

MATLAB-SIMULINK

Creazione di grafici 2D e 3D

Ing. Alessandro Pisano

apisano@unica.it

```
clear all
s='set(gcf, 'position', [0,0,400,350]);' %questa definizione viene impiegata in seguito
% per ovviare al fatto che di default il file pdf
% generato esportando un Live Script contiene le immagini di dimensione eccessiva
```

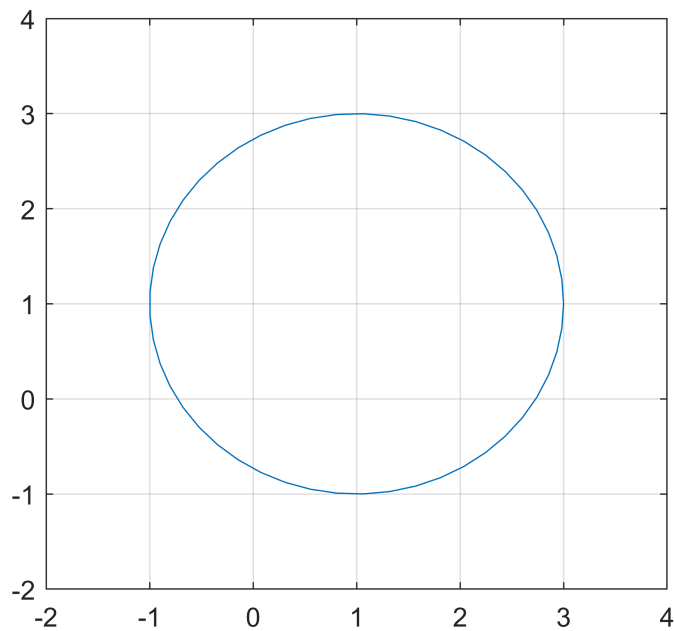
Parte seconda

Grafici di curve parametriche

La procedura di tracciamento di curve parametriche è analoga a quanto visto per i grafici di funzioni scalari di una variabile. Si devono generare due vettori **x** ed **y** contenenti le coordinate (x_i, y_i) di un numero sufficientemente elevato di punti della curva ed una volta fatto ciò è possibile impiegare il comando standard `plot`, che interpolerà tali punti mediante una spezzata.

Il seguente codice di esempio grafica nel piano la circonferenza centrata nel punto (1,1) e avente raggio pari a 2

```
close all % chiude tutte le finestre grafiche aperte
N=50; %numero di punti
t=linspace(0,2*pi,N);
x_c=1; %ascissa del centro
y_c=1; %ordinata del centro
R=2; %raggio
x=x_c+R*cos(t);
y=y_c+R*sin(t);
plot(x,y),grid
eval(s)
axis([-2 4 -2 4])
```

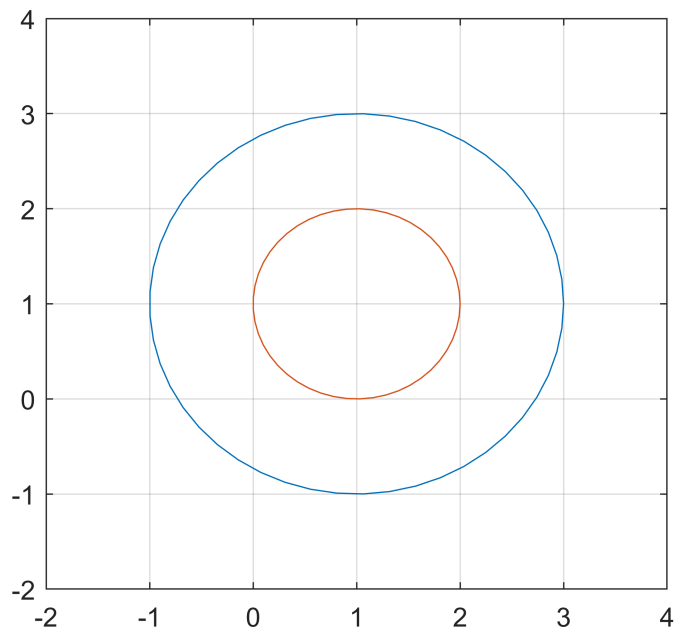


Si verifichi come inserendo un numero di punti inferiore (ad es. $N=10$) la circonferenza risulta approssimata da un poligono.

E' possibile creare grafici sovrapposti anche mediante il comando `hold on`. Successivamente alla creazione di un grafico, il comando `hold on` fa sì che tutti i grafici creati successivamente vadano ad essere inseriti all'interno della medesima finestra grafica. Il comando `hold off` disabilita tale funzione.

