Se si vogliono i veri indici degli elementi non nulli di una matrice **a** , la sintassi è la seguente:

```
» [id,jd]=find(a)
```

La matrice colonna **id** fornisce gli indici di riga e la matrice **jd** gli indici di colonna degli elementi cercati. Per esempio

```
» a=[0 5 2; 6 1 7; 0 1 -1];
» [id,jd]=find(a>5)
id =
     2
     2
jd =
     1
     3
```

Gli elementi maggiori di 5 della matrice **a** sono quelli di indici $(2,1)$ e $(2,3)$.

• Il massimo e il minimo elemento di un vettore **x** si ottengono con le funzioni

```
» max(x)
» min(x)
```

Se si vogliono conoscere anche le posizioni del massimo e del minimo, la sintassi è la seguente

```
» [m,id]=max(x)
```

e si ottengono **m** , il valore del massimo e **id** l'indice del massimo (il primo indice se più di uno degli elementi di **x** è massimo). Analogamente per il minimo.

• Per riordinare gli elementi di un vettore dal minimo al massimo si usa la funzione **sort** . Per esempio

```
» x=[0 8 3 7 5];
» sort(x)
ans =
     0     3     5     7     8
```

Se si vogliono conoscere anche le posizioni degli elementi in ordine di grandezza, la sintassi è

```
» [x1,id]=sort(x)
x1 =
     0     3     5     7     8
id =
     1     3     5     4     2
```

La matrice **id** fornisce le posizioni in cui si trovavano in **x** gli elementi della matrice riordinata.
Questo è utile per esempio, se si ha anche una matrice **y** i cui elementi sono in corrispondenza con quelli di **x** .
È possibile riordinare **y** mantenendo la corrispondenza con **x** , mediante la matrice **id** .

```
» x=[0 8 3 7 5]; y=[0 18 33 27 15];
» [x1,id]=sort(x);
» y(id)
ans =
     0    33    15    27    18
```

• Le funzioni **any** e **all** : la prima ritorna 1 se almeno un elemento della matrice è non nullo e 0 in caso contrario, la seconda ritorna 1 se tutti gli elementi della matrice sono non nulli e 0 in caso contrario. Le due funzioni sono solitamente usate in congiunzione con gli operatori relazionali.
Per comprenderne l'uso conviene osservare attentamente il seguente esempio:

```
» x=[1 2 3] ; y=[1 5 7] ; z=[10 11 12];
» any(x==y)
ans =
     1
```

```
» all(x==y)
ans =
     0
```

```
» any(x==z)
ans =
     0
```

```
» any(x~=y)
ans =
     1
```

```
» all(x~=y)
ans =
     0
```

```
» all(x~=z)
ans =
     1
```

```
» all(x<y)
ans =
     0
```

```
» all(x<z)
ans =
     1
```

INTRODUZIONE ALLA GRAFICA IN 2D

MatLab ha grosse capacità grafiche. Descriviamo il comando elementare **plot(x,y)** che è alla base di comandi più avanzati. Il comando **plot** ha in pratica la funzione di disegnare una spezzata nel piano. Per esempio per disegnare la spezzata che congiunge i punti di coordinate $(1,1)$ $(2,2)$ $(-1,1)$ $(3,0)$ $(2,1)$ occorre formare due matrici riga **x** e **y** , la prima con le ascisse dei cinque punti, la seconda con le ordinate.

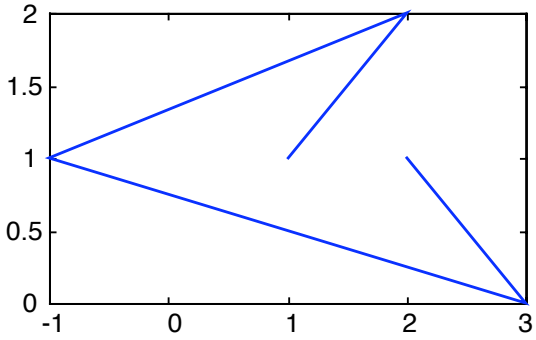
```
x=[1 2 -1 3 2];
y=[1 2 1 0 1];
```

Dopodiché il comando

```
» plot(x,y)
```

disegna la spezzata che appare nella finestra grafica. Se la finestra grafica non appare, la si può richiamare col comando **shg** (showgraph)

```
» shg
```



Il grafico appare non monometrico, senza assi e solo circondato da una cornice graduata avente come limiti i minimi e i massimi delle ascisse e delle ordinate dei punti. Nell'esempio le scale del grafico hanno passo 0.5, ma dipendono solo dalla grandezza della finestra grafica.
Altri comandi grafici consentono di impostare opzioni differenti.
Il comando

```
» axis=equal
```

consente per esempio di avere un grafico con assi monometrici.

