

TEORIA DEI SISTEMI E DEL CONTROLLO LM in Ingegneria Informatica e Ingegneria Elettronica

<http://www.dii.unimore.it/~lbiagiotti/TeoriaSistemiControllo.html>

INTRODUZIONE A MATLAB

Ing. Luigi Biagiotti

e-mail: luigi.biagiotti@unimore.it

<http://www.dii.unimore.it/~lbiagiotti>

Programma della lezione

- Che cos'è Matlab e obiettivo del corso
- Input/Output
- Principali comandi
- Costrutti principali
- M-files e M-function
- Conclusioni

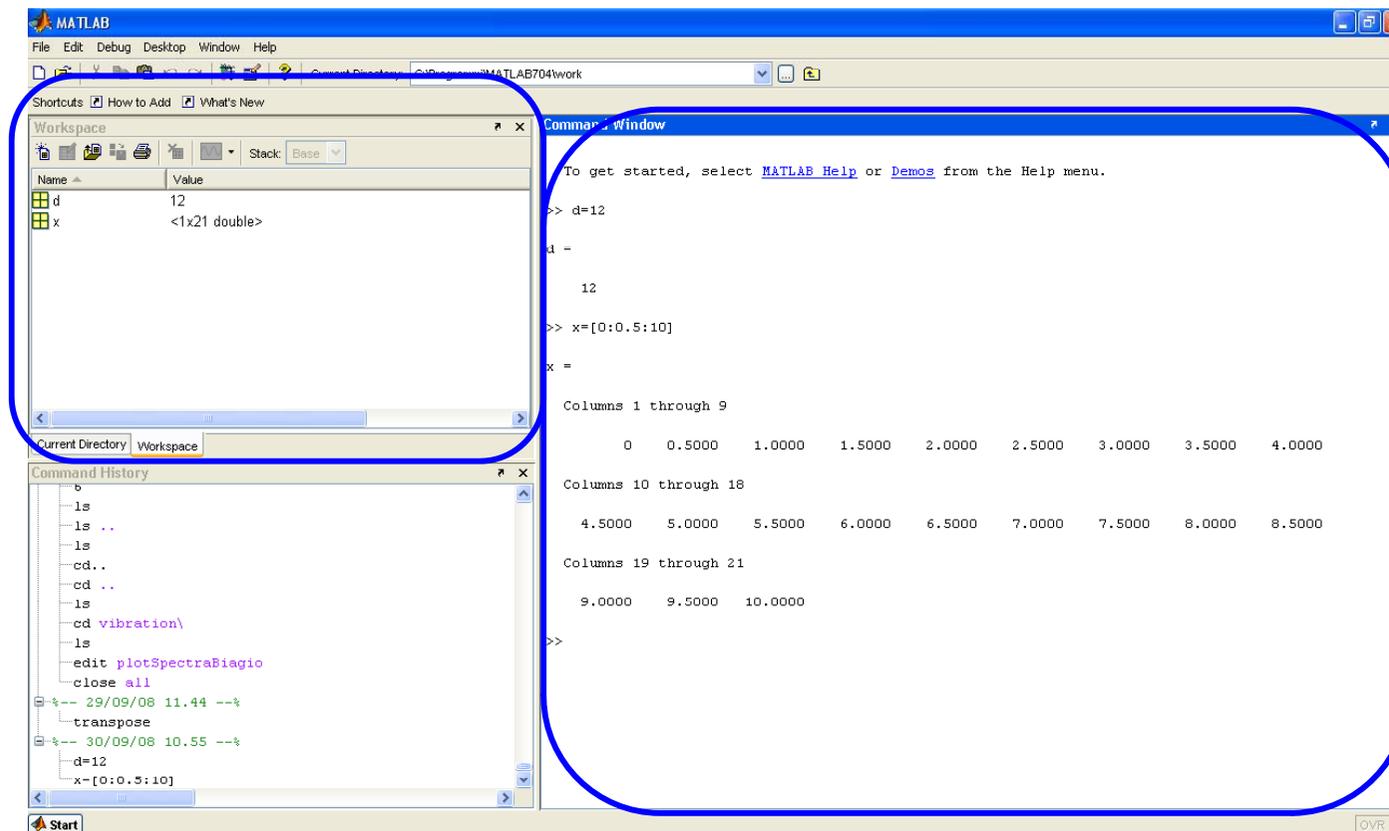
Matlab

- Matlab (**M**atrix **L**aboratory) è un programma per l'analisi numerica e la simulazione di sistemi dinamici.
- Esso contiene un nucleo di funzioni di base **general purpose**; esistono, poi, delle estensioni, i **Toolbox**, che consentono di estendere le del programma aggiungendo funzioni specialistiche.
- Durante il corso impareremo a utilizzare il programma sia per l'**Analisi** di sistemi dinamici che per la **Sintesi** di sistemi di controllo. Utilizzeremo il toolbox **Control**.

Matlab

- All'apertura il programma si presenta nel seguente modo:
 - Finestra principale con il **prompt dei comandi**
 - Finestre secondarie (tra cui si segnala la finestra di **Workspace**, che raccoglie tutte le variabili definite in Matlab)

Workspace



Command window con il prompt dei comandi

Command Window di Matlab



```
MATLAB Command Window
File Edit View Window Help

This version is for educational classroom use only.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, type tour or visit www.mathworks.com.

> pwd
ans =
C:\MATLABR11\work

> dir
.          motoreDC0.m      motoreDC@plot.m
..         motoreDC@nd1.mdl

> dir *.m
motoreDC0.m      motoreDC@plot.m

> motoreDC0
elapsed_time =
    0.4068

> who
Your variables are:

Anp      Kgf      U        h        rad
Cn       Kn       Ucen     lu       rpm
Cnax     Lne      Ui       a        s
Cr       N        Un       mAnp     t
Henry    Nm       Wmax     mHenry   th
lc0      Ohm      Wne      mN       th0
Inax     Rne0     Wne0     ng       tipo
lne      TABCR    bne      minuti   tout
lne0     TABTENPI cm       mm
Jne      TABVIN   g        nsec
KI       Tcsin    gr       nprova
Kg       Tfin     gradi    ora

> edit motoreDC0
> cd d:
> pwd
ans =
D:\

> clc
Ready
```

- I comandi in Matlab si inseriscono tramite il “CommandWindow”.
- Alcuni comandi di uso generale sono:
 - **pwd** restituisce la directory corrente
 - **dir** elenca i file della directory
 - **dir *.m** elenca i file .m
 - **cd <newdir>** cambia la directory di lavoro
 - **clc** pulisce la finestra
- Le variabili definite in Matlab sono raccolte nel “WorkSpace”. Il comando **who** elenca le variabili del workspace. Il comando **clear** cancella tutte le variabili definite finora.
- I file di comandi hanno estensione .m e contengono dei comandi di Matlab che sono eseguiti digitando il nome del file come se fosse un comando.
motoreDC0 è un file comandi.
I file di comandi si scrivono con il “Matlab Editor”: **edit motoreDC0**

In Matlab qualsiasi dato è trattato come una variabile

- Per introdurre una variabile basta assegnarle un nome e un valore in questo modo:

```
>> x=12;
```

La variabile **x** vale 12. Omettendo il ; viene stampato il nome della variabile e il suo valore, altrimenti non c'è *echo* dei comandi. Se si digita soltanto un valore e non lo si assegna a una variabile, Matlab assegna di default tale valore alla variabile **ans**

Per cominciare con Matlab

- Per una panoramica sui comandi di Matlab digitare:
`>> demo`
e seguire le istruzioni.
- Digitando `help` dal prompt di comando compare la lista completa dei toolbox presenti. Digitando
`>> help <nome toolbox>`
si ha l'elenco completo delle funzioni disponibili per quel toolbox. Digitando
`>> help <nome comando>`
si accede alla descrizione di quel comando.
- Per trovare un comando non noto riguardante un certo argomento digitare:
`>> lookfor keyword`
dove `keyword` è una parola relativa all'argomento di interesse. Esempio:
`>> lookfor transfer`
restituisce vari comandi relativi alle funzioni di trasferimento
- Molti comandi specifici per i controlli automatici:
`>> help control`

Vettori e Matrici

- Possiamo definire una matrice con la seguente sintassi:

```
>> A=[1,2,3;4,5,6;7,8,9]
```

- Gli elementi della stessa riga sono separati da `,` mentre le varie righe dal `;`. Per accedere a un elemento della matrice basta specificare la riga e la colonna dell'elemento.