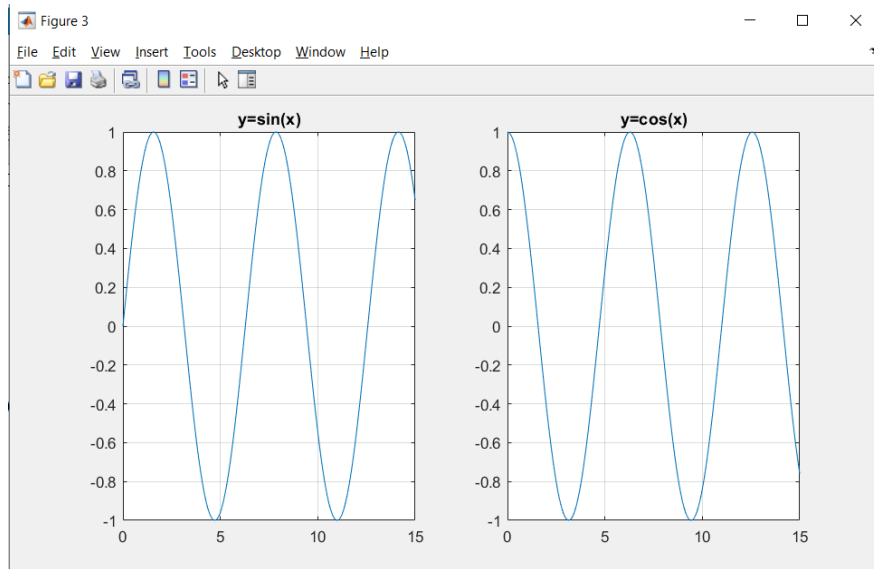
Testiamo tale funzionalità con un codice che vada a graficare due circonferenze di raggio differente.

```
t=linspace(0,2*pi,50);
x_c=1; %ascissa del centro
y_c=1; %ordinata del centro
R1=2; %raggio prima circonferenza
x1=x_c+R1*cos(t);
y1=y_c+R1*sin(t);
plot(x1,y1)
hold on
R2=1; %raggio seconda circonferenza
x2=x_c+R2*cos(t);
y2=y_c+R2*sin(t);
plot(x2,y2)
hold off
axis([-2 4 -2 4])
grid
```



Creazione di grafici multipli

Intendiamo per "grafici multipli" più grafici allocati all'interno di una unica finestra grafica che viene suddivisa in un certo numero di "sottofinestre". Un esempio di grafico multiplo è il seguente



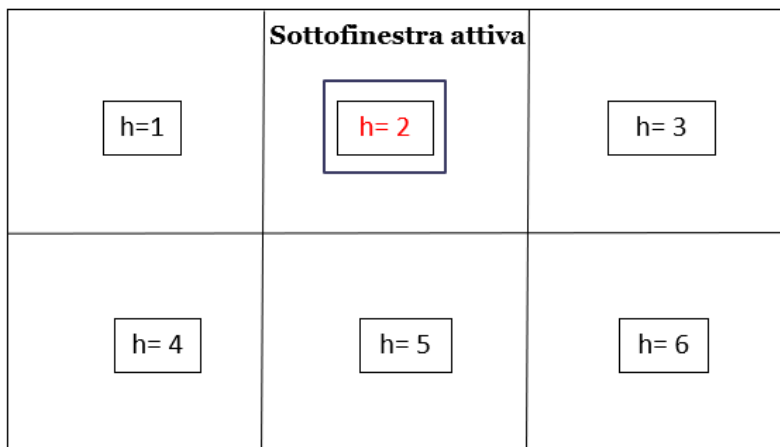
Per creare grafici multipli si utilizza, congiuntamente alla funzione `plot`, anche la funzione `subplot`.

La funzione `subplot` riceve in ingresso 3 argomenti: `subplot(n,m,h)`, l'ultimo dei quali è compreso fra 1 ed $n \cdot m$ ($1 \leq h \leq n \cdot m$).

L'istruzione `subplot` suddivide una finestra grafica rettangolare in $n \cdot m$ sottofinestre disposte su n righe ed m colonne, e "rende attiva" la sottofinestra individuata dal particolare valore dell'indice h . Alle sottofinestre della

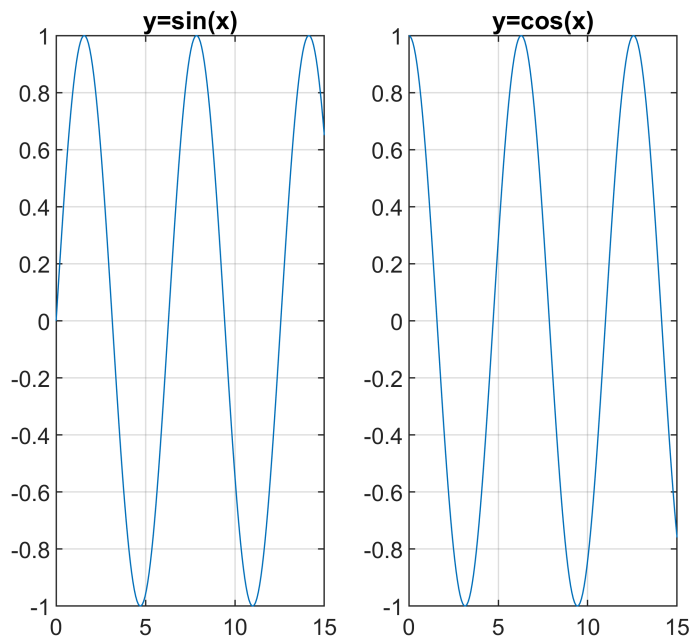
prima riga corrispondono i valori di h da 1 ad m , alle sottofinestre della seconda riga corrispondono i valori di h da $m+1$ a $2m$, e così via per le righe successive.

