Silvio Navaretti (Fedhman)

MATLAB: GUIDA PRATICA PER DISEGNARE FUNZIONI AD UNA VARIABILE

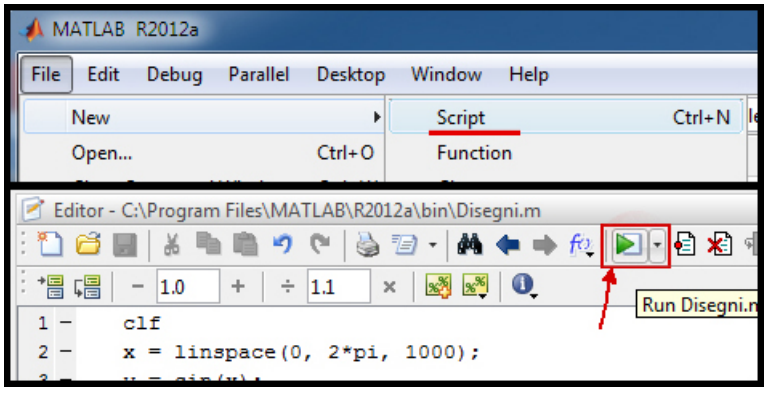
1 November 2013

Premesse

Questa breve guida si propone di spiegare come disegnare (*plottare*) funzioni ad una variabile con MATLAB. Non ha pretese di esaustività o di completezza.

È possibile che vi siano modi più rapidi per fare ciò che spiego nella guida; se li conoscete e ne avete voglia condivideteli col sottoscritto, sarò più che felice di apprenderli.

Potete chiamare le variabili come vi pare, x potete chiamarla pippo, y pluto, non cambia alcunchè. Sperimentate con uno script, è più comodo. Per farlo create uno script con File/New/Script e salvatelo da qualche parte (se vi stressa gli zebedei col "path" accettate quel che vi chiede). Potete modificare quando vi pare lo script ed eseguirlo subito con F5 o col pulsante "play" verde.



Creare ed eseguire uno script (MATLAB vi chiederà di salvarlo)

pi = π = 3.14159....

exp(x) = e^x

Dai vettori al grafico grezzo

Per disegnare un funzione servono:

- Vettore delle ascisse: chiamiamolo **x**. Lo possiamo usare per più grafici, anche contemporaneamente. Funge da ossatura su cui andremo a "modellare" i vettori delle ordinate.
- Vettore delle ordinate, ovvero la nostra funzione: lo chiamo **y**. Esso sarà una funzione di x, utilizzandola come variabile.

Li useremo in coppia col comando plot.

Creiamo x con il comando linspace:
Come funziona: vettore = linspace(InizioVettore, FineVettore, QuantiPuntiLoCompongono);

- *Esempio*: vettore x che inizia in 0, finisce in 8, è composto da 100 punti
  x = linspace(0, 8, 100);
- *Esempio*: vettore x che inizia in -12, finisce in 144, è composto da 39345 punti
  x = linspace(-12, 144, 39345);
- *Esempio*: vettore gatto che inizia in -56*pi, finisce in -18*pi, è composto da 271 punti
  gatto = linspace(-56*pi, -18*pi, 271);

Modelliamo y come funzione di x:
Vogliamo rappresentare una funzione del tipo y(x) = numero*f(x).
Il trucco è dire a MATLAB "prendi ogni punto del vettore x e facci qualcosa, buttando i risultati in y".

- Con il comando **y = qualcosa;** diciamo a MATLAB di creare un numero (ovvero un vettore con un singolo elemento).
- Evolviamolo in **y = x;** ovvero "crea un vettore y identico al vettore x che ho formato prima" (magari x l'ho formato con linspace, come negli esempi precedenti).
- E vualà, con **y = sin(x);** gli diciamo qualcosa del tipo "fammi un vettore y in cui *ogni punto* corrisponde *al valore del seno del corrispondente elemento* del vettore x". Da spiegare è un po' arzigogolato. La funzione sin(x) è solo un esempio, possiamo sbizzarrirci come vogliamo, **y = 3*x;** o **y = exp(x);** o qualsiasi altro tipo di funzione.

Abbiamo quindi ottenuto una sorta di funzione **y** che andremo a disegnare.

ATTENZIONE alle operazioni * e /, ci va un punto davanti se coinvolgono vettori. Se operate con qualcosa del tipo y(x) = x*sin(x) andate a scrivere in MATLAB **y = x.*sin(x)**

- *Esempio:* $y(x) = cos(x)$
  y = cos(x);
- *Esempio:* $y(x) = 3x$
  y = 3*x;
- *Esempio:* $y(x) = \frac{n}{e^{-nx}}$
  ATTENZIONE va definito **n** prima di calcolare y. Se **n** è un punto, ovvero un vettore composto da un singolo elemento, non v'è bisogno di inserire . nell'operazione in