- La chiamata:

```
>> element = A(1,2)
```

associa alla variabile `element` il valore dell'elemento di riga 1 e colonna 2 della matrice `A`. In particolare sarà `element=2`

Le wildcards

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

Ad esempio, la prima riga di **A** si seleziona con:

```
>> A(1,:)
```

```
ans =
```

```
    1    2    3
```

e la prima colonna di **A** con:

```
>> A(:,1)
```

```
ans =
```

```
    1
```

```
    4
```

```
    7
```

- Selezione di una sottomatrice

```
>> B=A(2:3,1:2)
```

```
B =
```

```
    4    5
```

```
    7    8
```

Nota che gli indici di riga e di colonna partono da 1 e non da 0.

Vettori e Matrici

- Un vettore è una particolare matrice

- `>> c=[4;5;6];`



Vettore colonna

- `>> r=[4,5,6];`



Vettore riga

- Si può far generare a Matlab un vettore automaticamente
- Il comando

```
>> t=[0:0.1:10]
```

genera un vettore riga con valori che vanno da 0 a 10 con passo 0.1.

- Possiamo operare con le variabili: possiamo costruire una variabile utilizzando i valori memorizzati in altre variabili. Possiamo costruire vettori con variabili scalari, matrici con vettori e matrici con altre matrici.

Vettori

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

I polinomi e le operazioni

- Definiamo il polinomio "pol" ($= 3s^2 + 2s + 1$) con l'istruzione:

```
>> pol = [3 2 1]
```

```
pol =
```

```
3 2 1
```

- **roots**: calcolo delle radici (pol=0):

```
>> roots(pol)
```

```
ans =
```

```
-0.3333 + 0.4714i
```

```
-0.3333 - 0.4714i
```

- **polyval**: valutazione in un punto:

```
>> polyval(pol,1)
```

```
ans =
```

```
6
```

I polinomi e le operazioni

- **Calcolo dei residui di una funzione razionale fratta:**

es.
$$\frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6} = \frac{-6}{s+3} + \frac{-4}{s+2} + \frac{3}{s+1} + 2$$

```
>> num = [2 5 3 6]; den = [1 6 11 6];
```

```
>> [r,p,k] = residue(num,den)
```

```
r =
```

```
    -6.0000
```

```
    -4.0000
```

```
     3.0000
```

```
p =
```

```
    -3.0000
```

```
    -2.0000
```

```
    -1.0000
```

```
k =
```

```
     2
```

I polinomi e le operazioni

- **Prodotto di polinomi** ($\text{pol3}=(s+1)(s+1)$):

```
>> pol1=[1 1]; pol2=[1 1];  
>> pol3=conv(pol1,pol2)  
pol3 =  
1 2 1
```

- **Divisione di polinomi** ($(s^2+2s+2)=q(s)(s+1)+r(s)$):

```
>> pol1=[1 2 2]; pol2=[1 1];  
>> [q,r]=deconv(pol1,pol2)  
q =  
1 1  
r =  
0 0 1
```

Vettori e Matrici

Esistono comandi che generano automaticamente alcune matrici notevoli

`A=eye(n);`



A è la matrice identità di ordine n

`A=zeros(n);`



A è una matrice quadrata di ordine n
i cui elementi sono zero

`A=ones(n)`



A è una matrice quadrata di ordine n i cui
elementi sono uno

Esempio: Costruzione di una matrice

Costruire una matrice 6 x 6 del tipo:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

