Master in "Calcolo Scientifico"

Diparimento di Matematica

# Laboratorio di Visualizzazione

# Grafica in Matlab

M. Rorro, M. Sagona, S. Tozza

Sapienza Università di Roma

A.A. 2016-2017

# Prefazione

Queste dispense vogliono essere un supporto per gli studenti del corso *Laboratorio di Visualizzazione*, previsto all'interno del Master di II Livello in Calcolo Scientifico. Sulla base del materiale raccolto nell'a.a. 2005-2006 da M. Rorro e M. Sagona in una prima versione di note per il corso, tali dispense sono state riprese, ampliate ed aggiornate da S. Tozza, docente titolare del corso per il corrente anno accademico 2016-2017.

Roma, 4 Maggio 2017

# Indice

# Capitolo 1

# Grafica in Matlab

## 1.1 Grafici in due dimensioni

### 1.1.1 Grafici elementari in due dimensioni

#### plot

plot 2D lineare

Sintassi

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
plot(...,'PropertyName',PropertyValue,...)
plot(axes_handle,...)
h = plot(...)
```

Descrizione

`plot(Y)` disegna le colonne di Y in funzione dei suoi indici se Y é reale. Se Y é complesso, `plot(Y)` é equivalente a `plot(real(Y),imag(Y))`. In tutti gli altri casi la parte immaginaria di Y viene ignorata.

`plot(X1,Y1,...)` disegna le colonne di Yn in funzione delle colonne di Xn. Se solo Xn o Yn é una matrice, é disegnata la colonna o righa di Yn che coincide con la lunghezza di Xn.

`plot(X1,Y1,LineSpec,...)` disegna tutte le linee definite da Yn in funzione di Xn con il tipo di linea, il simbolo per i punti ed il colore specificato da Linespec (vedi Tabelle 1.1,1.2,1.3). Ad esempio, `plot(X1,Y1,X2,Y2,'-ro',X3,Y3)` disegnerá Y2 in funzione di X2 in rosso unendo i punti, marcati dal simbolo 'o' con una linea continua mentre (X1,Y1) e (X3,Y3) saranno disegnati con il Linespec di default.

`plot(...,'PropertyName',PropertyValue,...)` configura la proprietà 'PropertyName' al valore PropertyValue per tutti gli oggetti creati dal plot. Ad esempio,
`plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g', ... 'MarkerSize',10)`.

`plot(axes_handle,...)` disegna negli assi indicati da axes_handle invece che negli assi correnti.

`h = plot(...)` assegna ad h un vettore di puntatori ad ogni linea creata dal plot.

| Simbolo | Tipo di linea |
|---|---|
| – | continua (default) |
| – – | trattegiata |
| : | punteggiata |
| –. | tratto punto |

Tabella 1.1: Tipi di linea

| Simbolo | colore | RGB value |
|---|---|---|
| r | rosso | [1 0 0] |
| g | verde | [0 1 0] |
| b | blu | [0 0 1] |
| c | ciano | [0 1 1] |
| m | magenta | [1 0 1] |
| y | giallo | [1 1 0] |
| k | nero | [0 0 0] |
| w | biaco | [1 1 1] |

Tabella 1.2: colori

| Simbolo | indicatore |
|---|---|
| + | segno più |
| o | cerchio |
| * | asterisco |
| . | punto |
| x | croce |
| 'square' o s | quadrato |
| 'diamond o d | rombo |
| $\wedge$ | triangolo $\triangle$ |
| v | triangolo $\nabla$ |
| $>$ | triangolo $\triangleright$ |
| $<$ | triangolo $\triangleleft$ |
| 'pentagram' o p | stella a 5 punte |
| 'hexagram' o h | stella a 6 punte |

Tabella 1.3: Tipi di punti

E' possibile inoltre specificare

- LineWidth – la larghezza (in punti) della linea
- MarkerEdgeColor – il colore dei simboli o del loro contorno per il cerchio, quadrato, rombo, stelle e triangoli
- MarkerFaceColor – il colore di riempimento dei simboli
- MarkerSize – la dimensione in punti dei simboli

In tutte le espressioni che richiedono un colore, e' possibile indicare invece della stringa il suo valore RGB. Ciò é utile se si vogliono utilizzare colori non descritti dalle strighe nella Tabella 1.2.

## fplot

Disegna una funzione in un intervallo specificato

Sintassi

```
fplot(fun,limits)
fplot(fun,limits,LineSpec)
fplot(fun,limits,tol)
fplot(fun,limits,tol,LineSpec)
fplot(fun,limits,n)
fplot(axes_handle,...)
[X,Y] = fplot(fun,limits,...)
```

Descrizione

`fplot(fun,limits)` disegna `fun` nell'intervallo specificato da `limits`. `limits` é un vettore che contiene i limiti dell'asse x (`[xmin xmax]`), oppure dell'asse x e y (`[xmin xmax ymin ymax]`). `fun` é

- Il nome di una funzione M-file
- Una stringa con la variabile x che può essere passata ad `eval`, e.g. 'sin(x)', 'diric(x,10)', o '[sin(x),cos(x)]'
- Un puntatore a funzione per una funzione M-file (`fhandle = @functionname`) o una funzione anonima `sqr = @(x) x.^2`

`fplot(fun,limits,LineSpec)` disegna `fun` come specificato in LineSpec

`fplot(fun,limits,tol)` disegna `fun` con un errore di tolleranza relativo `tol` (il default é 2e-3, cioè lo 0.2%)

`fplot(fun,limits,tol,LineSpec)` disegna `fun` come specificato in LineSpec e con un errore di tolleranza relativo `tol`

`fplot(fun,limits,n)` con $n >= 1$ disegna `fun` con un minimo di $n+1$ punti. Di default $n$ é 1. Il passo massimo é ristretto in modo da essere $(1/n) * (xmax - xmin)$.

`plot(fun,lims,...)` accetta combinazioni in qualsiasi ordine di `tol`, `n` e `LineSpec`.

`fplot(axes_handle,...)` disegna negli assi specificati in `axes_handle` invece che negli assi correnti.

`[X,Y] = fplot(fun,limits,...)` assegna ad `X` ed `Y` rispettivamente le ascisse e le ordinate della funzione `fun`. Non disegna alcun grafico.

Osservazione

`fplot` usa un passo adattivo concentrando i punti dove la derivata della funzione é più grande.

Esempi

Disegna la tangente iperbolica da $-2$ a 2

```
fnch = @tanh;
fplot(fnch,[-2 2])
```

Crea un M-file, myfun, che ritorni una matrice con due colonne

```
function Y = myfun(x)
Y(:,1) = 200*sin(x)./x;
Y(:,2) = x.^2;
```

crea un puntatore alla funzione myfun

```
fh = @myfun;
```

disegna la funzione

`fplot(fh,[-20 20])`

disegna la funzione `sin(1/x)` in `[.01 .1]` creata tramite funzione anonima

```
sn = @(x) sin(1./x);
fplot(sn,[.01 .1])
```

## ezplot

Plotter di funzione facile da usare

Sintassi

```
ezplot(FUN)
ezplot(FUN2)
ezplot(FUN,[A,B])
ezplot(FUN2,[A,B])
ezplot(FUN2,[XMIN,XMAX,YMIN,YMAX])
ezplot(FUNX,FUNY)
ezplot(FUNX,FUNY,[TMIN,TMAX])
ezplot(FUN,[A,B],FIG)
ezplot(FUN2,[XMIN,XMAX,YMIN,YMAX],FIG)
ezplot(FUNX,FUNY,[TMIN,TMAX],FIG)
ezplot(AX,...)
H = ezplot(...)
```

Descrizione

`ezplot(FUN)` disegna `FUN` sul dominio di default $-2 * PI < X < 2 * PI$, dove FUN(X) é una funzione di X definita esplicitamente.

`ezplot(FUN2)` disegna la funzione definita implicitamente FUN2(X,Y) = 0 sul dominio di default $-2 * PI < X < 2 * PI$ e $-2 * PI < Y < 2 * PI$.

`ezplot(FUN,[A,B])` disegna FUN(X) con $A < X < B$.

`ezplot(FUN2,[A,B])` disegna FUN2(X,Y) = 0 su $A < X < B$ e $A < Y < B$.

`ezplot(FUN2,[XMIN,XMAX,YMIN,YMAX])` disegna FUN2(X,Y) = 0 su $XMIN < X < XMAX$ e $YMIN < Y < YMAX$.

`ezplot(FUNX,FUNY)` traccia la curva planare definita parametricamente FUNX(T) e FUNY(T) sul dominio di default $0 < T < 2 * PI$.

`ezplot(FUNX,FUNY,[TMIN,TMAX])` disegna FUNX(T) e FUNY(T) su TMIN ¡ T ¡ TMAX.

`ezplot(FUN,[A,B],FIG)`, `ezplot(FUN2,[XMIN,XMAX,YMIN,YMAX],FIG)`, oppure `ezplot(FUNX,FUNY,[TMIN,TMAX],FIG)` disegna la funzione sul dominio specificato nella finestra di figura `FIG`.

`ezplot(AX,...)` disegna in `AX` invece che negli assi correnti (`GCA` = Get handle to current axis) o `FIG`.

`H = ezplot(...)` restituisce in `H` handles all'oggetto disegnato.

Esempi

Il piú semplice modo di esprimere una funzione é tramite una stringa:

```
ezplot('x^2 - 2*x + 1')
```

Una tecnica di programmazione é quella di vettorizzare l'espressione stringa usando gli operatori di array $.*\,(TIMES), ./\,(RDIVIDE), .\backslash\,(LDIVIDE), .\hat{}\,(POWER)$. Questo rende l'algoritmo piú efficiente in quanto puó eseguire valutazioni multiple di funzioni in una sola volta.

```
ezplot('x.*y + x.^2 - y.^2 - 1')
```

É inoltre possibile utilizzare una funzione handle ad una funzione esistente. Le funzioni handle sono piú potenti ed efficienti delle espressioni di stringhe.

```
ezplot(@humps)
ezplot(@cos,@sin)
```

`ezplot` disegna le variabili in espressioni stringhe in ordine alfabetico.