MATLAB.

```
n = 14;
y = n/(exp(-n*x))
```

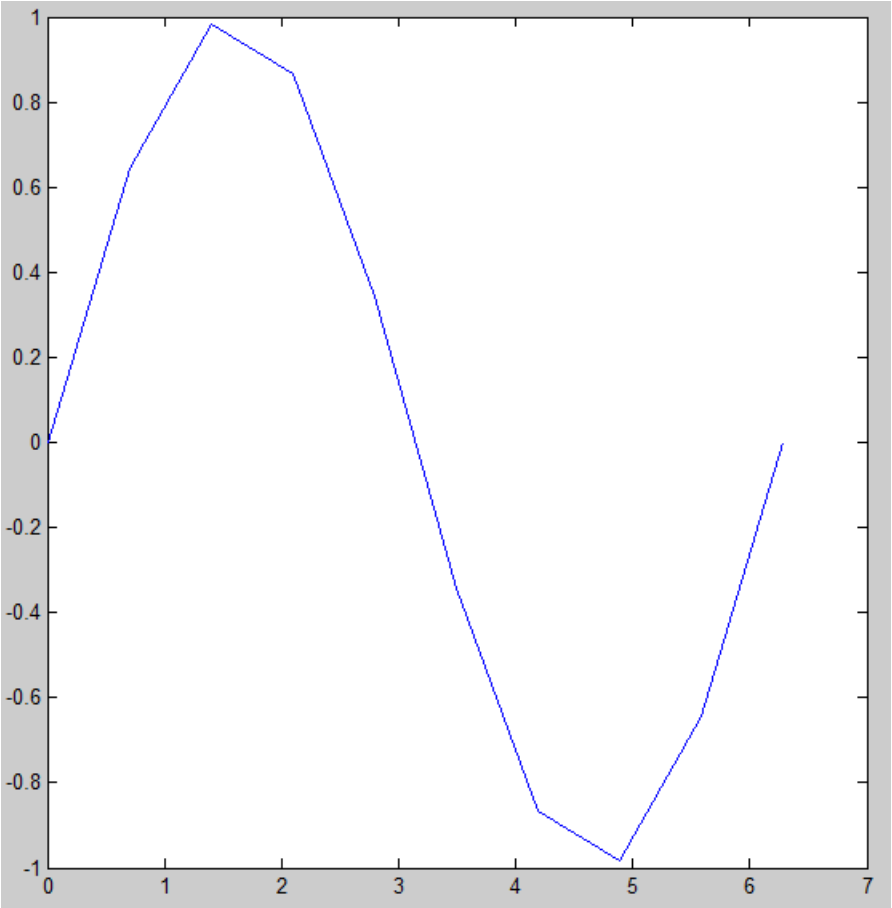
- *Esempio:* $y(x) = xtg(x)$
  Siccome moltiplichiamo un vettore per un altro vettore (nonostante siano entrambi x) dobbiamo inserire . nell'operazione.

```
y = x.*tan(x);
```

Formiamo il grafico base, senza opzioni:
Come funziona: plot(x, y)
NOTA: non inserite ; al termine della riga di plot, altrimenti non apparirà il grafico.
NOTA2: inserite il comando **clf** all'inizio del vostro script. In tal modo il grafico verrà resettato ogniqualvolta avvierete lo script, permettendovi di variare rapidamente i parametri di x o y.
NOTA3: a scanso di equivoci, prima del comando plot vanno definiti x e y.

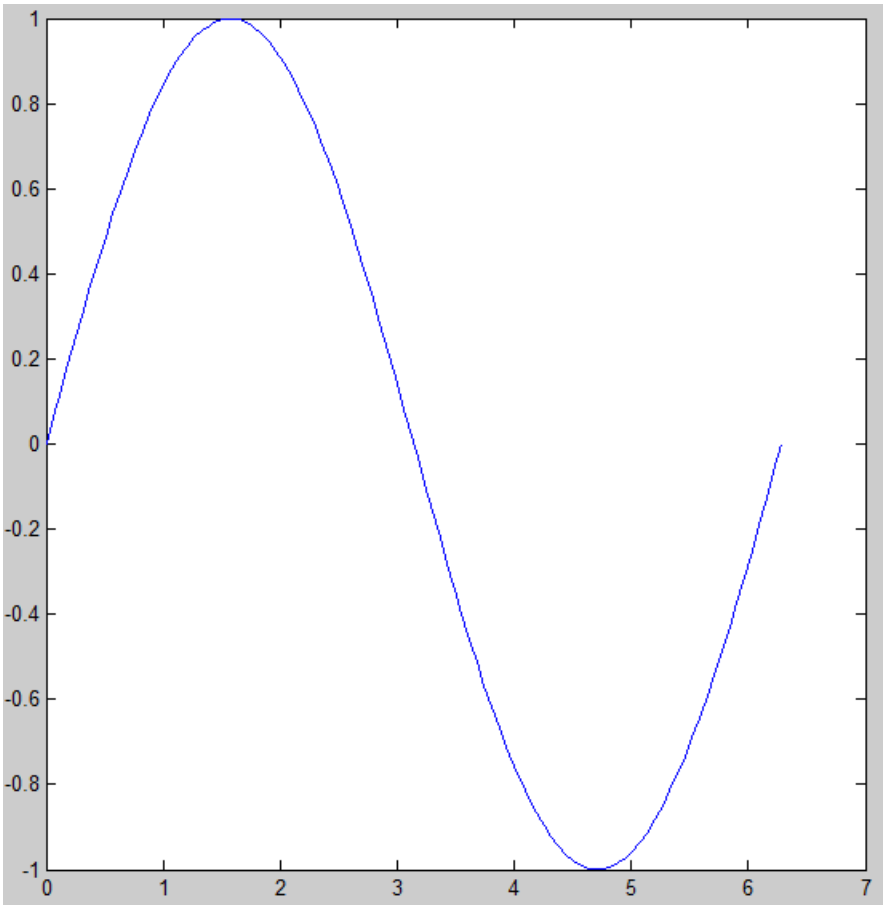
- *Esempio:*

```
clf
x = linspace(0, 2*pi, 10);
y = sin(x);
plot(x, y)
```



*Con x = linspace(0, 2*pi, 10); il grafico è obbrobrioso.*

Una schifezza. I contorni sono spigolosi perchè i punti di x non bastano, sono soltanto 10. Proviamo dando 100 punti a x.



*Settando x = linspace(0, 2*pi, 100); otteniamo un risultato migliore.*

Va molto meglio! C'è da dire che gli assi sono settati in modo strano.

