Calcolo Numerico

A.A. 2019-2020

Ingegneria chimica

Equazioni non lineari:

Schemi iterativi e Sistemi non lineari

D. Vitulano

Esercizio

Scrivere uno script Matlab che implementi il metodo delle approssimazioni successive per la soluzione di un'equazione non lineare usando come criterio di arresto il criterio di arresto a posteriori.

Quali parametri devono essere inseriti dall'utente?

Stampare il numero di iterazioni eseguite e l'approssimazione prodotta.

Stampare e disegnare il vettore contenente la stima dell'errore commesso ad ogni iterazione.

Esercizio

```
% script punto_unito
%
% cerca il punto unito xn della funzione f con precisione eps
% scegliendo x0 come punto iniziale. Il procedimento iterativo
% si interrompe se si raggiunge il numero massimo di iterazioni
% maxiter; in caso contrario calcola il numero di iterazioni
% effettuate n_iter
%
%
% INPUT
% f = espressione della funzione di cui si vuole cercare il punto unito
% x0 = punto iniziale
% eps = limite superiore dell'errore da usare come criterio di arresto
% (err<eps con err = |x(n)-x(n-1)|)
% maxiter = numero massimo di iterazioni consentite
%
```

```
format long;

f = input(‘introduci la funzione di iterazione f = ‘);

x0 = input(‘introduci il punto iniziale x0 = ’);

eps = input(‘introduci la precisione richiesta eps = ’);

maxiter = input('introduci il numero massimo di iterazioni consentite
maxiter = ');

if maxiter<=0 ,
    error(‘maxiter deve essere un numero positivo’),
end

figure,   hold on % lo script grafica l’errore ad ogni iterazione
```

```
for n_iter = 1:maxiter
       xn = f(x0);
       err = abs(xn-x0);
       plot(n_iter,err,'*')
       if err<=eps,
            break,
       else,
            x0 = xn;
       end
end

fprintf('n_iter xn errore \n‘)
fprintf('%3d \t %15.10f \t %11.10f \n',n_iter, xn, err)
if (n_iter == maxiter) && (err>eps)
     fprintf('La soluzione non ha raggiunto la precisione richiesta in
                  %5d iterazioni',maxiter)
end
```

Nota: per usare un criterio di arresto a posteriori è stato forzato l’arresto della ripetizione con il comando break
(in alternativa si usi il while)

Esercizio

Rappresentare graficamente la funzione $f(x; y) = x\, e^{-x^2-y^2}$ sul dominio $D = [-2, 2]x[-2, 2]$.

Soluzione

- Definire una griglia sul dominio D (matrice di punti):

- Definire la funzione f(x) e valutarla nei punti della griglia:

- Tracciare il grafico di z

Grafici

- Definire una griglia sul dominio D:

```
>> [x,y]=meshgrid(-2:.5:2,-2:.5:2);
```

La funzione [x,y]=meshgrid(xvett,yvett) genera una griglia di punti le cui coordinate sono contenute nelle matrici x e y. I valori degli elementi di x e y sono dati dai vettori di punti equidistanti xvett e yvett

Grafici

x =

-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000
-2.0000	-1.5000	-1.0000	-0.5000	0	0.5000	1.0000	1.5000	2.0000

y =

-2.0000	-2.0000	-2.0000	-2.0000	-2.0000	-2.0000	-2.0000	-2.0000	-2.0000
-1.5000	-1.5000	-1.5000	-1.5000	-1.5000	-1.5000	-1.5000	-1.5000	-1.5000
-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
-0.5000	-0.5000	-0.5000	-0.5000	-0.5000	-0.5000	-0.5000	-0.5000	-0.5000
0	0	0	0	0	0	0	0	0
0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.5000	1.5000	1.5000	1.5000	1.5000	1.5000	1.5000	1.5000	1.5000
2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000

Grafici

- Definire f(x) e valutarla nei punti della griglia:

```
>> f='x.*exp(-x.^2-y.^2)';
>> z=eval(f);
```

Sono le stesse istruzioni usare per definire funzioni di una sola variabile
L'istruzione è equivalente a

```
>> f=@(x,y)[x.*exp(-x.^2-y.^2)];
>> z=f(x,y);
```

