

Controlli Automatici T

Introduzione a Matlab

Prof. Guido Carnevale

Department of Electrical, Electronic, and Information Engineering
Alma Mater Studiorum Università di Bologna
guido.carnevale@unibo.it

Queste slide sono ad uso interno del corso
Controlli Automatici T dell'Università di Bologna a.a. 25/26.

Informazioni

Periodo lezioni

Venerdì, 12.30 - 15.30, Lab 9 e Lab 4.

Richieste ricevimento e comunicazioni con il docente

email: guido.carnevale@unibo.it

IMPORTANTE oggetto email: [CAT] “oggetto email da inviare”

Materiale didattico

slide e codice verranno resi disponibili su Virtuale

Tutor

Dr. Andrea Tramaloni, email: andrea.tramaloni@unibo.it

Dr. Andrea Drudi, email: andrea.drudi@unibo.it

Installazione

Creare un account Mathworks (<https://it.mathworks.com>) con la propria mail istituzionale @studio.unibo.it

Scaricare il software usando il proprio account

Seguire il processo di installazione guidata seguendo le indicazioni per i possessori di licenza

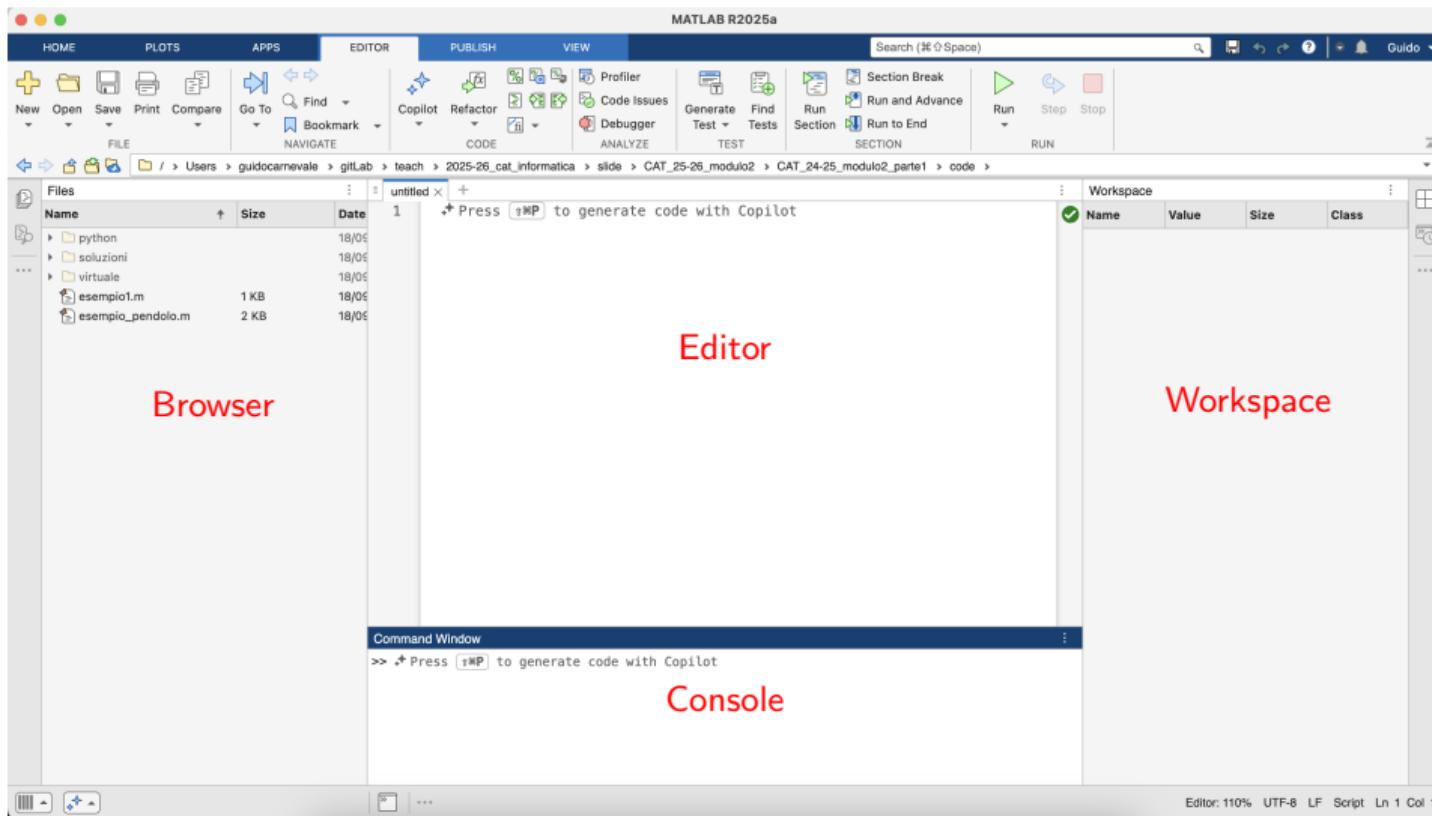
MATLAB: MATrix LABoratory

Matlab è una piattaforma di programmazione e calcolo numerico

- funzioni generali di base
- varie estensioni chiamate “toolbox” (es. Control System Toolbox e Simulink)

Esistono alternative open-source come GNU-Octave, Scilab, ...

Schermata principale



Iniziamo!

Matlab è un linguaggio interpretato e può anche essere scritto in maniera interattiva sulla console (come ad es. Python)

In linea di principio la console può essere usata come una calcolatrice...

```
>> log(sqrt(5*2) + 2^2)
```

```
ans =
```

```
1.9688
```

Nota: l'ultimo calcolo effettuato viene memorizzato nella variabile `ans`

Operatori di base: + - * / ^

```
help ops % Operatori e caratteri speciali
```

Variabili

L'oggetto di base su matlab è la matrice (array)

```
x=1          % uno scalare (matrice 1x1)

