

La grafica in MATLAB

1. Introduzione

La potenzialità di MATLAB si evidenzia, senza alcun dubbio, nelle rappresentazioni grafiche dei dati. I grafici consentono di visualizzare enormi quantità di dati in modo sintetico e significativo. È per questo che è stata dedicata particolare attenzione allo sviluppo di un gran numero di funzioni in grado di generare grafici bidimensionali e tridimensionali, dotati di caratteristiche modificabili dall'utente.

Per cominciare a conoscere le possibilità grafiche offerte da MATLAB è essenziale trattare i seguenti argomenti:

- tipi di grafici (**plot**) e funzioni relative;
- caratteristiche dei grafici;
- *'editing'* dei grafici, ovvero modifica da parte dell'utente dei grafici prodotti;
- salvataggio, stampa ed *'esportazione'* del grafico, per poterlo utilizzare con altre applicazioni, ad esempio di videoscrittura.

- Disegniamo i nostri dati

Molte volte un esempio è il metodo più efficiente per introdurre argomenti complessi e vasti come quelli che riguardano le funzioni grafiche di MATLAB ed il loro utilizzo. Per questo esamineremo il processo di costruzione di un grafico cercando di introdurre le tematiche principali associate alla creazione di rappresentazioni di dati. Suddivideremo il lavoro in passi logici alla fine dei quali il grafico 'apparirà' come risultato del processo.

Passo 1: preparazione dei dati.

```
x = 0:0.2:12;          % Creazione del vettore x o variabile indipendente.  
y1 = bessel(1,x);      % Creazione dei vettore y1, y2, y3 in funzione di x.  
y2 = bessel(2,x);  
y3 = bessel(3,x);
```

Il passo iniziale consente di creare un vettore x contenente i valori della variabile indipendente 'distribuiti' uniformemente tra 0 e 12 con incremento di 0.2. Questi valori, inseriti nelle funzioni di Bessel di primo, secondo e terzo ordine, generano i vettori y1, y2 e y3.

Passo 2: selezione della finestra ed eventualmente della porzione di essa in cui si vuole rappresentare il grafico.

```
figure(1)              % Fa comparire una finestra grafica vuota numerata nella  
                        % quale sarà inserito il grafico.
```

Una volta calcolati i vettori y1, y2, y3 si utilizza la funzione **figure**(1) per 'dare un contenitore' al grafico che costruiremo. In fig. 1 viene mostrato il risultato della funzione: appare una finestra vuota denominata **Figure No.1**, che conterrà in seguito il grafico. È possibile anche, mediante la funzione **subplot** specificare la porzione di finestra che vogliamo utilizzare, nel caso di rappresentazione di più grafici in una stessa finestra.

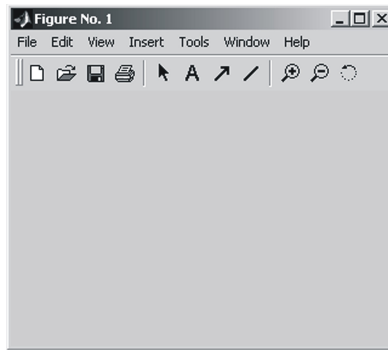


Fig. 1 Finestra vuota ottenuta con la funzione **figure(1)**.

Passo 3: plot dei dati (esecuzione del grafico).

```
h = plot(x,y1,x,y2,x,y3);           % Disegna il grafico bidimensionale
```

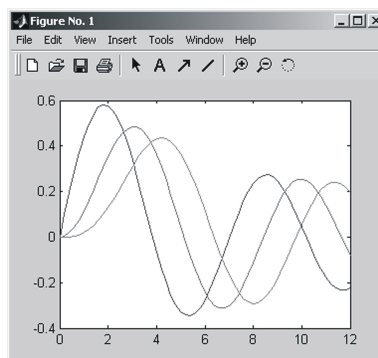


Fig. 2 Disegno delle tre funzioni di Bessel.

La funzione **plot** consente di disegnare il grafico delle tre funzioni. Si noti come la sintassi della funzione sia estremamente semplice: per ogni grafico basta inserire come parametro il vettore indipendente e quello dipendente. Da notare anche che la funzione **plot** sceglie automaticamente la scala del disegno in modo tale da massimizzare il grafico rispetto alle dimensioni della finestra.

Passo 4: modifica delle caratteristiche delle linee.

```
set(h,'LineWidth',1,{'LineStyle'},{'--' ; ':' ; '-.'})    % Definisce spessore e stile di linea
set(h,{'Color'},{'r' ; 'g' ; 'b'})                       % Definisce il colore delle linee
```

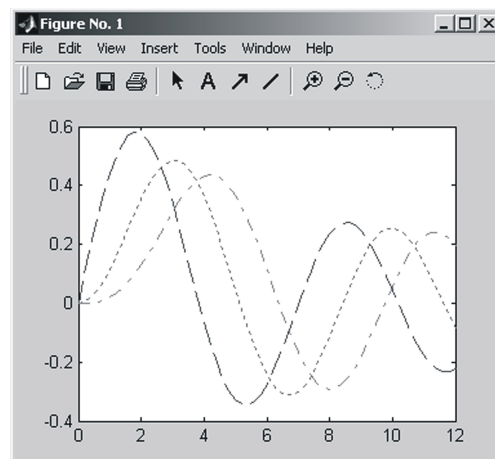


Fig. 3 Modifica dello stile delle linee.