```
subplot(1,2,1), ezplot('1./z - log(z) + log(-1+z) + t - 1')
```

Per evitare questa ambiguitá, specificare l'ordine con una funzione anonima:

```
subplot(1,2,2), ezplot(@(z,t)1./z - log(z) + log(-1+z) + t - 1).
```

Se la funzione dispone di parametri aggiuntivi, ad esempio `k` in `myfun`:

```
function z = myfun(x,y,k)
z = x.^k - y.^k - 1;
```

allora é possibile utilizzare una funzione anonima per specificare questo parametro:

```
ezplot(@(x,y)myfun(x,y,2))
```

## plotyy

Grafico con tacche asse y sia a destra che a sinistra

Sintassi

```
plotyy(x1,y1,x2,y2)
plotyy(x1,y1,x2,y2,fun)
plotyy(x1,y1,x2,y2,fun1,fun2)
```

Descrizione

`plotyy(x1,y1,x2,y2)` disegna y1 in funzione di x1 con tacche sulla sinistra e y2 in funzione di x2 con tacche sulla destra

`plotyy(x1,y1,x2,y2,fun)` disegna (x1,y1) e (x2,y2) usando la funzione specificata nella stringa `fun` anzichè `plot`. `fun` può essere : `plot`, `semilogx`, `semilogy`, etc ...

`plotyy(x1,y1,x2,y2,fun1,fun2)` usa `fun1(x1,y1)` per il grafico relativo a `(x1,y1)`, asse y sinistro e `fun2(x2,y2)` per il grafico relativo a `(x2,y2)`, asse y destro.

## loglog

Grafico con scala logaritmica asse x e y

## semilogx

Grafico con scala logaritmica asse x

## semilogy

Grafico con scala logaritmica asse y

## polar

Grafico in coordinate polari

Sintassi

```
polar(theta,rho)
polar(theta,rho,s)
```

Descrizione

`polar(theta,rho)` usa le coordinate polari dell'angolo theta, in radianti, in funzione del raggio rho.

`polar(theta,rho,s)` usa lo stile specificato dalla stringa s.

### 1.1.2 Grafica 2-D specializzata

## area

Grafico con area

Sintassi

```
area(x,y)
area(y)
area(x,y,level) o area(y,level)
```

Descrizione

`area(x,y)` produce un grafico con area colorata, analogo di plot(x,y) ma l'area tra 0 e y viene riempita. x deve essere monotono

`area(y)` usa il valore di default x=1:dim(y).

`area(x,y,level)` o `area(y,level)` stabilisce al livello y=level la base per il grafico dell'area. default : level=0.

Esempi

```
t=linspace(0,2*pi,100) ;
x=cos(t) ;
y=sin(t) ;
area(x,y)
area(y,0.5)
```

## bar

Grafico a barre

Sintassi

```
bar(X,Y)
bar(Y)
bar(x,y,width)
bar(...,'grouped')
bar(...,'stacked')
bar(...,s)
```

Descrizione

`bar(X,Y)` disegna le colonne della matrice Y, MxN come M gruppi di N barre verticali. Il vettore X deve essere monotono crescente (o decrescente). In caso contrario, MATLAB lo ordinerá.

NOTA: Il vettore X non deve avere valori duplicati.

`bar(Y)` usa il valore di default X=1:M. Se X,Y sono vettori `bar(x,y)` o `bar(y)` disegna `dim(y)` barre. I colori sono settati dalla funzione colormap.

`bar(x,y,width)` o `bar(y,width)` dove `width` specifica lo spessore delle barre. Se `width` é maggiore di 1 abbiamo barre sovrapposte. Default `width=0.8`

`bar(...,'grouped')` default: barre verticali raggruppate.

`bar(...,'stacked')` barre verticali sovrapposte.

`bar(...,s)` usa il colore specificato in s ('rgbymckw').

## barh

Grafico orizzontale a barre. Funzionamento analogo a `bar`.

## pie

Grafico a torta

Sintassi

```
pie(X)
pie(X,EXPLODE)
pie(...,LABELS)
pie(AX,...)
H = pie(...)
```

Descrizione

`pie(X)` disegna un grafico a torta dei dati contenuti nel vettore X. I valori in X sono normalizzati come $X/SUM(X)$ per determinare l'area di ogni fetta di torta.
Se $SUM(X) <= 1.0$, i valori in X specificano direttamente l'area delle fette di torta. Solo una parziale torta sará rappresentata se $SUM(X) < 1$.

`pie(X,EXPLODE)` é usato per specificare fette di torta che devono essere messi in evidenza, tirati fuori dalla torta. Il vettore `EXPLODE` deve avere stessa dimensione di X. Le fette in corrispondenza di valori diversi da zero in `EXPLORE` saranno messi in evidenza come distaccati dal resto della torta.

`pie(...,LABELS)` é usato per etichettare ogni fetta di torta con l'array di celle `LABELS`. `LABELS` deve avere stessa dimensione di X e puó contenere solo stringhe.

`pie(AX,...)` grafica in `AX` invece che nei correnti assi (`GCA`).

`H = pie(...)` restituisce un vettore contenente patch and text handles.

Esempio

```
pie([2 4 3 5],{'North','South','East','West'})
```

## hist

Istogramma

Sintassi

```
N = hist(Y)
N = hist(Y,M)
N = hist(Y,X)
[N,X] = hist(...)
hist(...)
hist(AX,...)
```

Descrizione

`N = hist(Y)` rappresenta gli elementi di Y in 10 contenitori equispaziati e restituisce in output il numero di elementi di ogni contenitore. Se Y é una matrice, `hist` lavora per colonne.

`N = hist(Y,M)`, dove M é uno scalare, usa M contenitori (bins).

`N = hist(Y,X)`, dove X é un vettore, restituisce la distribuzione di Y tra contenitori centrati nei valori specificati da X. Il primo contenitore include dati tra `-inf` e il primo centro, l'ultimo bins include dati tra l'ultimo contenitore e `inf`.

`[N,X] = hist(...)` restituisce anche la posizione dei centri dei contenitori in X.

`hist(...)` senza argomenti di output produce un istogramma a barre verticali dei risultati. I bordi del primo e dell'ultimo contenitore si possono estendere per coprire il minimo e il massimo dei dati, a meno che non venga fornita una matrice di dati.

`hist(AX,...)` grafica in `AX` invece che in `GCA`

## stairs

Grafico a scale

Sintassi

```
stairs(Y)
stairs(X,Y)
stairs(...,STYLE)
stairs(AX,...)
H = stairs(X,Y)
[XX,YY] = stairs(X,Y)
```

Descrizione

`stairs(Y)` disegna un grafico a scale degli elementi del vettore Y

`stairs(X,Y)` disegna un grafico a scale degli elementi del vettore Y nelle posizioni specificate da X.

`stairs(...,STYLE)` usa lo stile di linea per il grafico specificato dalla stringa `STYLE`.

`stairs(AX,...)` grafica in `AX` invece che negli assi correnti (`GCA`).

`H = stairs(X,Y)` restituisce un vettore di handles ai grafici a scalini

`[XX,YY] = stairs(X,Y)` non disegna alcun grafico, ma restituisce i vettori XX e YY tali che `plot(XX,YY)` é il grafico a scale.

## stem

Sequenza discreta o grafico a stelo.

Sintassi

```
stem(Y)
stem(X,Y)
stem(...,'filled')
stem(...,'LINESPEC')
stem(AX,...)
H = stem(...)
```

Descrizione

`stem(Y)` disegna la sequenza di dati Y come steli che partono dall'asse x e terminano con cerchi in corrispondenza del valore dato. Se Y é una matrice, allora ogni colonna di Y é disegnata come una serie separata.

`stem(X,Y)` grafica la sequenza di dati Y nei valori specificati da X.

`stem(...,'filled')` produce un grafico a stelo con i marcatori riempiti.

`stem(...,'LINESPEC')` usa il tipo di linea specificato per gli steli e i marcatori. Si faccia riferimento a `plot` per le varie possibilitá.

`stem(AX,...)` disegna negli assi con handle `AX`. Utilizzare `GCA` per ottenere l'handle agli assi correnti o per crearne uno se non esiste.

`H = stem(...)` restituisce un vettore di handle di serie di stem in H, un handle per colonna di dati in Y.

## compass

Grafico a bussola.

Sintassi

```
compass(U,V)
compass(Z)
compass(U,V,LINESPEC)
compass(AX,...)
H = compass(...)
```

Descrizione

`compass(U,V)` disegna un grafico che mostra i vettori di componenti (U,V) come frecce sprigionate dall'origine.

`compass(Z)` é l'equivalente di `compass(REAL(Z),IMAG(Z))`.

`compass(U,V,LINESPEC)` e `compass(Z,LINESPEC)` usa la linea specificata in `LINESPEC` (si veda `plot` per le possibilitá)

`compass(AX,...)` grafica in `AX` invece che negli assi correnti (`GCA`).

`H = compass(...)` restituisce in H gli handles agli oggetti linea.

### rose

Grafico istogramma ad angolo.

Sintassi

```
rose(THETA)
rose(THETA,N)
rose(THETA,X)
rose(AX,...)
H = rose(...)
[T,R] = rose(...)
```

Descrizione

`rose(THETA)` grafica l'istogramma ad angolo per gli angoli definiti in THETA. Gli angoli nel vettore THETA devono essere espressi in radianti.

`rose(THETA,N)` dove N é uno scalare, usa N bins (contenitori, recipienti) equispaziati da 0 a $2 * PI$. Il valore di default per N é 20.

`rose(THETA,X)` dove X é un vettore, disegna l'istogramma usando i bins specificati in X.

`rose(AX,...)` grafica in AX invece che negli assi correnti (GCA).

`H = rose(...)` restituisce un vettore di handles di linea.

`[T,R] = rose(...)` restituisce i vettori T e R tali che `POLAR(T,R)` é l'istogramma. Nessun grafico é mostrato in tal caso.

## 1.2 Alcune funzioni di gestione assi e piano

### box

Box degli assi.

Sintassi

```
box on
box off
```

Descrizione

`box on` aggiunge un box agli assi (default)

`box off` lo toglie

### grid

Linea della griglia

Sintassi