Per far sì che la finestra grafica, anziché essere compresa tra i minimi e i massimi, sia delimitata da un qualsiasi rettangolo $[x_0, x_1] \times [y_0, y_1]$ occorre il comando

```
» axis([x0 x1 y0 y1])
```

GRAFICO UNA CURVA NEL PIANO

Cominciamo col disegnare il grafico di una funzione **y = f(x)** in un intervallo **[a,b]** .
Osserviamo che ciò equivale a disegnare la spezzata che ha come vertici vari punti del grafico.
Di solito conviene suddividere l'intervallo in parti uguali scegliendo un passo più o meno ampio a seconda del dettaglio che si vuole ottenere.
Per esempio, se l'intervallo è $[-3, 2]$ e il passo scelto è 0.1 allora si può porre

```
» x=-3:0.1:2;
```

e si ottiene una matrice riga con 51 elementi.
Se, a titolo di esempio, vogliamo disegnare il grafico della funzione **y = log (x^3-3x+2)** nell'intervallo suddetto, occorre tabulare la funzione, cioè calcolarla in tutti i punti scelti dell'intervallo in questione.
Costruiamo quindi una matrice 1×51 **y** con tutti i valori corrispondenti ai valori di **x** .

```
» y=log(x.^3-3*x+2);
```

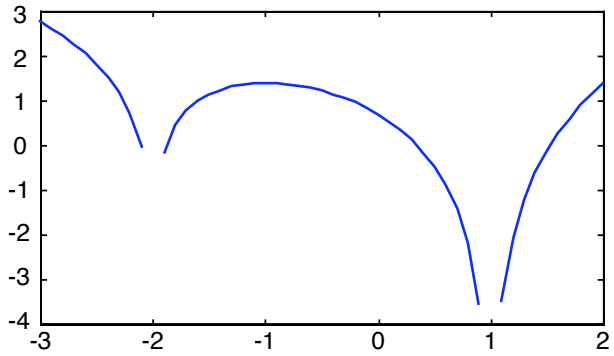
Notiamo il segno **.^** (potenza puntuale) per la potenza. Invece per moltiplicare **x** per lo scalare **3** non occorre il segno **.*** e anche la somma per uno scalare si ottiene semplicemente col segno **+** . Avendo posto il segno **;** la matrice **y** non è riportata sul display, comunque compare la scritta

```
Warning:  Log of zero.
```

che significa: Attenzione: log di zero. Questo perché tra i punti in cui è calcolata la funzione ci sono -2 e 1 dove la funzione non è definita.
In realtà la funzione non sarebbe definita anche tra -3 e -2 dato che si tratta del logaritmo di un numero negativo, ma in questo caso l'effetto è solo quello di ottenere valori non reali.
Ora possiamo procedere a disegnare il grafico con

```
» plot(x,y)
```

e si ottiene nella finestra grafica il disegno seguente:

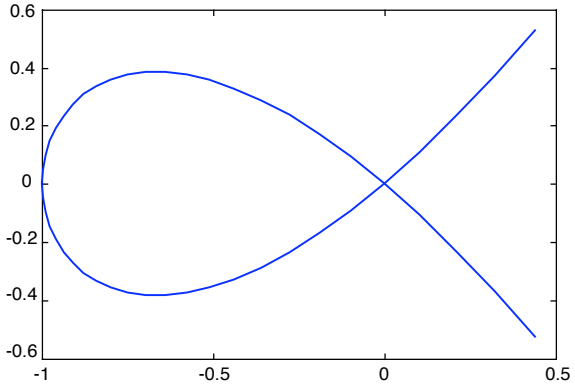


Ovviamente la funzione non viene disegnata tra -2.9 e -1.9 e tra 0.9 e 1.1, dato che la matrice **y** non è definita in -2 e in 1. Per quanto riguarda il pezzo di grafico tra -3 e -2 dove la funzione non è definita, esso è in realtà il grafico della funzione $y = \mathrm{Re}\,(\log(x^3 - 3x + 2))$ (parte reale). Se si lavora con funzioni di variabile reale il pezzo va trascurato.