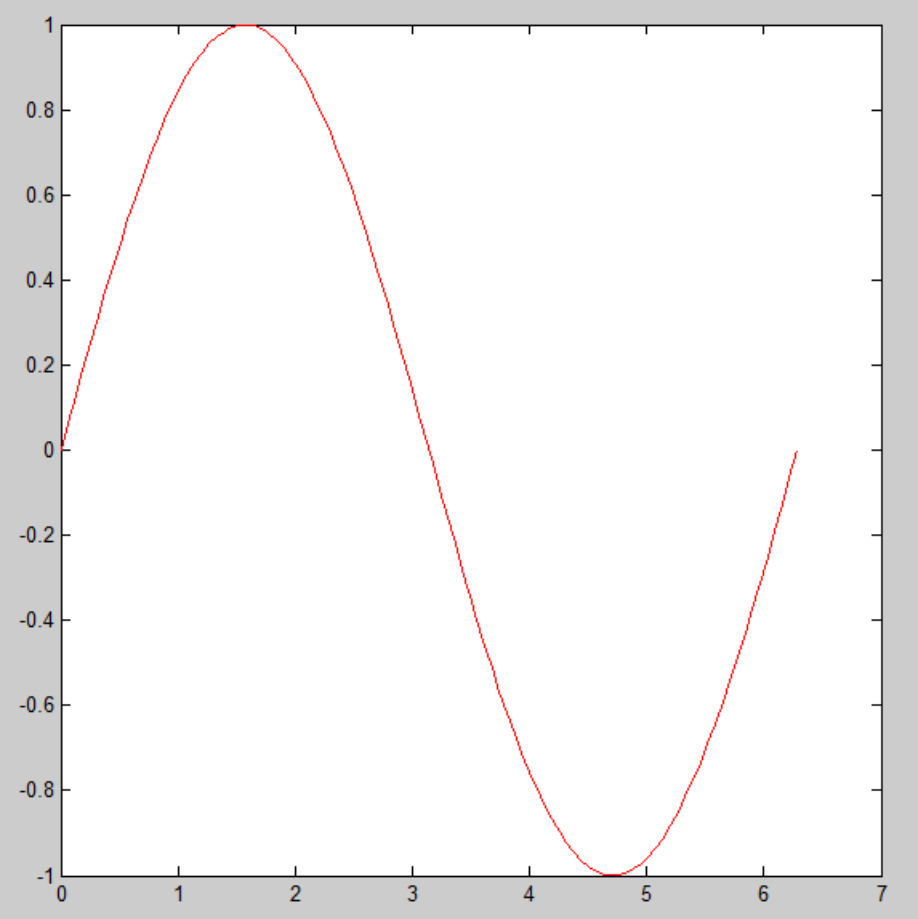
Opzioni di plot: colore, spessore

Come funziona: plot(x, y, NomeOpzione1, ValoreOpzione1, NomeOpzione2, ValoreOpzione2);
NOTA: Potete inserire anche più di 2 opzioni.

***Colore*:** il NomeOpzione è il nome del colore tra apici, il ValoreOpzione non va inserito.
Il colore standard è il blu.

- *Esempio*, colore rosso:

```
clf
x = linspace(0, 2*pi, 100);
y = sin(x);
plot(x, y, 'red')
```

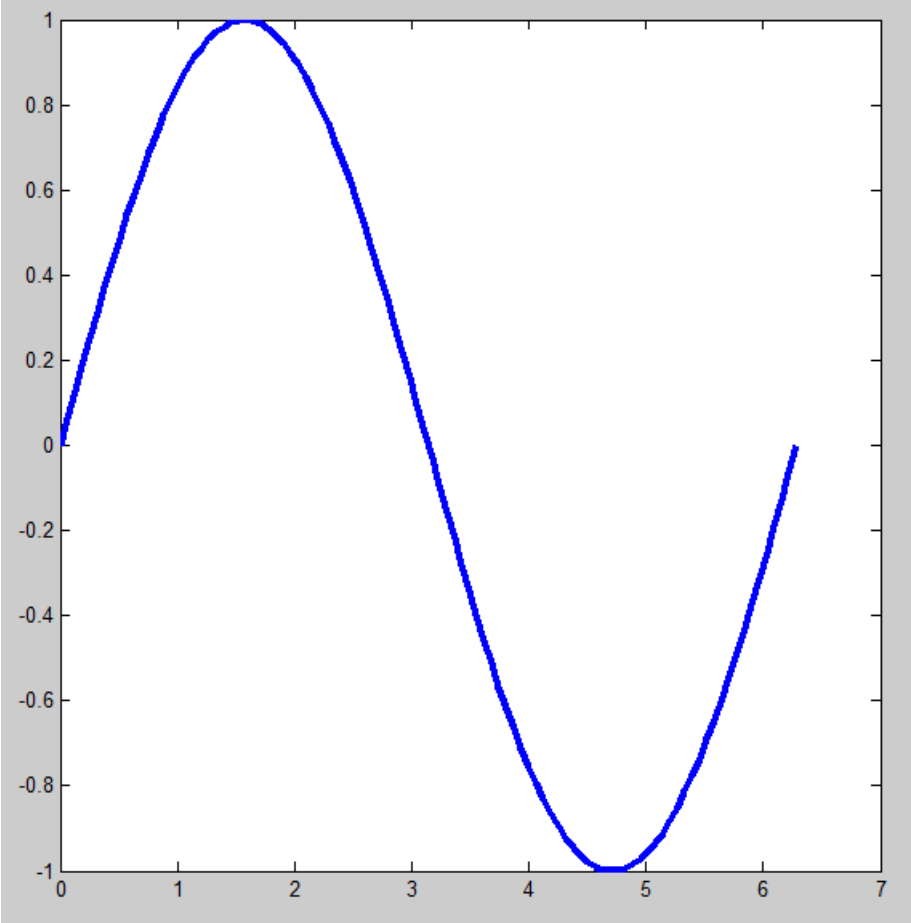


Colore rosso

***Spessore*:** il NomeOpzione è 'LineWidth', il ValoreOpzione è un numero da 1 a...quanto vi serve. Lo spessore standard è 1.