```
grid on
grid off
grid minor
grid
grid(axes_handle,...)
```

Descrizione

`grid on` disegna le linee di griglia negli assi correnti

`grif off` elimina le linee di griglia negli assi correnti

`grid minor` disegna delle linee di griglia più fini negli assi correnti o le elimina se già presenti

`grid` equivale a `grid on` se non sono presenti delle linee di griglia e a `grid off` in caso contrario

`grid(axes_handle,...)` specifica in quali assi eseguire la funzione grid, ad esempio, `grid(h,'minor')`

### colordef

Definizione dei default del colore

Sintassi

```
colordef white
colordef black
colordef none
```

Descrizione

`colordef white` definisce il colore dello sfondo degli assi bianco, le linee degli assi e le labels nero ed il colore di sfondo della figure grigio.

`colordef black` definisce il colore dello sfondo degli assi nero, le linee degli assi e le labels bianco ed il colore di sfondo della figure grigio scuro.

`colordef none` definisce stesso colore per lo sfondo degli assi e quello della figure. Il colore definito puó variare a seconda delle versioni tra nero (Matlab2012) e grigio (Matlab2016).

## 1.2.1 Annotazioni sul grafico

### xlabel

Label sull'asse x (analogo `ylabel`, `zlabel`)

Sintassi

```
xlabel('stringa')
```

Descrizione

`xlabel('stringa')` etichetta l'asse x con la stringa

## title

Titolo del grafico

Sintassi

```
title('stringa')
```

Descrizione

`title('stringa')` aggiunge il titolo stringa in alto nella finestra corrente

## text

Annotazioni di testo

Sintassi

```
text(x,y,s)
text(x,y,z,s)
```

Descrizione

`text(x,y,s)` aggiunge il testo stringa s nella posizione (x,y) del piano corrente. Se x ed y sono vettori (stessa dim), `text` scrive in tutte i punti dati. Se s un vettore stringa tale che dim(s)=dim(x)=dim(y), `text` scrive in ogni punto la corrispondente riga di s.

`text(x,y,z,s)` testo in coordinate 3D.

## gtext

Posizionamento del testo con mouse

Sintassi

```
gtext(s)
```

Descrizione

`gtext(s)` mostra il grafico e aspetta il posizionamento del testo s tramite bottone del mouse o tastiera.

## legend

Legenda del grafico

Sintassi

```
legend(string1,string2,...)
legend('off')
legend(...,'Location',location)
```

Descrizione

`legend(string1,string2,...)` posiziona una legenda nel grafico usando le stringhe come label.

`legend('off')` elimina la legenda dagli assi correnti

`legend(...,'Location',location)` usa `location` per determinare la posizione della legenda. `location` può essere un vettore di quattro componenti (`sinistra basso larghezza altezza`) o una stringa (e.g., N, NE, NEO, etc.).

Spostare la legenda

Per spostare la legenda si può premere il bottone sinistro del mouse sulla legenda e trascinarla nella posizione voluta. Con il doppio click su una label la si può editare.

Esempio

```
title('Esempio di 3 funzioni')
fplot('[sin(x) cos(x) tan(x)]',[0 4*pi -3 3]);
legend('First','Second','Third');
legend('First','Second','Third','NEO');
text(2,1,'Esempio di 3 funzioni')
```

### hold

Tiene attivo il grafico corrente

Sintassi

```
hold on
hold off
hold
hold(axes_handle,...)
```

Descrizione

`hold on` tiene attive le proprietà del grafico corrente di modo che i successivi comandi siano aggiunti ai preesistenti

`hold off` ripristina il default

`hold` equivale a `hold on` se hold é off e viceversa

`hold(axes_handle,...)` specifica gli assi a cui applicare hold

## 1.2.2 Creazione e controllo assi

Quando viene creato un grafico, MATLAB automaticamente seleziona il limite degli assi ed il passo. Si possono specificare dei propri valori con i seguenti comandi

## axes

Crea gli assi

Sintassi

```
axes
axes('PropertyName',PropertyValue,...)
axes(h)
h = axes(...)
```

Descrizione

`axes` crea gli assi

`axes('PropertyName',PropertyValue,...)` crea gli assi con la proprietà 'PropertyName' settata al valore PropertyValue

`axes(h)` definisce gli esistenti assi h come assi correnti

`h = axes(...)` assegna ad h un puntatore agli assi creati

## axis

Gestione della scala e aspetto degli assi

Sintassi

```
axis([xmin xmax ymin ymax])
v = axis

axis auto
axis manual

axis equal
axis square
axis normal

axis off
axis on
axis(axes_handles,...)
```

Descrizione

`axis([xmin xmax ymin ymax])` definisce i limiti degli assi del grafico corrente

`v = axis` assegna a `v` un vettore riga contenente i limiti degli assi

`axis auto` i limiti degli assi sono calcolati automaticamente sulla base dei valori massimi e minimi disegnati

`axis manual` and `axis(axis)` congela i limiti degli assi, così che se hold é on i grafici seguenti non modificano i limiti degli assi

`axis equal` setta gli assi di modo che l'unità di misura sia la stessa in ogni direzione

`axis square` rende il box degli assi quadrato

`axis normal` ridimensiona il box degli assi e la scala di modo che il grafico si adatti in maniera ottimale alle dimensioni della figura

`axis off` elimina gli assi

`axis on` ripristina gli assi

`axis(axes_handles,...)` applica il comando `axis` agli assi specificati da axes_handles, ad esempio `axis([h1 h2],'square')`

## cla

Cancella gli assi correnti

## gca

Ottiene l'handle agli assi correnti

## subplot

Crea e controlla più assi in formato matrice

Sintassi

```
subplot(m,n,p)
```

Descrizione

`subplot(m,n,p)` divide la finestra in mxn parti (matrice) in cui posiziona degli assi e attiva i p-esimi assi selezionati. Gli assi vengono contati per righe

## xlim

Limiti asse x (analogo asse y e asse z)

Sintassi

```
xlim
xlim([xmin xmax])
```

Descrizione

`xlim` ritorna il limite degli assi correnti

`xlim([xmin xmax])` assegna ai limiti degli assi correnti i valori xmin xmax

## 1.3 Grafici in tre dimensioni

### 1.3.1 Grafici elementari in tre dimensioni

### plot3

Grafico di curve in 3-D

Sintassi

```
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h = plot3(...)
```

Descrizione

`plot3(X1,Y1,Z1,...)` con `X1`, `Y1` e `Z1` vettori o matrici, disegna una o più linee in tre dimensioni le cui cordinate sono gli elementi di `X1`, `X2` e `X3`

`plot3(X1,Y1,Z1,LineSpec,...)` crea e visualizza tutte le linee definita da `Xn`, `Xn` e `Xn` con il tipo di linea, simbolo e colore specificato da `LineSpec`

`plot3(...,'PropertyName',PropertyValue,...)` configura la proprietà `PropertyName` al valore `PropertyValue` per tutti gli oggetti creati da `plot3`

Esempio L'elica

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t,'c*-');
```

### mesh, meshc

Grafico di griglie rettangolari

Sintassi

```
mesh(X,Y,Z)
mesh(Z)
mesh(X,Y,Z,C)
mesh(...,'PropertyName',PropertyValue,...)
mesh(axes_handles,...)
meshc(...)
h = mesh(...)
```

Descrizione

`mesh(X,Y,Z)` disegna un griglia con colore determinato da `Z` e quindi proporzionale all'altezza. Se `X` e `Y` sono vettori, allora la lunghezza di `X` deve corrispondere alle colonne di `Z` e quella di `Y` alle righe di `Z`. In questo caso i nodi della mesh sono definiti da `(X(j),Y(i),Z(i,j))`. Se `X`, `Y` e `Z` sono matrici, le loro dimensioni devono coincidere e i nodi della mesh sono definiti da `(X(i,j),Y(i,j),Z(i,j))`.

`mesh(Z)` usa `X=1:n` e `Y=1:m` con `[m,n]=size(Z)`

`mesh(X,Y,Z,C)` disegna una griglia con colore determinato dalla matrice `C` delle stesse dimensioni di `Z` attraverso una trasformazione lineare sui valori di `C` dalla corrente `colormap`.

`mesh(...,'PropertyName',PropertyValue,...)` configura la proprietà `PropertyName` al valore `PropertyValue`.

`mesh(axes_handles,...)` disegna negli assi puntati da `axes_handles`

`meshc(...)` disegna delle linee di livello sotto il grafico.

`h=mesh(...)` ritorna un puntatore all'oggetto grafico creato.

## surf, surfc

Grafico di superfici (su griglie rettangolari)

Sintassi

```
surf(X,Y,Z)
surf(Z)
surf(X,Y,Z,C)
surf(...,'PropertyName',PropertyValue,...)
surf(axes_handles,...)
surfc(...)
h = surf(...)
```

Descrizione

`surf(X,Y,Z)` disegna una superficie con colore determinato da `Z` e quindi proporzionale all'altezza. Se `X` e `Y` sono vettori, allora la lunghezza di `X` deve corrispondere alle colonne di `Z` e quella di `Y` alle righe di `Z`. In questo caso i nodi della mesh sono definiti da `(X(j),Y(i),Z(i,j))`. Se `X`, `Y` e `Z` sono matrici, le loro dimensioni devono coincidere e i nodi della mesh sono definiti da `(X(i,j),Y(i,j),Z(i,j))`.

`surf(Z)` usa `X=1:n` e `Y=1:m` con `[m,n]=size(Z)`

`surf(X,Y,Z,C)` disegna una superficie con colore determinato dalla matrice `C` delle stesse dimensioni di `Z` attraverso una trasformazione lineare sui valori di `C` dalla corrente `colormap`.

`surf(...,'PropertyName',PropertyValue,...)` configura la proprietà `PropertyName` al valore `PropertyValue`.

`surf(axes_handles,...)` disegna negli assi puntati da `axes_handles`

`surfc(...)` disegna delle linee di livello sotto il grafico.

`h=surf(...)` ritorna un puntatore all'oggetto grafico creato.

