



UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Corso di Laurea:
Insegnamento:
Lezione:
Docente:

Ingegneria Informatica e dell'Automazione (Magistrale)
Manutenzione preventiva per la robotica e l'automazione intelligente
E – Introduzione a MATLAB e Simulink
Alessandro Freddi

Manutenzione preventiva per la robotica e l'automazione intelligente

E – Introduzione a MATLAB e Simulink

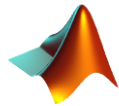


INTRODUZIONE

- ❑ MATLAB è un linguaggio di alto livello e un ambiente interattivo per il calcolo numerico, la visualizzazione e la programmazione prodotto dalla Mathworks.
 - ❑ Il linguaggio, gli strumenti e le funzioni matematiche integrate permettono di risolvere problemi in maniera intuitiva e più “semplice” rispetto ai tradizionali linguaggi di programmazione quali il C/C++ o il Java.
 - ❑ MATLAB può essere usato in numerosi campi applicativi: processamento di segnali e telecomunicazioni, analisi di immagini e video, sistemi di controllo, modellazione, diagnosi, test, misura, finanza, biologia, ecc ...
 - ❑ Caratteristiche chiave:
 - ❑ linguaggio di alto livello per il calcolo numerico, la visualizzazione e lo sviluppo di applicazioni;
 - ❑ ambiente interattivo per l'esplorazione iterativa, la visualizzazione e la risoluzione di problemi;
 - ❑ funzioni matematiche per l'algebra lineare, la statistica, l'analisi di Fourier, il filtraggio, l'ottimizzazione, l'integrazione numerica e la risoluzione di equazioni differenziali ordinarie;
 - ❑ grafica integrata per la visualizzazione di dati e strumenti di creazione di grafici;
 - ❑ strumenti di sviluppo per il miglioramento della qualità e manutenibilità del codice;
 - ❑ strumenti per lo sviluppo di applicazioni con interfacce personalizzabili;
 - ❑ funzioni per l'integrazione di algoritmi sviluppati in MATLAB con linguaggi di applicazione esterna, quali C, Java, .NET e Microsoft Excel.
-



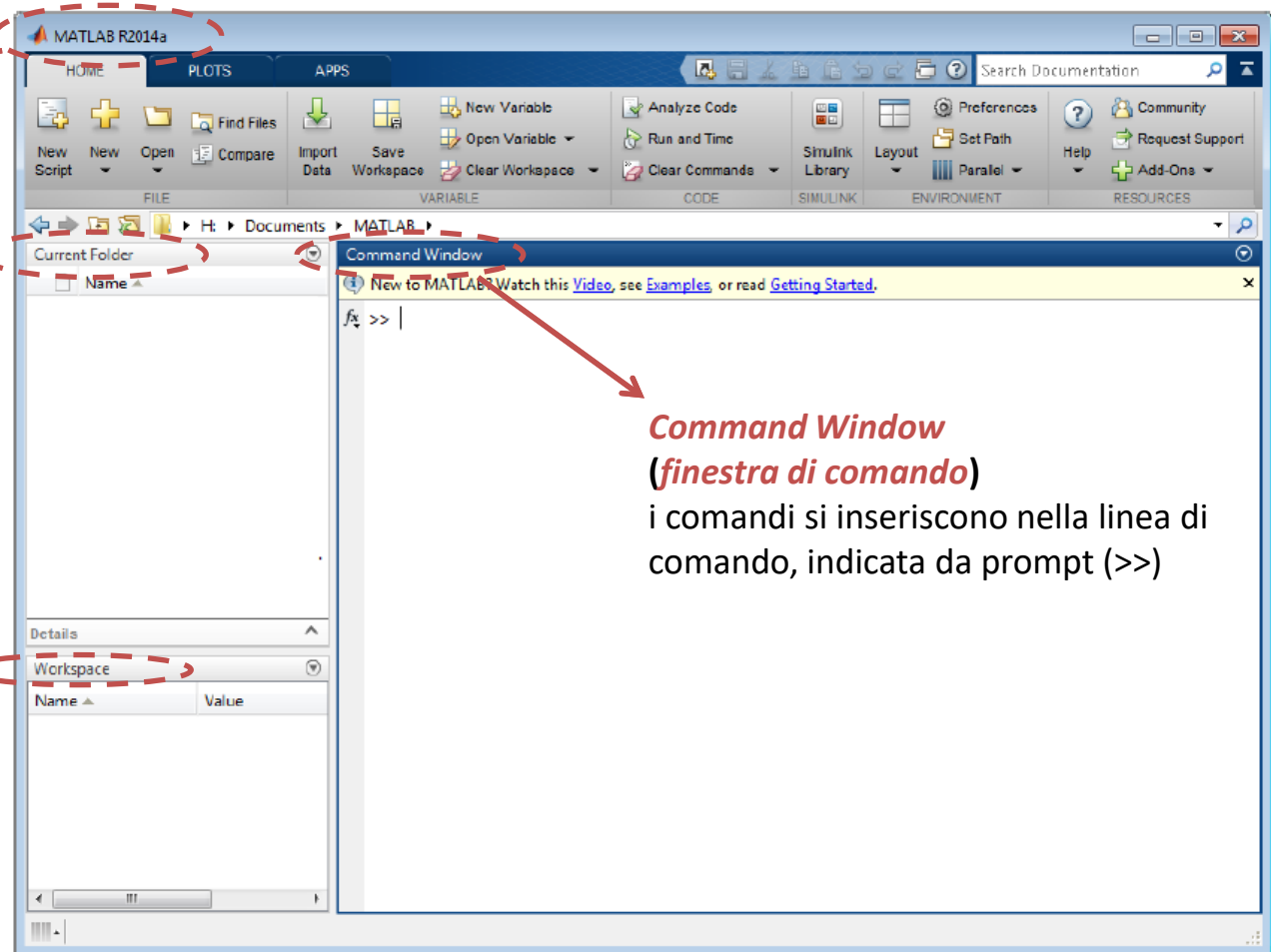
BASI DEL DESKTOP: L'INTERFACCIA GRAFICA



Simbolo e versione

Current Folder
(*directory corrente*)
permette di accedere ai propri file

Workspace
(*spazio di lavoro*)
esplora i dati creati o importati da file



Command Window
(*finestra di comando*)
i comandi si inseriscono nella linea di comando, indicata da prompt (>>)