La funzione `plot` disegna *per default*, cioè in modo predefinito, le linee con le seguenti caratteristiche: linea continua di colore differente per ogni funzione tracciata (l'effetto del colore non è purtroppo rappresentabile nel testo). Se per esigenze di chiarezza del grafico si vogliono modificare le caratteristiche delle linee è possibile operare come indicato nel codice.

La funzione **set** permette di:

- selezionare, all'interno del grafico **h** appena disegnato, lo spessore della linea mediante i parametri `'LineWidth',1;`
- scegliere lo stile delle linee che rappresentano le diverse funzioni (come il tratteggio, ecc.) con i parametri `{'LineStyle'},{'--'; ':' ; '-.'};`
- impostare i colori delle linee con i parametri `{'Color'},{'r';'g';'b'}`, (dove r, g e b stanno rispettivamente per red, green e blue).

Passo 5: impostazione dei limiti e denominazione degli assi, griglia e legenda.

```
axis([0 12 -0.5 1]) % Impostazione limiti degli assi.
grid on             % Attivazione della griglia.
xlabel('Tempo'), ylabel('Ampiezza') % Visualizzazione dei nomi degli assi x e y.
legend(h,'Prima','Seconda','Terza') % Legenda per riconoscere le funzioni
title('Funzioni di Bessel')          % Impostazione del titolo del grafico
[y,ix] = min(y1); % Determinazione di un punto notevole : il minimo della funzione y1
text(x(ix),y,'Primo minimo \rightarrow','HorizontalAlignment','right')
```

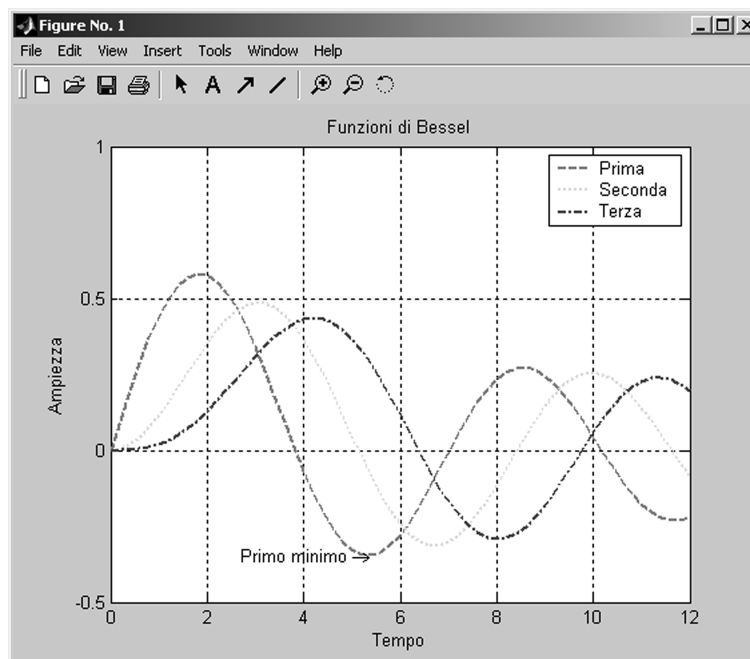


Fig. 4 Grafico completo di griglia, assi, legenda e con indicazione di un punto notevole come il primo minimo raggiunto dalle funzioni.

Le funzioni citate nel passo 5 consentono di completare l'aspetto informativo del grafico con l'inserimento di testi esplicativi. In particolare:

- **axis** modifica gli intervalli di *plotting* del grafico;
- **grid on** rende visibile una griglia;
- **xlabel** e **ylabel** assegnano i nomi agli assi;
- **legend** crea e visualizza una legenda che permette, sulla base dello stile delle linee usato, di riconoscere le diverse funzioni rappresentate;

- **title** imposta il nome dell'intero grafico;
- **min** determina il minimo della funzione indicata;
- **text** inserisce una casella di testo in un punto del grafico.

Passo 6: esportazione del grafico su file con formato **.eps** (Encapsulated PostScript) che è uno dei formati grafici standard. Il grafico è quindi inseribile nei documenti prodotti con gli applicativi (programmi) normalmente utilizzati come i programmi di videoscrittura.

```
print -depsc -tiff -r300 grafico      % Il comando crea un file grafico.eps
                                     % con definizione di 300 pixel per pollice (2.54 cm).
```

- Le funzioni grafiche principali

MATLAB fornisce una varietà di funzioni in grado di rappresentare dati di vettori in modo grafico. La tab. 1 riporta le funzioni principali, sintetizzando il loro funzionamento. Le principali differenze tra queste funzioni riguardano le scale utilizzate negli assi.