Dove:

$$\mathbf{A}_{11} = [v_1, v_2, v_3]$$

$$\mathbf{A}_{12} = \mathbf{0}_3$$

$$\mathbf{A}_{21} = \mathbf{I}_3$$

$$\mathbf{A}_{22} = [v_3, v_2, v_1]$$

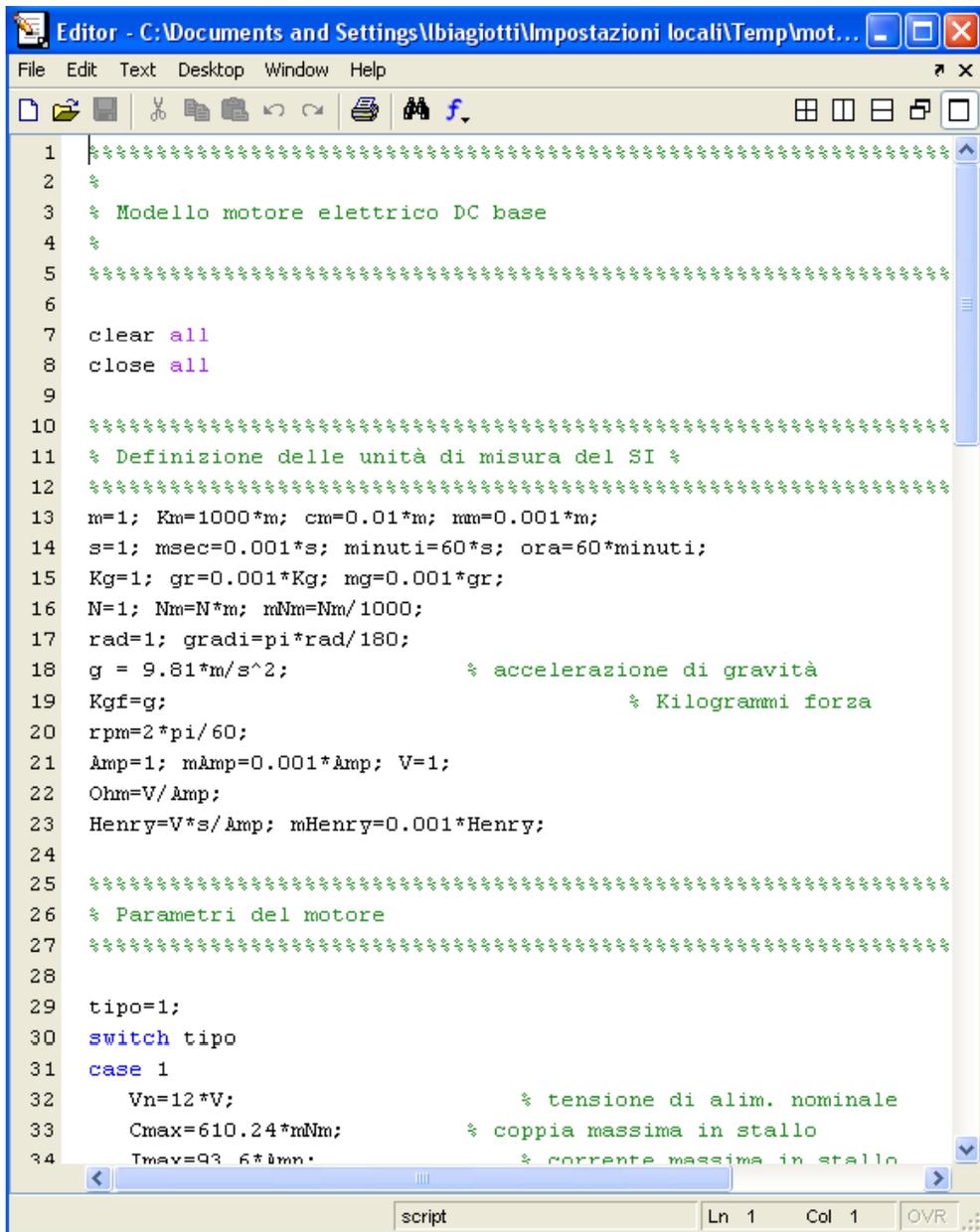
Dove v_1 , v_2 e v_3 sono vettori colonna definibili dall'utente

M-files e Matlab editor

- Matlab dispone di un editor a cui si accede mediante il comando `>> edit`
- L'editor di Matlab permette di scrivere file di comandi (con estensione `.m`) che contengono una successione di comandi che saranno eseguiti da Matlab quando si digita il nome del file (senza `.m`) nel command window.

