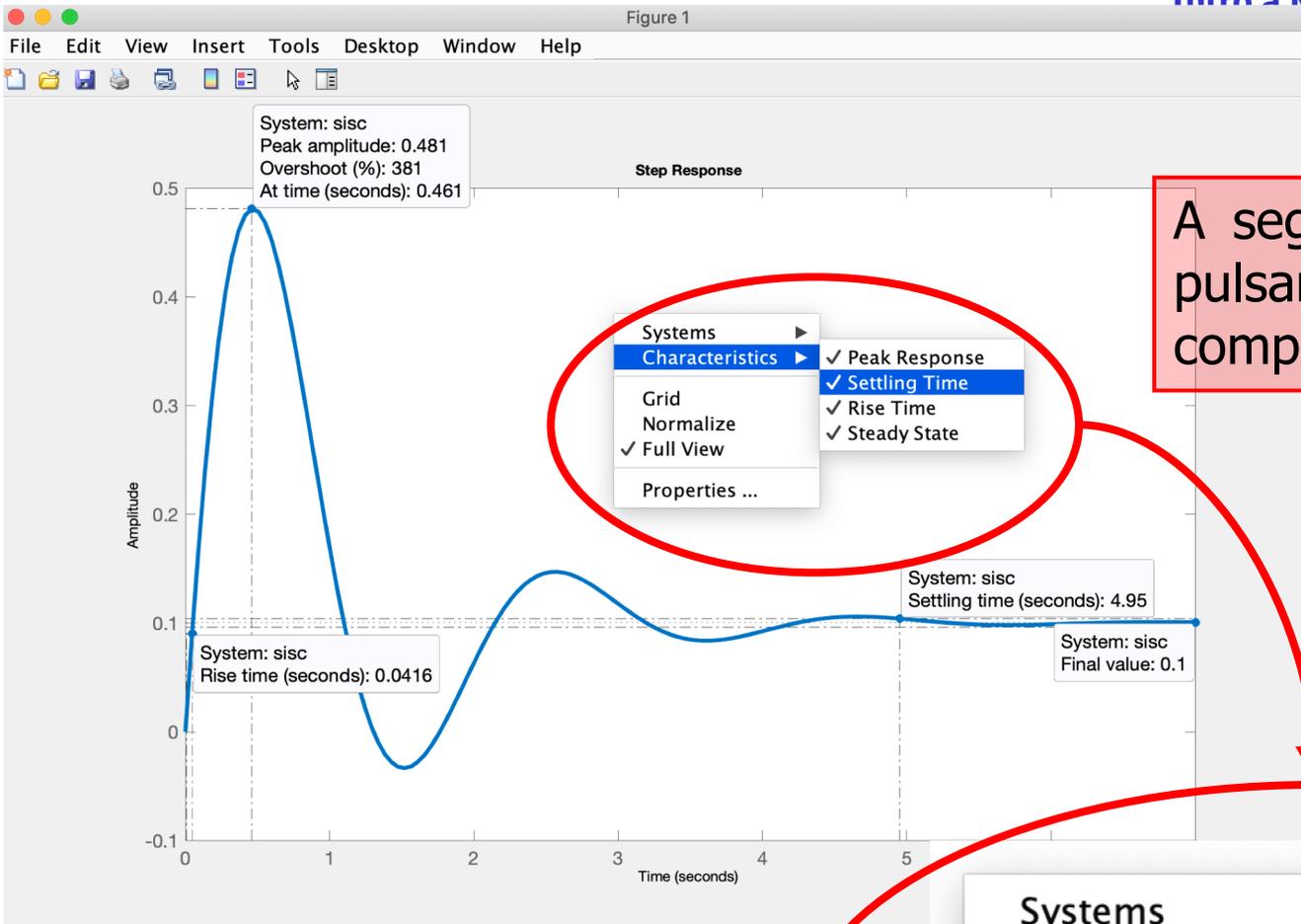
- Analisi della risposta allo scalino unitario:

» **step(sistema);**

- Interagendo tramite il mouse nella finestra che visualizza l'andamento della risposta allo scalino è possibile ottenere informazioni relative a
 - Sovraelongazione della risposta
 - Valore di regime
 - Tempo di salita (*rise time*)
 - Tempo di assestamento (*settling time*)

Esempio: analisi della risposta allo scalino

- Esempi di simulazione della risposta allo scalino unitario per sistemi dinamici LTI a tempo continuo ed a tempo discreto:
 - » *% sistema a tempo continuo*
 - » *sisc = tf([2 1],[1 2 10]);*
 - »
 - » *% sistema a tempo discreto*
 - » *sisd = tf([2 1],[1 0.2 0.5], 0.1);*
 - »
 - » *figure; step(sisc);*
 - » *figure;step(sisd);*