È altrettanto semplice disegnare una curva data mediante la sua rappresentazione parametrica, creando l'array **x** delle ascisse e quello **y** delle ordinate.
Per esempio per raffigurare la curva avente la rappresentazione parametrica a lato, va innanzitutto scelta la porzione più significativa, quella che si ottiene per $t \in [-1.2, 1.2]$. Il grafico viene generato dai seguenti comandi

$$\begin{cases} x = t^2 - 1 \\ y = t(t^2 - 1) \end{cases}$$

```
» t=-1.2:.05:1.2;
» x=t.^2-1
» y=t.*(t.^2-1)
» plot(x,y)
```



Osserviamo che, mentre è semplice disegnare una curva assegnata mediante rappresentazione parametrica, assai più complesso è il disegno di una curva piana nota attraverso la sua rappresentazione cartesiana, se una delle due coordinate non è facilmente esplicitabile. Qualche indicazione in proposito verrà data nel paragrafo sulla grafica tridimensionale.

GRAFICA IN 2D: OPZIONI E GRAFICI MULTIPLI

Il comando **plot(x,y)** ha molte opzioni. Esiste la capacità di cambiare l'aspetto del grafico (che, ricordiamo, è sempre una spezzata). Mediante il comando

```
» help plot
```

si ottiene l'elenco delle opzioni principali.

Le opzioni vanno assegnate mediante una stringa di caratteri racchiusa da due apici. Per esempio

```
» plot(x,y,'r:')
```

impone che il grafico sia rosso (`'r'`) e punteggiato (`':'`). Le due opzioni vanno combinate in un'unica stringa.

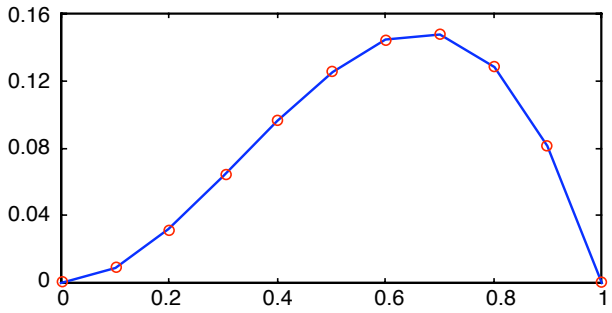
Esiste la possibilità di sovrapporre due grafici distinti con un solo comando. Per esempio

```
» plot(x,y,'r:',x1,y1,'b--')
```

disegnerà due spezzate, la prima rossa e punteggiata relativa alle array **x** e **y** e la seconda blu e tratteggiata relativa alle array **x1** e **y1** . I grafici possono essere più di due e dotati di opzioni o no.
Un altro esempio è il seguente:

```
x=0:.1:1;
y=x.^2-x.^3;
» plot(x,y,x,y,'or')
```

Viene disegnata la stessa funzione, la prima volta col colore e il tratto di default (blu, tratto continuo), dato che non sono state specificate opzioni, la seconda volta senza tratto, ma coi vertici segnati mediante pallini (`'o'`) di colore rosso (`'r'`).



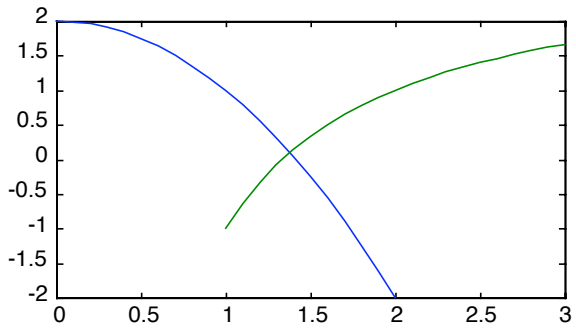
Un altro modo di sovrapporre due grafici differenti è quello di usare il comando **hold** .
Infatti, normalmente ogni comando **plot** cancella il precedente grafico e disegna il nuovo. Ma scrivendo

```
» hold on
```

si disabilita questa opzione e i grafici successivi verranno sovrapposti a quelli già esistenti. Per esempio possiamo disegnare due curve differenti anche come dominio e sovrapporre i grafici

```
» x=0:.1:2; y=2-x.^2;
» plot(x,y)
» x1=1:.1:3; y1=(3*x1-4)./(x1);
» hold on
» plot(x1,y1)
```

Come risultato abbiamo due grafici sovrapposti



Per ripristinare l'opzione che il grafico venga cancellato ad ogni **plot** si scrive

```
» hold off
```

Comunque per cancellare la finestra grafica, indipendentemente dallo status di **hold** basta il comando

```
» clf
```

C'è anche la posibilità di disegnare più grafici, non sovrapposti, ma affiancati nella stessa finestra, usando il comando **subplot** .
In pratica è possibile dividere la finestra in una matrice $n \times m$ in cui ogni elemento della matrice è un grafico.