Un m-file è solo un modo di raggruppare i comandi. Le modifiche sulle variabili non sono locali ma si riflettono direttamente sul workspace

M-files e Matlab editor



```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Modello motore elettrico DC base
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 clear all
8 close all
9
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 % Definizione delle unità di misura del SI %
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 m=1; Km=1000*m; cm=0.01*m; mm=0.001*m;
14 s=1; msec=0.001*s; minuti=60*s; ora=60*minuti;
15 Kg=1; gr=0.001*Kg; mg=0.001*gr;
16 N=1; Nm=N*m; mNm=Nm/1000;
17 rad=1; gradi=pi*rad/180;
18 g = 9.81*m/s^2;           % accelerazione di gravità
19 Kgf=g;                   % Kilogrammi forza
20 rpm=2*pi/60;
21 Amp=1; mAmp=0.001*Amp; V=1;
22 Ohm=V/Amp;
23 Henry=V*s/Amp; mHenry=0.001*Henry;
24
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 % Parametri del motore
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29 tipo=1;
30 switch tipo
31 case 1
32     Vn=12*V;               % tensione di alim. nominale
33     Cmax=610.24*mNm;       % coppia massima in stallo
34     Imax=93.6*amp;         % corrente massima in stallo
```

- I file comandi sono molto utili per facilitare tutte le operazioni con Matlab. In particolare è possibile ripetere lunghe sequenze di comandi senza doverli riscrivere direttamente nel Command Window.
- La possibilità di modificare facilmente il valore delle variabili permette di effettuare velocemente calcoli, simulazioni e confronti.
- I commenti sono preceduti da %
% questo è un commento

M-function

Possiamo aggiungere alle funzioni preesistenti, funzione costruite da noi per risolvere problemi specifici.

Le variabili definite all'interno di una function sono LOCALI

Sintassi:

```
function [output]= nomefunction(input)  
    istruzioni;
```

All'interno del blocco di istruzioni le variabili output vengono settati e il loro valore viene ritornato al termine della funzione stessa.

Non occorre usare **return** come in C.

Anche le function vengono salvate con estensione .m; il nome del file che le contiene deve essere lo stesso di **nomefunction**

Operatori matematici

- Sono definiti gli operatori matematici standard tra matrici (e vettori):
 - somma $+$
 - Differenza $-$
 - Prodotto $*$
 - divisione (a destra e a sinistra) $/$ \backslash
- Per quanto riguarda il prodotto tra matrici (e tra vettori), mentre l'operatore $*$ definisce l'operazione standard riga per colonna (**attenzione alle dimensioni**), è definito anche un operatore prodotto elemento per elemento, indicato con $.*$. Discorso analogo vale per la funzione potenza $^$, che nel caso elemento per elemento è indicata con $.^$.

Esempio:

```
>> v = [1 2 3].*[1 2 3]
```

```
v =
```

```
    1    4    9
```

Funzioni matematiche elementari

Trigonometric.

| | |
|--------------------|----------------------------------|
| <code>sin</code> | - Sine |
| <code>sinh</code> | - Hyperbolic sine. |
| <code>asin</code> | - Inverse sine. |
| <code>cos</code> | - Cosine. |
| <code>cosh</code> | - Hyperbolic cosine. |
| <code>acos</code> | - Inverse cosine. |
| <code>tan</code> | - Tangent. |
| <code>tanh</code> | - Hyperbolic tangent. |
| <code>atan</code> | - Inverse tangent. |
| <code>atan2</code> | - Four quadrant inverse tangent. |

Exponential.