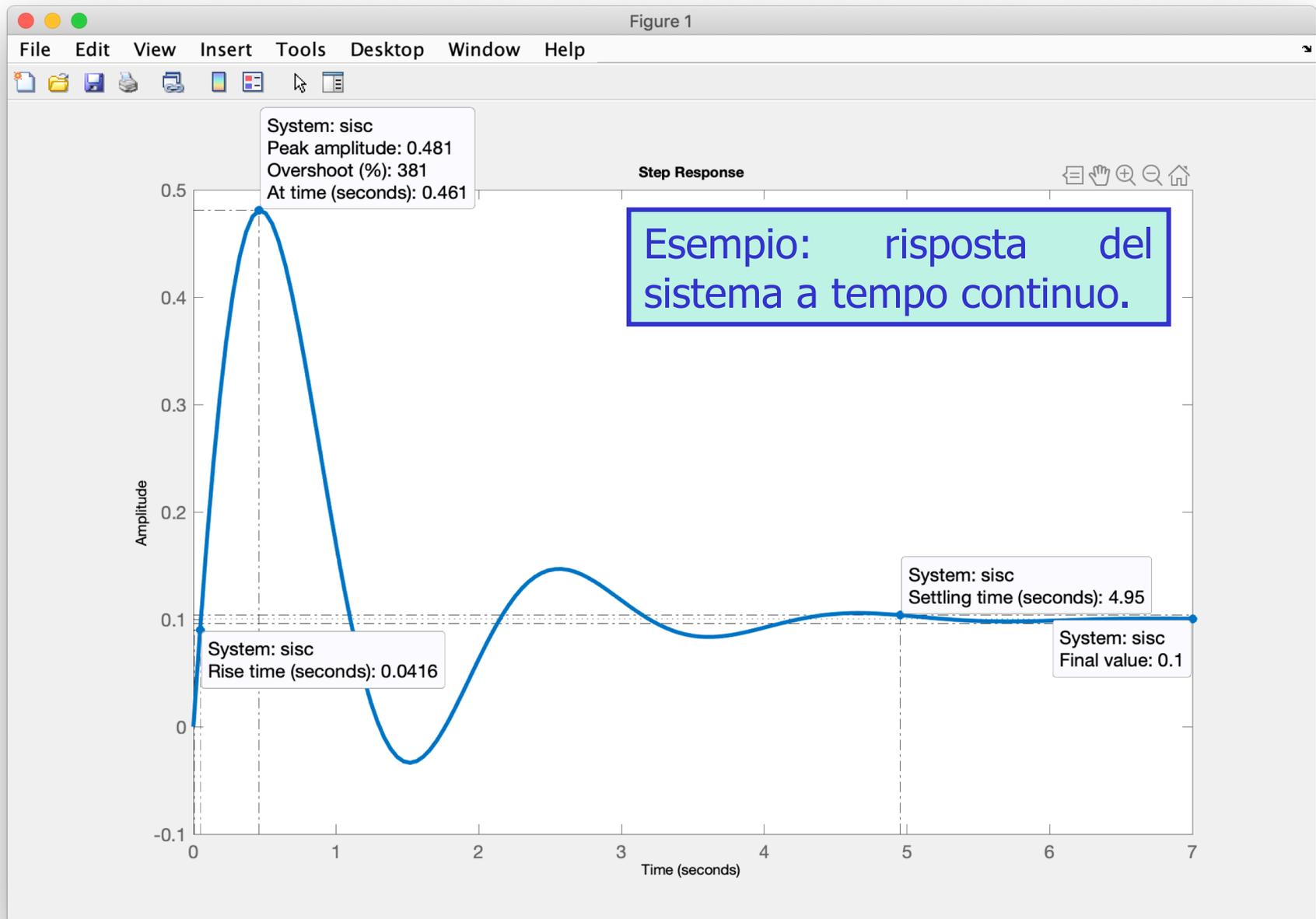


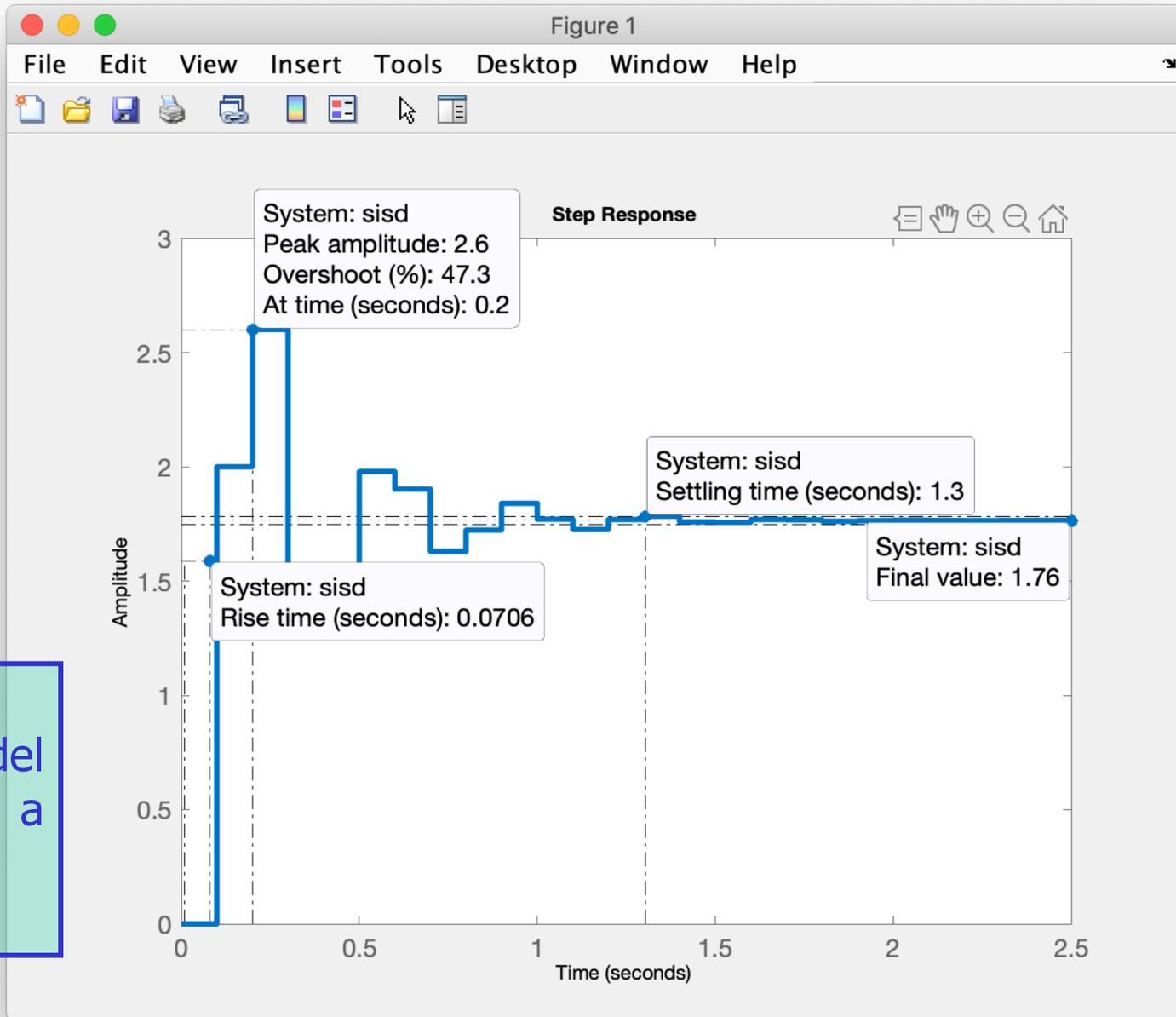
A seguito di un "click" del pulsante destro del mouse compare un menu ...