- *Esempio*, spessore 3:

```
clf
x = linspace(0, 2*pi, 100);
y = sin(x);
plot(x, y, 'LineWidth', 3)
```

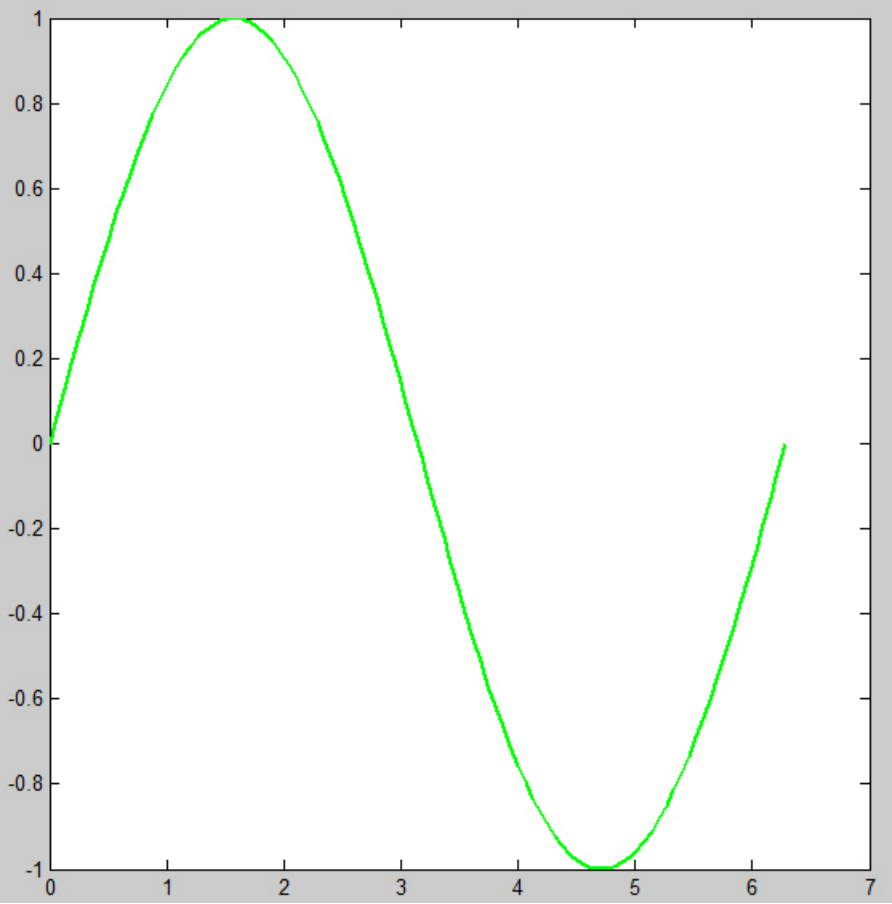


Spessore 3

È possibile rappresentare grafici colorati dello spessore desiderato, mischiando le opzioni.
NOTA: si possono unire anche altre opzioni di cui non parlo nella guida.
ATTENZIONE: le opzioni vanno inserite nell'ordine corretto. Prima il colore, poi lo spessore.

- *Esempio*, grafico verde di spessore 2:

```
clf
x = linspace(0, 2*pi, 100);
y = sin(x);
plot(x, y, 'green', 'LineWidth', 2)
```



Colore verde, spessore 2

Assi

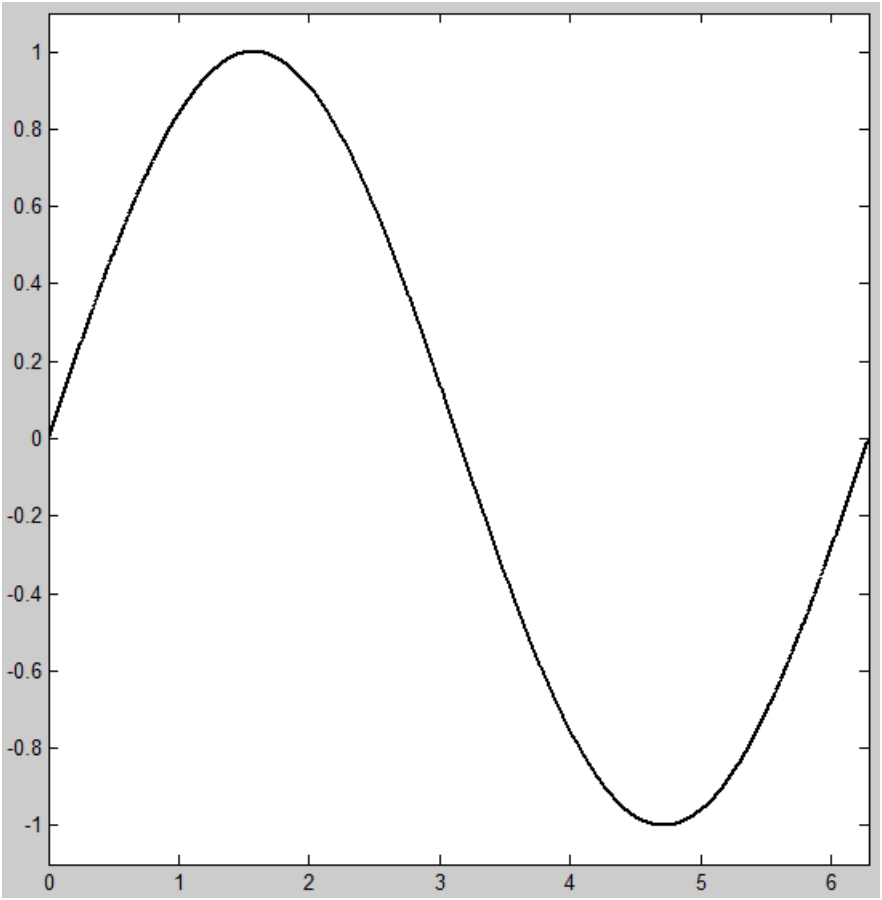
Nei grafici precedenti gli assi sono stati settati da MATLAB stesso. Le ascisse vanno da 0 a 7 mentre le ordinate sono comprese tra -1 e 1. Insomma MATLAB ha dato un'occhiata alla nostra funzione ed ha fatto del suo meglio.
Avanza però quello spazietto fastidioso tra 2*pi e 7; vorremmo anche visualizzare più chiaramente il massimo ed il minimo della funzione allargando un po' le ordinate.