| | |
|--------------------|-------------------------------|
| <code>exp</code> | - Exponential. |
| <code>log</code> | - Natural logarithm. |
| <code>log10</code> | - Common (base 10) logarithm. |
| <code>sqrt</code> | - Square root. |

Complex.

| | |
|--------------------|-------------------|
| <code>abs</code> | - Absolute value. |
| <code>angle</code> | - Phase angle. |

Rounding and remainder.

| | |
|--------------------|--|
| <code>floor</code> | - Round towards minus infinity. |
| <code>ceil</code> | - Round towards plus infinity. |
| <code>round</code> | - Round towards nearest integer. |
| <code>mod</code> | - Modulus (signed remainder after division). |
| <code>rem</code> | - Remainder after division. |
| <code>sign</code> | - Signum. |

Ecco una lista (non esaustiva) delle principali funzioni matematiche disponibili in Matlab, che includono funzioni trigonometriche, esponenziali, funzioni per la manipolazione di numeri complessi e numeri reali/interi.

NOTA: Le funzioni trigonometriche lavorano con angoli espressi in radianti

Per una lista più completa delle funzioni matematiche di base

>> help elfun

Operatori comparativi

In Matlab il valore **0** rappresenta il valore booleano **FALSE** mentre tutti gli altri valori numerici rappresentano il valore booleano **TRUE**. Si dispone di 6 operatori:

| Tipo | Sintassi |
|-------------------------|----------------------|
| Uguaglianza | eq $o ==$ |
| Disuguaglianza | ne $o \sim =$ |
| Minore di... | lt $o <$ |
| Maggiore di... | gt $o >$ |
| Maggiore o uguale di... | geq $o >=$ |
| Minore o uguale di... | leq $o <=$ |

Se l'espressione è vera ritorna un 1 altrimenti 0.

Un paragone tra due matrici viene eseguito elemento per elemento.

Operatori Logici

Operatori Binari: Il formato dell'operazione logica è $Ris=operatore(A,B)$. Gli operatori sono:

| Tipo | Sintassi | Vero se |
|------------|----------|-------------------------------------|
| AND | and o & | Entrambe le variabili sono vere |
| OR | or o | Almeno una delle due variabili vera |
| XOR | xor | Solo una delle due variabili vera |

Operatori Monari: Il formato dell'operazione logica è $Ris=operatore(A)$. Gli operatori sono:

| Tipo | Sintassi | Vero se |
|------------|----------|---|
| NOT | not | La variabile falsa |
| ANY | any | Se tutti i componenti sono veri (vettori) |
| ALL | all | Almeno uno dei componenti vero (vettori) |

Principali comandi sulle matrici

Dimensioni

```
>> [m,n]=size(A)
```

Assegna alla variabile **m** il numero di righe di **A** e alla variabile **n** il numero di colonne. Per la lunghezza di un vettore si veda il comando **length**.

Trasposta

```
>>B=A' ( oppure >>B=transpose(A) )
```

Assegna a **B** il valore \mathbf{A}^T

Determinante

```
>> d=det(A)
```

Assegna alla variabile **d** il valore del determinante di **A**, se **A** è quadrata

Principali comandi sulle matrici

Inversa

```
>> I=inv(A)
```

Assegna a **I** l'inversa della matrice **A** (se esiste!)

Rango

```
>> r=rank(A)
```

Assegna alla variabile **r** il valore del rango di **A**.

Autovalori

```
>> e=eig(A)
```

Assegna alla variabile **e** un vettore contenente gli autovalori di **A**.

Esempio: Risoluzione di un sistema lineare

Si risolva il seguente sistema:

$$\begin{cases} x_1 + x_2 + x_3 - x_4 = 1 \\ x_1 + x_2 - x_3 = 2 \\ x_1 - x_2 + x_3 = 0 \\ x_1 + 2x_2 - 3x_3 = 2 \end{cases}$$

Esempio: Risoluzione di un sistema lineare

- Abbiamo un sistema del tipo $Ax=b$.
- Passi per la risoluzione:
 - Costruire A e b
 - Verificare se A è invertibile
 - Trovare la soluzione
- Realizzare una funzione che fornisca la soluzione del sistema lineare per qualunque coppia (A,b)

Costrutti per la programmazione MATLAB

- Utilizzati per una programmazione evoluta
- Molto simili ai costrutti del C
- Consentono elaborazioni complesse dei dati

Costrutti fondamentali:

- **IF**
- **FOR**
- **WHILE**

IF

- Utilizzato quando l'esecuzione di un certo numero di istruzioni è vincolato dal soddisfacimento di un certa espressione logica.