r=[1,2]      % un vettore riga 1x2
r=[1 2]      % si puo' anche omettere la virgola

c=[1;2]      % un vettore colonna 2x1

M=[1,2;3,4]  % una matrice 2x2
M=[r;r]      % un'altra matrice 2x2...
```

Nota: definizione di una matrice con le parentesi *quadre*.

Per accedere all'elemento (i,j) si usano invece le parentesi *tonde*:

```
M(1,1)      % accedere all'elemento 1,1 della matrice
```

Nota: il punto e virgola alla fine della riga ; non è necessario ma evita che il risultato venga stampato

Funzioni di base

Per registrare la sequenza di comandi inseriti sulla console

```
diary mydiary.txt  
% ... comandi  
diary off
```

Altre funzioni

```
clear % reset del workspace  
clc % pulisce la console  
cd directory % per cambiare la current working directory  
format long % mostra piu' cifre decimali  
format short % mostra meno cifre decimali
```

Visualizzare il valore di una variabile

```
disp(x) % alternativa 1  
fprintf('valore di x(1): %.4f\n', x(1)) % alternativa 2
```

Operazioni su matrici

```
r1=[-1,5]          % vettore riga 1x2
r2=r1'            % trasposto di r1, vettore colonna 2x1

sum(r1)           % somma delle componenti di r1
max(r1)           % massimo valore tra le componenti di r1
norm(r1)          % norma (euclidea) di r1

M=[1,2;3,4]        % una matrice 2x2
det(M)             % determinante di M
jordan(M)          % forma di Jordan di M
eig(M)             % autovalori di M
[V D] = eig(M)    % autovettori/autovalori della matrice M

inv(M)             % inversa di M
```

Operazioni tra matrici

```
% somma di due vettori riga
r1=[1,2]
r2=[3,4]
r1+r2

% somma di due matrici
M1=[1,2;2,3]
M2=[4,5;6,7]
M1+M2

% somma di uno scalare e un non-scalare
x = 5
r1 + x      % somma 'x' ad ogni componente del vettore
M1 + x      % somma 'x' ad ogni componente della matrice
```