Come esempio disegniamo 6 grafici disposti in due righe e tre colonne, usando le due funzioni sopra definite e una

terza definita da **x2** e **y2** , dove **x2=0:.1:pi;** e **y2=sin(x2);** .
Il comando **subplot** per creare il primo grafico stabilisce innanzitutto che la matrice dei grafici è 2×3 e che il grafico ottenuto col seguente **plot** è il primo

```
» subplot(2,3,1) , plot(x,y)
```

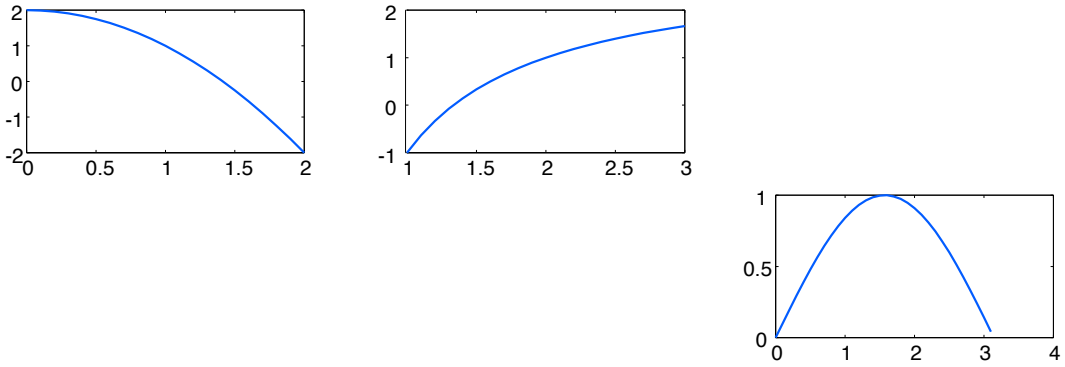
Il secondo grafico si ottiene con

```
» subplot(2,3,2) , plot(x1,y1)
```

Per l'ultimo grafico (il sesto), il comando è

```
» subplot(2,3,6) , plot(x2,y2)
```

Il risultato è il seguente. Avendo disegnato solo tre grafici, rimane posto per altri tre grafici

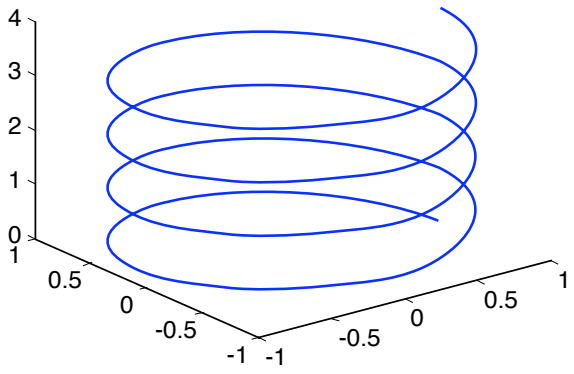


GRAFICA IN 3D: LINEE

Il comando **plot3** è perfettamente analogo al comando **plot** , ma consente di usare tre coordinate e ottenere il disegno di una spezzata elementare in 3 dimensioni.

Per esempio in questo modo si ottiene una porzione di elica cilindrica a passo costante

```
» t=0:.05:8*pi;
» x=cos(t) ;y=sin(t); z=t/(2*pi);
» plot3(x,y,z)
```



Ovviamente si tratta della raffigurazione assonometrica di un oggetto tridimensionale.
Se non si forniscono altre indicazioni, **MatLab** sceglie come angolo di visione (azimuth) rispetto agli assi x, y l'angolo -37.5^0 (la misura dell'angolo di azimuth parte dalla parte negativa dell'asse y), mentre l'altezza (elevazione) è di 30^0.
Nella figura, la retta graduata a destra ha la direzione dell'asse x, quella a sinistra ha la direzione dell'asse y e quella verticale dell'asse z. Non sono esattamente gli assi coordinati, perché l'origine delle coordinate è al centro dell'elica.