Funzione	Descrizione
plot	Grafico 2-D (bi-dimensionale) con scale lineari sui due assi
plot3	Grafico 3-D (tri-dimensionale) con scale lineari su tutti gli assi
logplot	Grafico con scale logarithmiche su entrambi gli assi
semilogx	Grafico con scala logarithmica sull'asse x e lineare sull'asse y
semilogy	Grafico con scala logarithmica sull'asse y e lineare sull'asse x
plotyy	Disegna due grafici con medesimo asse delle x e differenti assi delle y posizionati a destra e a sinistra (serve per disegnare contemporaneamente due funzioni con scale diverse)

Tab. 1 Tabella delle funzioni grafiche.

Le sintassi della funzione **plot** sono le seguenti:

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,Specifiche della linea,...)
h = plot(...)
```

Descrizione

plot(Y)

Disegna i dati della variabile Y considerando come ascisse i corrispondenti numeri ordinali degli elementi. Se ad esempio il vettore Y possiede 500 elementi il grafico parte dall'ascissa 1 e termina al valore 500.

plot(X1,Y1,...)

Permette di disegnare più grafici all'interno di una finestra assegnando i valori delle ascisse e delle ordinate.

plot(X1,Y1,Specifiche della linea,...)

È come il comando precedente nel quale però è possibile aggiungere le caratteristiche della linea mediante una stringa contenente simboli speciali che specificano:

- **tipo di linea.** I codici sono indicati in tab. 2

Specificatori	Stile linea
-	Linea continua(default)
--	Linea tratteggiata
:	Linea puntiforme
-.	Linea tratto e punto

Tab. 2 Tabella degli specificatori di linea.

- **tipo di marcatori.** Esistono molti marcatori come '.', '+', '*', 's', ecc. che indicano un particolare simbolo usato per rappresentare la funzione disegnata. Ad esempio il marcatore 's' traccia la funzione per mezzo di piccoli quadrati.
- **colore.** I codici dei colori riprendono l'iniziale della parola inglese: 'r' per red (rosso), 'g' per green (verde) e così via.

Il comando seguente: **plot(x,y,'-sg')** traccerebbe il grafico con linea continua verde alla quale sono sovrapposti quadratini anch'essi verdi.

h = plot(...)

Restituisce un vettore colonna di **handles** (collegamenti) agli oggetti grafici

ESEMPIO 1

Questo esempio mostra come sia possibile visualizzare più grafici contemporaneamente, situati in una matrice di grafici come in fig. 5 nella quale la matrice è di 2 per 2.

Supponiamo di tracciare i grafici di quattro funzioni di tipo trigonometrico indicate come y1, y2, y3 e y4.

Le istruzioni per generare la figura sono le seguenti:

```
>>x=linspace(0, 2*pi, 1000);      % Generazione del vettore indipendente x.

>>y1=sin(x); y2=cos(x);    % Calcolo delle quattro funzioni da rappresentare.
>>y3=sin(x) .* cos(5*x);  y4=sin(5*x) ./ (cos(x)+2);

>>subplot(2,2,1), plot(x, y1)      % Grafico di y1.
>>xlabel ('variabile indipendente x'), ylabel ('grafico di sin(x)')

>>subplot(2,2,2), plot(x, y2)      % Grafico di y2.
>>xlabel ('variabile indipendente x'), ylabel ('grafico di cos(x)')

>>subplot(2,2,3), plot(x, y3)      % Grafico di y3.
>>xlabel ('variabile indipendente x'), ylabel ('grafico di sin(x)cos(5x)')
```