Operazioni tra matrici (II)

```
c1 = [1;2] % vettore colonna 2x1
c2 = [3;4] % vettore colonna 2x1
c1-c2      % differenza di due vettori colonna
5*c1       % scalare per vettore (colonna)

M1*M2       % prodotto tra due matrici (riga per colonna)
c1'*c2     % vettore riga per vettore colonna (prodotto scalare)
c1*c2'     % colonna per riga (l'output e' una matrice 2x2)

c1*c2       % errore: incorrect dimensions for matrix multiplication

c1.*c2      % prodotto elemento per elemento
M1.*M2      % prodotto elemento per elemento, diverso da M1*M2
```

Funzioni aritmetiche di base

Alcuni esempi

sin	%	- Seno
sind	%	- Seno dell'argomento in gradi
sinh	%	- Seno iperbolico
asin	%	- Arcoseno
cos	%	- Coseno
tan	%	- Tangente
atan	%	- Arcotangente
atan2	%	- Arcotangente (4 quadranti)
deg2rad	%	- Converte da gradi a radianti
rad2deg	%	- Converte da radianti a gradi

Esempio di utilizzo

```
sin(pi/2)    % 1
sind(90)     % 1
deg2rad([30 60 90]) % si possono anche applicare a vettori o matrici
```

Funzioni aritmetiche di base (II)

```
help elfun % Elementary math functions
```

Altri esempi

abs	%	- Valore assoluto
exp	%	- Esponentiale
sqrt	%	- Radice quadrata
log	%	- Logaritmo naturale
log10	%	- Logaritmo in base 10

Esempio di utilizzo: trasformare un numero in decibel

```
20*log10(3) % 3 in dB
```

Numeri complessi

```
i % simbolo per l'unita' immaginaria
j % altro simbolo per l'unita' immaginaria
5 + 1i % un numero complesso
2*exp(i*pi) % un numero complesso in modulo e fase

real(2*exp(i*pi)) % parte reale
imag(2*exp(i*pi)) % parte immaginaria
abs(5 + 1i) % modulo (norma)
angle(5 + 1i) % argomento (angolo)

% variabili complesse
x = 5 + 1i
y = 2*exp(i*pi)

% le operazioni si possono effettuare normalmente
x+y
x*y
```

Altro sulle matrici

Informazioni sulle variabili

```
whos          % - Stampa informazioni su una variabile  
size          % - Dimensione di un array  
length        % - Lunghezza di un vettore
```

Esempio

```
x=[1,2]  
length(x)      % restituisce 2  
size(x)        % restituisce (1,2)
```

Manipolazioni di matrici

```
reshape % Cambia le dimensioni di un array  
diag    % Crea matrice diagonale o estraе diagonale di una matrice  
blkdiag % Crea una matrice diagonale a blocchi
```

Alcune matrici elementari

```
help elmat % Elementary math functions
```

Alcuni esempi

zeros	% - Array di zeri
ones	% - Array di uni
eye	% - Matrice identità
repmat	% - Ripetizione di una matrice
linspace	% - Vettore con spaziatura lineare
logspace	% - Vettore con spaziatura logaritmica

Per generare una matrice casualmente

rand	% - Numeri casuali con distribuzione uniforme
randn	% - Numeri casuali con distribuzione normale

Matrici e Indici

```
1:5          % numeri da 1 a 5 (come linspace)
1:0.5:5     % numeri da 1 a 5 a passo 0.5

M(1,1)       % estrae l'elemento 1,1
M(1:2,1)     % estrae gli elementi 1,1 e 2,1
M(1:0.5:2,1) % errore: non si possono usare indici decimali

M(end,1)     % estrae l'ultimo elemento della colonna 1
M(end-1,1)   % estrae il penultimo elemento della colonna 1
M(:,1)        % estrae tutta la colonna 1
M(:end-1,1)  % estrae tutta la colonna 1 tranne l'ultimo elemento
```

Attenzione! A differenza di altri linguaggi in Matlab gli indici iniziano da 1 (e non da 0)

Concatenazione di matrici

```
M = [eye(3),zeros(3,2); ones(1,3),0,2]
```

Funzioni di algebra lineare

```
help matfun % Matrix functions - numerical linear algebra.
```

Alcuni esempi

```
rank          % - Rango  
det           % - Determinante  
trace         % - Traccia (somma degli elementi sulla diagonale)  
null          % - Kernel  
orth          % - Ortogonalizzazione
```