### surfl

Grafico di superfici illuminate

Sintassi

```
surfl(X,Y,Z)
surfl(Z)
surfl(...,'light')
surfl(...,s)
surfl(X,Y,Z,s,k)
h = surfl(...)
```

Descrizione

`surfl(X,Y,Z)` e `surfl(Z)` come `surf` con una direzione della sorgente luminosa e delle caratteristiche di riflessione dell'oggetto di default

`surfl(...,s)` specifica la direzione della luce. `s = [sx sy sz]` oppure `s = [azimuth elevation]`. Il default è $45^o$ in senso antiorario rispetto alla corrente vista.

## 1.3.2 Controllo del colore

### colormap

mappa dei colori

Sintassi

```
colormap(map)
colormap('default')
cmap = colormap
```

Descrizione

Una matrice mappa di colore (tavolozza) é una matrice $n \times 3$. Ogni riga ha 3 valori reali in [0,1] che definiscono un colore nella mappa RGB

`colormap(map)` assegna alla figura corrente la mappa definita in map

`colormap('default')` assegna alla figura corrente la mappa definita di default ( jet)

`map = colormap` assegna a `map` la `colormap` corrente

mappe dei colori predefinite

- `autumn` varies smoothly from red, through orange, to yellow.
- `bone` is a grayscale colormap with a higher value for the blue component. This colormap is useful for adding an electronic look to grayscale images.
- `colorcube` contains as many regularly spaced colors in RGB colorspace as possible, while attempting to provide more steps of gray, pure red, pure green, and pure blue.

- `cool` consists of colors that are shades of cyan and magenta. It varies smoothly from cyan to magenta.
- `copper` varies smoothly from black to bright copper.
- `flag` consists of the colors red, white, blue, and black. This colormap completely changes color with each index increment.
- `gray` returns a linear grayscale colormap.
- `hot` varies smoothly from black through shades of red, orange, and yellow, to white.
- `hsv` varies the hue component of the hue-saturation-value color model. The colors begin with red, pass through yellow, green, cyan, blue, magenta, and return to red. The colormap is particularly appropriate for displaying periodic functions.
- `jet` ranges from blue to red, and passes through the colors cyan, yellow, and orange. It is a variation of the hsv colormap.
- `lines` lines produces a colormap of colors specified by the axes ColorOrder property and a shade of gray.
- `pink` contains pastel shades of pink. The pink colormap provides sepia tone colorization of grayscale photographs.
- `prism` repeats the six colors red, orange, yellow, green, blue, and violet.
- `spring` consists of colors that are shades of magenta and yellow.
- `summer` consists of colors that are shades of green and yellow.
- `white` is an all white monochrome colormap.
- `winter` consists of colors that are shades of blue and green.

## colorbar

Visualizza la barra dei colori (scala colori)

Sintassi

```
colorbar
colorbar(h)
colorbar(...,'location')
```

Descrizione

`colorbar` visualizza o aggiorna la barra di colori nella figura corrente.

`colorbar(h)` visualizza o aggiorna la barra di colori nella figura puntata da `h`

`colorbar(...,'location')` la stringa `location` specifica come posizionare la barra e i possibili valori sono: 'N','S','E','W','NO','SO','EO','WO'.

### shading

Modalità colore ombreggiatura (effetto visibile con `surf`, non con `mesh`)

Sintassi

```
shading flat
shading faceted
shading interp
```

Descrizione

`shading` controlla l'ombreggiatura del colore della superficie graficata

`shading flat` ombreggiatura piatta

`shading faceted` come `shading flat` ma con le linee della griglia in nero (default)

`shading interp` ombreggiatura interpolata

### hidden

Rimuove le linee di griglia nascoste dai grafici di griglie (visibile con `mesh`, non con `surf`)

Sintassi

```
hidden on
hidden off
hidden
```

Descrizione

`hidden on` attiva la modalità secondo la quale non vengono disegnate le le linee di griglia nascoste (default)

`hidden off` disegna anche le linee di griglia nascoste

`hidden` corrisponde `hidden off` se `hidden` é `on` e viceversa

## 1.3.3 Controllo del punto di visualizzazione

### view

specifica il punto di visualizzazione in 3D

Sintassi

```
view(az,el)
view([az,el])
view([x,y,z])
view(2)
view(3)
[az,el]=view
```

Descrizione

`view(az,el)` e `view([az,el])` definiscono l'angolo del punto di visualizzazione dell'osservatore. `az` controlla l'azimut o la rotazione orizzontale, `el` controlla l'elevazione verticale (espresse in gradi)

`view([x,y,z])` definisce l'angolo in coordinate cartesiane

`view(2)` corrisponde a `view(0,90)`

`view(3)` corrisponde a `view(-37.5,30)`

`[az,el]=view` assegna ad `az` ed `el` i valori correnti

### 1.3.4 Contour

## contour

Grafico di curve di livello

Sintassi

```
contour(Z)
contour(Z,n)
contour(Z,v)
contour(X,Y,Z)
contour(X,Y,Z,n)
contour(X,Y,Z,v)
contour(...,LineSpec)
[C,h]=contour(...)
```

Descrizione

`contour(Z)` grafico di curve di livello della matrice `Z` ai livelli `v` scelti automaticamente.

`contour(Z,n)` grafico di curve di livello della matrice `Z` ad `n` livelli `v` scelti automaticamente.

`contour(Z,v)` disegnano `dim(v)` curve di livello alle altezze specificate in `v`. Per disegnare un unica curva di livello `i` si usa `contour(Z,[i i])`.

`contour(X,Y,Z)`, `contour(X,Y,Z,n)` e `contour(X,Y,Z,v)` grafico di curve di livello della matrice `Z` con `X` e `Y` che specificano le coordinate della superficie.

`contour(...,LineSpec)` per disegnare le curve di livello con colore e tipo di linea specificato da `LineSpec`.

`[C,h]=contour(...)` assegna a `C` la matrice dei dati delle curve di livello e a `h` un vettore di puntatori alle curve. Entrambi sono usati da `clabel` per etichettare le curve di livello con il loro valore.

Esempio

```
[C,h] = contour(peaks(50));
clabel(C,h)
```

### contourf

Grafico di curve di livello come `contour` ma l'area tra diverse linee di livello viene riempita con colori costanti a seconda della `colormap` corrente.

Esempio

```
z=peaks;
contourf(z), hold on, shading flat
[c,h]=contour(z,'k-');
clabel(c,h), colorbar
```

### contour3

Grafico di curve di livello come `contour` ma le linee di livello sono disegnate in tre dimensioni alla loro corrispondente altezza.

Esempio

```
contour3(peaks(25))
```

### clabel

Etichettatura delle curve di livello.

Sintassi

```
clabel(C,h)
clabel(C,h,v)
clabel(C,h,'manual')
```

Descrizione

`clabel(C,h)` inserisce delle etichette con l'altezza delle linee di livelo in mezzo alle curve

`clabel(C,h,v)` etichetta solo le linee indicate nel vettore `v`

`clabel(C,h,'manual')` posiziona le etichette con il mouse

## 1.3.5 Grafica 3-D specializzata

### bar3

Grafico a barre 3-D (analogo a `bar` in 2-D)

Sintassi

```
bar3(Y)
bar3(x,Y)
```

Descrizione

`bar3(Y)` disegna un grafico a barre 3-D, una barra per ogni elemento di `Y`

`bar3(x,Y)` come `bar3(Y)` con `x` monotono che specifica le coordinate dell'asse `y`

## bar3h

Grafico a barre orizzontali 3-D (analogo a `barh` in 2-D)

## pie3

Grafico a torta 3-D (analogo a `pie` in 2-D)

## stem3

Grafico con steli 3-D (analogo a `stem` in 2-D)

## hist3

Istogramma tridimensionale di dati bivariati.

Esempio

```
load carbig
X = [MPG,Weight];
hist3(X,[7 7]);
xlabel('MPG'); ylabel('Weight');

hist3(X,[7 7],'FaceAlpha',.65);
xlabel('MPG'); ylabel('Weight');
set(gcf,'renderer','opengl');

hist3(X,[7 7]);
xlabel('MPG'); ylabel('Weight');
set(gcf,'renderer','opengl');
set(get(gca,'child'),'FaceColor','interp','CDataMode','auto');

cnt = hist3(X, {0:10:50 2000:500:5000});
```

## triplot

Traccia una triangolazione 2D

Sintassi

```
triplot(TRI,X,Y)
triplot(DT)
triplot(...,COLOR)
H = triplot(...)
triplot(...,'param','value','param','value'...)
```

Descrizione

`triplot(TRI,X,Y)` mostra i triangoli definiti nella matrice $M \times 3$ `TRI`. Una riga di `TRI` contiene gli indici in X, Y che definisce un singolo triangolo. Il colore di linea di default é il blu.

`triplot(DT)` mostra i triangoli prodotti dalla triangolazione di Delaunay `DT`, dove `DT` é un `DelaunayTri`.

`triplot(...,COLOR)` usa la stringa `COLOR` come colore di linea.

`H = triplot(...)` restituisce un vettore di handles ai triangoli mostrati.

`triplot(...,'param','value','param','value'...)` permette di aggiungere coppie parametri/valori da essere usati quando si crea il plot.

Esempi

```
Esempio 1:
     X = rand(10,2);
     dt = DelaunayTri(X);
     triplot(dt)

Esempio 2:
     X = rand(10,2);
     dt = DelaunayTri(X);
     tri = dt(:,:);
     triplot(tri, X(:,1), X(:,2));
```

## trimesh

Grafico di griglie triangolari

Sintassi

```
trimesh(Tri,X,Y,Z)
trimesh(Tri,X,Y,Z,C)
trimesh(...,'PropertyName',PropertyValue}
h = trimesh(...)
```

Descrizione

`trimesh(Tri,X,Y,Z)` visualizza la superficie con facce triangolari definite dalla matrice $m \times 3$ `tri`. Una riga di `tri` definisce un triangolo e contiene gli indici dei vettori o delle matrici `X`, `Y` e `Z`.

`trimesh(Tri,X,Y,Z,C)` specifica il colore attraverso il vettore o la matrice `C`.

`trimesh(...,'PropertyName',PropertyValue)` configura la proprietà `'PropertyName'` al valore `PropertyValue`

`h = trimesh(...)` ritorna un puntatore all'oggetto grafico creato

## trisurf

Grafico di superfici definite su griglie triangolari, come `trimesh`

Esempi

```
[x,y]=meshgrid(1:15,1:15);
tri = delaunay(x,y);
z = peaks(15);
trisurf(tri,x,y,z)

c=linspace(1,256,size(z,1)*size(z,2));
map=bone(256);  colormap(map);
trimesh(tri,x,y,z,c)
```

### waterfall

Grafico a cascata, come `mesh` ma le linee di colona non sono disegnate.