This block shows a detailed view of the 'Characteristics' menu. The menu is open, and the 'Characteristics' option is selected. The sub-menu is visible, showing the following options:

- ✓ Peak Response
- ✓ Settling Time
- ✓ Rise Time
- ✓ Steady State

The menu also includes 'Systems', 'Grid', 'Normalize', 'Full View', and 'Properties ...'.





Esempio:
risposta del
sistema a
tempo
discreto.

Considerazioni riassuntive

- Come tutti i risultati di **operazioni di calcolo numerico**, anche i sistemi LTI, descritti con le istruzioni MATLAB viste, possono essere affetti da **errori** (errata o mancante cancellazione di termini nella FdT ecc. ...).
- Consiglio: consultare la sezione “**Reliable Computations**” e l’argomento “**Choice of LTI Model**” nella documentazione del **Control Toolbox**.

Sistemi lineari interconnessi

Definizioni, proprietà, applicazioni

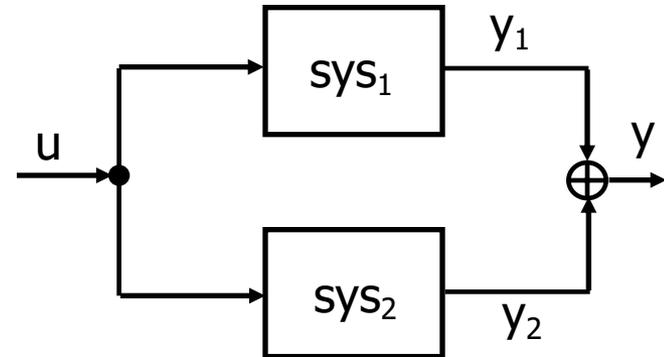
Interconnessione di sistemi

- Agli *oggetti* sistemi lineari si possono applicare i normali operatori $+$, $*$, $/$, \backslash con il seguente significato:
 - $+$ connessione in **parallelo**;
 - $*$ connessione in **serie**;
 - $/$, \backslash usati per definire l'operazione di inversione (a sx, a dx) per **sistemi quadrati**.

Esempi di interconnessioni elementari

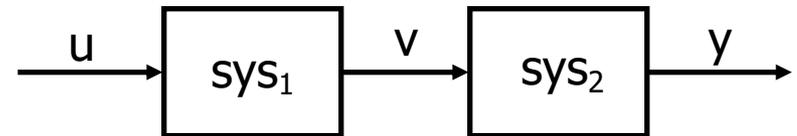
- **Connessione parallelo**

$$sys = sys_1 + sys_2$$



- **Connessione serie**

$$sys = sys_1 \cdot sys_2$$



Altri esempi di operazioni elementari

- **Inversione** (per sistemi quadrati)

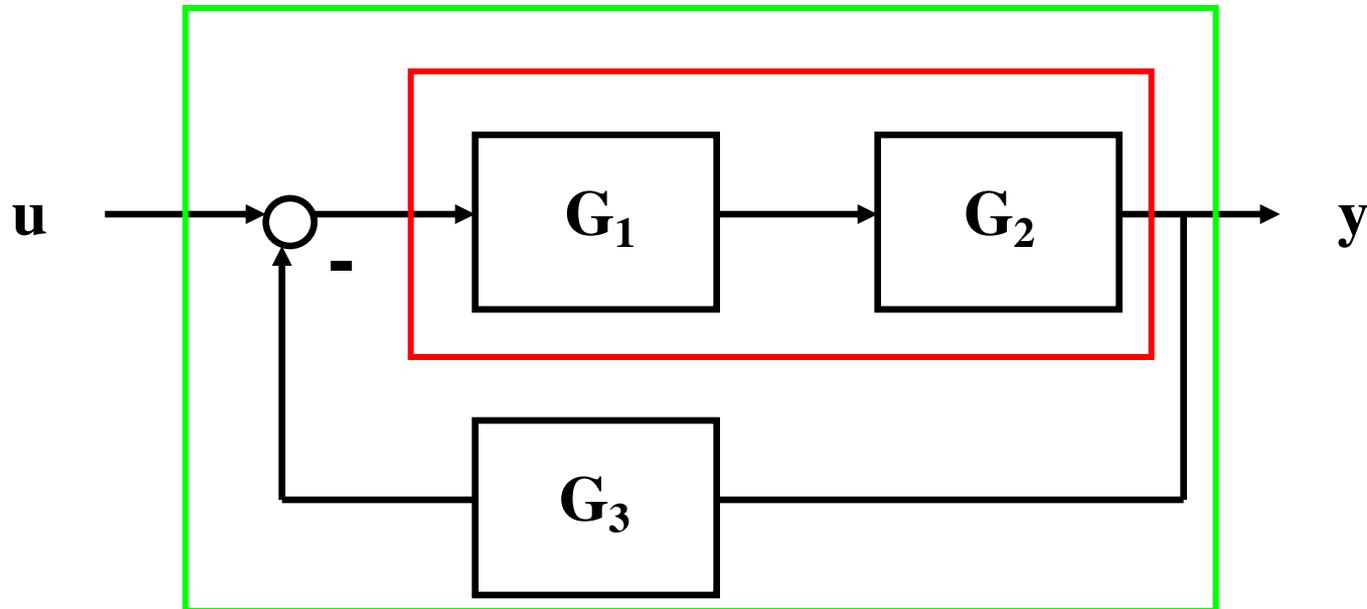
$$sys_1 \backslash sys_2 \iff inv(sys_1) \cdot sys_2$$

$$sys_1 / sys_2 \iff sys_1 \cdot inv(sys_2)$$

- **Trasposizione**

$$sys^T \iff sys.'$$

Esempio di connessione



andata = $g1 * g2$; **retroazione** = $andata / (1 + andata * g3)$

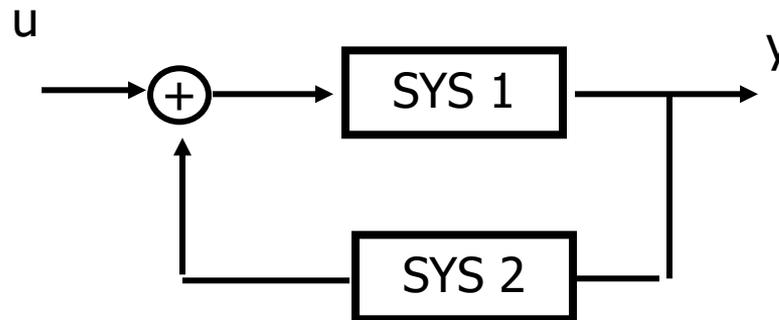
Interconnessioni tra sistemi

- Funzioni che permettono di descrivere interconnessioni tra sistemi dinamici:
 - concatenazione `,` (orizz.) ; (vert.)
 - connessione diagonale a blocchi **append**
 - connessione parallelo di due blocchi **parallel**
 - connessione serie di due blocchi **series**
 - connessione in retroazione **feedback**
 - connessione generica **connect**