Esempio di utilizzo

```
v=rand(3,1)  
rank(v*v')  
null(v*v')
```

Soluzione di sistemi lineari

Supponiamo di voler risolvere il seguente sistema lineare

$$\begin{cases} x + y + z = 1 \\ x - y - z = 1 \\ x - 2y + 3z = -5 \end{cases}$$

o equivalentemente

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -5 \end{bmatrix}$$

Per risolvere numericamente su Matlab:

```
A = [1 1 1; 1 -1 -1; 1 -2 3];
b = [1; 1; -5];

inv(A)*b      % metodo 1: tramite inversa di A
A\b          % metodo 2: come metodo 1 ma internamente ottimizzato
linsolve(A, b) % metodo 3: funzione 'linsolve'
```

Grafici

Grafico della funzione seno

```
t = 1:100; % punti campionati sull'asse orizzontale  
y = sin(t); % valori sull'asse y  
figure % crea una nuova finestra  
plot(y) % disegna il grafico  
close % chiude la finestra  
  
figure  
plot(t,y) % doppio argomento per specificare i valori sull'asse x
```

Per rendere le figure più fruibili

```
figure  
plot(t,y, 'b--') % linea tratteggiata rossa  
  
plot(t,y, 'LineStyle',':') % linea punteggiata  
plot(t,y, 'LineWidth',3) % spessore 3pt  
  
plot(t,y, 'Color','r') % 'r' sta per red  
plot(t,y, 'Color',[0,0,1]) % codice RGB
```

Figure con grafici multipli

```
t = 1:0.1:100;
y_sin = sin(t);
y_cos = cos(t);

figure
plot(t,y_sin, 'b-.') % blu tratto punto
hold on % comando per i grafici multipli
plot(t,y_cos, 'r') % rosso
```

Ulteriori dettagli di una figura

```
title('Funzioni Trigonometriche')
xlim([10,20])
ylim([0,0.5])
xlabel('asse orizzontale')
ylabel('asse verticale')
legend('sin', 'cos')
```

Grafici con scale non-lineari

Funzioni per generare i grafici

```
plot      % - grafico con scala lineare per entrambi gli assi  
loglog    % - grafico con scala logaritmica per entrambi gli assi  
semilogx  % - grafico con scala logaritmica per x e lineare per y  
semilogy  % - grafico con scala logaritmica per y e lineare per x
```

Esempio

```
x = linspace(1,10)  
y1 = exp(x)  
y2 = exp(-x)  
figure  
    plot(x,y1)  
    hold on  
    plot(x,y2)  
figure  
    semilogy(x,y1)  
    hold on  
    semilogy(x,y2)
```

Esportare una figura

```
fig = figure % handle della figura
plot(t,y);

savefig('miaFigura'); % salva la figura come 'miaFigura.fig'

%% Esportare in formato pdf
fig = gcf; % gcf: Get handle to Current Figure
fig.PaperUnits = 'centimeters';
fig.Units = 'centimeters';
fig.PaperPosition = [0 0 10 3]; % 10cm x 3cm
fig.PaperSize = [10 3];
print(fig, 'miaFiguraPdf', '-dpdf', '-r200'); % miaFiguraPdf.pdf
```

Script

Un gruppo di istruzioni da eseguire in sequenza può essere racchiuso in un file con estensione .m. Lo script può essere eseguito con run.

```
%  
% Esempio 1  
%  
clc      % pulisce la command window  
clear all % pulisce il workspace  
close all % chiude le finestre aperte  
  
t=1;  
y=2;  
v = [t,y];  
fprintf('Norma del vettore [1,2]: %d.\n', norm(v));  
  
fprintf('Vettore: %d.\n', v); % viene ripetuto per ogni entry  
  
str = 'CAT25-26';  
fprintf('\tStringa: %s.\n', str);
```

Operatori logici

```
x=1          % dichiara la variabile e le assegna valore 1 (double)
logical(x)   % converte x in una variabile logica/booleana
x==1         % la variabile x vale 1? Risultato booleano: 1/0, TRUE/
              FALSE

if x==1
    fprintf('x vale 1.\n');
end
```