## 1.4 Creazione e controllo della finestra figura

### figure

Crea la finestra della figura

Sintassi

```
figure
figure(h)
h = figure(...)
```

Descrizione

`figure` crea una nuova finestra e il suo indirizzo

`figure(h)` rende attiva la finestra di indirizzo `h` se esiste. Se la figura `h` non esiste, ne crea una nuova di indirizzo `h`

`h = figure(...)` assegna l'indirizzo della figura creata ad `h`

### gcf

Ritorna l'indirizzo della finestra corrente

Sintassi

```
h = gcf
```

Descrizione

`h=gcf` ritorna in `h` l'indirizzo della figura corrente

## clf

Pulisce la finestra corrente

Sintassi

```
clf
clf(fig)
```

Descrizione

`clf` Pulisce la finestra corrente

`clf(fig)` Pulisce la finestra di indirizzo `fig`

## refresh

ridisegna la figura corrente

Sintassi

```
refresh
refresh(fig)
```

Descrizione

`refresh` Pulisce e ridisegna la finestra corrente

`refresh(fig)` Pulisce e ridisegna la finestra di indirizzo `fig`

## drawnow

Esegue le istruzioni grafiche in coda e aggiorna la figura.

Sintassi

```
drawnow
```

## close

chiude la figura corrente

Sintassi

```
close
close(fig)
close all
```

Descrizione

`close` Chiude la figura corrente

`close(fig)` Chiude la figura di indirizzo `fig`

`close all` Chiude tutte le figure

### 1.4.1 Utilizzo operazioni di grafica

#### set

configura le proprietà degli oggetti grafici

Sintassi

```
set(h,'PropertyName',PropertyValue)
a= set(h)
a = set(h,'PropertyName')
```

Descrizione

`set(h,'PropertyName',PropertyValue)` configura la proprietà `PropertyName` al valore `PropertyValue` per l'oggetto puntato da `h`

`a= set(h)` restituisce i possibili valori di tutte le proprietà dell'oggetto di indirizzo `h`

`a = set(h,'PropertyName')` ritorna i possibili valori della proprietà `PropertyName` dell'oggetto di indirizzo `h`

Esempio

assegnare il colore rosso come default per il testo della figura corrente

```
set(gcf,'DefaultTextColor','red')
```

#### get

Ottiene le proprietà degli oggetti

Sintassi

```
get(h)
a = get(h,'PropertyName')
```

Descrizione

`get(h)` restituisce le proprietà e i loro valori per l'oggetto `h`

`a = get(h,'PropertyName')` ritorna in `a` il valore della proprietà `PropertyName` dell'oggetto `h`

#### reset

Resetta le proprietà degli oggetti

Sintassi

```
reset(h)
```

Descrizione

`reset(h)` resetta tutte le proprietà dell'oggetto `h` riportandole al default.

### delete

Cancella file o oggetti grafici

Sintassi

```
delete filename
delete(h)
delete('filename')
```

Descrizione

`delete filename` cancella `filename` dal disco

`delete(h)` cancella l'oggetto puntato da `h`

`delete('filename')` cancella `filename` dal disco

## 1.5 Importazione e esportazione di figure e dati

### hgload

Carica oggetti grafici da file

Sintassi

```
h=hgload('filename')
```

Descrizione

`h=hgload('filename')` Carica una figura in formato MATLAB (`.fig`) con tutti i suoi oggetti

### hgsave

Salva oggetti grafici

Sintassi

```
hgsave('filename')
hgsave(h,'filename')
```

Descrizione

`hgsave('filename')` salva la figura corrente in formato MATLAB (`.fig`).

`hgsave(h,'filename')` salva la figura puntata da `h`

### print

Stampa o salva una figura in formato grafico

Sintassi

```
print
print filename
print -ddriver
print -dformat filename
print ... -options
```

Descrizione

`print` stampa la figura corrente utilizzando le impostazioni di default

`print filename` salva la figura in formato PostScript

`print -ddriver` stampa la figura usando il driver `driver`

`print -dformat filename` salva la figura nel formato specificato da `format` (-dbmp, -deps, -depsc, -deps2, -depsc2, -djpeg, -dpdf, -dpng, -dppm, -dtiff etc.)

`print ... -options` stampa o salva usando le opzioni `options` tra cui

- `-append` salva la figura di seguito ad una già esistente
- `-fhandle` stampa o salva la figura specificata da `handle`
- `-noui` non stamba i controlli dell'interfaccia utente
- `-Pprinter` usa la stampante `printer`
- `-r number` usa la risoluzione `numbers` punti per pollici

Osservazioni

Alcune proprietà possono essere configurate tramite il comando `set`, in particolare `set(gcf,'PaperPositionMode','auto')` permette di salvare o stampare la figura nel formato in cui appare sullo schermo. Se `PaperPositionMode` ha il valore di default `'manual'` allora la figura é stampata o salvata nel formato determinato dalla proprietà `PaperPosition` il cui valore `[left, bottom, width, height]` é di default uguale a `[0.2500 2.5000 8.0000 6.0000]` pollici.

### saveas

Salva una figura nel formato output desiderato

Sintassi

```
saveas(h,'filename')
saveas(h,'filename','format')
```

Descrizione

`saveas(h,'filename')` salverá la figure puntata dall'handle `h` nel file denominato `filename`. Il formato del file é determinato dall'estensione di `filename`.

`saveas(h,'filename','format')` salverá la figure puntata dall'handle `h` nel file denominato `filename` nel formato specificato da `format`. `format` puó coincidere con l'estensione di `filename`. Nel caso essi non coincidano, il formato specificato da `format` sovrascrive l'estensione del `filename`.

Esempi

Scrivere l'attuale figure in un file nel formato fig di MATLAB

```
saveas(gcf, 'output', 'fig')
```

Scrivere l'attuale figure in un file bitmap

```
saveas(gcf, 'output', 'bmp')
```

## load

Carica le variabili sul workspace (dal disco)

Sintassi

```
load
load filename
load filename X Y Z ...
load -ascii filename
load -mat filename
s = load(...)
```

Descrizione

`load` carica tutte le variabili dal file `matlab.mat`

`load filename` (o, equivalentemente, `load('filename')`) carica tutte le variabili dal file `filename`. Se `filename` non ha estensioni `load` cerca il file `filename.mat` e lo tratta come un file-MAT binario. Se `filename` ha un'estensione diversa da `.mat`, `load` lo tratta come un ASCII file e carica i dati in una variabile con il nome del file senza estensione

`load filename X Y Z ...` carica dal MAT-file solo le variabili specificate

`load -ascii filename` e `load -mat filename` forza `load` a considerare il file di tipo ASCII o MAT.

`s=load(...)` memorizza in s il contenuto del file.

## save

Salva le variabili del workspace nel disco

Sintassi

```
save
save('filename')
save('filename','X','Y','Z',...)
save('filename','format')
save filename X Y Z ...
```

Descrizione

`save` salva tutte le variabili nella directory corrente nel file `matlab.mat`

`save('filename')` salva tutte le variabili nella directory corrente nel file `filename.dat`. Per salvare in una directory specifica si può assegnare il percorso alla variabile `filename`.

`save('filename','X','Y','Z',...)` salva in `filename.mat` solo le variabili specificate.

`save('filename','format')` salva in filename.dat con il formato specificato da format, `-ascii` (8-digit ASCII format), `-ascii -double` (16-digit ASCII format), `-mat` (Binary MAT-file format, default)

`save filename X Y Z ...` come `save('filename','X','Y','Z',...)`.

## ginput

Input grafico dal mouse.

Sintassi

```
[X,Y] = ginput(N)
[X,Y] = ginput
```

Descrizione

`[X,Y] = ginput(N)` acquisisce N punti dai correnti assi e restituisce le coordinate xy dei punti salvandole nei vettori X e Y che hanno lunghezza N. Il cursore si posiziona usando il mouse. I punti dati vengono acquisiti premendo un pulsante del mouse nella figure o qualsiasi tasto sulla tastiera, tranne il tasto `return`, che termina l'ingresso prima che siano acquisiti gli N punti.

`[X,Y] = ginput` raccoglie un numero illimitato di punti finché non viene premuto il tasto `return`.

Esempi

```
[x,y] = ginput;

[x,y] = ginput(5);
```

## fscanf

Legge dati formattati da file

Sintassi

```
A = fscanf(fid, format)
[A,count] = fscanf(fid, format, size)
```

Descrizione

`A = fscanf(fid, format)` importa in `A` tutti i dati dal file specificato da `fid` secondo il formato specificato in `format`. `fid` é un intero ottenuto da `fopen`.

`[A,count] = fscanf(fid, format, size)` assegna ad `A` la quantità di dati specificata in `size` secondo il formato `format`, ed assegna a `count` il numero degli elementi letti. `size` può essere:

`n` - vengono letti `n` elementi e assegnati ad un vettore colonna

`inf` vengono letti tutti gli elementi e assegnati ad un vettore colonnna

[m,n] - vengono letti `m` per `n` elementi e assegnati ad una matrice `m` per `n` in ordine colonna (`n` può essere `inf`)

La stringa `format` é composta da uno o più caratteri iniziali, `%`, un intero (opzionale) che specifica la lunghezza del campo da leggere, e da un formato di conversione ( `e,f,g` per numeri floating-point, `i` per gli interi, `c` per i caratteri, `s` stringhe senza spazi, etc.. (vedi le referenze della routine `fscanf` in un manuale del linguaggio C per maggiori informazioni).

Esempio

```
fid=fopen('prova')
A=fscanf(fid,'%*s %i',2)
B=fscanf(fid,'%i')
fclose(fid)
```

Apre il file prova:

```
intestazione 1
intestazione 2
3
4
5
```

tralascia la sequenza di caratteri e assegna ad `A` i due interi 1 e 2, poi assegna a `B` i rimanenti interi fino alla fine del file.