```
>>subplot(2,2,4), plot(x, y4)      % Grafico di y4.
>>xlabel ('variabile indipendente x'), ylabel ('grafico di sin(5x) / (cos(x)+2)')
```

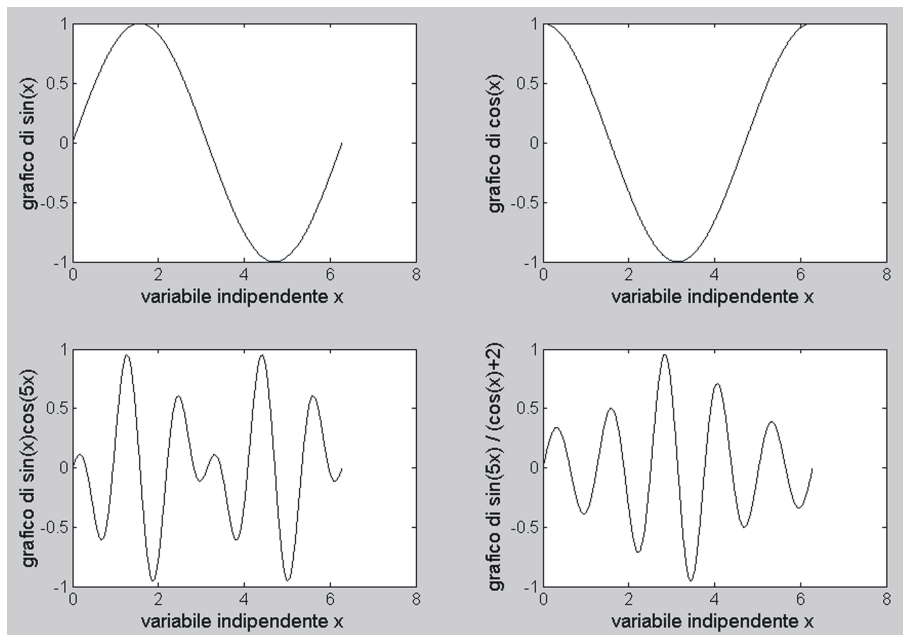


Fig. 5 Grafico multiplo o matrice di grafici 2 per 2.

2. Le funzioni grafiche speciali

MATLAB fornisce il supporto necessario per la costruzione di una grande varietà di grafici il cui utilizzo dipende dalla natura dei dati. Limiteremo il nostro studio ad alcune funzioni speciali a due dimensioni.

Prendiamo in considerazione l'elenco delle tipologie grafiche speciali al fine di specificarne le caratteristiche essenziali. I tipi di grafici possibili sono:

- **Grafici a barre e ad area.** Sono usati in genere per rappresentazioni indipendenti dal tempo, per comparare risultati e visualizzare i contributi individuali rispetto all'ammontare totale.
- **Grafici a torta.** Mostrano i contributi individuali rispetto all'ammontare totale.
- **Istogrammi.** Evidenziano la distribuzione dei valori dei dati.
- **Grafici a dati discreti.** Per rappresentazioni di dati campionati.
- **Rappresentazioni vettoriali di dati.** Ogni dato viene rappresentato come un vettore.
- **Rappresentazioni del contorno di regioni.** Disegnano linee a dati equivalenti ovvero linee passanti in punti in cui i dati sono uguali.
- **Grafici interattivi.** Si utilizzano per selezionare dei punti del grafico mediante un puntatore grafico.
- **Animazioni.** Permettono di rappresentare sequenze di grafici come immagini ottenendo effetti di movimento.

Grafici a barre e ad area

Questi grafici visualizzano dati di vettori o matrici, in particolare per comparare risultati e visualizzare i contributi individuali rispetto all'ammontare totale. I grafici a barre sono utilizzati spesso per rappresentare dati discreti, mentre quelli ad area per visualizzare i dati continui.

Numerose sono le funzioni associate a questo tipo di grafici. La tab. 3 sintetizza il loro funzionamento.

Funzione	Descrizione
bar	Visualizza le colonne di una matrice m per n come m gruppi di n barre verticali
barh	Visualizza le colonne di una matrice m per n come m gruppi di n barre orizzontali
bar3	Visualizza le colonne di una matrice m per n come m gruppi di n barre verticali in 3-D
bar3h	Visualizza le colonne di una matrice m per n come m gruppi di n barre verticali in 3-D
area	Visualizza i dati del vettore come area

Tab. 3 Tabella delle funzioni per i grafici a barre e ad area.

Tutte le funzioni inserite in tab. 3 possono essere utilizzate con varie sintassi. Noi mostreremo solo alcuni esempi applicativi, rimandando l'analisi dettagliata di tutte le altre sintassi all'Help di MATLAB che le illustra ampiamente.

ESEMPIO 2

Supponiamo di rappresentare i dati di una matrice Y di 5 righe e 3 colonne:

```
>> Y=[5 2 1; 8 7 3; 9 8 6; 5 5 5; 4 3 2];
Y =
     5     2     1
     8     7     3
     9     8     6
     5     5     5
     4     3     2
>> bar(Y)
```

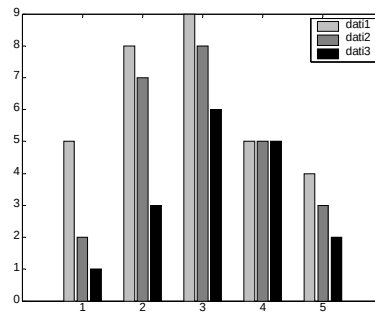


Fig. 6 Grafico a barre della matrice Y.

La rappresentazione ottenuta nella figura 6 evidenzia i 5 gruppi di barre verticali.

ESEMPIO 3

Una rappresentazione tridimensionale (3-D) della stessa matrice si può ottenere con la funzione

```
>>bar3(Y)
```

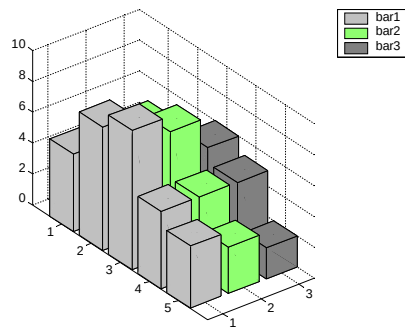


Fig. 7 Grafico a barre 3-D della matrice Y.

ESEMPIO 4

I grafici ad area rappresentano in modo cumulativo e relativo gli elementi di una matrice. Ogni colonna della matrice è rappresentata con un'area di colore diverso con i valori dei vertici uguali agli elementi delle righe. Questo tipo di grafico è molto utilizzato per visualizzare i diversi contributi di dati diversi al valore globale.