BASI DEL DESKTOP: I COMANDI

- ❑ All'avvio di MATLAB, nella sezione di inserimento dei comandi (i.e. la *Command Window*), appare il prompt “>>” all’interno della quale eseguire i comandi.
- ❑ Vi sono due tipi di comandi:
 - assegnamenti** | >> variabile = espressione
asigna il valore “espressione” a “variabile”;
 - valutazione di espressioni** | >> espressione
genera una matrice che viene assegnata alla variabile indicata.
- ❑ Quando nell'istruzione non si specifica la variabile a cui assegnare il risultato, la valutazione dell'espressione viene assegnata alla variabile di sistema *ans* (abbreviazione di “answer”).
- ❑ Se un’espressione non termina con il punto e virgola il risultato della sua valutazione viene mostrato anche sullo schermo (si consiglia di utilizzare sempre il punto e virgola nella programmazione in modo da non visualizzare continuamente le uscite a schermo e rallentare l’esecuzione dei programmi).
- ❑ E’ possibile richiamare comandi mediante le frecce direzionali, su linea vuota oppure dopo aver specificato l’iniziale del comando.
- ❑ In MATLAB le variabili non devono essere dichiarate.
- ❑ MATLAB è case-sensitive.

```
>> 3+2  
ans =  
5
```

```
>> a = 3*ans  
a =  
15
```



ARRAY: DEFINIZIONE DI UN VETTORE

- ❑ MATLAB è l'abbreviazione di *MATrix LABoratory*: tutte le variabili di MATLAB sono array multidimensionali, a prescindere dal tipo di dato in essi contenuto.
- ❑ Un **vettore riga** (array monodimensionale) può essere definito inserendo gli elementi tra parentesi quadre, separati da una virgola oppure da uno spazio:

```
>> a = [1,2,3]
```

```
a =
```

```
1      2      3
```

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

```
a =
```

```
1      2      3
```

- ❑ **a : step : b** crea un vettore riga di estremi *a* e *b*; il parametro *step* è opzionale e indica l'intervallo tra ciascun elemento del vettore (valore di default pari a 1).

```
>> a = 0:2:6
```

```
a =
```

```
0      2      4      6
```

```
>> a = 0:6
```

```
a =
```

```
0      1      2      3      4      5      6
```



ARRAY: DEFINIZIONE DI UN VETTORE

- ❑ `linspace(a,b,N)` crea un vettore riga di estremi a e b , costituito da N punti equispaziati.

```
>> a = linspace(0,6,1)
```

```
a =
```

```
6
```

```
>> a = linspace(0,6,7)
```

```
a =
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

- ❑ Un **vettore colonna** può essere definito separando gli elementi con un punto e virgola o un cambio di riga:

```
>> a = [1;2;3]
```

```
a =
```

```
1
```

```
2
```

```
3
```

```
>> a = [1
```

```
2
```

```
3]
```

```
a =
```

```
1
```

```
2
```

```
3
```



ARRAY: DEFINIZIONE DI UNA MATRICE

- Per creare una **matrice** (array bidimensionale) è necessario separare i vettori riga con il punto e virgola oppure un cambio di riga:

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

A =

1	2	3
4	5	6
7	8	9

```
>> A = [1 2 3
```

```
4 5 6
```

```
7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

- MATLAB permette di operare su tutti gli elementi di una matrice con un singolo operatore:

```
>> B = A+1
```

B =

2	3	4
5	6	7
8	9	10

```
>> C = sin(A)
```

C =

0.8415	0.9093	0.1411
-0.7568	-0.9589	-0.2794
0.6570	0.9894	0.4121



ARRAY: DEFINIZIONE DI UNA MATRICE

- ❑ `zeros(n,m)` matrice di “0” di dimensione $n \times m$;
- ❑ `ones(n,m)` matrice di “1” di dimensione $n \times m$;
- ❑ `eye(n,m)` matrice identità di dimensione $n \times m$;

```
>> Z = zeros(2,3)
```

Z =

0	0	0
0	0	0

```
>> O = ones(2,3)
```

O =

1	1	1
1	1	1

```
>> I = eye(2,2)
```

I =

1	0
0	1

- ❑ `rand(n,m)` matrice di numeri casuali di dimensione $n \times m$;
- ❑ `diag([a11,a22,a33,...,ann])` matrice diagonale i cui elementi sono $a_{11} \dots a_{nn}$.

```
>> R = rand(2,3)
```

R =

0.8147	0.1270	0.6324
0.9058	0.9134	0.0975

```
>> D = diag([1,2])
```

D =

1	0
0	2



ARRAY: ACCESSO AI SINGOLI ELEMENTI (INDEXING)

□ Data la matrice A di dimensione $n \times m$ è possibile accedere/estrarre i singoli elementi mediante i seguenti comandi:

- $A(i, j)$ elemento (i, j) della matrice A ;
- $A(k)$ k -esimo elemento, contato in ordine colonna, della matrice A ;
- $A(n, :)$ n -esima riga della matrice A ;
- $A(:, m)$ m -esima colonna della matrice A .

```
>> A = [1, 2; 3, 4]

A =

     1     2
     3     4
```

```
>> A(1, 2)

ans =

     2
```

```
>> A(2)

ans =

     3
```

```
>> A(2, :)

ans =

     3     4
```

```
>> A(:, 1)

ans =

     1
     3
```

□ Se si prova ad accedere ad un elemento al di fuori delle dimensioni n o m si ottiene un messaggio di errore:

```
>> A(3, 1)
Index exceeds matrix dimensions.
```



WORKSPACE

- ❑ *Workspace* contiene le variabili che l'utente crea o importa in MATLAB.
- ❑ Le variabili nel workspace sono visualizzabili con il comando `whos` oppure nel pannello workspace presente sul desktop:

```
>> A = magic(4);  
B = rand(3,5,2);
```

Name	Value	Min	Max
A	4x4 double	1	16
B	3x5x2 double	0.0357	0.9706

```
>> whos
```

Name	Size	Bytes	Class
Attributes			
A	4x4	128	double
B	3x5x2	240	double

- ❑ Le variabili presenti nel workspace non rimangono all'uscita del MATLAB: pertanto è necessario salvare i dati mediante il comando `save` prima di uscire, il quale salva tutte le variabili contenute nel workspace in un file di estensione `.mat`; i dati così salvati sono poi recuperabili al successivo avvio di MATLAB mediante il comando `load`:

```
>> save myfile.mat
```

```
>> load myfile.mat
```

- ❑ Per cancellare le variabili dal workspace si usa il comando `clear`.