## fprintf

Scrive dati formattati su file

Sintassi

```
count = fprintf(fid, format, A,...)
```

Descrizione

`count = fprintf(fid, format, A,...)` scrive la parte reale di `A` nel file specificato da `fid` secondo il formato specificato in `format`. `fid` é un intero ottenuto da `fopen`.

Esempio

```
fid=fopen('prova','w')
fprintf(fid,'intestazione %i\n',1)
fprintf(fid,'intestazione %i\n',2)
fprintf(fid,'%i\n',3)
```

```
fprintf(fid,'%i\n',4)
fprintf(fid,'%i\n',5)
fclose(fid)
```

scrive il file prova dell'esempio precedente.

### 1.5.1 Lettura e scrittura di files audio

#### wavread

Legge microsoft WAVE (.wav) file.

Sintassi

```
y=wavread('filename')
[y,Fs,bits]=wavread('filename')
[y,Fs,bits,opts]=wavread(...)
```

Descrizione

`y=wavread('filename')` legge il file WAVE `filename` e lo assegna ad `y`

`[y,Fs,bits]=wavread('filename')` ritorna oltre ai dati in `y`, la frequenza di campionamento `Fs` in Hertz e il numero di bits `bits`.

`[y,Fs,bits,opts]=wavread(...)` restituisce, oltre ai dati in `y`, alla frequenza `Fs` e al numero di bits `bits`, una struttura `opts` di informazioni aggiuntive contenute nel WAVE file. Il contenuto di questa struttura varia da file a file. Campi tipici includono '.fmt' (informazioni sul formato audio) and '.info' (testo che puó descrivere, per esempio, il titolo, l'autore, ecc.)

Esempio

```
[y,Fs,bits]=wavread('ding.wav')
sound(y,Fs,bits)
```

#### audioread

Dalla versione MATLAB 2016, il comando `wavread` non é piú disponibile. É a disposizione dell'utente il piú generale `audioread`, che legge file audio.

Sintassi

```
[y,Fs]=audioread('filename') % per leggere l'array di dati y e la frequenza Fs
INFO=audioinfo('filename')   % per avere altre info
```

#### wavwrite

Scrive microsoft WAVE (.wav) file.

Sintassi

```
wavwrite(y,'filename')
wavwrite(y,Fs,bits,'filename')
```

Descrizione

`wavwrite(y,'filename')` scrive `y` nel file WAVE `filename`.

`wavwrite(y,Fs,bits,'filename')` scrive `y` nel file WAVE `filename` e specifica la frequenza di campionamento in Hertz `Fs` e il numero di bits `bits`.

## audiowrite

Dalla versione MATLAB 2016, il comando `wavwrite` non é piú disponibile. É a disposizione dell'utente il piú generale `audiowrite`, che scrive generici file audio.

Sintassi

```
audiowrite('filename', y, Fs)
```

## audioplayer

Audio player object.

Sintassi

```
audioplayer(Y, Fs)
audioplayer(Y, Fs, nbits)
```

Descrizione

`audioplayer(Y, Fs)` crea un oggetto audioplayer per il segnale `Y`, usando la frequenza di campionamento `Fs`. Un handle all'oggetto viene restituito in output.

`audioplayer(Y, Fs, nbits)` crea un oggetto audioplayer e usa `nbits` bits per campione per il segnale a virgola mobile `Y`. Valori validi per `nbits` sono 8, 16 e 24. Di default, il numero di bits é 16.

## play

Riproduce campioni audio nell'oggetto del riproduttore audio (audioplayer).

Sintassi

```
play(OBJ)
play(OBJ, START)
play(OBJ, [START STOP])
```

Descrizione

`play(OBJ)` riproduce campioni audio dall'inizio.

`play(OBJ, START)` riproduce campioni audio dal campione `START`.

`play(OBJ, [START STOP])` riproduce campioni audio dal campione `START` al campione `STOP`.

Esempio

```
load handel;
p = audioplayer(y, Fs);
play(p);
```

### 1.5.2 Lettura e scrittura di immagini

## imread

Legge un'immagine da un file grafico

Sintassi

```
A = imread('filename','fmt')
[X,map] = imread('filename','fmt')
[...] = imread(...,idx)
```

Descrizione

`A = imread('filename','fmt')` legge l'immagine dalla stringa `filename` di formato `fmt` (stringa) e la memorizza in `A` di classe uint8 (unsigned 8-bit integers). `A` é un array 2-dim se `A` é un immagine in toni di grigio, mentre é un array 3-dim se l'immagine é truecolor (o RGB).

`[X,map] = imread('filename','fmt')` legge immagini di indici da `filename` di formato `fmt` e le memorizza in `X` e nell'associata mappa di colori `map` (classe double).

`[...] = imread(...,idx)` legge un'immagine da un'immagine che ne contiene più di una. idx é un intero che specifica quale immagine acquisire dal file. Omettendo idx, `imread` legge la prima immagine.

Esempi

```
[X,map] = imread('forest.tif');
image(X)
colormap(map)

load clown
imagesc(X)
colormap(map)
```

## imwrite

Scrive un'immagine in un file grafico

Sintassi

```
imwrite(A,'filename','fmt')
imwrite(X,map,'filename','fmt')
```

Descrizione

`imwrite(A,'filename','fmt')` scrive l'immagine `A` nel file `filename` nel formato `fmt`.

`imwrite(X,map,'filename','fmt')` scrive l'immagine `X` e la mappa di colori associata `map` nel file `filename` nel formato `fmt`.

### 1.5.3 Visualizzazione e manipolazione di immagini

## image

Visualizza un'immagine

Sintassi

```
image(C)
image(x,y,C)
```

Descrizione

`image(C)` visualizza la matrice C come immagine. C può essere una matrice `MxN` o `MxNx3`.

`image(x,y,C)` come `image(C)` dove `x` ed `y` definiscono i limiti degli assi `x` e `y`.

## imagesc

Riscala i dati e visualizza un'immagine

Sintassi

```
imagesc(C)
imagesc(x,y,C)
```

Descrizione

`imagesc(C)` riscala i dati al range della colormap corrente e visualizza la matrice C come immagine. C può essere una matrice `MxN` o `MxNx3`.

`imagesc(x,y,C)` come `imagesc(C)` dove `x` ed `y` definiscono i limiti degli assi `x` e `y`.

## imshow

Display image in Handle Graphics figure.

Sintassi

```
imshow(I)
imshow(I,[LOW HIGH])
imshow(RGB)
imshow(BW)
imshow(X,MAP)
imshow(FILENAME)
HIMAGE = imshow(...)
```

Descrizione

`imshow(I)` visualizza l'immagine in toni di grigio `I`.

`imshow(I,[LOW HIGH])` visualizza l'immagine in toni di grigio `I`, specificando l'intervallo di visualizzazione per I in `[LOW HIGH]`. Il valore `LOW` (e ogni valore minore di `LOW`) visualizza come nero, il valore `HIGH` (e ogni valore maggiore di `HIGH`) si mostra come bianco. Valori tra di essi sono visualizzati come intermedie sfumature di grigio, usando il numero di livelli di grigio di default. Se si utilizza una matrice vuota ([ ]) per `[LOW HIGH]`, `imshow` usa l'intervallo `[min(I(:))  max(I(:))]`; cioé, il minimo valore di I é visualizzato come nero, il massimo come bianco.

`imshow(RGB)` visualizza l'immagine truecolor `RGB`.

`imshow(BW)` mostra l'immagine binaria (immagine in bianco e nero) `BW`. `imshow` visualizza i pixels con valore 0 come nero, e i pixels con valore 1 come bianco.

`imshow(X,MAP)` mostra l'immagine indicizzata `X` con la corrispondente mappa dei colori `MAP`.

`imshow(filename)` visualizza l'immagine memorizzata nel file grafico `filename`. Il file deve contenere un'immagine che puó essere legga da `imread` o da `dicomread`. `imshow` chiama `imread` o `dicomread` per leggere l'immagine dal file, ma non memorizza il dato immagine nel workspace di MATLAB. Se il file contiene piú immagini, solo la prima verrá visualizzata. Il file deve essere nella directory corrente o nel path di matlab.

`HIMAGE = imshow(...)` restituisce l'handle all'oggetto immagine creato da `imshow`.

## rgb2gray

Converte l'immagine RGB o la colormap in scala di grigi

Sintassi

```
I = rgb2gray(RGB)
NEWMAP = rgb2gray(MAP)
```

Descrizione

`I = rgb2gray(RGB)` converte l'immagine truecolor `RGB` nell'immagine di intensit in scala di grigi `I`.

`NEWMAP = rgb2gray(MAP)` restituisce una mappa di colori in scala di grigi `NEWMAP` equivalente a `MAP`.

Esempio

```
Esempio 1:
   Img = imread('board.tif');
   J = rgb2gray(Img);
   figure(1), imshow(Img),
   figure(2), imshow(J);

Esempio 2: rgb2gray applicato anche a colormap
```

`restituisce una colormap in toni di grigio (gray) equivalente a map`

```
[X,map] = imread('trees.tif');
gmap = rgb2gray(map);
figure(3), imshow(X,map);
figure(4), imshow(X,gmap);
```

## gray2ind

gray2ind converte un'immagine in toni di grigio (o bianco e nero) in immagine indicizzata. gray2ind scala, e poi approssima, un'immagine in toni di grigio per produrre un'equivalente immagine indicizzata.

Sintassi

```
[X,MAP] = gray2ind(I,N)
[X,MAP] = gray2ind(BW,N)
```

Descrizione

`[X,MAP] = gray2ind(I,N)` converte l'immagine di intensitá `I` in una immagine indicizzata `X` con colormap gray(N). Se N é omesso, di default viene usato N= 64.

`[X,MAP] = gray2ind(BW,N)` converte l'immagine binaria in bianco e nero `BW` (0=nero, 1=bianco) in una matrice indicizzata `X` con colormap gray(N). Se `N` é omesso, di default viene usato N= 2.
N deve essere un intero tra 1 e 65536.
`I` puó essere logical, uint8, uint16, int16, single, o double e puó avere qualsiasi dimensione.
La matrice di output `X` sará di classe uint8 SE la lunghezza della colormap é <= 256 (unsigned intero a 8 bit, ossia assume valori interi tra 0 e $2^8 - 1$, cioé tra 0 e 255) altrimenti `X` é uint16 (assume valori tra 0 e $2^{16} - 1$).