```
>> area(Y)
```

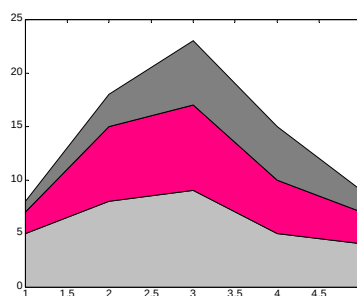


Fig. 8 Grafico ad area.

Nota. Si osservi come ad esempio nel punto di ascissa 3 il grafico abbia il valore $9+8+6 = 23$ pari alla somma degli elementi della terza riga della matrice Y.

Istogrammi

Solitamente gli istogrammi vengono utilizzati per rappresentare distribuzioni di dati, come quelle tipiche della statistica. Le funzioni sono sinteticamente descritte in tab. 4. La differenza tra le funzioni risiede nel tipo di sistema di coordinate utilizzato nel grafico.

Funzione	Descrizione
Hist	Visualizza i dati in un sistema di coordinate cartesiane
Rose	Visualizza i dati in un sistema di coordinate polari

Tab. 4 Tabella delle funzioni per i grafici a barre e ad area.

Nota. Le coordinate polari di un punto sono il modulo e l'argomento (angolo) del vettore che unisce l'origine degli assi al punto stesso.

La funzione **hist** conta il numero di elementi contenuti in un 'range' (intervallo) e rappresenta ogni range come un rettangolo di altezza pari al numero di elementi appartenenti al range stesso.

ESEMPIO 5

```
>> y = randn(10000,1);    % Crea un vettore con elementi casuali con distribuzione
                           % di probabilità normale.
>> x = min(y):0.2:max(y); % Determina il minimo ed il massimo del vettore e imposta
                           % la dimensione ed il numero dei range rappresentati.
>> hist(y,x)              % Disegna l'istogramma.
```

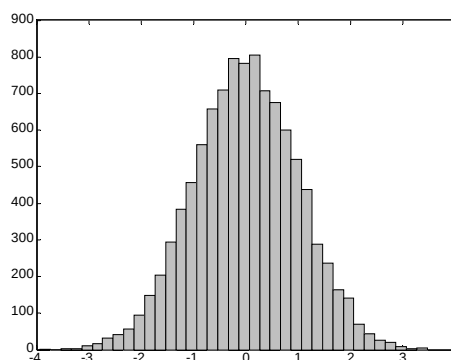


Fig. 9 Istogramma della funzione a campana o curva normale.

3. Grafici a dati discreti

Il principale utilizzo di questo tipo di grafici è legato alle rappresentazioni di dati campionati. In altre parole una funzione continua può essere ‘campionata’, nel senso di prelevare per certi valori dell’ascissa (ad istanti prefissati se la funzione è dipendente dal tempo) campioni della funzione stessa, per cui la funzione viene rappresentata come un insieme di linee verticali di altezza proporzionale al valore dei campioni. Questi dati formano, si dice, una sequenza numerica discreta. Nell’es.6 verrà generato un vettore e disegnato prima nel modo classico, per mostrare l’andamento della funzione, poi in modo discreto.

Le funzioni speciali di questo tipo sono elencate in tab. 5.

Funzione	Descrizione
stem	Visualizza i dati di una sequenza discreta inserita in un vettore Y dipendente da un asse X
stem3	Visualizza i dati di una sequenza discreta inserita in un vettore Z dipendente da un piano XY
stairs	Visualizza i dati di una sequenza discreta inserita in un vettore Y come scalini dipendenti da un asse X

Tab. 5 Tabella delle funzioni per i grafici a dati discreti.

Anche queste funzioni inserite in tab. 5 possono essere utilizzate con varie sintassi (vedi l’Help in linea).

ESEMPIO 6

Costruiamo una funzione $y = f(x)$ e rappresentiamola, dapprima in modo continuo (fig. 10), poi in modo discreto (figg. 11-12-13).