STRINGHE DI CARATTERI

- Una **stringa** di caratteri è una **qualunque sequenza di caratteri racchiusa tra apici**: si può assegnare una stringa ad una variabile.

```
>> mytext = 'Hello, World'

mytext =

Hello, World
```

- Una variabile contenente una stringa è sempre un array in MATLAB, la cui classe (o tipo di dato) è `char` (abbreviazione di *character*).

```
>> whos mytext
```

Name	Size	Bytes	Class
Attributes			
mytext	1x10	20	char

- Per tradurre valori numerici in stringhe è possibile avvalersi delle funzioni integrate di MATLAB, quale ad esempio `num2str`.



CHIAMARE LE FUNZIONI

- ❑ MATLAB fornisce un ampio numero di funzioni che eseguono elaborazioni.
- ❑ Le funzioni sono equivalenti a “subroutines” o “metodi” in altri linguaggi di programmazione.
- ❑ Per richiamare una funzione, ad esempio `max`, è necessario racchiudere i suoi argomenti tra parentesi tonde.
- ❑ Se la funzione ha più argomenti è necessario separarli con la virgola.
- ❑ Per memorizzare il valore di uscita di una funzione lo si assegna ad una variabile.
- ❑ Se la funzione restituisce uscite multiple, allora si assegnano tali uscite ad un vettore riga.

```
>> A = [1 3 5];  
max(A)
```

```
ans =  
  
5
```

```
>> B = [10 6 4];  
max(A,B)
```

```
ans =  
  
5      10      6
```

```
>> maxA = max(A)  
  
maxA =
```

```
5
```

```
>> [maxA,location] = max(A)  
  
maxA =
```

```
5
```

```
location =
```

```
3
```

- ❑ Per richiamare una funzione senza ingressi e uscite basta scrivere il nome della funzione (e.g. `clc` pulisce la finestra di comando).



OPERATORI E FUNZIONI MATEMATICHE ELEMENTARI PER SCALARI

□ I principali **operatori aritmetici** presenti in MATLAB sono:

- **+** e **-** somma e differenza;
- ***** e **/** prodotto e quoziente,
- **^** elevamento a potenza.

```
>> x = 2 * ((3+2-4) ^2 /13)

x =

    0.1538
```

□ Le **funzioni matematiche** elementari in MATLAB sono:

- **abs** modulo (anche di un numero complesso);
- **angle** fase di un numero complesso;
- **conj** complesso coniugato;
- **exp** elevamento a potenza in base *e*;
- **real** parte reale di un numero complesso;
- **imag** parte immaginaria di un numero complesso;
- **log** logaritmo naturale;
- **log10** logaritmo in base 10;
- **sqrt** radice quadrata.

```
>> z = 10 + 4i

z =

    10.0000 + 4.0000i
```

```
>> Z =
log10(real(z))+sqrt(imag(z))

Z =

     3
```



FUNZIONI TRIGONOMETRICHE ELEMENTARI PER SCALARI E COSTANTI

□ Le principali **funzioni trigonometriche** in MATLAB sono:

- `sin` seno;
- `cos` coseno;
- `tan` tangente;
- `asin` arcoseno;
- `acos` arcocoseno;
- `atan` arcotangente.

```
>> id=asin(sin(1))+acos(cos(1))-sin(1)^2-cos(1)^2  
  
id =  
  
1
```

□ Le principali **costanti** definite in MATLAB sono:

- `i o j` unità complessa;
- `e` costante di Eulero;
- `pi` π ;
- `Inf` ∞ ;
- `NaN` “not a number”;
- `realmax` massimo numero esprimibile;
- `realmin` minimo numero esprimibile;
- `eps` precisione macchina.

} dipendono dal
calcolatore

```
>> z = 1/0  
  
z =  
  
Inf
```

```
>> z = 0/0  
  
z =  
  
NaN
```

```
>> eps  
  
ans =  
  
2.2204e-16
```



OPERATORI E FUNZIONI MATEMATICHE ELEMENTARI PER MATRICI

- ❑ Le funzioni elementari per matrici sono le stesse viste per gli scalari, ad eccezione di:
 - ❑ `'` trasposizione complessa coniugata;
 - ❑ `\` divisione sinistra (i.e. $X = A \backslash B$ è soluzione dell'equazione $A * X = B$).
- ❑ L'operazione di somma o di sottrazione è definita tra matrici aventi le stesse dimensioni.
- ❑ Se uno dei due operandi è uno scalare, esso viene sommato o sottratto a tutti gli elementi della matrice.
- ❑ `.*`, `./` e `.^` effettuano le corrispondenti operazioni sui singoli elementi delle matrici coinvolte.
- ❑ Le funzioni matematiche elementari e trigonometriche, quando applicate alle matrici, si riferiscono ai singoli elementi della matrice.
- ❑ Le principali funzioni per matrici sono:
 - ❑ `size` dimensioni;
 - ❑ `det` determinante;
 - ❑ `rank` rango;
 - ❑ `eig` autovalori.

```
>> A = [1,3;4,2]
```

```
A =
```

```
1    3  
4    2
```

```
>> det(A)
```

```
ans =
```

```
-10
```

```
>> size(A)
```

```
ans =
```

```
2    2
```