Esempio

```
I = imread('cameraman.tif');
figure, imshow(I);
[X, map] = gray2ind(I, 16);   % gray(16) come colormap
                              % per ottenere X=I occorre usare gray(256)
figure, imshow(X, map);
```

## ind2gray

ind2gray converte un'immagine indicizzata in un'immagine in toni di grigio. Funzionamento analogo a `gray2ind`.

Esempio

```
[X,map] = imread('trees.tif');
I = ind2gray(X,map);
figure, imshow(X,map), figure, imshow(I);
```

## imfinfo

Fornisce informazioni su un file grafico.

Sintassi

```
INFO = imfinfo(filename,fmt)
INFO = imfinfo(filename)
INFO = imfinfo(URL,...)
```

Descrizione

`INFO = imfinfo(filename,fmt)` restituisce una struttura i cui campi contengono informazioni su un'immagine in un file grafico. `filename` é una stringa che specifica il nome del file grafico, e `fmt` é una stringa che specifica il formato del file. Il file deve essere nella directory corrente o in una directory nel path di MATLAB. Se `imfinfo` non riesce a trovare un file chiamato `filename`, cerca un file chiamato `filename.fmt`. I possibili valori per `fmt` possono essere visualizzati tramite il comando `imformats`.

`INFO = imfinfo(filename)` tenta di dedurre il formato del file dal suo contenuto.

`INFO = imfinfo(URL,...)` legge l'immagine da una `URL`. La `URL` deve includere il tipo di protocollo (per esempio, http://).

L'insieme dei campi in `INFO` dipende dal singolo file e dal suo formato. Comunque, i primi nove campi sono sempre gli stessi e sono i seguenti:

- `Filename` Una stringa contenente il nome del file o la URL internet data
- `FileModDate` Una stringa contenente la data di modifica del file o la data in cui stata scaricata l'immagine da un URL
- `FileSize` Un intero che indica la dimensione del file in bytes
- `Format` Una stringa contenente il formato del file, come specificato da `fmt`; per formati con piú di una possibile estensione ( per esempio, files JPEG o TIFF), la prima variante nel registro é restituita
- `FormatVersion` Una stringa o un numero che specifica la versione del formato del file
- `Width` Un numero intero che indica la larghezza dell'immagine in pixels
- `Height` Un numero intero che indica l'altezza dell'immagine in pixels
- `BitDepth` Un numero intero che indica il numero di bits per pixel
- `ColorType` Una stringa che indica il tipo di immagine. Questo potrebbe essere, per esempio, `'truecolor'` per un'immagine a colori RGB, `'grayscale'`, per un'immagine in scala di grigi o `'indexed'` per un'immagine indicizzata.

Esempio

```
info = imfinfo('ngc6543a.jpg');
```

## imnoise

Aggiunge rumore ad un'immagine.

Sintassi

```
J = imnoise(I,type,...)
```

Descrizione

`J = imnoise(I,type,...)` aggiunge rumore di un dato tipo specificato da `type` all'immagine di intensitá `I`. `type` é una stringa che puó assumere uno dei seguenti valori:

- `'gaussian'` Rumore bianco gaussiano con costante media e varianza
- `'localvar'` Rumore bianco gaussiano con media zero e una varianza dipendente dall'intensitá
- `'poisson'` Rumore di Poisson
- `'salt & pepper'` "On and Off" pixels
- `'speckle'` Rumore moltiplicativo

A seconda del tipo `type` scelto, si possono specificare ulteriori parametri. NOTA: Tutti i parametri numerici sono normalizzati; essi corrispondono ad operazioni con immagini che hanno intensitá che varia da 0 ad 1.

- `J = imnoise(I,'gaussian',M,V)` aggiunge rumore bianco gaussiano di media `M` e varianza `V` all'immagine `I`. Se non specificato, i valori di default per `M` e `V` sono rispettivamente 0 e 0.01.
- `J = imnoise(I,'localvar',V)` aggiunge rumore gaussiano con media zero e varianza `V` all'immagine `I`. `V` é un vettore della stessa dimensione di `I`.
- `J = imnoise(I,'localvar',IMAGEINTENSITY,VAR)` aggiunge all'immagine `I` rumore gaussiano con media zero e varianza locale del rumore che é una funzione dei valori di intensitá dell'immagine `I`. `IMAGEINTENSITY` e `VAR` sono vettori della stessa dimensione, e `plot(IMAGEINTENSITY,VAR)` grafica la relazione funzionale tra varianza del rumore e intensitá dell'immagine. NOTA BENE: `IMAGEINTENSITY` deve contenere valori di intensitá normalizzati che assumono valori tra 0 e 1.
- `J = imnoise(I,'poisson')` genera rumore di Poisson dai dati, invece di aggiungere rumore artificiale ai dati. Se `I` é double, allora i valori dei pixel di input vengono interpretati come medie di distribuzioni di Poisson scalate da `1e12`. Ad esempio, se un pixel di input ha valore pari a `5.5e-12`, allora il corrispondente pixel di output sará generato da una distribuzione di Poisson con una media di 5.5 e poi scalato indietro da `1e12`. Se `I` é un single, il fattore di scala utilizzato é `1e6`. Se `I` é uint8 o uint16, vengono utilizzati i valori dei pixel di input direttamente senza riscalamento.
- `J = imnoise(I,'salt & pepper',D)` aggiunge un rumore "sale e pepe" all'immagine `I`, dove `D` é la densitá del rumore. Ció influenza circa `D * numel(I)` pixels. Il valore predefinito per `D` é `0.05`.

- `J = imnoise(I,'speckle',V)` aggiunge rumore moltiplicativo all'immagine `I`, usando l'equazione $J = I + n * I$, dove $n$ é un rumore random distribuito uniformemente con media 0 e varianza V. Il valore predefinito per `V` é `0.04`.

Osservazioni

I parametri di media e varianza per rumori di tipo 'gaussian', 'localvar', e 'speckle' sono sempre specificati come se trattassimo con un'immagine double nell'intervallo $[0, 1]$. Se l'immagine di input é di classe uint8 o uint16, la funzione `imnoise` converte l'immagine in double, aggiunge rumore in accordo al tipo e ai parametri specificati, e poi riconverte l'immagine ottenuta nella stessa classe di quella di input. Sarebbe dunque preferibile convertire l'immagine in toni di grigio di input in double con valori compresi tra 0 e 1 prima di utilizzare il comando `imnoise`.

Esempi

```
Esempio 1:
  I = imread('eight.tif');
  J = imnoise(I,'salt & pepper', 0.02);
  figure, imshow(I), figure, imshow(J)

Esempio 2:
   Img1 = imread('lena.jpg');
   % trasformo in double e normalizzo per ottenere valori in [0,1]
   I=double(Img1);
   I=I./255;
   figure(1)
   imshow(I);
   title('immagine originaria');
   Isporca=imnoise(I,'salt & pepper');
   figure(2)
   imshow(Isporca);
   title('Immagine sporcata');
```

## imfilter

Filtro a densitá neutra [1] (ND filter) di immagini multidimensionali.

Sintassi

```
B = imfilter(A,H)
B = imfilter(A,H,OPTION1,OPTION2,...)
```

Descrizione

[1] Un filtro a densitá neutra o filtro ND (dall'inglese Neutral Density), in fotografia e, piú in generale, in ottica, é un filtro di colore grigio che riduce l'intensitá della luce in maniera uniforme su tutte le lunghezze d'onda. Si veda `https://it.wikipedia.org/wiki/Filtro_a_densit%C3%A0_neutra` per maggiori informazioni a riguardo.

`B = imfilter(A,H)` filtra l'array a piú dimensioni `A` con il filtro multidimensionale `H` `A` puó essere di tipo logical o un array numerico non sparso di qualsiasi dimensione e classe. Il risultato, `B`, ha stessa dimensione e classe di `A`.

`B = imfilter(A,H,OPTION1,OPTION2,...)` esegue un filtraggio multidimensionale in base alle opzioni specificate. Gli argomenti di opzione possono avere i seguenti valori:

- Opzioni al bordo
  - `X` I valori dell'array di input al di fuori dei limiti dell'array assumono implicitamente il valore X. Quando non viene specificata un'opzione al bordo, `imfilter` utilizza X = 0.
  - `'symmetric'` I valori dell'array di input al di fuori dei limiti dell'array vengono calcolati con una riflessione a specchio dell'array attraverso il bordo dell'array.
  - `'replicate'` I valori di input al di fuori dei limiti dell'array sono eguagliati al valore al bordo pi vicino.
  - `'circular'` I valori di input al di fuori dei limiti dell'array sono calcolati assumendo implicitamente che l'array di input é periodico.
- Output size options (Output size options for imfilter are analogous to the SHAPE option in the functions CONV2 and FILTER2.)
  - `'same'` The output array is the same size as the input array. This is the default behavior when no output size options are specified.
  - `'full'` The output array is the full filtered result, and so is larger than the input array.
- Correlation and convolution
  - `'corr'` imfilter performs multidimensional filtering using correlation, which is the same way that FILTER2 performs filtering. When no correlation or convolution option is specified, imfilter uses correlation.
  - `'conv'` imfilter performs multidimensional filtering using convolution.

Esempio

```
originalRGB = imread('peppers.png');
h = fspecial('motion',50,45);  % 50 pixels della camera, 45 gradi in senso antiorario
filteredRGB = imfilter(originalRGB,h);
figure, imshow(originalRGB), figure, imshow(filteredRGB)
boundaryReplicateRGB = imfilter(originalRGB,h,'replicate');
% si utilizza l'opzione replicate per eliminare il problema del bordo scuro.
% I valori fuori dai bounds dell'array assumono valore
% uguale al piu' vicino valore al bordo.
figure, imshow(boundaryReplicateRGB)
```

### medfilt2

Filtro mediano 2-D. Questo tipo di filtro si utilizza per sopprimere un particolare tipo di rumore, il rumore 'sale e pepe' . Tale filtro é molto efficace per togliere effetti come i 'bad pixel' o raggi cosmici nelle immagini astronomiche.