```
>> x=linspace(0, 2*pi, 100); % Genera un vettore con 100 elementi equispaziati.  
>> y=sin(5*x) ./ (cos(x)+2); % Funzione dipendente.  
>> plot(x, y) % Traccia il grafico continuo.
```

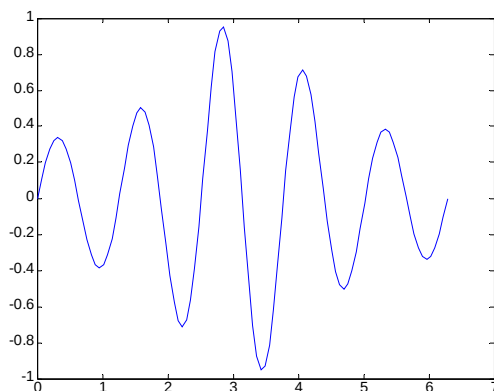


Fig. 10 Rappresentazione continua della funzione y dell’es. 6

Utilizzando la funzione

```
>> stem(x,y) % Traccia il grafico discreto.
```

si ottiene il grafico di fig. 11. Si osservi come la funzione sia definita solo in alcuni punti dell'asse x.

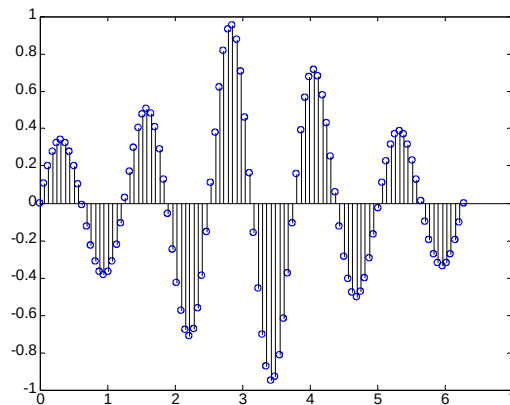


Fig. 11 Rappresentazione discreta della funzione y dell'es. 29.

Se utilizziamo la funzione

```
>> stairs(x,y) % Traccia il grafico discreto a gradinata.
```

Otterremo il grafico di fig. 12, nel quale ogni singolo valore della sequenza discreta costituisce 'un gradino' di una scalinata.

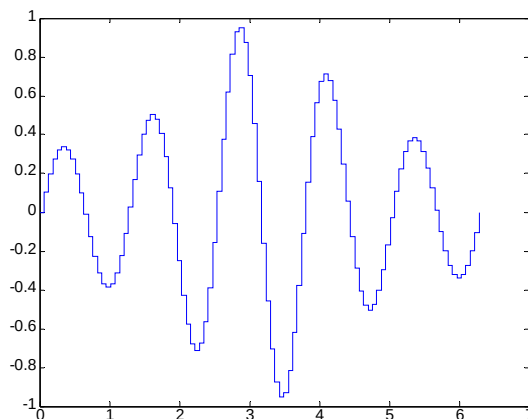


Fig. 12 Grafico a scalinata ottenuto con la funzione **stairs**.

Nota. Mediante il comando **hold on** è possibile disegnare un nuovo grafico sovrapponendolo al grafico precedente. Per cui inserendo i comandi