Sintassi del comando feedback

Sintassi semplice del comando

SYS = feedback(SYS1,SYS2) fornisce il sistema LTI corrispondente al semplice ciclo di reazione seguente

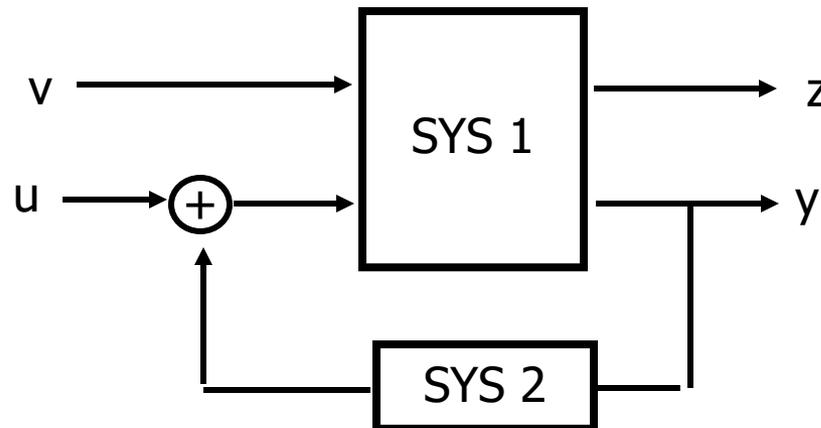


Si presuppone che sia uno schema a **retroazione negativa**. Nel caso in cui si voglia una retroazione positiva, va specificato nel comando

SYS = feedback(SYS1,SYS2,+1).

Sintassi completa

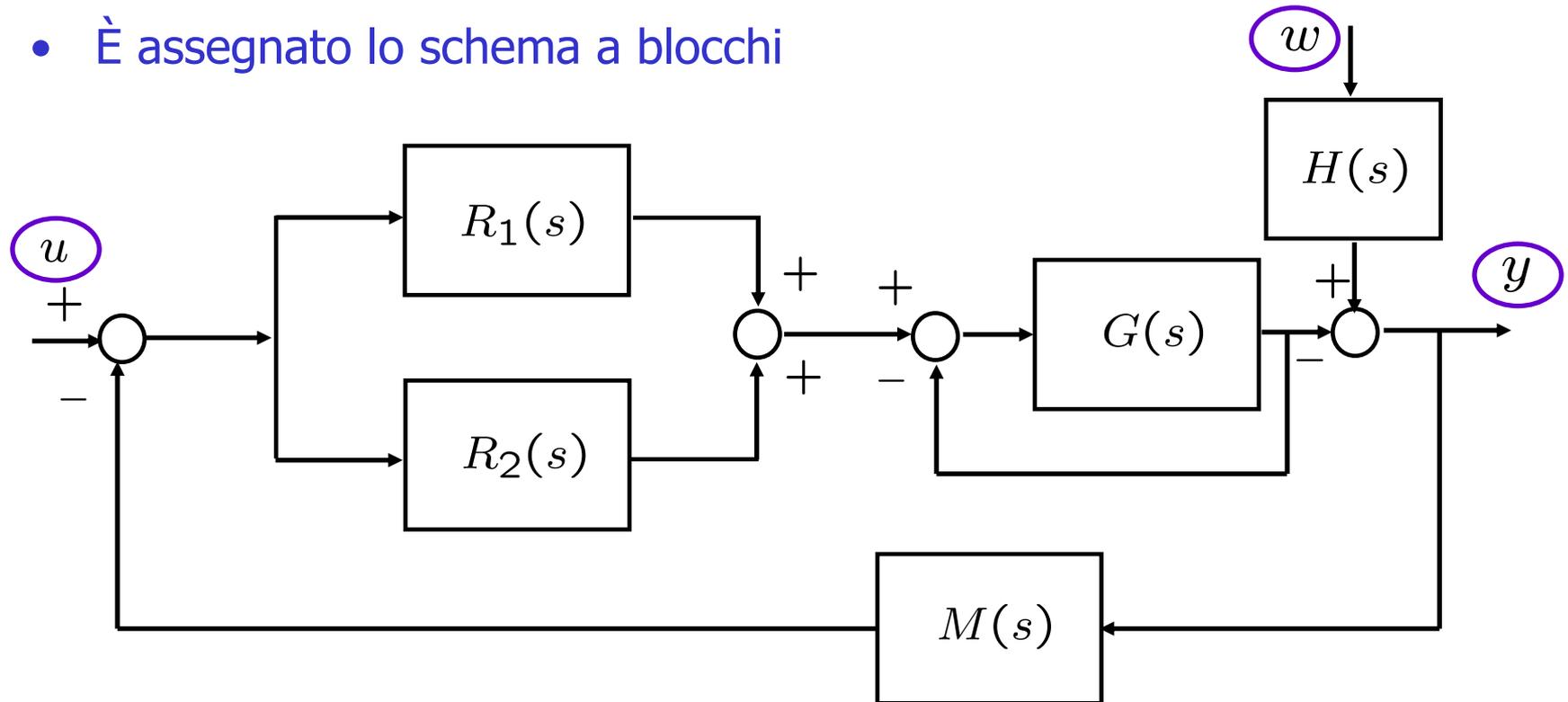
SYS = feedback(SYS1, SYS2, FEEDIN, FEEDOUT, SIGN) fornisce il sistema LTI corrispondente alla struttura:



I vettori **FEEDIN** e **FEEDOUT** contengono rispettivamente gli indici degli ingressi e delle uscite del sistema SYS1 coinvolti nella retroazione. La variabile **SIGN** indica se si tratta di retroazione positiva (**SIGN** pari a +1) oppure negativa (**SIGN** pari a -1 oppure non assegnato).