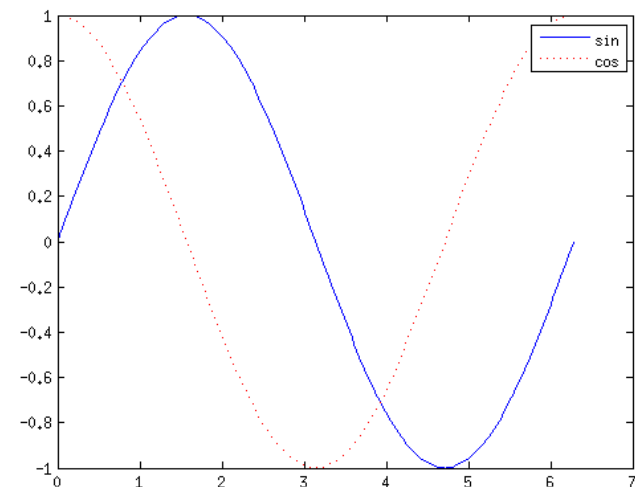


REALIZZARE GRAFICI BIDIMENSIONALI

- ❑ Per creare grafici bidimensionali si usa la funzione `plot`.
- ❑ Il comando `plot` può essere integrato mediante argomenti aggiuntivi oppure ulteriori funzioni grafiche:

- ❑ `hold on` rappresenta sullo stesso grafico più curve;
- ❑ `xlabel` e `ylabel` etichette sugli assi x e y;
- ❑ `grid on` imposta una griglia;
- ❑ `'r:'` argomento che specifica colore rosso della linea (r) e formato a punti (:);
- ❑ `legend` crea una legenda;
- ❑ `ecc ..`

```
x = 0:pi/100:2*pi;  
y = sin(x);  
plot(x,y)  
hold on  
y2 = cos(x);  
plot(x,y2,'r:')  
legend('sin','cos')
```

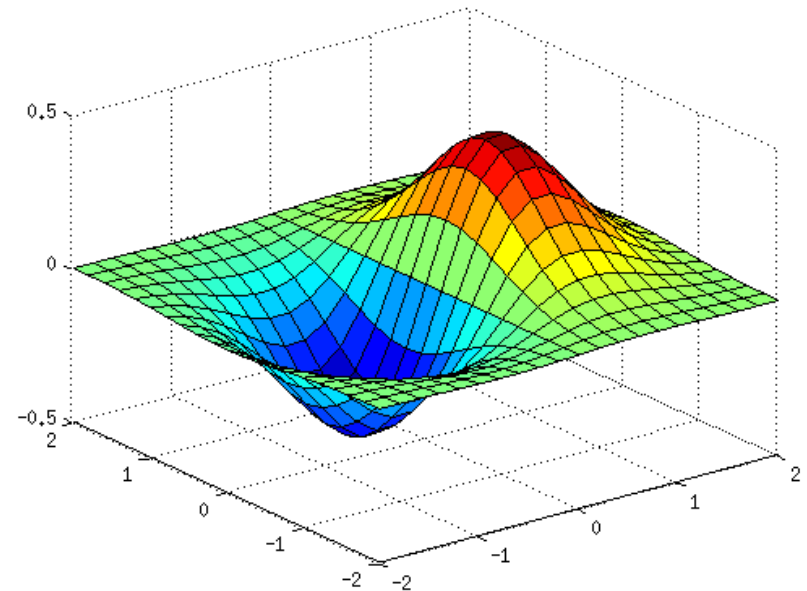




REALIZZARE GRAFICI TRIDIMENSIONALI

- ❑ Un grafico tridimensionale raffigura tipicamente una superficie definita come funzione di due variabili, e.g. $z = f(x,y)$.
- ❑ Per disegnare z è prima necessario definire il set di punti (x,y) del dominio della funzione con `meshgrid` (analogamente a come si definisce in due dimensioni il dominio della funzione con un vettore riga); poi si definisce z in funzione di (x,y) e si usa la funzione `surf`.

```
>> [X,Y] = meshgrid(-2:.2:2);  
Z = X .* exp(-X.^2 - Y.^2);  
surf(X,Y,Z)
```





PROGRAMMAZIONE E SCRIPT

- ❑ L'esempio più semplice di programma MATLAB è detto *script*: **uno script è un file con estensione .m che contiene linee multiple e sequenziali di comandi e chiamate a funzione MATLAB.**
 - ❑ Si può eseguire uno script digitando il suo nome nella linea di comando oppure posizionandosi su di esso con il cursore del mouse all'interno della directory corrente e cliccando esegui (o "F9" da tastiera).
 - ❑ Ogni volta che si scrive codice in MATLAB è buona pratica aggiungere commenti che descrivano il codice: il simbolo che denota i commenti in MATLAB è il `%`.
 - ❑ All'interno di uno script è possibile eseguire cicli di codice avvalendosi delle espressioni condizionali **for**, **while**, **if** e **switch**.
 - ❑ Affinché MATLAB possa eseguire uno script è necessario che esso si trovi nella directory corrente, oppure all'interno del percorso di ricerca che è modificabile dall'utente.
 - ❑ Di default la directory di installazione di MATLAB è inserita nel percorso di ricerca: per utilizzare programmi installati in altre directory e aggiungerle al percorso è sufficiente selezionare la cartella di interesse, cliccare con il tasto destro del mouse e selezionare **Add to Path**.
-



HELP E DOCUMENTAZIONE

- ❑ Tutte le funzioni MATLAB hanno una documentazione di supporto che include esempi e descrive gli argomenti, le uscite e la sintassi.
- ❑ Ci sono molteplici modi di accedere alla documentazione:
 - ❑ `doc` apre la documentazione in una pagina separata;
 - ❑ `help` visualizza una versione ridotta della documentazione direttamente nella finestra di comando;
 - ❑ digitare il nome della funzione seguito da parentesi tonda per ottenere suggerimenti in linea.

```
>> help sin
sin      Sine of argument in radians.
sin(X) is the sine of the elements of X.

See also asin, sind.

Overloaded methods:
codistributed/sin

Reference page in Help browser
doc sin
More Help...

fx >> sin(
```



INTRODUZIONE

- ❑ Simulink è un ambiente grafico a blocchi per la progettazione basata su modello, la simulazione e l'analisi di processi appartenenti a molteplici domini di applicazione.
 - ❑ E' integrato con MATLAB, permette di incorporare algoritmi MATLAB all'interno di modelli e di esportare i risultati delle simulazioni all'interno di MATLAB per ulteriori analisi.
 - ❑ Le sue caratteristiche fondamentali sono:
 - ❑ editor grafico per costruire e gestire diagrammi a blocchi di natura gerarchica;
 - ❑ librerie predefinite per la modellazione di sistemi sia a tempo continuo sia a tempo discreto;
 - ❑ motore di simulazione con risolutori per le equazioni differenziali ordinarie (*Ordinary Differential Equation, ODE*);
 - ❑ memorizzazione e visualizzazione dei risultati della simulazione;
 - ❑ strumenti per la gestione dei dati e del progetto;
 - ❑ strumenti per l'analisi dei modelli;
 - ❑ blocchi per l'importazione di codice MATLAB all'interno dei modelli Simulink;
 - ❑ blocchi per l'importazione di codice C e C++ all'interno dei modelli Simulink;
 - ❑ blocchi per la progettazione di sistemi dinamici in linea.
-