Come funziona: axis([MinimoAscisse MassimoAscisse MinimoOrdinate MassimoOrdinate]);
il comando axis va posto DOPO il comando plot.
ATTENZIONE: i parametri NON vanno separati da una virgola, attenti ad inserire correttamente le parentesi tonde e quadre.

- *Esempio:* colore nero, spessore 2, ascisse da 0 a 2*pi, ordinate da -1.2 a 1.2, mille punti di x.

  clf

```
x = linspace(0, 2*pi, 1000);
y = sin(x);
plot(x, y, 'black', 'LineWidth', 2)
axis([0 2*pi -1.1 1.1]);
```



Assi

Ma dov'è esattamente π? Più o meno sappiamo dov'è, indichiamolo però con precisione.

Cambiare la spaziatura degli assi

Come funziona:

Asse delle ascisse: set(gca, 'XTick', primoPunto : step : ultimoPunto);

Asse delle ordinate: set(gca, 'YTick', primoPunto : step : ultimoPunto);

NOTA: step indica ogni quanto inserire una tacca, andando dal primo punto all'ultimo.

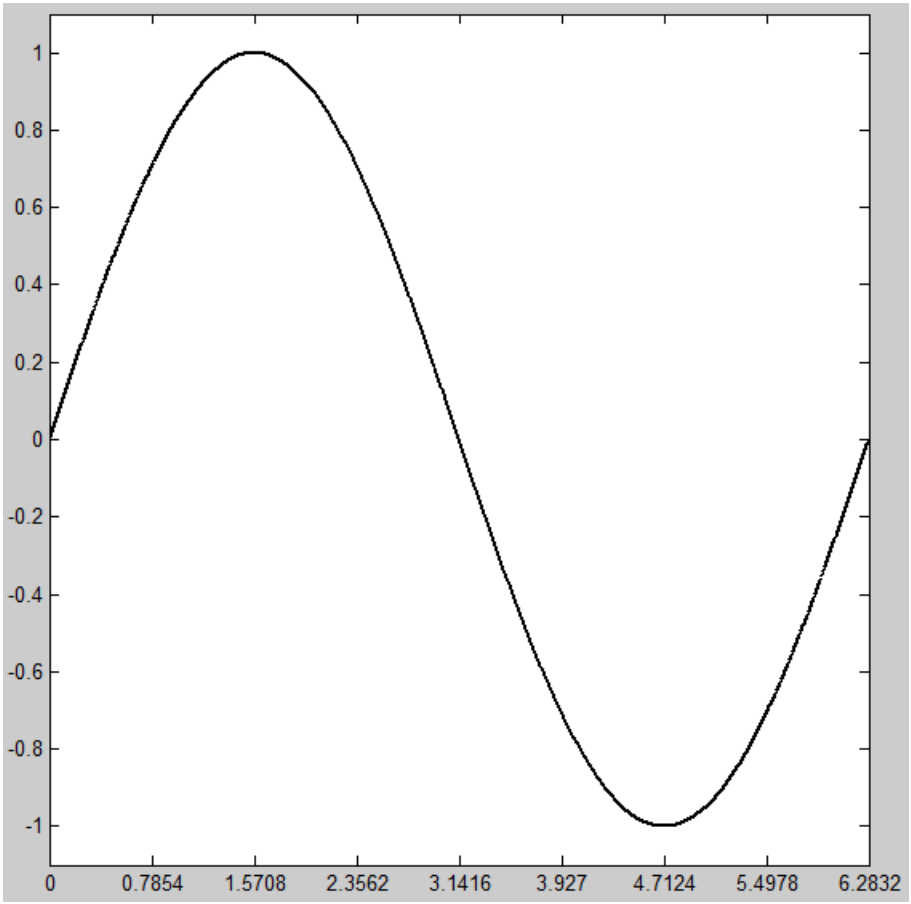
NOTA2: il comando va inserito DOPO plot. Potete inserirlo prima o dopo axis indifferentemente.

NOTA3: prima MATLAB si gestiva gli assi da solo. Ciò ha i suoi vantaggi; noi poveri umani potremmo inserire degli assi sballati, non vedendo alcunchè.

- *Esempio*: mettiamo una tacca ogni quarto di pi (0.25*pi)
  sull'asse delle ascisse dell'esempio precedente; primo punto 0, ultimo punto 2*pi.

L'asse delle ordinate mi sembra comodo, lasciamolo così. Avremmo potuto modificarlo.

```
clf
x = linspace(0, 2*pi, 1000);
y = sin(x);
plot(x, y, 'black', 'LineWidth', 2)
axis([0 2*pi -1.1 1.1]);
set(gca, 'XTick', 0 : 0.25*pi : 2*pi);
```



Una tacca ogni quarto di pi

Nonostante tracciare linee col dito sullo schermo sia affascinante, sarebbe comodo avere una griglia.