Ad esempio, l'istruzione `subplot(2,3,2)` suddivide la finestra grafica in 6 sottofinestre, disposte su 2 righe e 3 colonne, e rende attiva la seconda sottofinestra della prima riga (si veda la figura seguente)



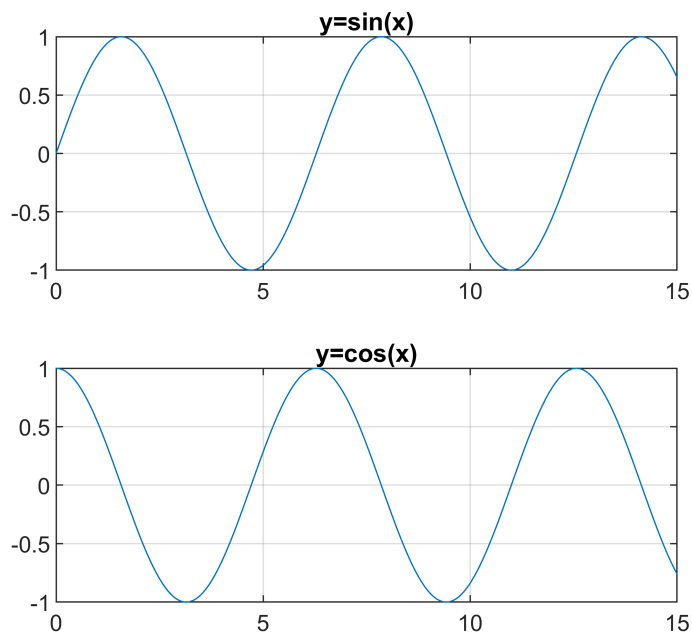
Vediamo alcuni esempi. Creiamo preliminarmente il grafico multiplo delle due funzioni oscillatorie riportato poco sopra.

```
x=linspace(0,15,200);  
y1=sin(x);  
y2=cos(x);  
subplot(1,2,1)  
plot(x,y1),grid  
title('y=sin(x)')  
subplot(1,2,2)  
plot(x,y2),grid  
title('y=cos(x)')
```



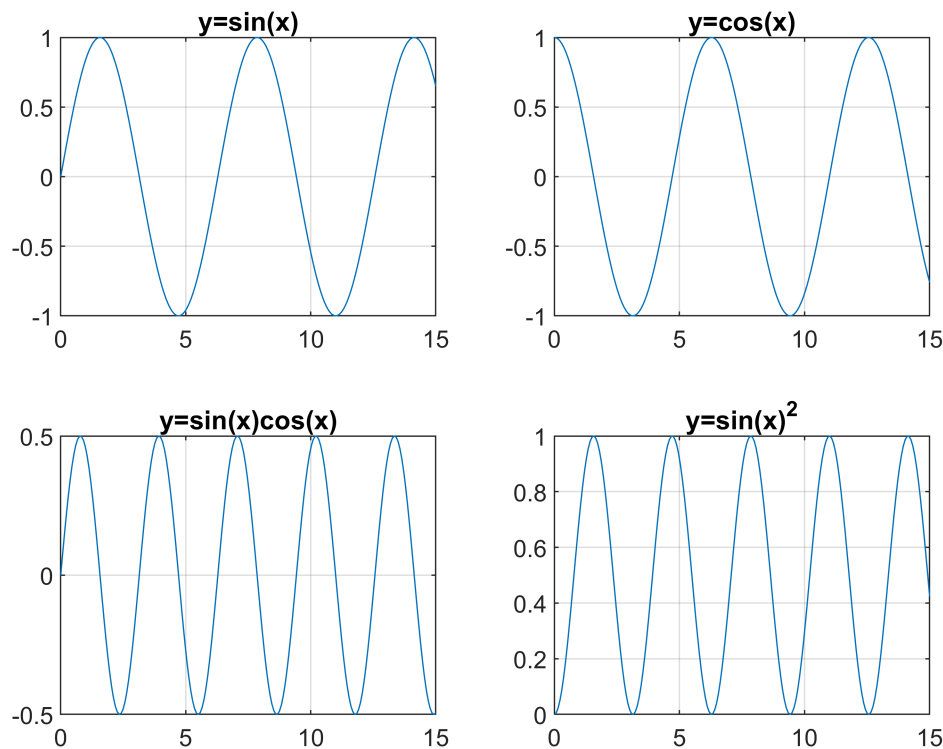
Ora creiamo un grafico multiplo in cui le medesime curve siano disposte l'una sopra l'altra (2 righe e 1 colonna) anziche, come nel caso precedente, l'una affianco all'altra. E' sufficiente modificare il precedente codice invertendo i primi due argomenti di ingresso della funzione subplot.

```
x=linspace(0,15,200);  
y1=sin(x);  
y2=cos(x);  
subplot(2,1,1)  
plot(x,y1),grid  
title('y=sin(x)')  
subplot(2,1,2)  
plot(x,y2),grid  
title('y=cos(x)')
```