STRUMENTI PER LA PROGETTAZIONE, LA SIMULAZIONE E L'ANALISI

Progettazione

- ❑ Con Simulink è possibile modellare sistemi di diversa tipologia mediante un'interfaccia grafica che permette l'accesso a **librerie costituite da blocchi prefediniti** (e.g.: sorgenti, segnali, componenti lineari, componenti nonlineari, connettori, funzioni matematiche, ecc ...).
- ❑ Dopo aver selezionato i blocchi di interesse, il progettista ha il compito di **connetterli tra di loro nella maniera desiderata**, in modo da decidere le modalità con cui i dati devono transitare all'interno di ogni blocco.
- ❑ Se i blocchi messi a disposizione da Simulink non soddisfano le esigenze del progettista, è possibile impiegare **blocchi personalizzabili all'interno dei quali scrivere il codice desiderato**.
- ❑ I **modelli** generati sono **gerarchici**: facendo doppio click su un blocco di livello superiore si accede al blocco (o ai blocchi) di livello inferiore, fino ad arrivare al livello più basso.

Simulazione

- ❑ Dopo aver definito il modello è possibile simularlo impiegando **differenti metodi di integrazione** selezionabili sia dai menu in Simulink sia mediante comando in linea MATLAB:
 - ❑ la simulazione gestita mediante menu è comoda per un approccio interattivo;
 - ❑ la simulazione mediante comandi in linea MATLAB è adatta ad un approccio di tipo *batch* (e.g. Monte Carlo).
 - ❑ Mediante il blocco **Scope** è possibile visualizzare i risultati di una simulazione in tempo reale.
 - ❑ I risultati della simulazione possono essere sempre salvati, comunque, nello spazio di lavoro MATLAB per un'analisi successiva.
-

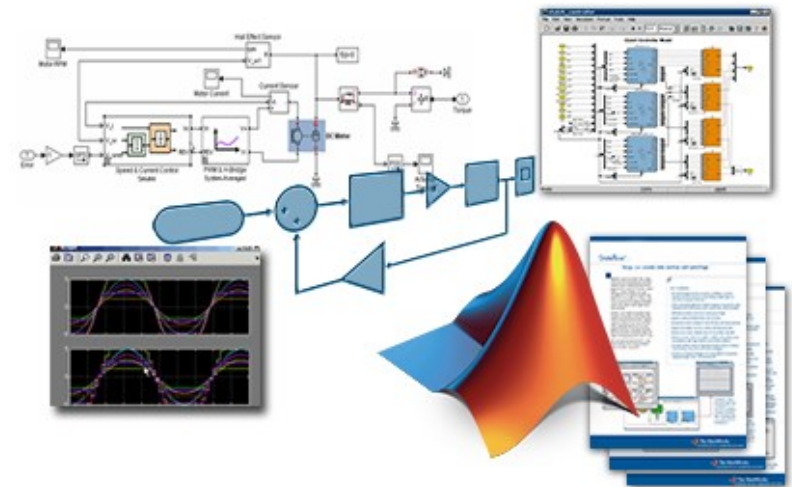
STRUMENTI PER LA PROGETTAZIONE, LA SIMULAZIONE E L'ANALISI

Analisi

- ❑ Gli strumenti per l'analisi includono la linearizzazione, il trimming e molti altri.
- ❑ L'integrazione tra Simulink e MATLAB, inoltre, permette di simulare, analizzare e revisionare i modelli in ogni loro componente e ad ogni istante di simulazione.

Integrazione con MATLAB

- ❑ Simulink è fortemente integrato con l'ambiente MATLAB e richiede MATLAB per funzionare, in quanto dipende da esso per la definizione e il calcolo dei parametri di blocchi e modelli.
- ❑ Questo permette di sfruttare in Simulink molte delle funzionalità di MATLAB, ad esempio
 - ❑ definire gli ingressi del modello;
 - ❑ conservare le uscite del modello per una loro visualizzazione o analisi;
 - ❑ chiamare le funzioni o gli operatori di MATLAB all'interno di un blocco Simulink.