Un esempio

- È assegnato lo schema a blocchi



- Determinare le FdT tra gli ingressi u , w e l'uscita y (cfr. Parte 5)

- Come FdT dei vari blocchi elementari consideriamo le seguenti

$$R_1(s) = \frac{3}{2} \cdot \frac{1 + 10s}{1 + \frac{s}{25}} \quad R_2(s) = \frac{49}{2} \cdot \frac{s + 2}{s(s + 10)}$$

$$G(s) = \frac{90.0}{s^2 + 8.2s + 36.0} \quad H(s) = \frac{1 + s}{1 + \frac{s}{50}}$$

$$M(s) = 125 \frac{s + 2}{(s + 10)(s + 25)}$$

- Dall'analisi svolta in *Parte 5* (#20-21) del materiale del corso sappiamo che le FdT cercate hanno le espressioni

$$F_1(s) = \frac{Y(s)}{U(s)} = \frac{-[R_1(s) + R_2(s)] \frac{G(s)}{1 + G(s)}}{1 - [R_1(s) + R_2(s)] \frac{G(s)}{1 + G(s)} M(s)}$$

$$F_2(s) = \frac{Y(s)}{W(s)} = \frac{H(s)}{1 - [R_1(s) + R_2(s)] \frac{G(s)}{1 + G(s)} M(s)}$$

- Ed in MATLAB? Come fare ad ottenere le due FdT cercate?

```

s = tf('s');
% definisco le FdT dei blocchi elementari
R1s = (3/2)*(1+10*s)/(1+s/25);
R2s = (49/2)*(s+2)/s/(s+10);
Gs = 90/(s^2+8.2*s+36.0);
Hs = (1+s)/(1+s/50);
Ms = 125 * (s+2)/(s+10)/(s+25);
% -----

% costruisco alcune FdT intermedie,
% utili nei calcoli
R1R2 = R1s + R2s;
% parallelo dei blocchi R1s ed R2s
FG = feedback(Gs,1,-1);
% Gs chiusa in semplice loop di reazione
R1R2FG = R1R2 * FG;
% serie dei blocchi appena determinati

% ----- calcolo le FdT finali -----
F1s = -R1R2FG / (1 - R1R2FG * Ms);

F2s = Hs / (1 - R1R2FG * Ms);
% -----

```

Attenzione! Il comando `tf` non esegue alcuna semplificazione per cancellazione zero/polo!

```
>> zpk(F1s)
```

```
ans =
```

$$\frac{-33750 s (s+25)^2 (s+10)^2 (s+9.92) (s^2 + 0.2454s + 0.3293)}{(s^2 + 8.2s + 126)}$$

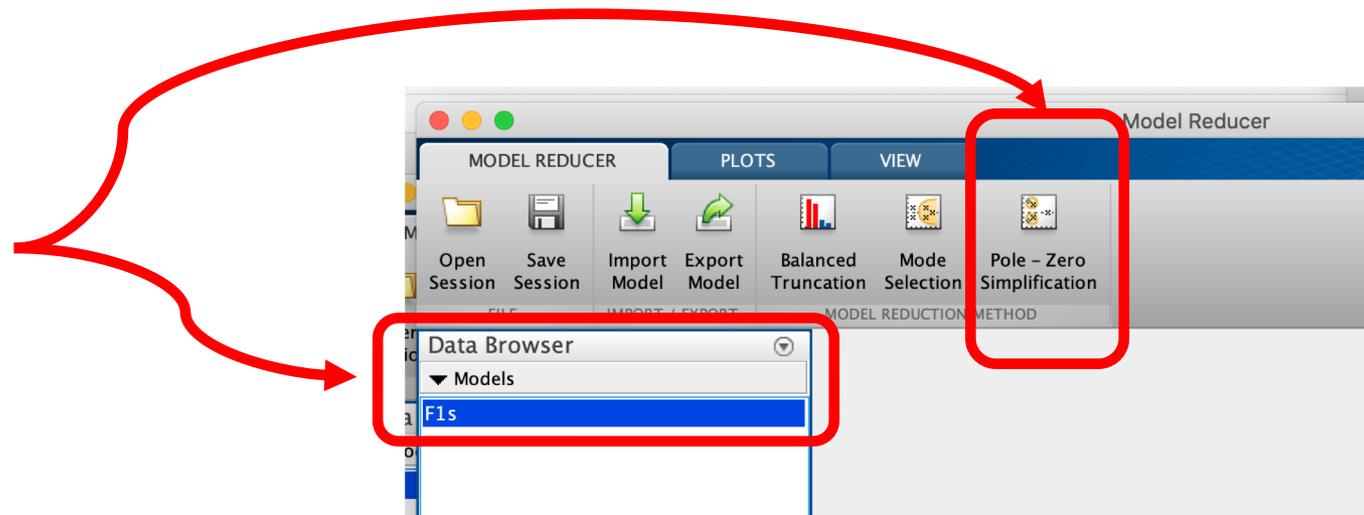
$$\frac{s (s-139.2) (s+25) (s+10) (s+9.92) (s+2.058) (s^2 + 0.1524s + 0.3216) (s^2 + 8.2s + 126)}{(s^2 + 205.3s + 3.017e04)}$$