Come ultimo esempio, allochiamo 4 grafici in una finestra suddivisa in due righe e due colonne

```
figure
linspace(0,15,200);
y1=sin(x);
y2=cos(x);
y3=sin(x).*cos(x);
y4=sin(x).^2;
subplot(2,2,1)
plot(x,y1),grid
title('y=sin(x)')
subplot(2,2,2)
plot(x,y2),grid
title('y=cos(x)')
subplot(2,2,3)
plot(x,y3),grid
title('y=sin(x)cos(x)')
subplot(2,2,4)
plot(x,y4),grid
title('y=sin(x)^2')
```



Impostazione del colore della linea

Per impostare arbitrariamente il colore delle varie curve (o anche di una sola curva) si utilizza una sintassi più complicata. In cui oltre ai due "soliti" vettori si passa come ulteriore argomento di ingresso alla funzione `plot` una stringa di testo che contiene una lettera rappresentativa del colore desiderato, secondo la seguente tabella.

Color	Description
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

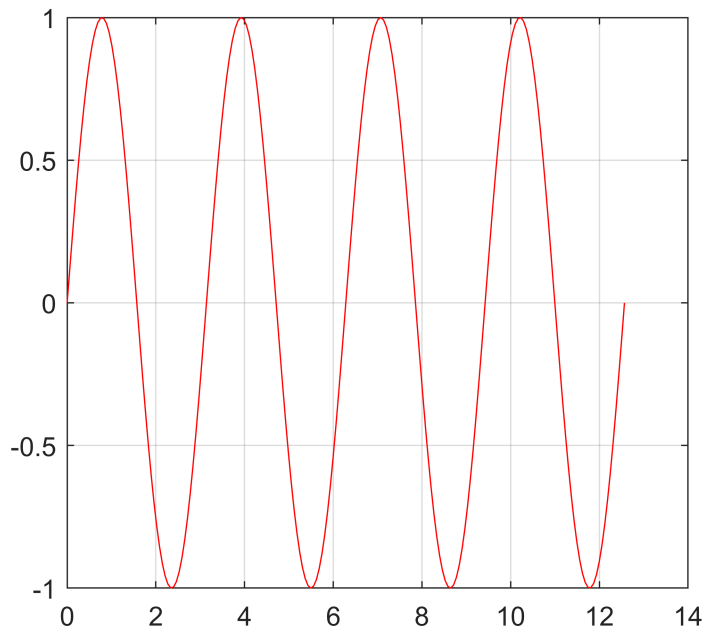
Vediamo un semplice esempio inerente il grafico della funzione $y = \sin(2x)$ nell'intervallo $I = [0, 4\pi]$, con la curva da tracciare in rosso

```
x=linspace(0,4*pi,200);
```

```

y=sin(2*x);
figure
plot(x,y,'r'),grid
eval(s)

```



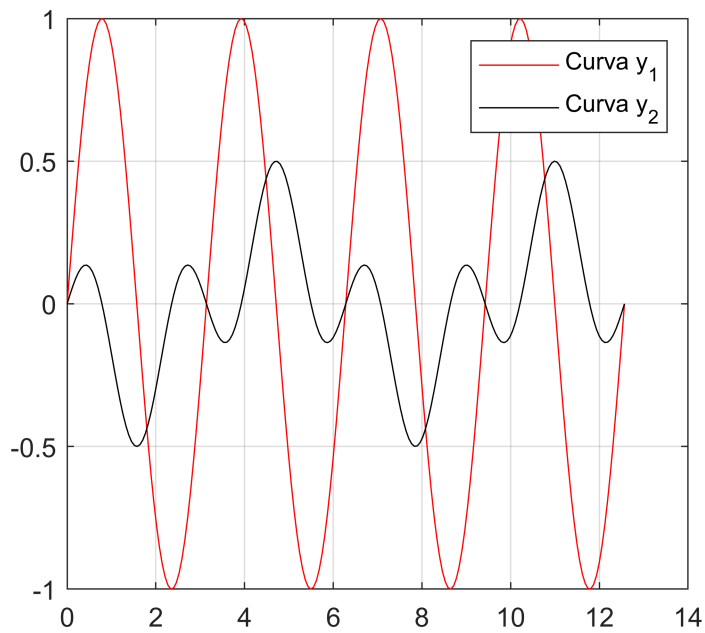
Ora vediamo come procedere per impostare il colore delle curve in grafici con curve sovrapposte (funzioni

$$y_1 = \sin(2x) \text{ ed } y_2 = \frac{1}{2} \cos(2x) \sin(x)$$

```

x=linspace(0,4*pi,200);
y1=sin(2*x);
y2=0.5*cos(2*x).*sin(x);
plot(x,y1,'r',x,y2,'k'),grid
legend('Curva y_1','Curva y_2')