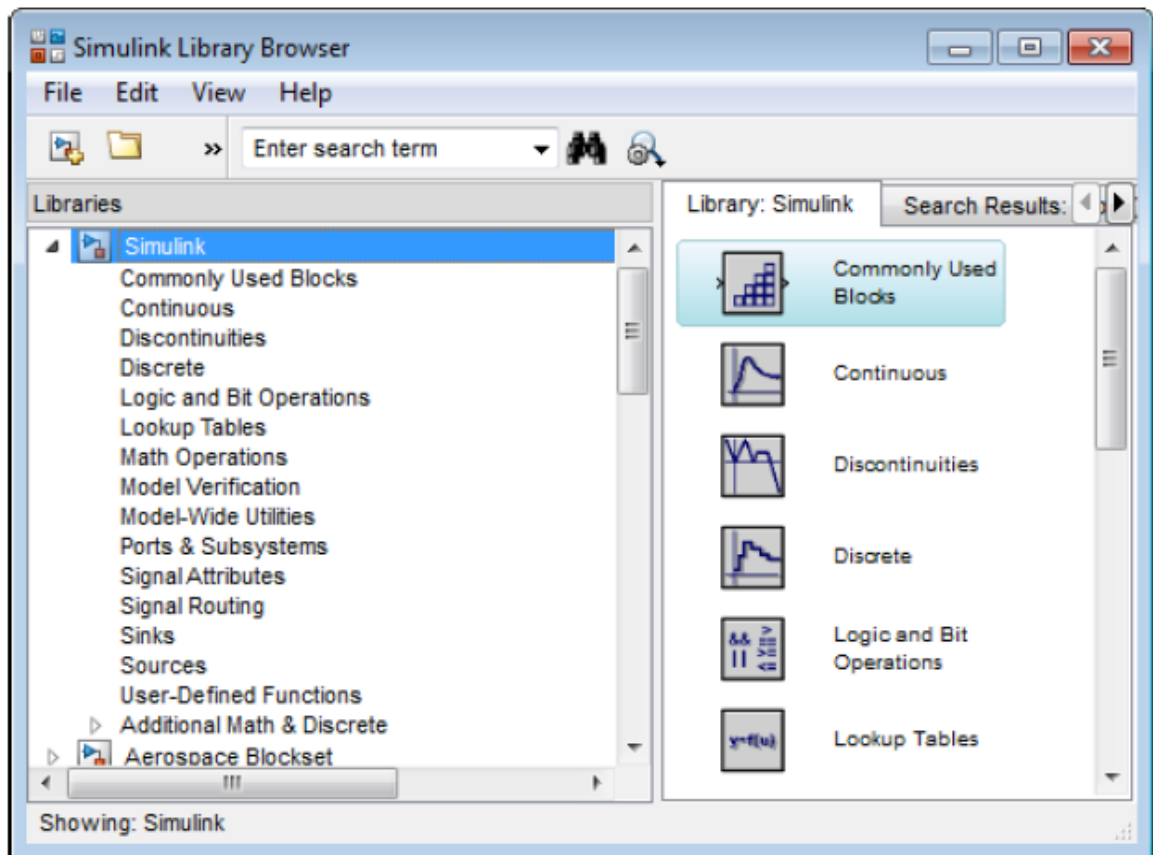


APRIRE SIMULINK

E' necessario aver avviato MATLAB prima di poter aprire il *Simulink Library Browser* (gestore librerie Simulink).

Per accedere al browser ci sono due possibilità:

1. digitare il comando `simulink` all'interno della command window di MATLAB;
2. cliccare sul bottone *Simulink Library* (libreria Simulink) all'interno di MATLAB.





CREARE UN NUOVO MODELLO

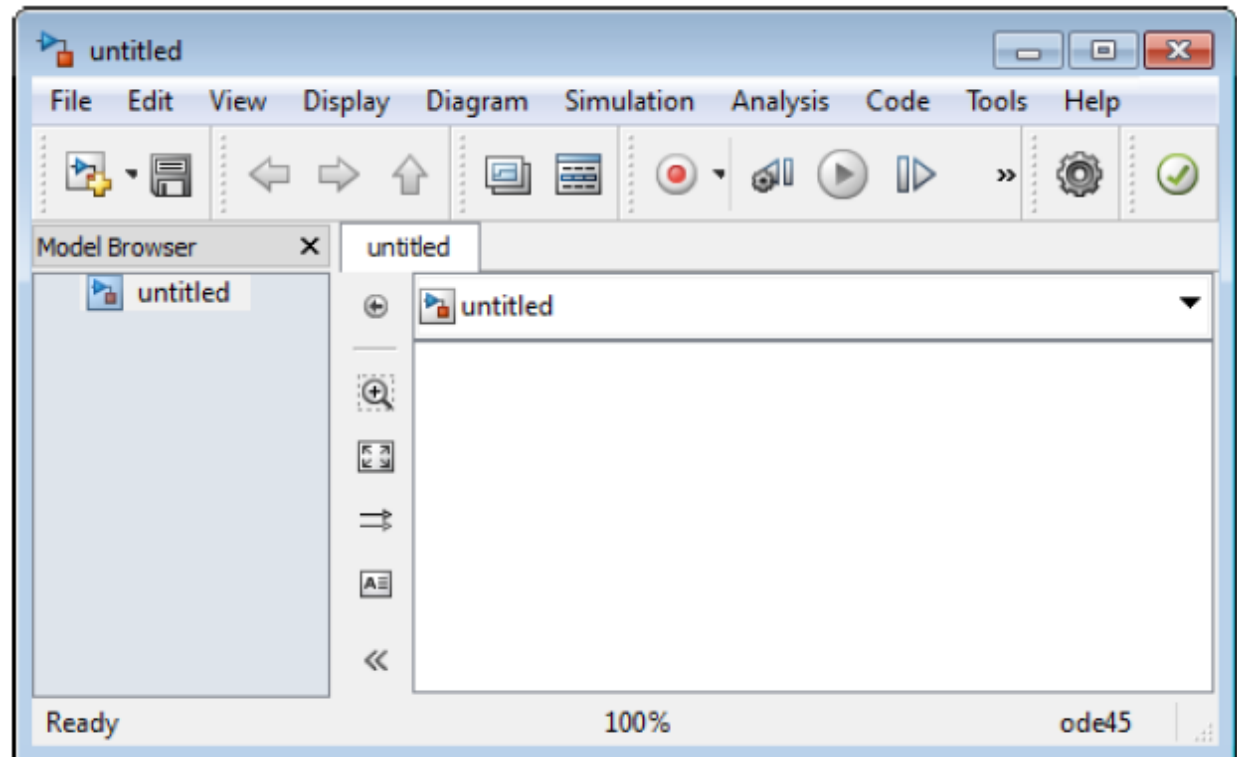
- ❑ Selezionare **File > New > Model** dal Simulink Library Browser per creare un modello vuoto.
- ❑ Selezionare **File > Save** dal *Simulink Editor* e successivamente inserire il nome che si vuole attribuire al modello per salvarlo.