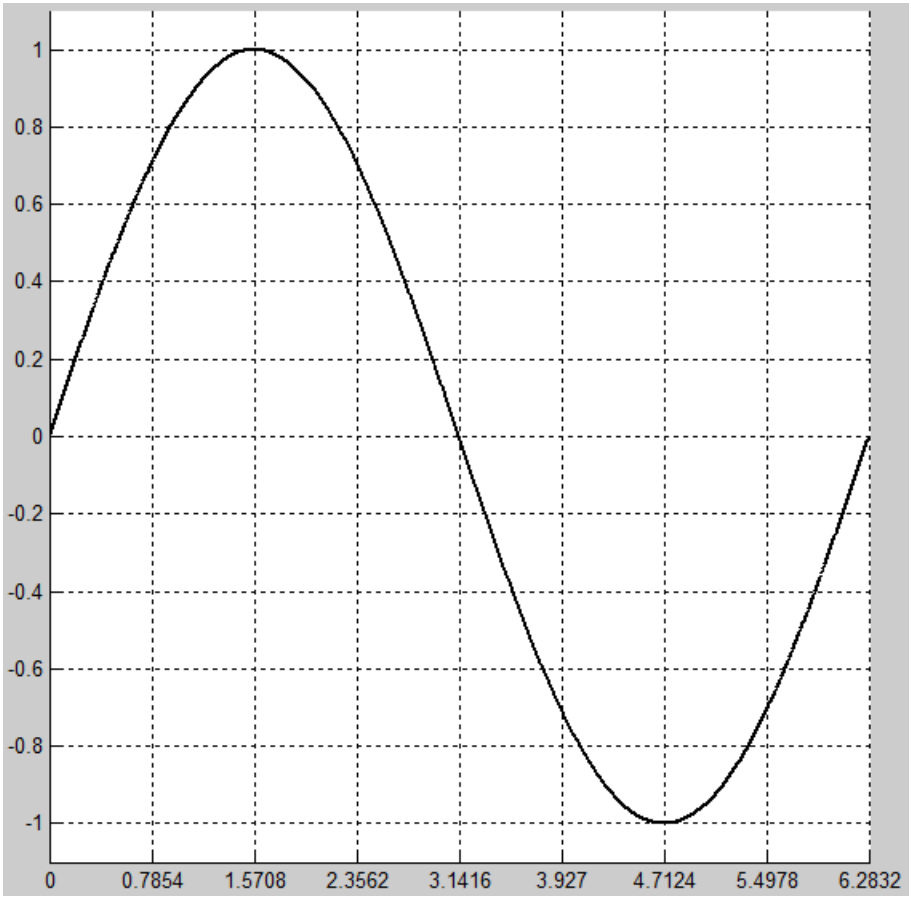
Hold on, Griglia

Per inserire la griglia ci serve il comando **hold on**. Serve a rappresentare sullo stesso grafico più funzioni. Se non lo si inserisce la griglia non viene visualizzata.
Il comando per visualizzare la griglia è **grid on**.

Ripeto, se non si inserisce **hold on** il comando **grid on** non serve ad alcunchè.
Inseriamo **hold on**, **grid on** ed infine plot con le sue menate.
NOTA: l'ordine con cui li inseriamo è importante.
NOTA2: al fondo delle righe di **hold on** e **grid on** NON vanno inserite le ;

- *Esempio*: grafico precedente con la griglia.

```
clf
x = linspace(0, 2*pi, 1000);
y = sin(x);
hold on
grid on
plot(x, y, 'black', 'LineWidth', 2)
axis([0 2*pi -1.1 1.1]);
set(gca, 'XTick', 0 : 0.25*pi : 2*pi);
```



Con la griglia è più leggibile

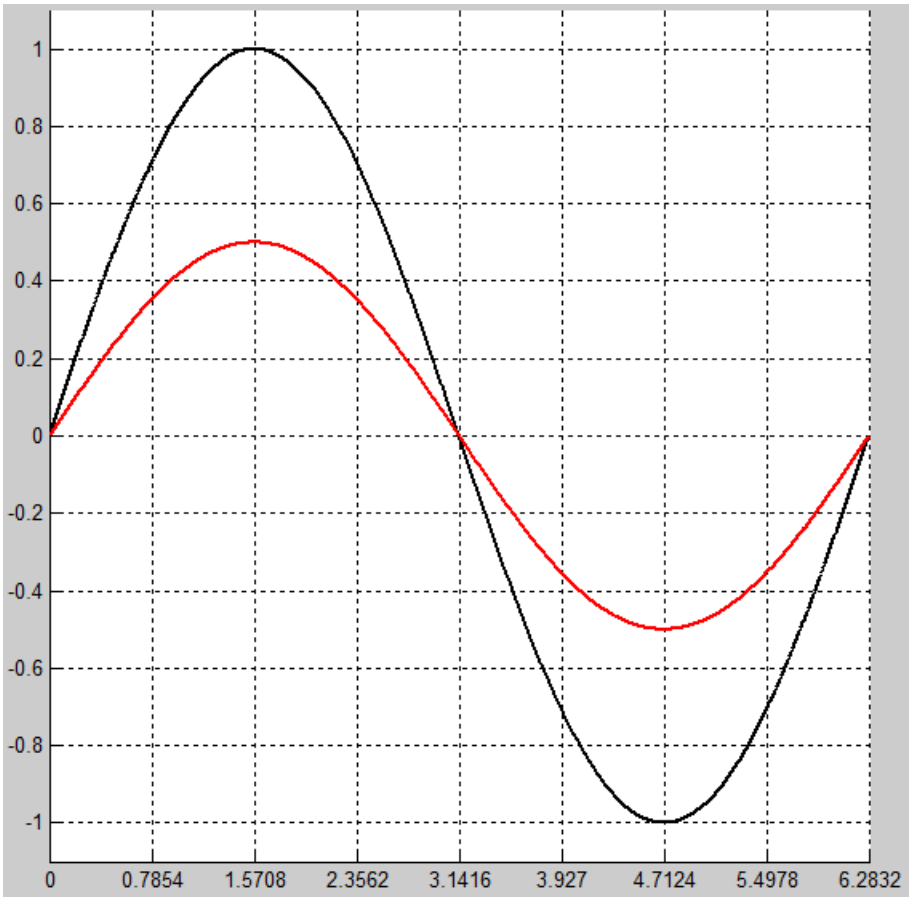
Disegnare più funzioni sullo stesso grafico

Tratto solo questo modo semplice, si potrebbe alternativamente utilizzare il comando hold on.
Come funziona:
plot(x, y1, coloreDiY1, x, y2, coloreDiY2, nomeOpzione1, valoreOpzione1)

y1 ed y2 sono due funzioni differenti della stessa variabile x, andranno definite entrambe prima del comando plot.
Possiamo impostare y1 ed y2 di colori differenti ma dobbiamo impostarle dello stesso spessore, 'LineWidth' ed il suo valore andranno al fondo della parentesi di plot.
NOTA: il comando senza opzioni è plot(x, y1, x, y2);
ATTENZIONE a come ho inserito le opzioni: prima metto x, y1, poi il colore di y1, poi inserisco x, y2, poi il colore di y2 ed infine le opzioni complessive come lo spessore.