```

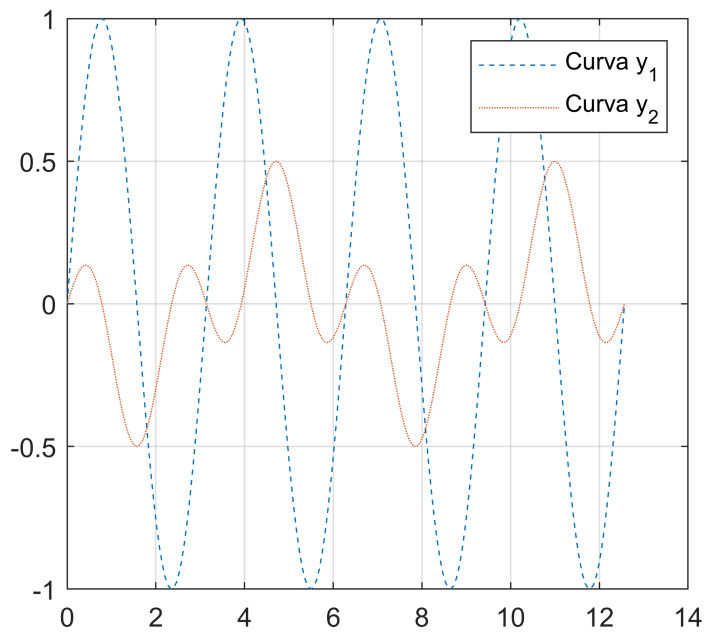



Impostazione del tipo di linea

Il tipo di linea può essere modificato con modalità analoga a quanto fatto per il colore, e cioè mediante una stringa aggiuntiva che contiene uno fra i caratteri inseriti nella seguente Tabella.

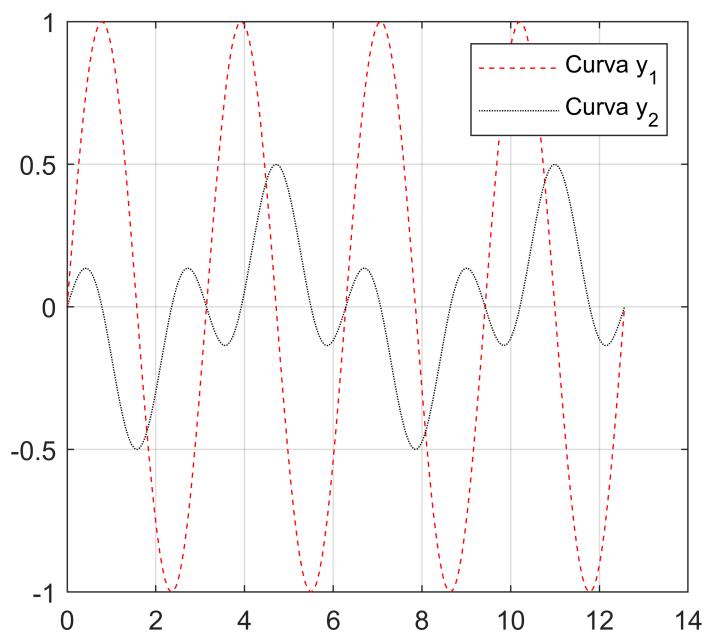
Line Style	Description	Resulting Line
'_'	Solid line	_____
'--'	Dashed line	- - - - -
':'	Dotted line
'-.'	Dash-dotted line	- . - . - .
'none'	No line	No line

```
x=linspace(0,4*pi,200);
y1=sin(2*x);
y2=0.5*cos(2*x).*sin(x);
plot(x,y1,'--',x,y2,':'),grid
legend('Curva y_1','Curva y_2')
```



Per impostare simultaneamente il colore ed il tipo di linea si impiega una stringa che contiene sia la lettera associata al colore che il carattere associato al tipo di linea (l'ordine è interscambiabile)

```
x=linspace(0,4*pi,200);
y1=sin(2*x);
y2=0.5*cos(2*x).*sin(x);
plot(x,y1, 'r--',x,y2, 'k:'),grid
legend('Curva y_1', 'Curva y_2')
```



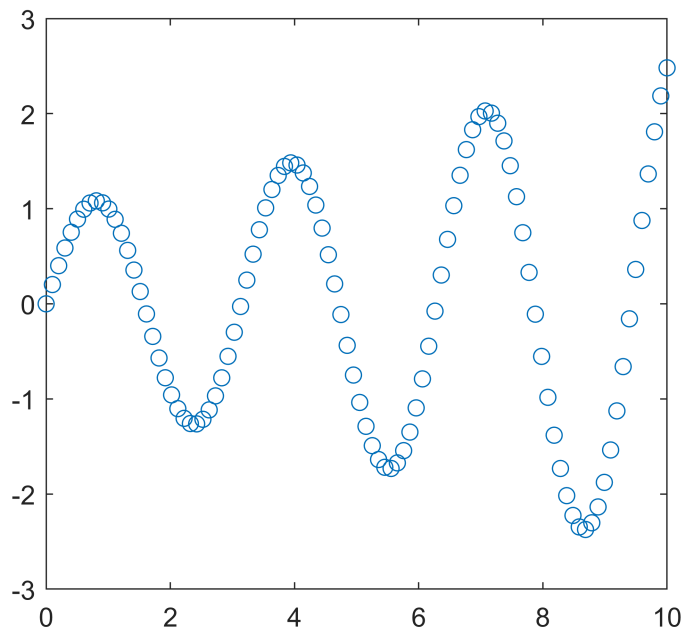
Impostazione di un marker

Per particolari esigenze di visualizzazione, si può fare in modo che ai punti della curva sia associato un marker. La sintassi è analoga: si passa alla funzione plot come argomento aggiuntivo una stringa che contiene uno dei simboli della seguente tabella:

Value	Description
'o'	Circle
'+'	Plus sign
'*'	Asterisk
'.'	Point
'x'	Cross
'_'	Horizontal line
' '	Vertical line
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h'	Six-pointed star (hexagram)
'none'	No markers

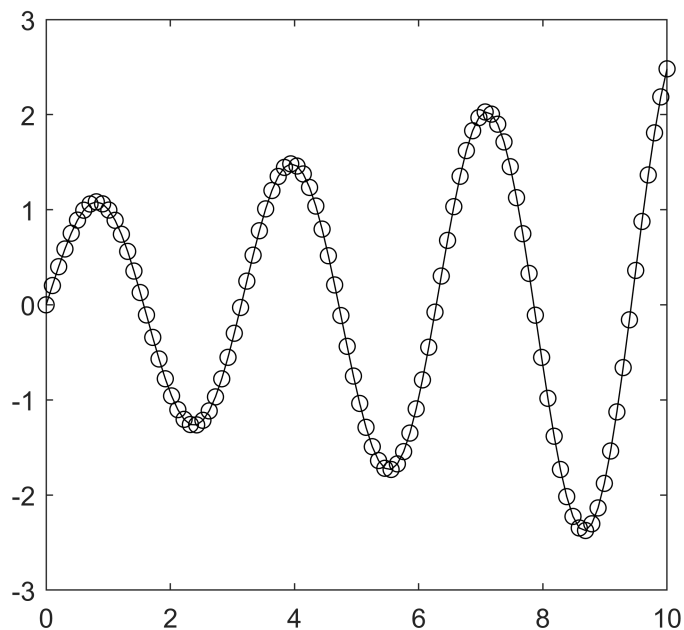
Come esempio, consideriamo la funzione $y(x) = e^{\frac{x}{10}} \sin(2x)$, plottata nell'intervallo $[0, 10]$ con $N=100$ punti. Si noti come la curva interpolante i punti non viene più tracciata.

```
x = linspace(0,10,100);
y = exp(x/10).*sin(2*x);
plot(x,y, 'o')
```



Per fare in modo che sia tracciata anche una curva interpolante nera a tratto continuo (Solid Line) utilizziamo la seguente sintassi

```
x = linspace(0,10,100);  
y = exp(x/10).*sin(2*x);  
plot(x,y,'-ko')
```

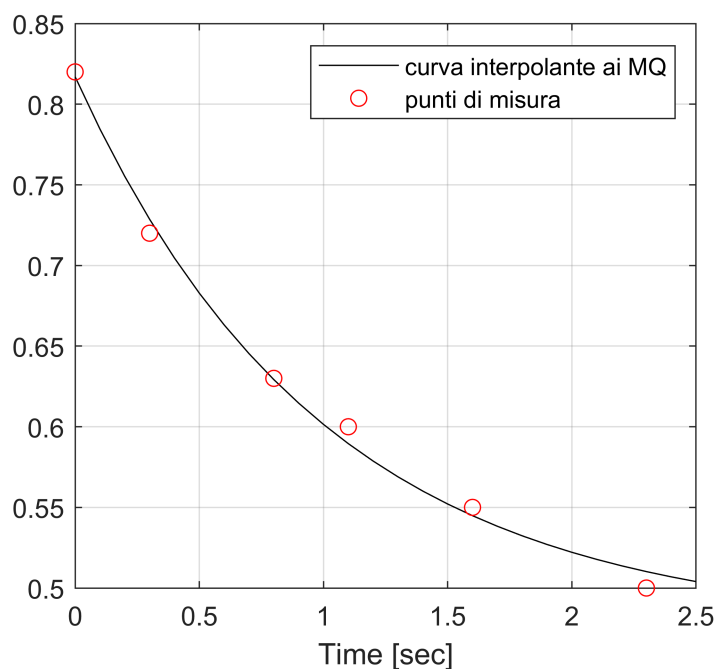


Nella stringa '-ko' utilizzata nel precedente esempio di codice il primo carattere definisce il tipo di linea (Solid, tratto continuo), il secondo definisce il colore (k=nero), ed il terzo il tipo di marker (un cerchietto).

Vediamo un ultimo esempio in cui sovrapponiamo fra loro due curve con caratteristiche di visualizzazione differenti.

Il grafico successivo intende sovrapporre ad un certo numero di punti del piano, le cui coordinate sono contenute nei vettori t (ascisse) ed y (ordinate), la curva $y_{\text{int}}(t) = 0.476 + 0.341e^{-t}$, che come mostrato in uno degli esempi della dispensa "Sistemi di equazioni lineari" è una curva interpolante ai minimi quadrati associata a tale insieme di punti.

```
t = [0 .3 .8 1.1 1.6 2.3]';  
y = [.82 .72 .63 .60 .55 .50]';  
int_funct=@(t)0.476+0.341*exp(-t);  
T = (0:0.1:2.5)';  
y_int=int_funct(T);  
figure  
plot(T,y_int,'-k',t,y,'ro'),grid  
eval(s)  
xlabel('Time [sec]')  
legend('curva interpolante ai MQ','punti di misura')
```



La stringa `'-k'` attribuisce alla curva interpolante ai MQ il tratto continuo ed il colore nero, mentre la stringa `'ro'` attribuisce ai punti di misura il marker circolare tracciato in rosso.