È possibile cambiare il punto di vista mediante il seguente comando (con **az** e **el** in gradi)

```
» view(az,el)
```

Scegliendo come angoli rispettivamente 0^0 e 90^0 si ottiene la vista dall'alto con gli assi x e y disposti nel modo solito, ovvero un grafico bidimensionale.

Comunque vari pulsanti nella finestra grafica consentono di cambiare interattivamente il punto di vista del disegno.

GRAFICA IN 3D: SUPERFICI

La rappresentazione di superfici è uno degli aspetti più spettacolari di **MatLab** . Le opzioni disponibili sono

parecchie. Illustriamo qui solo gli aspetti base, lasciando alla documentazione di **MatLab** il compito di descrivere tutte le innumerevoli possibilità di rappresentazione.

Vediamo quindi come è possibile disegnare la superficie grafico di una funzione di due variabili $z = f(x, y)$ in un dominio rettangolare $[x_0, x_m] \times [y_0, y_n]$.

È necessario innanzitutto calcolare la funzione f in vari punti del dominio.
Quindi occorre costruire una griglia dividendo gli intervalli $[x_0, x_m]$ e $[y_0, y_n]$ in un certo numero di punti.
Si avranno le successioni $x_0, x_1, x_2, ..., x_m$ e $y_0, y_1, y_2, ..., y_m$.

Di solito queste divisioni sono uniformi e ottenute scegliendo passi **h** e **k** e quindi generate con comandi tipo

```
» x=x0:h:xm;
» y=y0:k:yn;
```

La funzione dovrà essere calcolata nei punti aventi queste ascisse e queste ordinate. Le coordinate di questi punti possono formare una matrice.

$$\begin{array}{llll} (x_0,y_0) & (x_0,y_1) & (x_0,y_2) & \cdots \quad (x_0,y_n) \\ (x_1,y_0) & (x_1,y_1) & (x_1,y_2) & \cdots \quad (x_1,y_n) \\ \cdots & \cdots & & \ddots \\ (x_m,y_0) & (x_m,y_1) & (x_m,y_2) & \cdots \quad (x_m,y_n) \end{array}$$

In realtà le matrici sono due: una per le ascisse, una per le ordinate dei punti della griglia. Esiste una funzione che genera facilmente queste due matrici partendo dai vettori **x** e **y** .
Dato che la funzione ha due output, la sintassi sarà

```
» [xx,yy]=meshgrid(x,y)
```

e si ottengono due matrici **xx** e **yy** di formato $(m + 1) \times (n + 1)$.
Da notare che nella matrice **xx** tutte le colonne sono uguali, mentre nella matrice **yy** tutte le righe sono uguali.
Ora è possibile calcolare la funzione $f(x, y)$ in tutti i punti della griglia e generare una terza matrice $(m+1)\times(n+1)$ che chiameremo per esempio **zz**

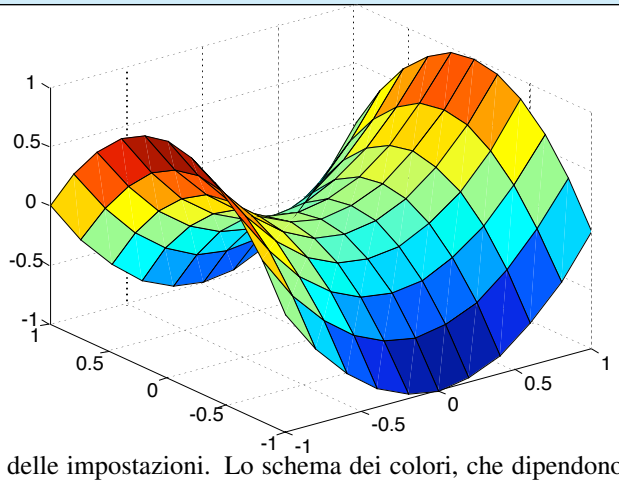
```
» zz=...% funzione di xx,yy
```

Per ottenere il grafico sono disponibili due comandi

```
» mesh(xx,yy,zz)
» surf(xx,yy,zz)
```

Il comando **mesh** rappresenta la funzione mediante una maglia costituita da quadrilateri.
Il comando **surf** rappresenta la funzione sempre mediante un maglia costituita da quadrilateri però riempiti di colore. I colori, senza ulteriori indicazioni, dipendono dalla quota dei singoli quadrilateri.
Il classico esempio è il paraboloide iperbolico nel dominio $[-1, 1] \times [-1, 1]$ generato coi seguenti comandi

```
» x=-1:.2:1;
» y=-1:.2:1;
» [xx,yy]=meshgrid(x,y)
» zz=xx.^2-yy.^2;
» surf(xx,yy,zz)
```