- *Esempio:* grafico precedente con l'aggiunta della funzione y2 = 0.5*sin(x).
  La y dell'esempio precedente è la nostra y1 della spiegazione, y2 è invece la seconda funzione da inserire. Ho inserito y2 rossa, lo spessore settato è 2.

```
clf
x = linspace(0, 2*pi, 1000);
y = sin(x);
y2 = 0.5*sin(x);
hold on
grid on
plot(x, y, 'black', x, y2, 'red', 'LineWidth', 2)
axis([0 2*pi -1.1 1.1]);
set(gca, 'XTick', 0 : 0.25*pi : 2*pi);
```

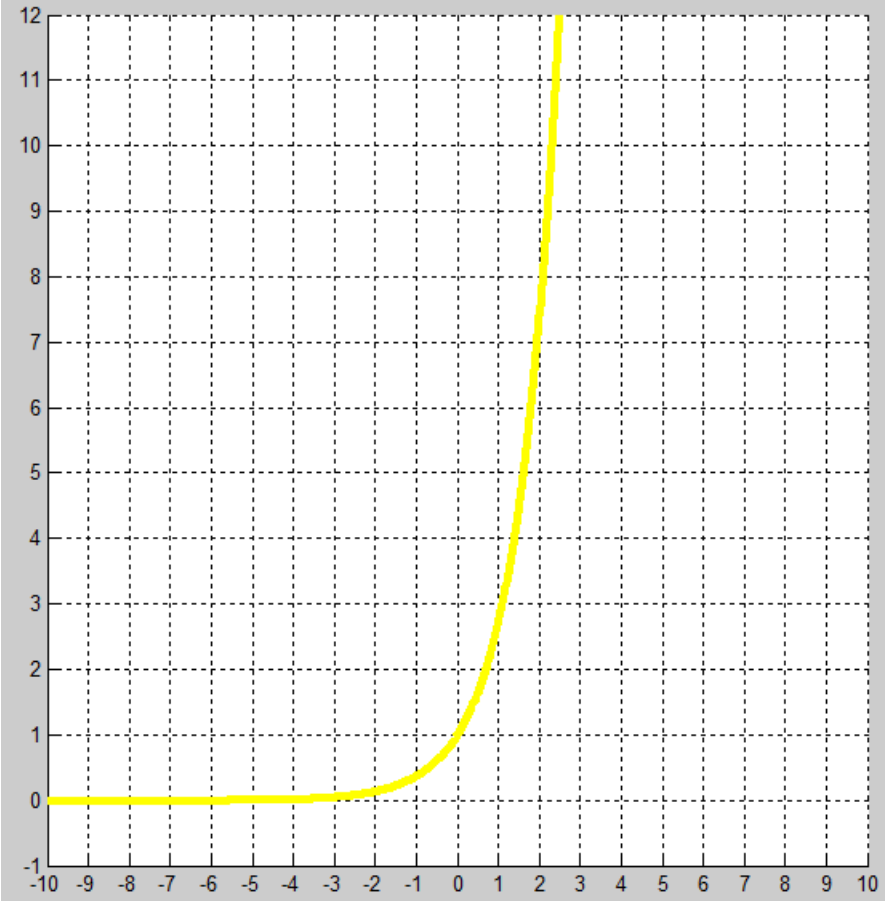


sin(x) e sin(x)/2

Esempi raffiguranti altre funzioni

- $y(x) = e^x$
  Ascisse da -10 a 10, ordinate da -1 a 12, tacche unitarie su entrambi gli assi, colore giallo e spessore 4. Griglia abilitata.

```
clf
x = linspace(-10, 10, 1000);
y = exp(x);
hold on
grid on
plot(x, y, 'yellow', 'LineWidth', 4)
axis([-10 10 -1 12]);
set(gca, 'XTick', -10 : 1 : 10);
set(gca, 'YTick', -1 : 1 : 12);
```