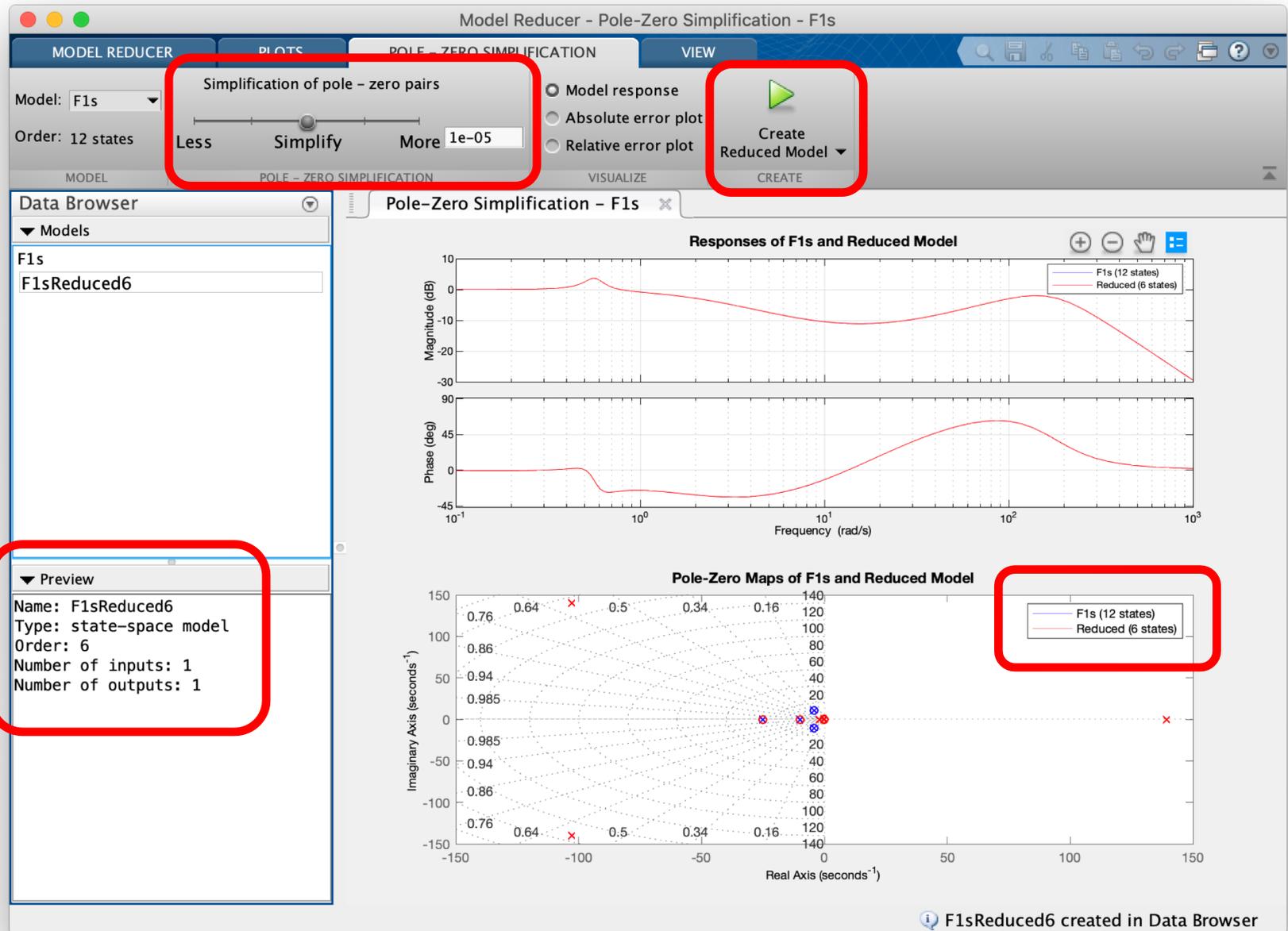
- Come fare per eseguire le cancellazioni?

Model Reducer App

- Permette di
 - ridurre l'ordine di un sistema, determinando modelli approssimati di ordine ridotto (cfr. concetto di "poli dominanti") [NB non approfondiamo!]
 - eseguire **semplificazioni tra zeri e poli**

```
>> modelReducer(F1s)
```





```
zpk(F1sReduced6)
```

```
ans =
```

$$-33750 (s+25) (s+10) (s^2 + 0.2454s + 0.3293)$$

$$(s-139.2) (s+2.058) (s^2 + 0.1524s + 0.3216) (s^2 + 205.3s + 3.017e04)$$

Analisi della risposta in frequenza

Risposta in frequenza per sistemi LTI a tempo continuo

- **risposta in frequenza** per un sistema LTI asintoticamente stabile a tempo continuo

$$F(\omega) = G(s)|_{s=j\omega}$$

dove $G(s)$ è la fdt del sistema.

Risposta in frequenza per sistemi LTI a tempo discreto

- **risposta in frequenza** per un sistema LTI asintoticamente stabile a tempo discreto

$$F(\vartheta) = G(z)|_{z=e^{j\vartheta}}$$

dove $G(z)$ è la fdt del sistema.

Grafici della risposta in frequenza

- Comandi MATLAB
 - **bode** : rappresentazione della risposta in frequenza tramite **diagrammi di Bode**
 - **nyquist** : rappresentazione della risposta in frequenza tramite **diagramma polare / diagramma di Nyquist**

Calcolo ed analisi della risposta in frequenza

- Comandi MATLAB
 - **freqresp** : **calcolo** di valori della risposta in frequenza di un sistema dinamico;
 - **ltiview** : strumento interattivo che permette l'**analisi** delle risposte nel tempo ed in frequenza di sistemi dinamici LTI.

Sintassi dei comandi (1)

- Sintassi generale dei comandi *bode*, *nyquist*:
 - *comando(SYS)* crea il diagramma voluto della risposta in frequenza del sistema *SYS*, creato tramite le istruzioni *ss,tf,zpk*. L'intervallo di frequenza ed il numero di punti utilizzati per i diagrammi sono scelti in modo automatico.
 - *comando(SYS,{WMIN,WMAX})* crea il diagramma per le frequenze comprese tra **WMIN** and **WMAX** [rad/s].

Sintassi dei comandi (2)

- ***comando(SYS, W)*** utilizza il vettore **W** specificato dall'utente come vettore delle frequenze (in *rad/s*) per le quali determinare la risposta in frequenza.

- ***[a,b]=comando(SYS,...)*** non visualizza alcun grafico, ma restituisce valori della risposta in frequenza, utilizzabili per tracciare successivamente il diagramma voluto.
 - ***[m,f]=bode(...)*** restituisce modulo e fase (in gradi)
 - ***[r,i]=nyquist(...)*** restituisce parte reale ed immaginaria dei punti della risposta in frequenza

Sintassi dei comandi (3)

- Per sistemi LTI a tempo discreto con periodo di campionamento T_s viene utilizzata la relazione

$$z = e^{j\Omega T_s}$$

per mappare la circonferenza unitaria con centro l'origine (in z) tramite la pulsazione Ω [rad/s] e la pulsazione massima che compare nei grafici è pari a

$$\Omega_{\max} = \frac{\pi}{T_s} = \Omega_N = \frac{\Omega_S}{2}$$

Se T_s non è assegnato, viene assunto un periodo di campionamento pari ad **1 s**.