Simulink Editor

Finestra all'interno della quale è possibile operare sul modello Simulink

Estensioni MATLAB/Simulink

.m / script e funzioni
.mdl / modello Simulink
.mat / variabili
.fig / figure



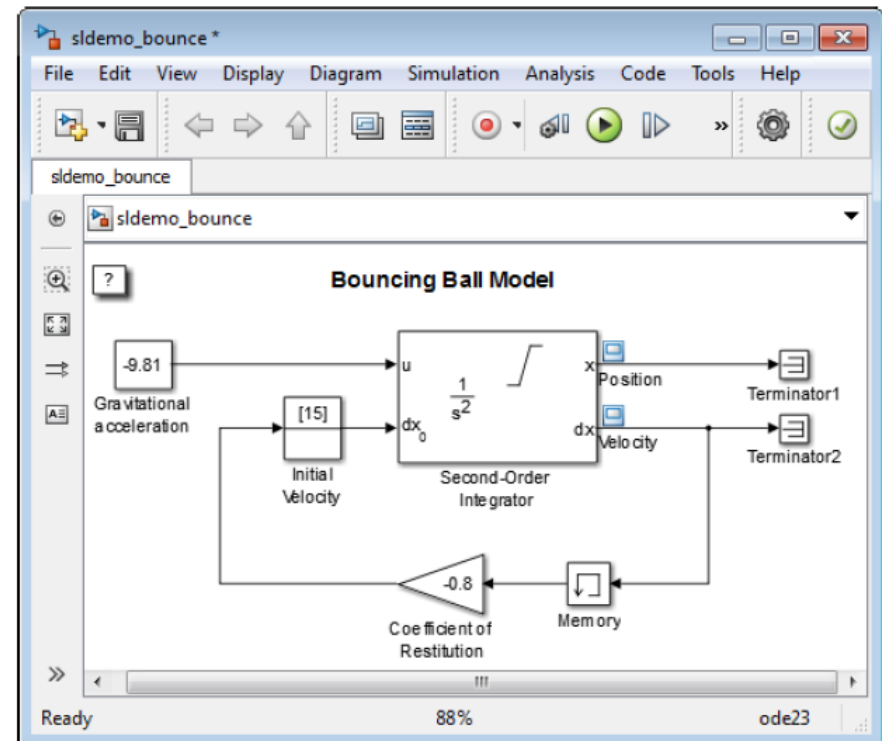


APRIRE UN MODELLO ESISTENTE

- ❑ Selezionare **File > Open** dal Simulink Library Browser e, nella finestra che segue, selezionare il modello desiderato.
- ❑ Il modello selezionato viene aperto all'interno del Simulink Editor (e.g. "Bouncing Ball Model").

Alternativa

E' possibile impostare la directory corrente su quella che contiene il modello di interesse, e digitare poi il nome del modello nella finestra di comando MATLAB per aprirlo.



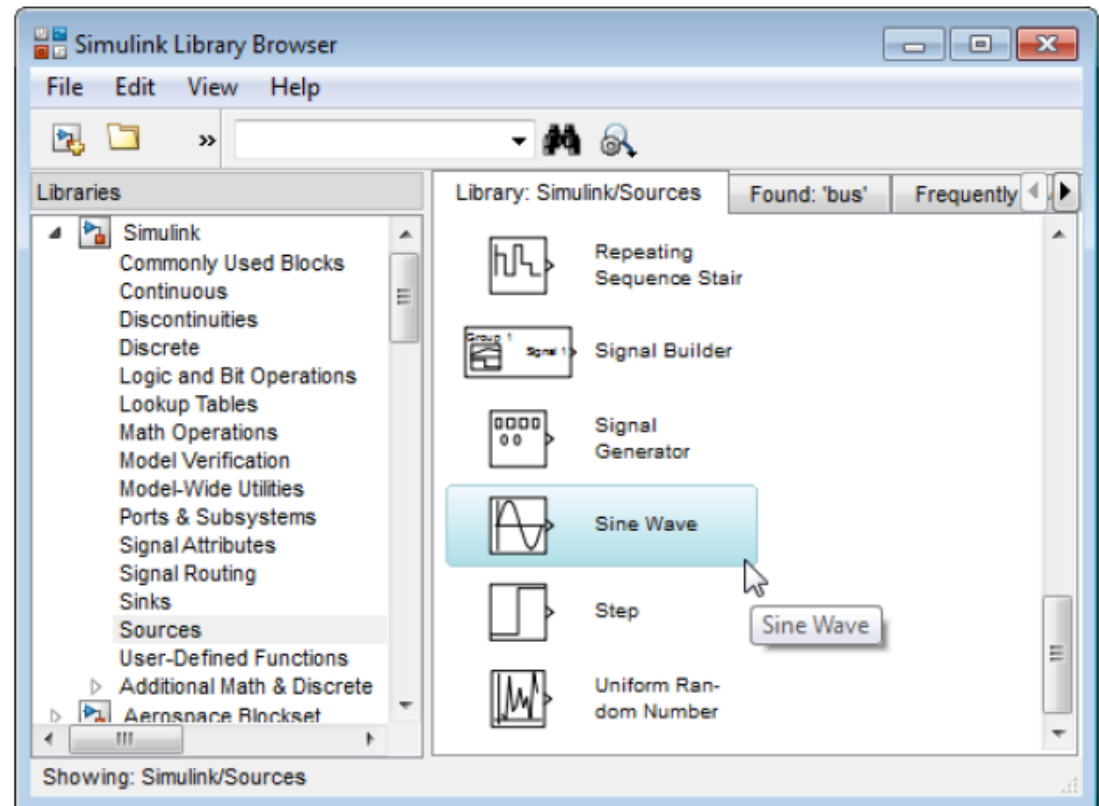


INTERFACCIA UTENTE: IL BROWSER

- ❑ Il Simulink Library Browser mostra le librerie dei blocchi che sono installate nel computer.
- ❑ Per iniziare a costruire un modello occorre copiare i blocchi da una libreria all'interno del Simulink Editor.


Esempio

- selezionare la libreria *Sources*
- selezionare il blocco *Sine Wave*





INTERFACCIA UTENTE: IL BROWSER

Compito	Azione da eseguire nel Library Browser
Visualizzare i blocchi in una libreria	Selezionare il nome della libreria nel pannello sinistro, oppure cliccare sull'icona della libreria nel pannello destro.
Cercare un blocco specifico	Digitare il nome del blocco nel campo di ricerca e cliccare sull'icona di ricerca 
Ottenere informazioni di riepilogo di un blocco	Selezionare View > Show Block Descriptions e poi selezionare il blocco corrispondente.
Ottenere informazioni dettagliate di un blocco	Selezionare il blocco e poi Help > Help for the Selected Block . L' <i>Help browser</i> si apre sulla pagina di riferimento per il blocco selezionato.
Visualizzare i parametri di un blocco	Tasto destro su un blocco, poi selezionare Block parameters .
Copia un blocco dal Library Browser su di un modello	Tasto sinistro, poi trascinare il blocco dal Library Browser al Simulink Editor.