Grafici con assi delle ordinate multipli

Capita di dover confrontare in un medesimo grafico due curve che abbiano degli ordini di grandezza differenti, e che in un grafico tradizionale non potrebbero essere visualizzate in simultanea. E' possibile creare un grafico con due curve sovrapposte a ciascuna delle quali corrisponde un diverso asse delle ordinate. Si utilizzano a tal fine i comandi `yyaxis left` ed `yyaxis right` come mostrato nel seguente codice di esempio.

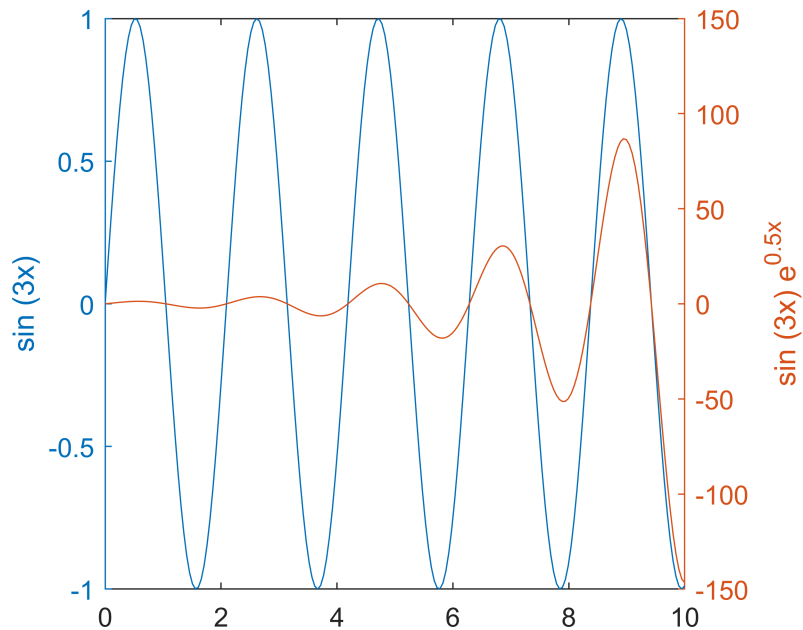
```
x = linspace(0,10,200);
```

```

y = sin(3*x);

figure
yyaxis left
plot(x,y)
eval(s)
ylabel('sin (3x)')
z = sin(3*x).*exp(0.5*x);
yyaxis right
plot(x,z)
ylabel('sin (3x) e^{0.5x}')
ylim([-150 150])

```



Formattazione avanzata delle Label

Per l'inserimento di **lettere greche** in una Label si possono utilizzare direttamente i seguenti identificatori

Lettere minuscole:

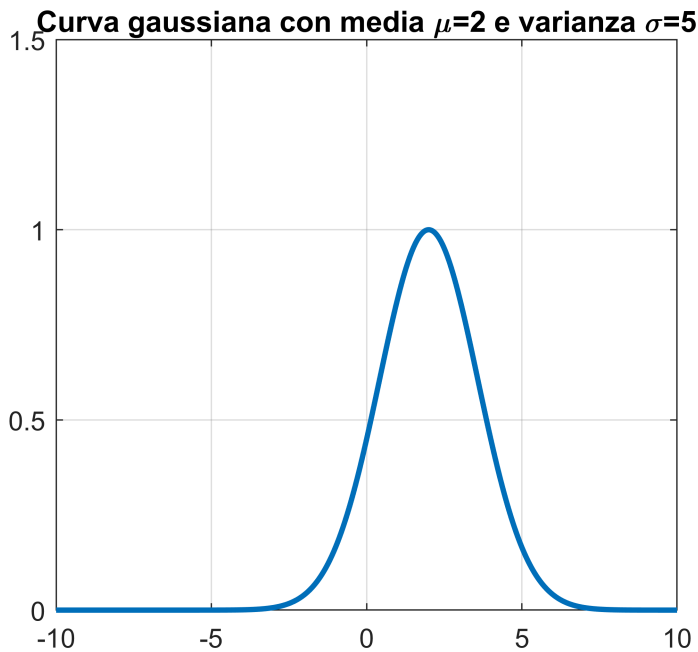
α	<code>\alpha</code>	θ	<code>\theta</code>	o	<code>o</code>	v	<code>\upsilon</code>
β	<code>\beta</code>	ϑ	<code>\vartheta</code>	π	<code>\pi</code>	ϕ	<code>\phi</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	φ	<code>\varphi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	χ	<code>\chi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	ψ	<code>\psi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ω	<code>\omega</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>		
η	<code>\eta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>		

Lettere maiuscole:

Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		

Vediamo un esempio associato alla creazione di un grafico della curva gaussiana $e^{-\frac{(x-\mu)^2}{\sigma}}$

```
mu=2;
sigma=5;
x=linspace(-10,10,1000);
y=exp(-(x-mu).^2 ./ sigma);
figure
plot(x,y,'LineWidth',2);
eval(s)
title('Curva gaussiana con media \mu=2 e varianza \sigma=5');
grid
axis([-10 10 0 1.5])
```



Per inserire in una label delle formule matematiche più complesse, ad esempio l'espressione analitica della funzione Gaussiana, si sfrutta la compatibilità di Matlab con le istruzioni del programma open source **LaTeX 2e** orientato alla redazione di documenti scientifici.