Esempio

```
Img1 = imread('lena.jpg');
% trasformo in double e normalizzo per ottenere valori in [0,1]
I=double(Img1);
I=I./255;
figure(1)
imshow(I);
title('immagine originaria');
Isporca=imnoise(I,'salt & pepper');
figure(2)
imshow(Isporca);
title('Immagine sporcata');
Ifiltrata=medfilt2(Isporca);
figure(3)
imshow(Ifiltrata);
title('immagine filtrata')
```

## 1.6 Filmati e animazioni

### getframe

Acquisisce un frame di un filmato

Sintassi

```
F= getframe
F= getframe(h)
F=getframe(h,rect)
```

Descrizione

`F= getframe` acquisisce in `F` un frame dagli assi correnti

`F= getframe(h)` acquisisce in `F` un frame dell'oggetto puntato da `h`

`F=getframe(h,rect)` specifica il rettangolo da acquisire. `rect` é un vettore di quattro elementi, `[left bottom width height]` e l'unità di misura é il pixel.

### movie

Visualizza i frames di un filmato

Sintassi

```
movie(M)
movie(M,n)
movie(M,n,fps)
movie(h,...)
```

Descrizione

`movie(M)` visualizza il filmato definito dalla matrice `M` le cui colonne sono i singoli frame.

`movie(M,n)` visualizza il filmato `n` volte. Se `n<0` il filmato viene mandato `n` volte avanti e indietro. Se `n` un vettore `n(1)` indica il numero di volte, i restanti elementi indicano i frames da utilizzare nel filmato.

`movie(M,n,fps)` visualizza il filmato con `fps` frames per secondo (default `fps=12`).

`movie(h,...)` visualizza il filmato negli assi `h`.

Esempio

```
for j=1:M
  plot_command
  M(j)=getframe;
end
movie(M)
```

### movie2avi

Converte un filmato MATLAB in AVI

Sintassi

```
movie2avi(mov,filename)
movie2avi(mov,filename,param,value,param,value...)
```

Descrizione

`movie2avi(mov,filename)` crea il filmato `filename` in formato AVI dal filmato MATLAB

`movie2avi(mov,filename,param,value,param,value...)` crea il filmato `filename` in formato AVI dal filmato MATLAB con i parametri `param` del valore `value`.

## 1.7 Creazione di Interfacce Grafiche

### uimenu

Crea un menu interfaccia utente

Sintassi

```
uimenu('PropertyName1',value1,'PropertyName2',value2,...)
uimenu(H,...)
```

Descrizione

`uimenu('PropertyName1',value1,'PropertyName2',value2,...)` crea un menu interfaccia utente che comparirá nella barra dei menú nella parte superiore della finestra figura corrente, e restituisce un handle per esso.

`uimenu(H,...)` crea un nuovo menú che ha `H` come "genitore". `H` deve essere un figure handle, menu handle, oppure un context menu handle.
Le proprietá del menú possono essere impostate all'atto della creazione dell'oggetto usando la coppia di argomenti PropertyName/PropertyValue in `uimenu`, oppure modificati in seguito tramite il comando `set`.
Eseguire `get(H)` per vedere la lista delle proprietá per l'oggetto `uimenu` e i loro valori attuali.

Esempio

Creo un menú chiamato MIOMENU, le cui scelte consentono agli utenti di creare una nuova finestra figure, salvare le variabili del workspace, uscire da matlab.
In aggiunta, definisco una accelerator key Q per l'opzione Quit.

```
surf(peaks(50))
f = uimenu('Label','MIOMENU');
uimenu(f,'Label','New Figure','Callback','figure');
uimenu(f,'Label','Save','Callback','save');
uimenu(f,'Label','Quit','Callback','exit',...
        'Separator','on','Accelerator','Q');

% Separator on e' per creare una linea di separazione nel menu'
% Accelerator e' per creare un percorso piu' rapido equivalente ad un
% comando. DEVE essere una stringa di un solo carattere
```

## uicontextmenu

Crea un menu interfaccia utente a scelta rapida

Sintassi

```
uicontextmenu
uicontextmenu('PropertyName1',value1,'PropertyName2',value2,...)
```

Descrizione

`uicontextmenu` crea un menú di scelta rapida e restituisce un handle ad esso. Per collegare un menú di scelta rapida ad un oggetto, impostare la proprietá uicontextmenu dell'oggetto sull'handle restituito dalla funzione `uicontextmenu`. Le voci di menú possono essere aggiunte al menú contestuale utilizzando la funzione `uimenu`.
Le proprietá del menú a scelta rapida possono essere impostate all'atto della creazione dell'oggetto usando la coppia di argomenti PropertyName/PropertyValue in `uicontextmenu`, oppure modificati in seguito tramite il comando `set`.
Esare il comando `get(H)` per vedere la lista delle proprietá per l'oggetto `uicontextmenu` e i loro valori attuali.

Esempio

```
% Crea assi e salva handle
hax = axes;
% Plot tre linee random
plot(rand(20,3));
% Definisco un context menu che non  assegnato a nulla al momento
hcmenu = uicontextmenu;
% Vogliamo per esempio definire callbacks per le voci del contextmenu
% che cambiano stile delle linee

% gco = Get handle to current object.
hcb1 = ['set(gco, ''LineStyle'', ''--'')'];
hcb2 = ['set(gco, ''LineStyle'', '':'')'];
hcb3 = ['set(gco, ''LineStyle'', ''-'')'];
% Definire le voci del contextmenu e i callbacks
item1 = uimenu(hcmenu, 'Label', 'dashed', 'Callback', hcb1);
item2 = uimenu(hcmenu, 'Label', 'dotted', 'Callback', hcb2);
item3 = uimenu(hcmenu, 'Label', 'solid',  'Callback', hcb3);
% Localizzare oggetti linea
hlines = findall(hax,'Type','line');
% Collegare il context menu per ogni linea
for line = 1:length(hlines)
    set(hlines(line),'uicontextmenu',hcmenu)
end

% Premere il tasto destro del mouse su una linea per far apparire il menu'
% (su un MAC premere Ctrl+fare click col mouse sulla linea desiderata).
```

## uicontrol

Crea oggetti grafici per interfacce

Sintassi

```
handle = uicontrol('PropertyName',PropertyValue,...)
handle = uicontrol(parent,'PropertyName',PropertyValue,...)
handle = uicontrol
uicontrol(uich)
```

Descrizione

`handle = uicontrol('PropertyName',PropertyValue,...)` crea un oggetto grafico il cui indirizzo é assegnato ad `handle`, con la proprietà `'PropertyName'` di valore `PropertyValue`. Alle proprietà non configurate viene assegnato un valore di default.

`handle = uicontrol(parent,'PropertyName',PropertyValue,...)` crea l'oggetto grafico nell'oggetto specificato dal puntatore `parent` (figura, uipanel o uibuttongroup).

`handle = uicontrol` crea un bottone, che é il defaul della proprietà `'style'`, con tutte le proprietà di default.

`uicontrol(uich)` rende attivo il controllo puntato da `uihc`

uicontrol style

Il tipo di controlo é definito assegnando alla proprietà `'Style'` uno dei seguenti valori:

- 'checkbox' caselle selezionabili
- 'edit' campi dove é possibile immettere del testo
- 'frame' rettangoli usati come contenitori di altri oggetti
- 'listbox' visualizza una lista definita dalla proprietà `'String'` e permette di selezionare uno o più elementi della lista.
- 'popup' crea un menu i cui campi sono definiti tramite la proprietà `'String'`
- 'pushbutton' crea un bottone
- 'radiobutton' simile a `'checkbox'`, ma se all'interno di un gruppo la selezione di uno deseleziona l'altro.
- 'slider' crea una barra scorrevole che permette di specificare dei valori di input in un determinato range
- 'text' caselle di testo, tipicamente usate per etichettare altri controlli
- 'togglebutton' simile a `pushbutton` ma il loro stato rimane evidenziato

Esempio

```
function guif
global n
n=50;
s=get(0,'ScreenSize');
f=figure('position',[s(3)/4 s(4)/4 s(3)/2 s(4)/2]);
ax=axes('units','pixel','position',[s(3)/8 s(4)/8 s(3)/4 s(4)/4]);
surf(peaks(n));
htxt=uicontrol('Style', 'text',...
       'String', 'N=',...
       'Position', [5*s(3)/32 s(4)/32 s(3)/16 s(4)/32]);

hed=uicontrol('Style', 'edit',...
       'String', num2str(n),...
       'Position', [7*s(3)/32 s(4)/32 s(3)/16 s(4)/32],...
       'Callback', {@setn});

hpb = uicontrol('Style', 'pushbutton',...
```

```
       'String', 'Draw',...
       'Position', [s(3)/32 13*s(4)/32 s(3)/16 s(4)/32],...
       'Callback', {@draw});

hpu=uicontrol('Style', 'popup',...
       'String', 'hsv|hot|cool|gray',...
       'Position', [13*s(3)/32 13*s(4)/32 s(3)/16 s(4)/32],...
       'Callback', {@setmap});

function draw(obj,eventdata)
global n
if isnan(n)
    errordlg('You must enter a numeric value','Bad Input','modal')
else
surf(peaks(n));
end

function setn(obj,eventdata)
global n
user_entry = get(obj,'string');
n=str2double(user_entry);
if isnan(n)
    errordlg('You must enter a numeric value','Bad Input','modal')
end

function setmap(obj,eventdata)
val = get(obj,'Value');
if val == 1
    colormap(hsv)
elseif val == 2
    colormap(hot)
elseif val == 3
    colormap(cool)
elseif val == 4
colormap(gray)
end
```

## uiwait, uiresume

Controlla l'esecuzione del programma

Sintassi

```
uiwait(h)
uiwait(h,timeout)
uiresume(h)
```

Descrizione

`uiwait(h)` blocca l'esecuzione del programma finchè non viene richiamato `uiresume` o la figura `h` é cancellata.

`uiwait(h,timeout)` come `uiwait(h)`, ma il programma viene sbloccato dopo `timeout` secondi.

`uiresume(h)` sblocca l'esecuzione del programma bloccata da `uiwait(h)`

# Bibliografia

[1] http://www.mathworks.com

# Indice analitico