oppure

```
>> z=x.*exp(-x.^2-y.^2);
```

Grafici

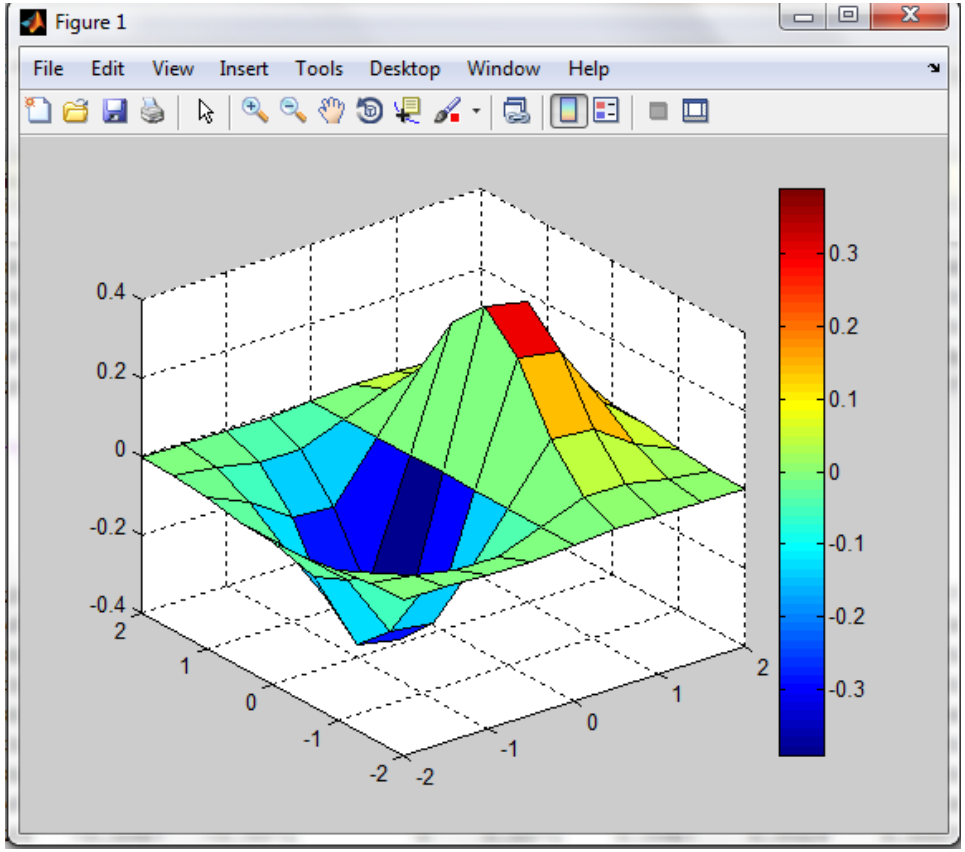
z =

-0.0007	-0.0029	-0.0067	-0.0071	0	0.0071	0.0067	0.0029	0.0007
-0.0039	-0.0167	-0.0388	-0.0410	0	0.0410	0.0388	0.0167	0.0039
-0.0135	-0.0582	-0.1353	-0.1433	0	0.1433	0.1353	0.0582	0.0135
-0.0285	-0.1231	-0.2865	-0.3033	0	0.3033	0.2865	0.1231	0.0285
-0.0366	-0.1581	-0.3679	-0.3894	0	0.3894	0.3679	0.1581	0.0366
-0.0285	-0.1231	-0.2865	-0.3033	0	0.3033	0.2865	0.1231	0.0285
-0.0135	-0.0582	-0.1353	-0.1433	0	0.1433	0.1353	0.0582	0.0135
-0.0039	-0.0167	-0.0388	-0.0410	0	0.0410	0.0388	0.0167	0.0039
-0.0007	-0.0029	-0.0067	-0.0071	0	0.0071	0.0067	0.0029	0.0007