INTERFACCIA UTENTE: LIBRERIE DEI BLOCCHI STANDARD

Libreria	Descrizione
Commonly Used Blocks	Blocchi di uso più frequente, quali Constant , In1 , Out1 , Scope e Sum . Ognuno dei blocchi contenuti in questa libreria fa anche parte di una libreria specifica.
Continuous	Per modelli a tempo continuo (e.g. blocchi Derivative e Integrator).
Discontinuities	Per la creazione di uscite che sono funzioni discontinue degli ingressi, quali ad esempio la saturazione (e.g. blocco Saturation).
Discrete	Per modelli a tempo discreto (e.g. blocco Unit Delay).
Logic and Bit Operations	Per le funzioni logiche o su bit (e.g. blocchi Logical Operator e Relational Operator).
Lookup Tables	Permettono di determinare le uscite a partire dai valori di ingresso e da relazioni tabellari.
Math Operations	Funzioni matematiche standard (e.g. blocchi Gain , Product e Sum).
Model Verification	Crea modelli auto validanti (e.g. blocco Check Input Resolution).



INTERFACCIA UTENTE: LIBRERIE DEI BLOCCHI STANDARD

Libreria	Descrizione
Model-Wide Utilities	Blocchi per fornire informazioni sul modello (e.g. blocco Model Info).
Ports & Subsystems	Per la creazione di sottosistemi e la connessione di ingressi e uscite (e.g. blocchi In1 , Out1 , Subsystem).
Signal Attributes	Per la modifica degli attributi dei segnali (e.g. blocco Data Type Conversion).
Signal Routing	Per la connessione tra blocchi e l'instradamento dei segnali (e.g. blocchi Mux e Switch).
Sinks	Visualizzazione o esportazione delle uscite (e.g. blocchi Out1 e Scope).
Sources	Generazione o importazione degli ingressi di sistema (e.g. blocchi Constant , In1 e Sine Wave).
User-Defined Function	Per la definizione di blocchi personalizzati (e.g. blocco MATLAB Function).
Additional Math & Discrete	Librerie aggiuntive per le funzioni matematiche e di elaborazione a tempo discreto.

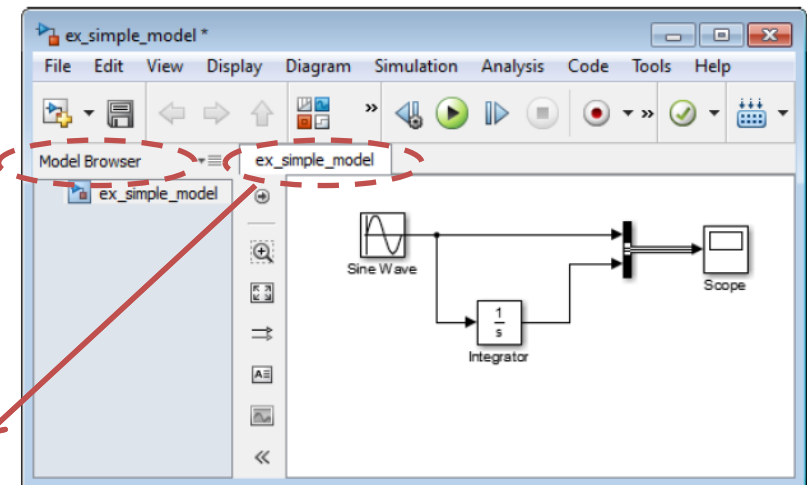


INTERFACCIA UTENTE: SIMULINK EDITOR

- ❑ Il Simulink Editor contiene il diagramma a blocchi del modello, costruito trascinando i blocchi dal Simulink Library Browser.
- ❑ Il modello si costruisce quindi posizionando i blocchi all'interno del Simulink Editor, connettendoli logicamente tra loro con le linee di segnale e impostando i parametri di simulazione per ciascuno dei blocchi.
- ❑ Il Simulink Editor può essere impiegato anche per:
 - ❑ impostare i parametri di configurazione del modello, incluso l'istante di inizio e fine simulazione, tipo di solver da impiegare, importazione/esportazione;
 - ❑ avviare e interrompere la simulazione;
 - ❑ salvare il modello;
 - ❑ stampare il diagramma a blocchi.

Model Browser
(esploratore di modello)

Model Window
(finestra di modello)





DOCUMENTAZIONE ED ESEMPI

- ❑ Simulink fornisce un'ampia **documentazione** che descrive in modo dettagliato le caratteristiche, i blocchi e le funzioni proprie del programma.
- ❑ L'accesso a tale documentazione può avvenire:
 - ❑ dal Simulink Library Browser, selezionando **Help > Simulink Help**;
 - ❑ dal Simulink Editor, selezionando **Help > Simulink > Simulink Help**;
 - ❑ cliccando con il tasto destro su un blocco Simulink e selezionando **Help**;
 - ❑ dal *Model Configuration Parameters* o dal *Block Parameters Dialog*, cliccando sull'etichetta relativa ad un parametro e selezionando **What's This?**
- ❑ Simulink fornisce una **varietà di esempi** che illustrano i concetti chiave della modellazione e delle funzionalità del programma.
- ❑ E' possibile accedere a tali esempi:
 - ❑ dal Simulink Editor, selezionando **Help > Simulink > Examples**;
 - ❑ dalla documentazione, cliccando **Examples** nella parte superiore della pagina.





ESERCIZI MATLAB

- ☐ Disegnare il grafico di $\tan(x)$ nell'intervallo $-\pi/2 \div \pi/2$.
 - ☐ Calcolare il massimo della funzione $y = \tan(x)$.
 - ☐ Assegnare l'etichetta ' x [rad]' all'asse delle x .
 - ☐ Disegnare la parabola $y = x^2$.
 - ☐ Creare la legenda.
 - ☐ Calcolare il minimo della funzione $y = x^2$.
 - ☐ Includere tutto il codice all'interno di uno script eseguibile e commentato.
-



ESERCIZIO SIMULINK

- ☐ Creare un modello capace di calcolare l'integrale e la derivata di un segnale sinusoidale e visualizzarne l'andamento insieme al segnale stesso.
 - ☐ Aggiungere rumore e vedere come cambia il comportamento.
-