```
>> hold on  
>> plot(x,y)
```

si ottengono due grafici della stessa funzione, quello continuo sovrapposta quello discreto. Nella fig. 13 si può vedere un ingrandimento di una zona della figura ottenuta ed osservare la relazione tra i due grafici.

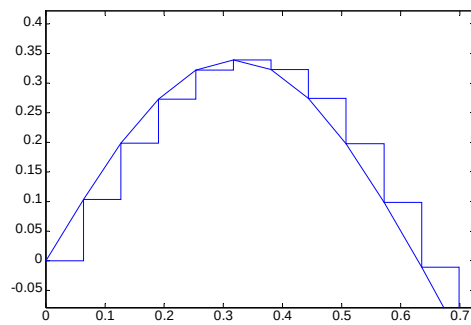


Fig. 13 Ingrandimento di una zona della funzione.

- Caricamento dei grafici

L'operazione inversa al salvataggio, il caricamento, permette di ripristinare uno spazio di lavoro precedentemente salvato.

La procedura per caricare lo spazio di lavoro è la seguente:

1. dal menu **File**, selezionare **Import data**: si aprirà la finestra di dialogo **Open**;
2. selezionare il file **.mat**;
3. premere il tasto **Open**.

4. Capacità grafiche aggiuntive

I grafici più comuni utilizzati dagli ingegneri e dagli scienziati sono quelli cartesiani. I dati che utilizziamo per generare il grafico vengono di solito letti da un file o calcolati nel nostro programma e memorizzati in vettori che possiamo chiamare x e y . Generalmente supponiamo che i valori di x rappresentino la variabile indipendente e che i valori di y rappresentino la variabile dipendente. I valori di y possono essere calcolati come funzione di x , o i valori di x e y possono essere misurati nel corso di un esperimento. Ora presentiamo alcuni modi diversi per rappresentare questa informazione.

La maggior parte dei grafici che generiamo suppongono che gli assi x e y siano suddivisi in intervalli ugualmente spazati; questi grafici sono chiamati grafici lineari. A volte, potremmo voler utilizzare invece una scala logaritmica su uno o entrambi gli assi. Una scala logaritmica (in base 10) è conveniente quando una variabile varia su molti ordini di grandezza poiché questo vasto intervallo di valori può essere rappresentato senza comprimere i valori più piccoli.

In MATLAB i comandi per disegnare grafici lineari e logaritmici dei vettori x e y sono i seguenti.

plot(x,y) Disegna un grafico lineare dei valori di x e y .

semilogx(x,y) Disegna un grafico dei valori di x e y utilizzando una scala logaritmica per x e lineare per y .

semilogy(x,y) Disegna un grafico dei valori di x e y utilizzando una scala lineare per x e una logaritmica per y .

loglog(x,y) Disegna un grafico dei valori di x e y utilizzando una scala logaritmica per entrambi gli assi.

È importante capire che il logaritmo di un numero negativo o dello zero non esiste.

Se i dati necessari per la creazione di un grafico semilogaritmico o logaritmico contengono valori negativi o degli zeri, MATLAB stamperà un messaggio che vi informa che questi dati sono stati omessi dal grafico.

Ognuno di questi comandi può anche essere eseguito con un solo argomento, come nel caso di `plot(y)`. In questo caso, i grafici saranno generati con i valori degli indici del vettore `y` utilizzati come se fossero dei valori di `x`.

- Grafici multipli

Un metodo per disegnare molte curve sullo stesso grafico è quello di utilizzare il comando `hold on`. Come già illustrato, un altro metodo è quello di utilizzare argomenti multipli, come in:

`plot(X,Y,W,Z)`

dove le variabili `X`, `Y`, `W` e `Z` sono vettori. Quando viene eseguito questo comando, verrà disegnata la curva di `Y` in funzione di `X` e poi la curva di `W` in funzione di `Z` verrà disegnata sullo stesso grafico. Il vantaggio di questa tecnica è che MATLAB sceglierà automaticamente tipi di linea e colori differenti in modo da distinguere i due grafici.

Se la funzione `plot` viene chiamata con una matrice come singolo argomento, MATLAB disegnerà grafici diversi per ogni colonna della matrice. L'asse `x` viene indicizzato con gli indici dei vettori riga `1:k`, dove `k` è il numero di righe nella matrice.

Se la funzione `plot` viene chiamata con due argomenti di cui uno è un vettore e l'altro una matrice, MATLAB disegnerà in successione una linea per ogni riga (o colonna) della matrice.

Esempio

Per esempio, digitate i seguenti comandi per creare alcuni dati per i grafici:

```
X=0.0:pi/100:2*pi;  
Y1 =cos(X)*2;  
Y2 =cos(X)*3;  
Y3 =cos(X)*4;  
Y4 =cos(X)*5;  
Z =[Y1;Y2;Y3;Y4 ];
```

Immettete il seguente comando `plot` e osservate i risultati:

```
plot(X,Y1,X,Y2,X,Y3,X,Y4)
```

La seguente istruzione `plot` produrrà lo stesso risultato utilizzando l'argomento matrice `Z`:

```
plot(X,Z)
```

5. Stili per le linee e i simboli

Il comando `plot(x,y)` disegna un grafico con una linea che connette i punti rappresentati dai vettori `x` e `y` con alcuni segmenti. Potete anche selezionare altri tipi di linea: tratteggiata, a punti, e a punti e trattini, oppure selezionare un grafico a dispersione, invece che uno a linea. Con un grafico a dispersione, i punti rappresentati dal vettore verranno marcati con un punto invece che con dei segmenti di linea. Potete anche selezionare simboli diversi dal punto; le altre scelte sono il segno

più, la stella, il cerchio, e la x. La Tabella 6 contiene queste opzioni diverse per le linee e i simboli.

Tipo di linea	Indicatore	Tipo di punto	Indicatore
continua	-	punto	.
tratteggiata	--	più	+
a puntini	:	stella	*
tratteggio e puntini	-.	cerchio	o
		x	x

Tab 6. Opzioni per le linee e i simboli.

Il seguente comando illustra l'utilizzo degli stili per le linee e i simboli. Esso genera un grafico con una linea dei punti rappresentati dai vettori x e y e poi disegna i punti stessi con dei cerchi:

```
plot(x,y,x,y,'o')
```

La scala degli assi

MATLAB scala automaticamente gli assi per adattarli ai valori dei dati. Tuttavia, potete cambiare questa scala con il comando `axis`. Ci sono numerose forme del comando `axis`.

axis Congela la scala attuale per i grafici seguenti. Una seconda esecuzione del comando ripristina il sistema di scalatura automatica.

axis(V) Specifica gli assi utilizzando un vettore V a quattro elementi che contiene i valori di scala [xmin,xmax,ymin,ymax].

Questi comandi sono utili specialmente quando volete confrontare curve di grafici diversi, infatti tale operazione può risultare difficile se le curve sono disegnate con assi diversi. Il comando `plot` precede il comando `axis` corrispondente.

Sottografici

Il comando `subplot` vi permette di dividere la finestra grafica in **sottofinestre**. Le possibili suddivisioni sono due o quattro sottofinestre. Due sottofinestre possono essere sistemate sia in alto o basso sia a destra o a sinistra. Una divisione in quattro finestre ha due sottofinestre in alto e due sottofinestre in basso. Gli argomenti di `subplot` sono tre interi *m*, *n*, *p*. Gli argomenti *m* e *n* specificano che la finestra grafica deve essere divisa in una griglia *m* per *n* di sottofinestre, e l'argomento *p* specifica la *p*-esima finestra per il grafico corrente. Le finestre sono numerate da sinistra a destra e dal basso verso l'alto. Perciò, i seguenti comandi specificano che la finestra grafica deve essere divisa in un grafico in alto e uno in basso e il grafico attuale deve essere posizionato in alto:

```
subplot(2,1,1), plot(x,y)
```

La Figura 14 contiene quattro grafici che illustrano il comando `subplot` insieme con i comandi per i grafici lineari e logaritmici. Questa figura è stata generata utilizzando le seguenti istruzioni:

```
% Disegna il grafico di polinomio.
%
x = 0:0.5:50;
```

```

y = 5*x.^2;
subplot(2,2,1),plot(x,y),
title('Polinomio - lineare/lineare'),
ylabel('y'),grid,
subplot(2,2,2),semilogx(x,y),
title('Polinomio - logaritmico/lineare'),
ylabel('y'),grid,
subplot(2,2,3),semilogy(x,y),
title('Polinomio - lineare/logaritmico'),
xlabel('x'),ylabel('y'),grid,
subplot(2,2,4),loglog(x,y),
title('Polinomio - logaritmico/logaritmico'),
xlabel('x'),ylabel('y'),grid

```

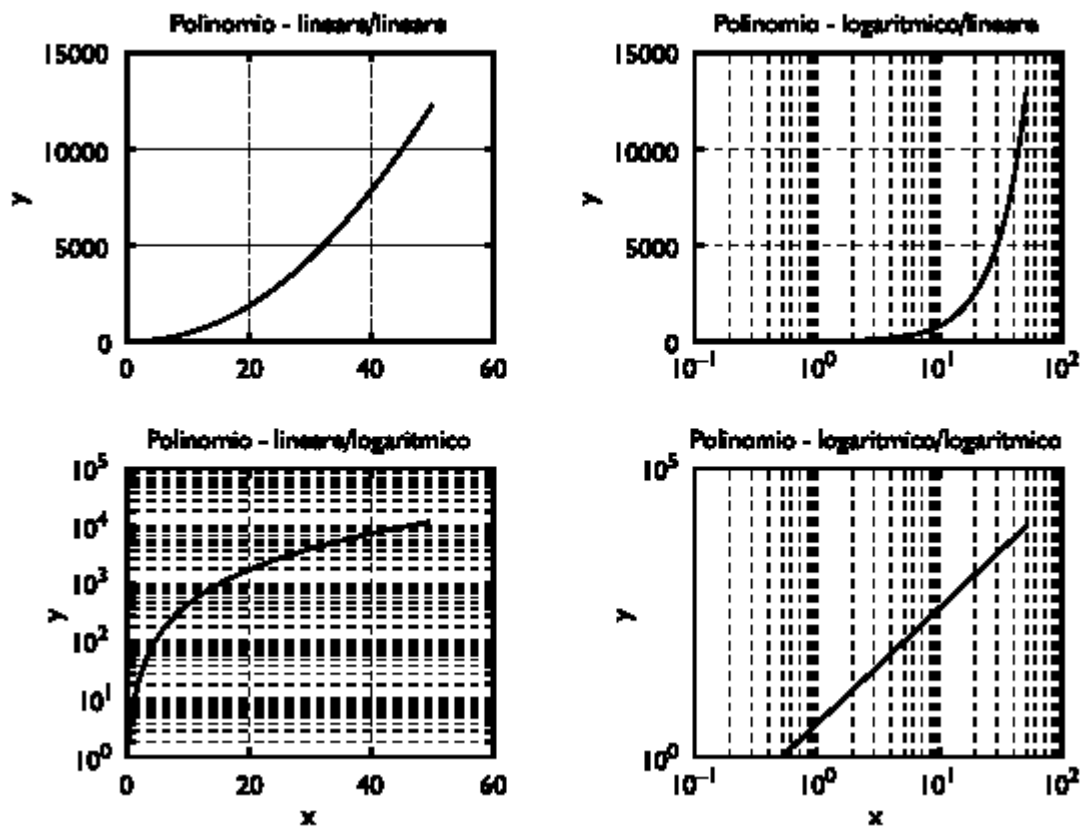


Figura 14 Grafici lineari e logaritmici