Grafici

- Tracciare il grafico di z

```
>> surf(x,y,z);
>> colorbar
```

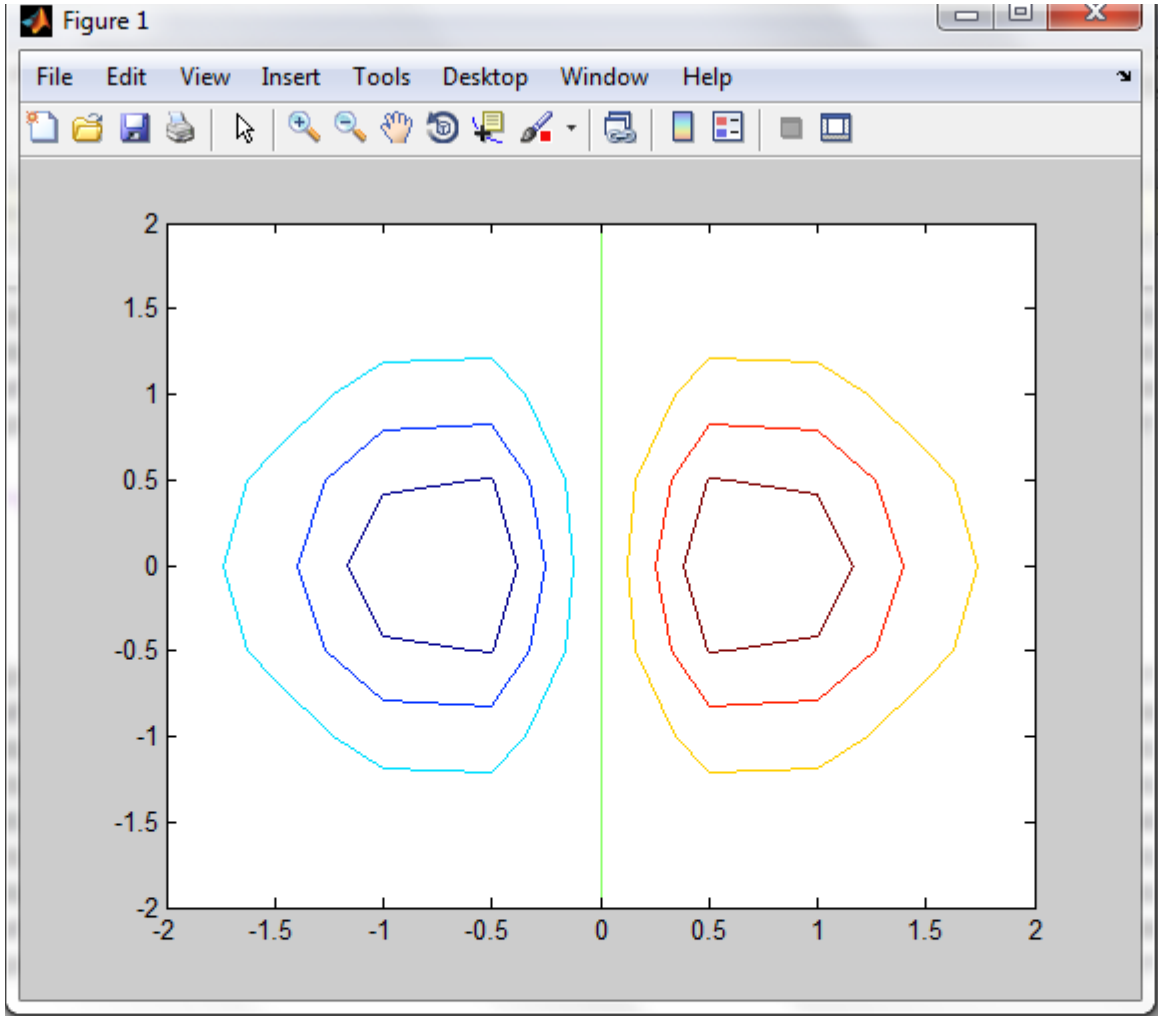


Il comando surf(x,y,z) disegna la superficie z=f(x,y) valutata nei punti della griglia le cui coordinate sono date dagli elementi corrispondenti delle matrici x e y

Colorbar mostra la barra dei colori che distinguono i valori assunti da z

Grafici

Il comando contour(x,y,z) disegna le linee di livello, cioè le curve dei punti in cui la superficie assume un valore fissato costante