Sintassi dei comandi (4)

- Calcolo della risposta in frequenza
 - $H = \text{freqresp}(\mathbf{SISTEMA}, W)$ calcola la risposta in frequenza H del sistema dinamico LTI descritto da $\mathbf{SISTEMA}$ in corrispondenza delle pulsazioni assegnate nel vettore W . Queste pulsazioni sono espresse in *rad/s*.
 H è un array di valori complessi.

Sintassi dei comandi (5)

- Analisi nel tempo e in frequenza di un sistema LTI
 - *ltiview(SISTEMA)* apre una finestra grafica interattiva, nella quale viene visualizzata, come scelta predefinita, la risposta al gradino unitario di **SISTEMA**, ma permette di analizzare sia l'evoluzione temporale che la risposta in frequenza di **SISTEMA**.

Sintassi dei comandi (6)

- ***ltiview*(grafico, **SISTEMA**)** specifica quale grafico debba venire visualizzato. In particolare grafico può essere una delle seguenti espressioni (oppure una combinazione di esse)
 - ***'step'*** risposta al gradino unitario
 - ***'impulse'*** risposta all' impulso unitario
 - ***'bode'*** diagrammi di Bode
 - ***'bodemag'*** diagramma di Bode del modulo
 - ***'nyquist'*** diagramma di Nyquist
 - ***'pzmap'*** mappa poli/zeri
 - altro ancora (si veda la sintassi completa del comando ...)

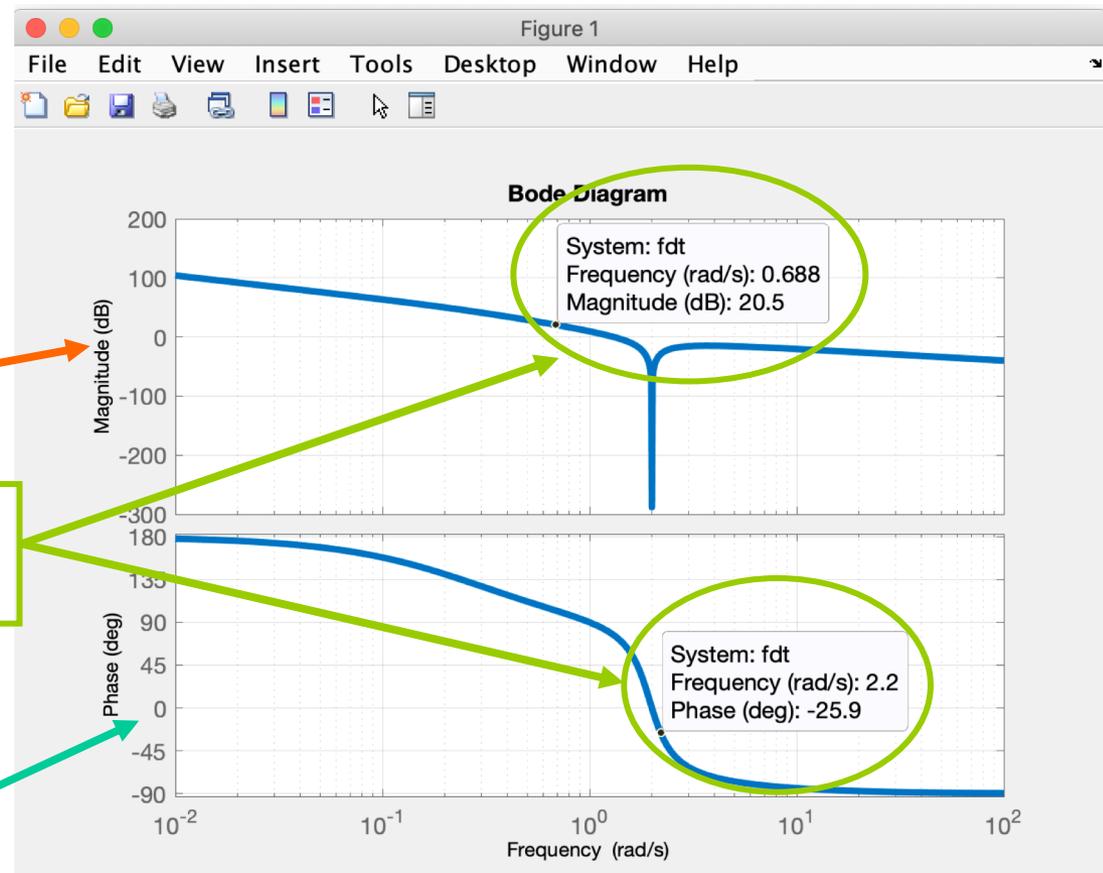
Esempi: diagrammi di Bode

- » `num=conv([1 0 4],[1 0 4]);`
- » `den=[1 1 4 1 0 0];`
- » `fdt=tf(num,den);`
- » `figure;bode(fdt)`

Diagramma del modulo

Esplorazione del diagramma con il mouse (pulsante sx premuto)