È possibile cambiare molte delle impostazioni. Lo schema dei colori, che dipendono dalla quota, inizialmente è quello predefinito col nome " **jet** ", ma per esempio col comando

```
» colormap copper
```

i quadrilateri vengono riempiti con una tonalità "rame".
Gli schemi di colore predefiniti sono **gray** , **hot** , **cool** , **bone** , **copper** , **pink** , **flag** , **prism** ,

jet , **hsv** . Mediante **help colormap** si ottengono le istruzioni per generare qualunque schema di colori.
Anche l'aspetto della superficie può essere variato col comando **shading** . Le tre opzioni sono

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

Il primo comando toglie le maglie e mostra solo i colori dei quadrilateri.
Il secondo comando toglie le maglie e interpola i colori dando così un aspetto calibrato piacevole alla superficie.
Il terzo ripristina l'opzione iniziale e quindi mostra sia le maglie che le colorazioni dei quadrilateri.

Come nel comando **plot3** , il punto di vista iniziale ha azimuth -37.5^0 e elevazione 30^0, ma può essere cambiato con **view** o mediante i pulsanti della finestra grafica.

GRAFICA IN 2D E 3D: LINEE DI LIVELLO E CURVE IMPLICITE

Una volta definita una funzione di due variabili mediante tre matrici **xx** , **yy** , **zz** come sopra, è possibile tracciare il grafico bidimensionale delle linee di livello di **zz** mediante il comando

```
» contour(xx,yy,zz)
```

che disegna nel piano (x, y) le proiezioni di alcune delle linee di livello.

Se si vogliono alcune ben precise linee di livello, il comando è

```
» contour(xx,yy,zz,[q0 q1 ... qn])
```

che disegnerà le linee di livello alle quote **q0,q1,...,qn**

Sono disponibili alcune delle opzioni grafiche dei grafici bidimensionali, tipo **'r'** e **':'**

In particolare, volendo una sola linea di livello, alla quota **q** , in colore blu, il comando dovrà essere

```
» contour(xx,yy,zz,[q q],'b')
```

La quota va indicata precisamente con **[q q]** e non semplicemente con **q** .

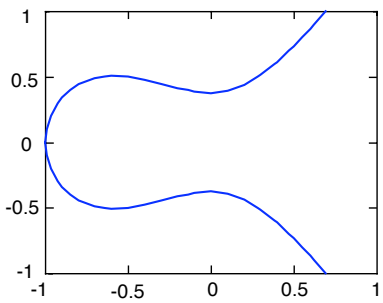
Il comando **contour** fornisce un grafico bidimensionale, ovvero le proiezioni sul piano $[xy]$ delle linee di livello. È però possibile disegnare le linee di livello in 3D, posizionandole alla loro vera quota. Questo fornisce un altro modo di rappresentare una superficie in 3D diverso da quello delle maglie. Il comando è

```
» contour3(xx,yy,zz)
```

Le opzioni per avere le linee desiderate sono le stesse di **contour** .
Normalmente per avere un buon disegno in questo modo è bene specificare un numero abbastanza alto di linee di livello.

Usando il comando **contour** è anche possibile disegnare, con una certa approssimazione, una curva piana assegnata mediante equazione implicita $f(x, y) = 0$.
Occorrerà infatti creare la funzione **zz** , come per disegnare il grafico della funzione $z = f(x, y)$, e disegnare la curva di livello a quota 0. Per esempio, per disegnare la porzione della curva di equazione $8x^3 - 7y^2 + 7x^2 + 1 = 0$ nel quadrato $[-1, 1] \times [-1, 1]$, si possono usare i comandi

```
» x=-1:.1:1;
» y=-1:.1:1;
» [xx,yy]=meshgrid(x,y);
» zz=8*xx.^3-7*yy.^2+7*xx.^2+1;
» contour(xx,yy,zz,[0 0],'b')
```



Da tenere presente che non si può pretendere la precisione assoluta da un grafico di questo tipo e spesso nei punti critici il disegno della curva può essere non del tutto affidabile.
Inoltre il problema spesso consiste nell'individuare un rettangolo contenente la porzione più interessante della curva.