Altri operatori logici

```
x>1          % Restituisce booleano: 1/0, TRUE/FALSE
x<=0         % Attenzione agli arrotondamenti numerici...
~x            % nega il valore di x (con x booleano)

x && y      % AND logico
x || y       % OR logico

x=[1,2];
x==0         % restituisce un vettore di booleani
```

Controllo di flusso

```
if x==1
    fprintf('se x vale 1... ');
elseif x==3
    fprintf('se x vale 3... ');
else
    fprintf('negli altri casi... ');
end
```

Esiste anche la funzione switch

```
switch x
case 1
    fprintf('se x vale 1... ');
case 3
    fprintf('se x vale 3... ');
otherwise
    fprintf('negli altri casi... ');
end
```

Ciclo for e ciclo while

```
for x=[1,2,3,4]
    x
end
for x=1:4
    x
end
v=1:4;
for x=v
    x
end
```

```
for x=1:4
    if x>2
        break % uscita dal ciclo for
    elseif x==1
        continue % vai alla prossima iterazione
    else
        ...
    end
    ...
end
```

Esiste anche il ciclo while

```
x = -1;

while x<10
    x = x+1;
end
```

Esercizio

- Scrivere uno script che faccia la moltiplicazione tra due matrici utilizzando il ciclo *for*.

Funzioni

Le operazioni viste finora sono delle *funzioni* (built-in o implementate in un toolbox)

```
len_of_x = length(x)
```

Si possono definire nuove funzioni salvandole in un file con estensione .m (n.b.: nome della funzione e nome del file .m devono coincidere (es.: `untitled.m`))

```
function [outputArg1,outputArg2] = untitled10(inputArg1,inputArg2)
%UNTITLED10 Summary of this function goes here
% Detailed explanation goes here
arguments (Input)
    inputArg1
    inputArg2
end

arguments (Output)
    outputArg1
    outputArg2
end

outputArg1 = inputArg1;
outputArg2 = inputArg2;
end
```

Esercizi con le funzioni

- Creare una funzione chiamata `miaFunzione` che
 - ▶ prenda in input due numeri
 - ▶ restituisca in output il massimo dei due numeri e la somma dei due numeri.

- Creare una funzione chiamata `prodottoScalare` che
 - ▶ prenda in input due vettori della stessa lunghezza `vec1` e `vec2`
 - ▶ restituisca in output il loro prodotto scalare calcolato con un ciclo `for`.

Esercizio

Creare uno script che crei 4 figure con i grafici dei seguenti segnali di base, con $t \in [0, 10]$:

- $y(t) = e^{-t}$
- $y(t) = a \sin(2\pi ft + \varphi)$
- $y(t) = at \sin(2\pi ft + \varphi)$
- $y(t) = a \sin(2\pi ft + \varphi)e^{-t}$

Valori dei parametri: $a = 5$, $f = 0.5$ hz, $\varphi = 0.2$ rad

Funzione anonime

Le funzioni anonime offrono un modo conciso per creare funzioni senza la necessità di un file separato. Sono particolarmente utili per i calcoli semplici e per passare funzioni come argomenti.

```
% dato v = [v_1;v_2] generiamo [sin(v_1); cos(v_2)]
fun1 = @(v) [sin(v(1)); cos(v(2))];
y    = fun1([pi/2;0]);
```

Le funzioni anonime possono avere uno o più argomenti di ingresso, separati da virgole.

L'ordine degli argomenti determina quale valore di ingresso corrisponde a quale argomento.

```
% dato t e v = [v_1;v_2] generiamo [sin(v_1)/t; cos(v_2)/t]
fun2 = @(t,v) [sin(v(1))/t; cos(v(2))/t];
y    = fun2(100,[pi/2;0]);
```

Funzione anonime

Le funzioni anonime offrono un modo conciso per creare funzioni senza la necessità di un file separato. Sono particolarmente utili per i calcoli semplici e per passare funzioni come argomenti.

```
% dato v = [v_1;v_2] generiamo [sin(v_1); cos(v_2)]
fun1 = @(v) [sin(v(1)); cos(v(2))];
y    = fun1([pi/2;0]);
```

Le funzioni anonime possono avere uno o più argomenti di ingresso, separati da virgole.