Diagramma della fase

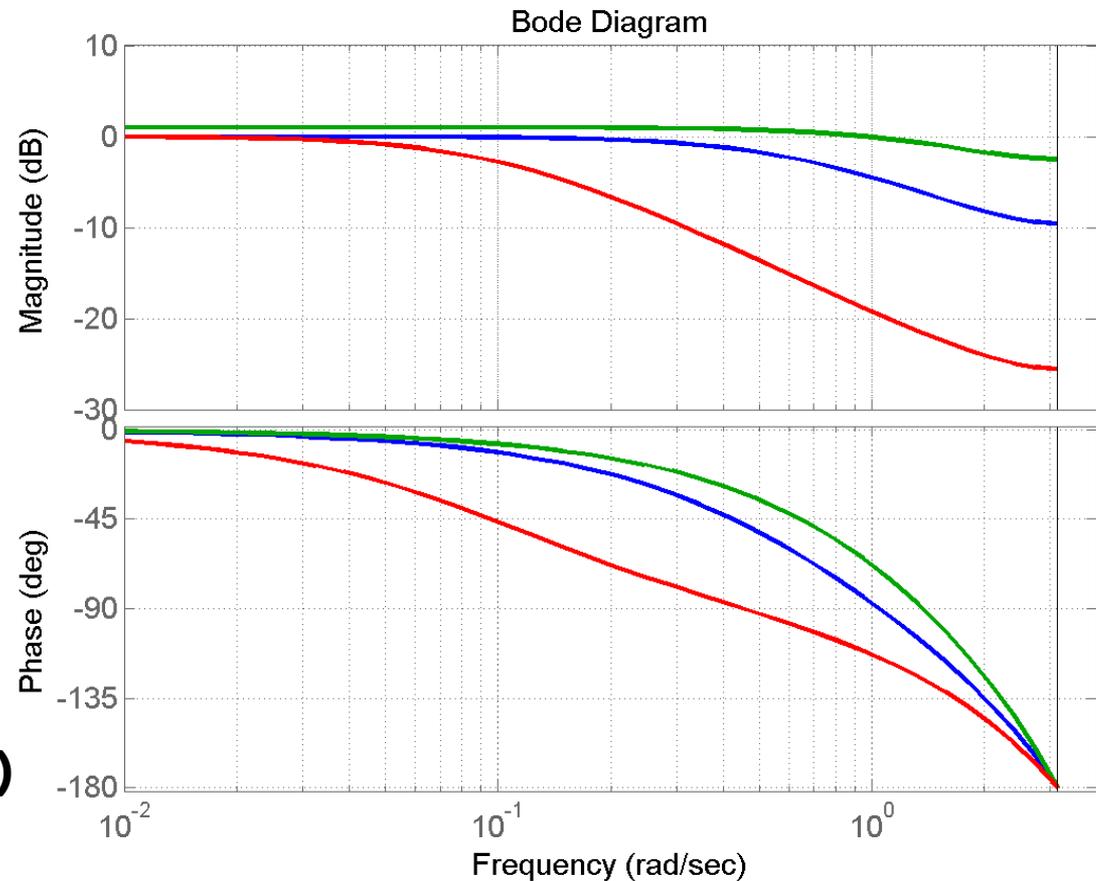


Esempi: diagrammi di Bode

```

» num = [0.5]
» den=[1 -0.5];
» fdt1=tf(num,den,-1);
» num = [0.9]
» den=[1 -0.2];
» fdt2=tf(num,den,-1);
» num = [0.1]
» den=[1 -0.9];
» fdt3=tf(num,den,-1);
» figure;bode(fdt1,fdt2,fdt3)

```



Diagrammi di Bode per sistemi LTI a tempo discreto

Esempi: diagrammi di Nyquist

- » `num1=1`
- » `den1=[1 2 1]`
- » `figure;`
- » `nyquist(tf(num1,den1))`

Verso di percorrenza
sul diagramma

In evidenza il punto $-1+j0$

Esplorazione del diagramma con il
mouse (pulsante sx premuto)

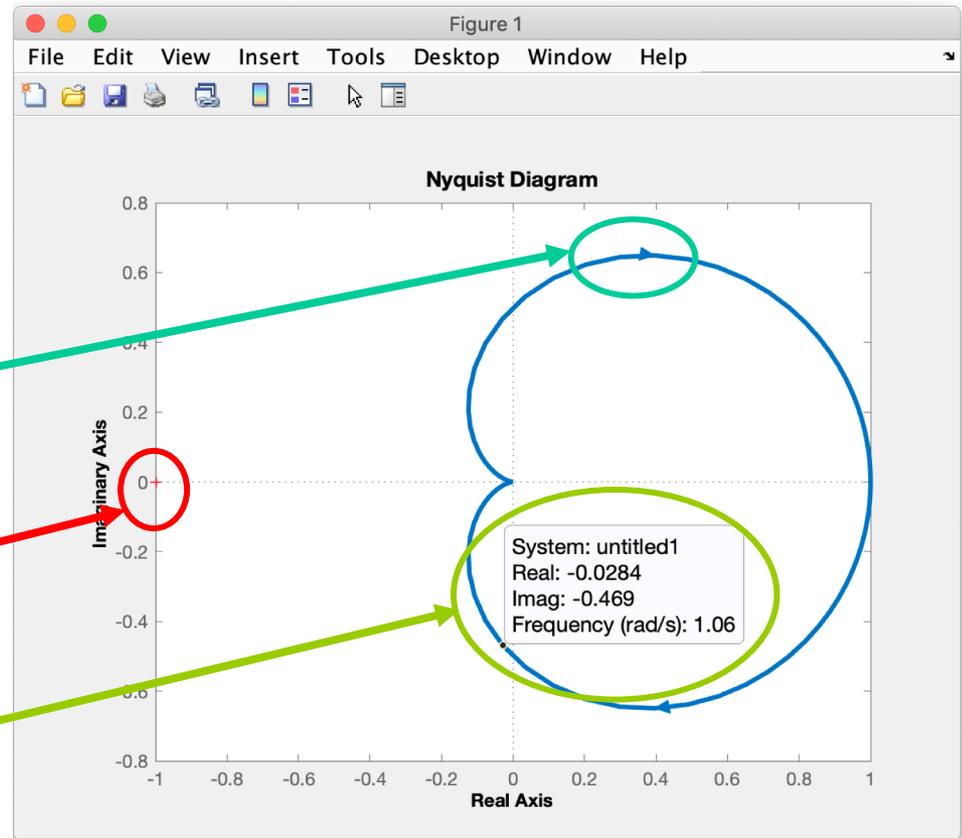


Diagramma di Nyquist COMPLETO

Diagrammi di Nyquist: casi particolari

- **Attenzione!** Il comando *nyquist(sistema)* traccia il diagramma completo di Nyquist della risposta in frequenza del sistema in esame.
- Ciò può portare ad una visualizzazione non ottimale dell'andamento del diagramma in condizioni particolari.
- In tal caso il diagramma va visualizzato scegliendo opportunamente l'intervallo di pulsazioni d'interesse. **NON approfondiamo:** si rimanda ai manuali del software per ulteriori dettagli!

Diagrammi di Nyquist: casi particolari

- nel caso di **sistema LTI a tempo continuo**
 - presenza di poli (semplici o multipli) nell'origine;
 - presenza di poli (semplici o multipli) immaginari puri;
- nel caso di **sistema LTI a tempo discreto**
 - Presenza di poli (semplici o multipli) complessi, di modulo unitario