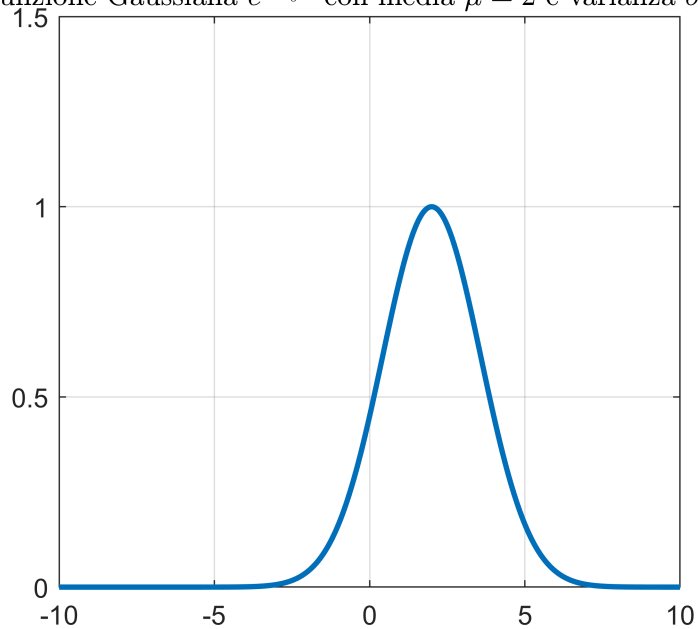
Una ottima guida in italiano può essere reperita sul web all'indirizzo:

<http://www.ctan.org/tex-archive/info/lshort/italian/itlshort.pdf>

Si veda in particolare il Capitolo 3 “*Scrivere formule matematiche*”. Modifichiamo il precedente codice inserendo un titolo che contiene l'espressione analitica della curva gaussiana.

```
clc
mu=2;
sigma=5;
x=linspace(-10,10,1000);
y=exp(-(x-mu).^2 ./ sigma);
figure
plot(x,y,'LineWidth',2);
eval(s)
title('Funzione Gaussiana  $e^{-\frac{x-\mu}{\sigma}}$  con media  $\mu=2$  e varianza  $\sigma=5$ ','
grid
axis([-10 10 0 1.5])
```


Funzione Gaussiana $e^{-\frac{x-\mu}{\sigma}}$ con media $\mu = 2$ e varianza $\sigma = 5$

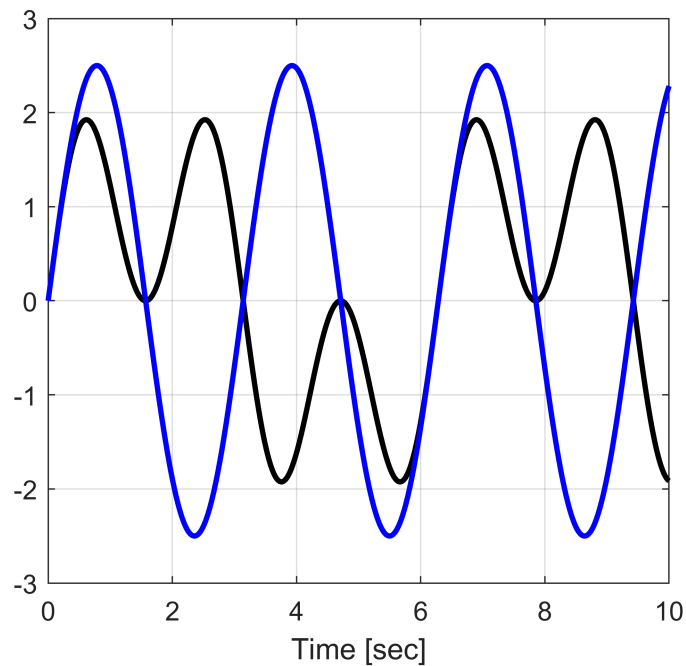


Come accedere ai dati da un file immagine con estensione .fig

Presentiamo ora un codice che consente di "estrarre" da un file immagine con estensione .fig i dati associati alle curve del grafico.

Si apra il file "Fig01.fig" con la seguente istruzione

```
h=hgload('fig01.fig');  
eval(s)
```



Le seguenti istruzioni salvano nei vettori `time1` e `time2` gli istanti di campionamento delle due curve, e nei vettori `data1` e `data2` i relativi campioni.

```
ch=get(h,'Children');
l=get(ch,'Children');
x=get(l,'Xdata');
y=get(l,'Ydata');
time1=x{1,1};
time2=x{2,1};
data1=y{1,1};
data2=y{2,1};
```

I dati contenuti nei vettori possono ora essere elaborati. Grafichiamoli per verificare la corrispondenza con le curve della figura originale

```
plot(time1,data1,time2,data2), grid
xlabel('Time [sec]')
```