L'ordine degli argomenti determina quale valore di ingresso corrisponde a quale argomento.

```
% dato t e v = [v_1;v_2] generiamo [sin(v_1)/t; cos(v_2)/t]
fun2 = @(t,v) [sin(v(1))/t; cos(v(2))/t];
y    = fun2(100,[pi/2;0]);
```

La funzione `ode45` (prossima slide) di default si aspetta come primo argomento una funzione anonima dove il primo argomento è il tempo ed il secondo l'incognita.

Integrazione numerica di sistemi dinamici

Consideriamo un generico sistema dinamico (tempo-invariante)

$$\dot{x}(t) = f(x(t), u(t)) \quad \text{equazione della dinamica}$$

$$x(0) = x_0 \quad \text{condizione iniziale}$$

Fissato un ingresso $u(t) = \bar{u}(t)$ e definita $\tilde{f}(x, t) = f(x, u(t))$, otteniamo il seguente sistema non forzato (tempo-variante)

$$\dot{x}(t) = \tilde{f}(x(t), t)$$

Per trovare la traiettoria di stato $x(t)$, $t \geq 0$, si può risolvere l'equazione differenziale con `ode45`:

```
% supponiamo di avere t_max, x_0
% e supponiamo di aver definito una funzione f_tilde(t, x)
% risolviamo l'equazione differenziale e plottiamo il risultato
[time, traj] = ode45(f_tilde, [0 t_max], x_0);
plot(time, traj);
```

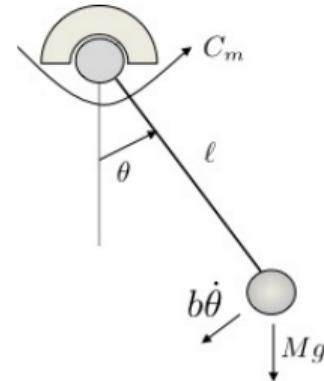
Esempio: pendolo

Equazione della dinamica

$$\ddot{\theta}(t) = -\frac{g}{\ell} \sin(\theta(t)) - \frac{b}{M\ell^2} \dot{\theta}(t) + \frac{1}{M\ell^2} C_m(t)$$

M, ℓ, b parametri fisici del pendolo

$C_m(t)$ momento torcente (input) applicato al pendolo



Definiamo $x_1 := \theta$ e $x_2 := \dot{\theta}$ (stato $x := [x_1 \ x_2]^T$) e $u := C_m$ (ingresso)

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} x_2(t) \\ -\frac{g}{\ell} \sin(x_1(t)) - \frac{b}{M\ell^2} x_2(t) + \frac{1}{M\ell^2} u(t) \end{bmatrix}}_{f(x(t), u(t))}$$

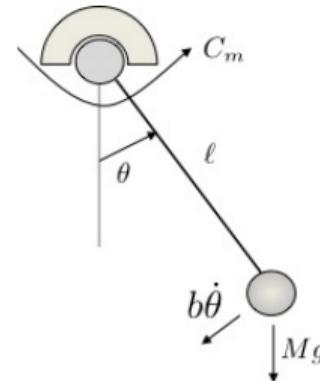
Esempio: pendolo

Equazione della dinamica

$$\ddot{\theta}(t) = -\frac{g}{\ell} \sin(\theta(t)) - \frac{b}{M\ell^2} \dot{\theta}(t) + \frac{1}{M\ell^2} C_m(t)$$

M, ℓ, b parametri fisici del pendolo

$C_m(t)$ momento torcente (input) applicato al pendolo



Definiamo $x_1 := \theta$ e $x_2 := \dot{\theta}$ (stato $x := [x_1 \ x_2]^T$) e $u := C_m$ (ingresso)

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} x_2(t) \\ -\frac{g}{\ell} \sin(x_1(t)) - \frac{b}{M\ell^2} x_2(t) + \frac{1}{M\ell^2} u(t) \end{bmatrix}}_{f(x(t), u(t))}$$

Esercizio: dato l'input costante $\bar{u}(t) = 2, t \geq 0$ e la condizione iniziale $\theta(0) = 30^\circ, \dot{\theta}(0) = 0$, creare uno script che calcoli la traiettoria dello stato nell'intervallo temporale $0 \leq t \leq 10$ e ne faccia il grafico.