Sintassi:

```
if (espressione logica)  
    istruzioni;  
elseif (espressione logica)  
    istruzioni  
else  
    istruzioni  
end
```

I blocchi **elseif** e **else** sono opzionali

Controllo di temperatura di una stanza

```
if (temperatura > 25)
    ariafredda = 1;
elseif (Temperatura > 20)
    ariafredda=0;
    ariacalda=0;
else
    ariacalda=1;
end
```

FOR

- Utilizzato quando un blocco di istruzioni deve essere ripetuto un ben determinato numero di volte.

Sintassi:

```
for indice=init:step:end  
  
    istruzioni;  
end
```

Step è l'incremento del contatore ad ogni ciclo. E' opzionale, se omesso vale 1

FOR: Esempio

Si vuole realizzare un vettore che contenga i valori da 0 a 5 passo 0.1

```
y=[];  
for t=0:0.1:5  
    y= [y t];  
end
```

Otteniamo $y=[0, 0.1, 0.2, \dots]$

WHILE

- Utilizzato quando un blocco di istruzioni deve essere ripetuto finché una condizione logica risulta vera.

Sintassi:

```
while (espressione logica)
```

```
    istruzioni;
```

```
end
```

WHILE: Esempio

Controllo di temperatura di una stanza

```
while (Temperatura<25)
    ariacalda=1;
end
```

BREAK

- Utilizzato quando si deve interrompere un ciclo (for o while) prima del previsto.

Esempio

```
for i=1:1000
    a(i)=y(i)
    if (a(i) > 1000)
        break;
    end
end
```

Per maggiori informazioni consultare l'help a matlab/lang

Grafici in Matlab

- Per graficare l'andamento di una variabile si utilizza il comando `plot`.
- `>> plot(x,y)` crea un grafico nella cui ordinata sono presenti i valori del vettore `y` e nella cui ascissa i valori del vettore `x`.

ATTENZIONE!!!!

I vettori `x` e `y` devono avere le stesse dimensioni!

- E possibile anche specificare lo stile del plottaggio. Ad esempio con `>> plot(x,y,'m--')`
- Esempi: plottare semplici funzioni come `sin(t)`, `cos(t)` per `t` da 0 a 10

Grafici in Matlab

- Il comando `plot` traccia un grafico sull'ultima figura aperta, cancellando il grafico precedente. Per creare una nuova figura si usa il comando `figure`
- Il comando `hold` consente di “bloccare” l'immagine sul grafico. Il prossimo comando `plot` disegnerà sulla stessa finestra ma **non** cancellerà il grafico precedente. In questo caso si può usare il comando `legend` per aggiungere una legenda
- Il comando `grid` consente di sovrapporre una griglia al grafico
- Con `xlabel`, `ylabel` si possono aggiungere etichette agli assi, e con il comando `axis` si possono cambiare i limiti del plottaggio
- È possibile tracciare grafici di diversi colori e con diverse tipologie di linea e disegnare più grafici nella stessa figura (comando `subplot`).
- Il comando `print` consente di salvare i grafici in figure di diversi formati (eps, jpeg, tiff).
- Per ulteriori dettagli: `help plot`

TEORIA DEI SISTEMI E DEL CONTROLLO LM in Ingegneria Informatica e Ingegneria Elettronica

<http://www.dii.unimore.it/~lbiagiotti/TeoriaSistemiControllo.html>

INTRODUZIONE A MATLAB

Ing. Luigi Biagiotti

e-mail: luigi.biagiotti@unimore.it

<http://www.dii.unimore.it/~lbiagiotti>