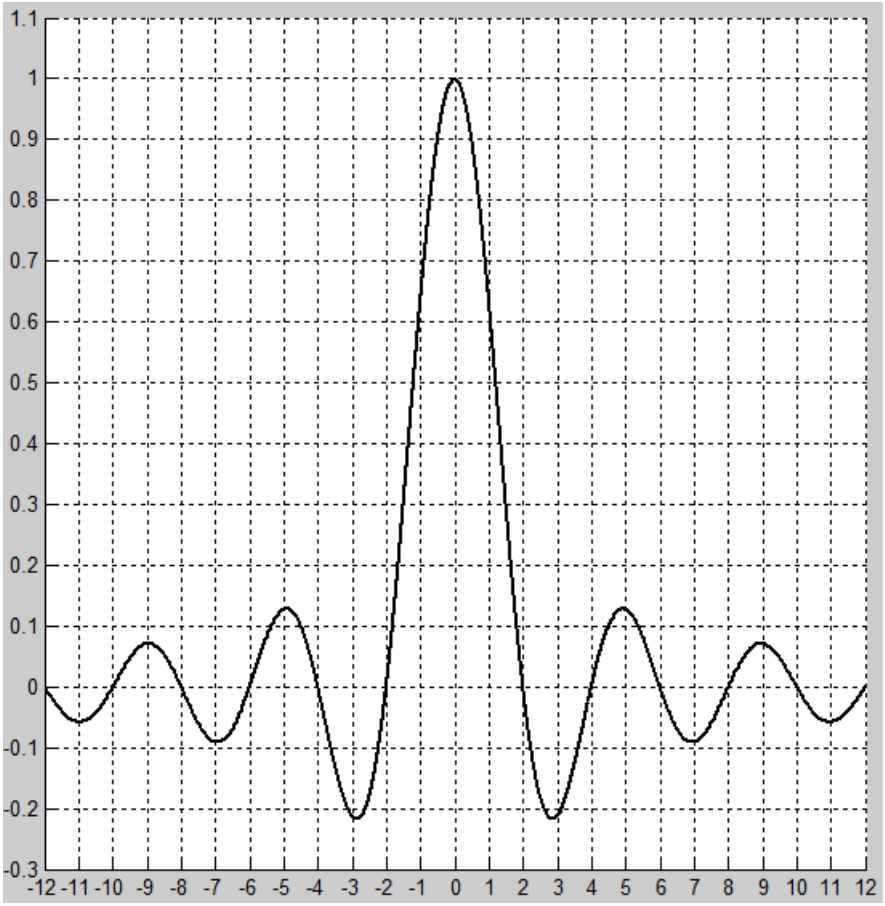
Esponenziale giallo

- $y(t) = sinc(\frac{t}{T}) = \frac{sin(\frac{\pi t}{T})}{\frac{\pi t}{T}}$

  Ho chiamato il vettore delle ascisse t ed aggiunto più punti (da 1000 a 10000) per visualizzarla in modo più definito. Ascisse da -12 a 12 a tacche unitarie, ordinate da -0.3 a 1.1 a tacche con step di 0.1. Colore nero, spessore 2, griglia abilitata.
  In questo esempio T è settato a 2.
  NOTA: in MATLAB non si possono usare le parentesi quadre nel definire le funzioni.

```
clf
t = linspace(-12, 12, 10000);
T = 2;
y = sin((pi*t)/T)./(pi*t/T);
hold on
grid on
plot(t, y, 'black', 'LineWidth', 2)
axis([-12 12 -0.3 1.1]);
```

```
set(gca, 'XTick', -12 : 1 : 12);
set(gca, 'YTick', -0.3 : 0.1 : 1.1);
```



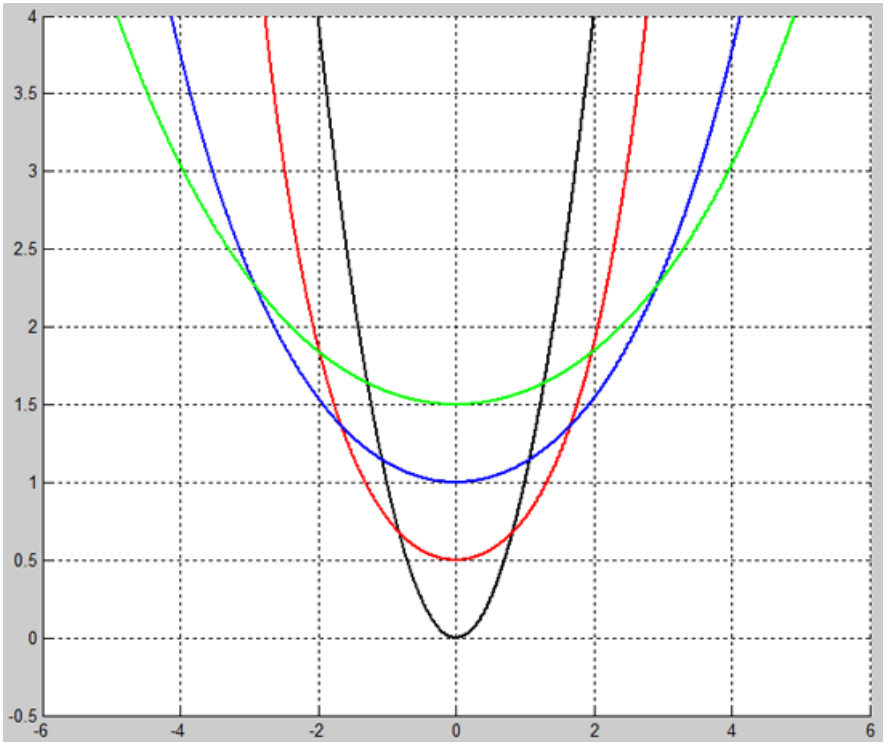
sinc(t/2)

- $y(x) = x^2, \quad y(x) = (\frac{a}{2})cosh(\frac{x}{a})$

  È possibile rappresentare anche più di due funzioni contemporaneamente. Questo esempio sfrutta la capacità del comando plot di rappresentare più funzioni allo stesso tempo; con un ciclo for, qualche if, hold on e più comandi plot potremmo disegnare parecchie catenarie al variare del parametro **a**.

  ```
  clf
  x = linspace(-6, 6, 10^4);
  y = x.^2;
  a = 1;
  y1 = (a/2)*cosh(x/a);
  a = 2;
  ```

```
y2 = (a/2)*cosh(x/a);
a = 3;
y3 = (a/2)*cosh(x/a);
hold on
grid on
plot(x, y, 'black', x, y1, 'red', x, y2, 'blue', x, y3, 'green', 'LineWidth', 2)
axis([-6 6 -0.5 4]);
```



Parabola (nera) e tre catenarie.

Conclusione

Sono stati trattati i seguenti comandi:

vettore = **linspace**(InizioVettore, FineVettore, QuantiPuntiLoCompongono);
plot(x, y, NomeOpzione1, ValoreOpzione1, NomeOpzione2, ValoreOpzione2);
axis([MinimoAscisse MassimoAscisse MinimoOrdinate MassimoOrdinate]);
set(gca, '**XTick**', primoPunto : step : ultimoPunto);
set(gca, '**YTick**', primoPunto : step : ultimoPunto);

Si è visto un uso basilare di **hold** e **grid**.

Il testo mira a far apprendere rapidamente i comandi di base senza trattare l'*estetica* dei grafici (titolo, titoli degli assi, ecc) e senza utilizzare cicli for o costrutti "complessi".

Mi scuso per il disordine ed i continui aggiornamenti.
Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Fedhman:guida-pratica-a-matlab-disegnare-funzioni-ad-una-variabile"