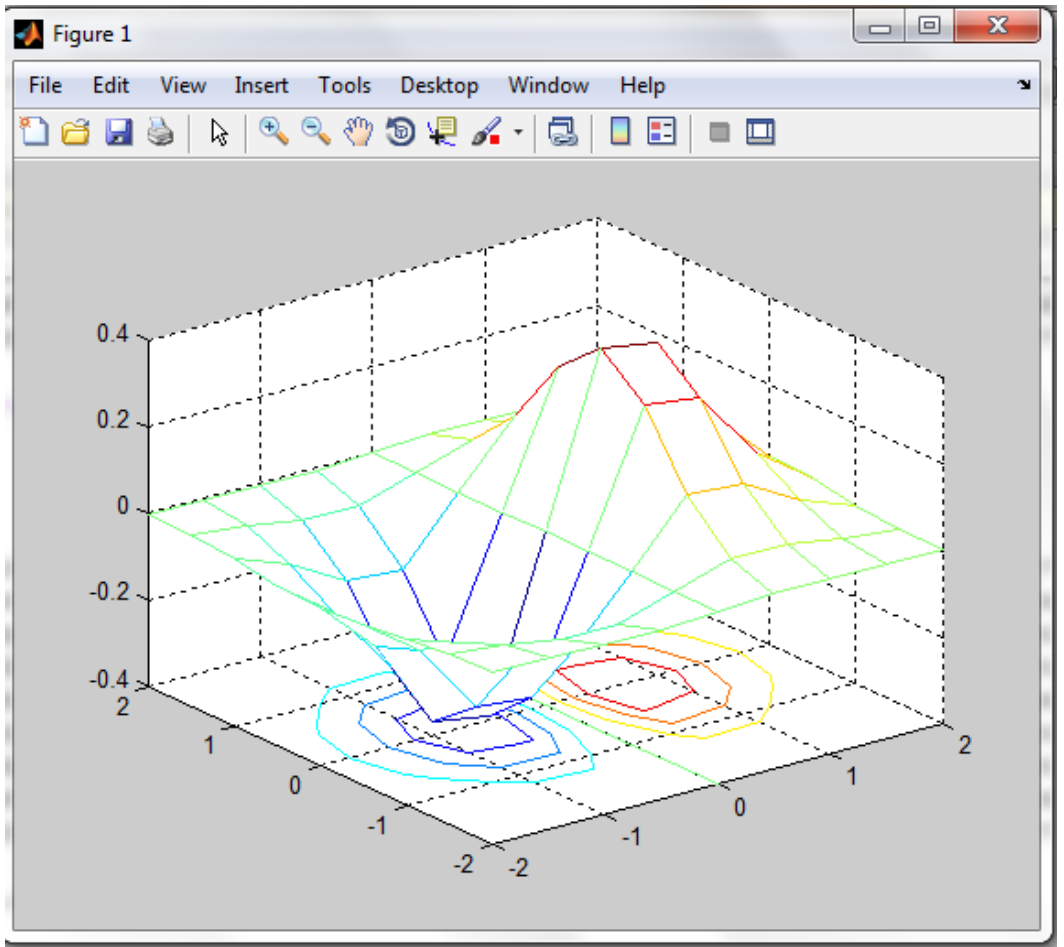


Grafici

Il comando mesh(x,y,z) si può usare in alternativa al comando surf

I comandi meshc(x,y,z) e surfc(x,y,z) disegnano contemporaneamente la superficie e le linee di livello

```
>> meshc(x,y,z)
```



Grafici

Il grafico si può completare aggiungendo etichette e titoli, esattamente come accade per il caso 1D

Altre funzioni utili sono

view	cambia l'orientamento del grafico (punto di vista)
colormap	cambia il colore del grafico
shading	cambia l'ombreggiatura del grafico

Usare lo help per stabilire quando usare le funzioni pcolor e plot3

Grafici: esempi

Per disegnare il grafico della seguente funzione $f(x,y) = x^2+y^2-2x-3$ sul dominio $D = [-3,3]$ è necessario

- definire la griglia (matrice) dei punti $\mathrm{X} = [\mathrm{x},\mathrm{y}]$ su cui è definita la funzione f

```
[x,y]=meshgrid(-3:.1:3,-3:.1:3);
```

dove le variabili di output x e y sono matrici

- definire la funzione di cui disegnare il grafico

```
f=@(x,y)[x.^2+y.^2-2*x-3];
```

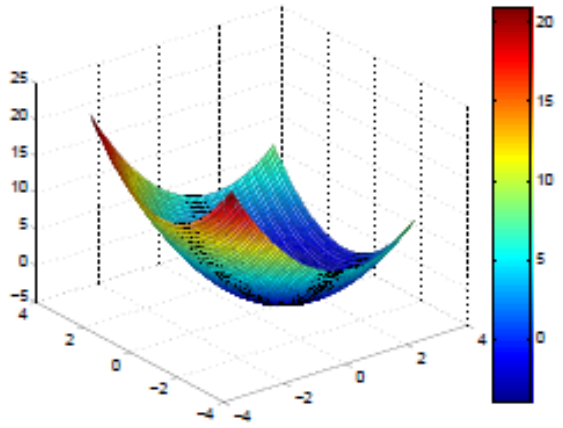
- valutare la funzione nei punti della griglia

```
z = f(x,y);
```

Grafici: esempi

- disegnare la funzione valutata nei punti $X = [x, y]$

```
surf(x,y,z); colorbar
```



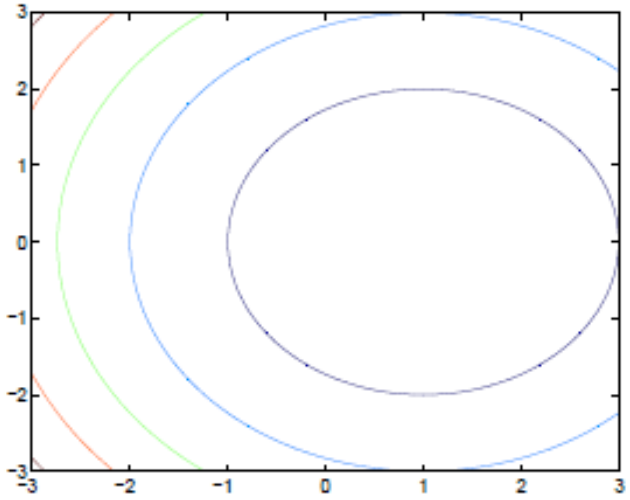
I punti della superficie risultano colorati diversamente secondo il valore assunto. La colorbar riporta la scala dei colori

In alternativa si può usare il comando

```
mesh(x,y,z)
```

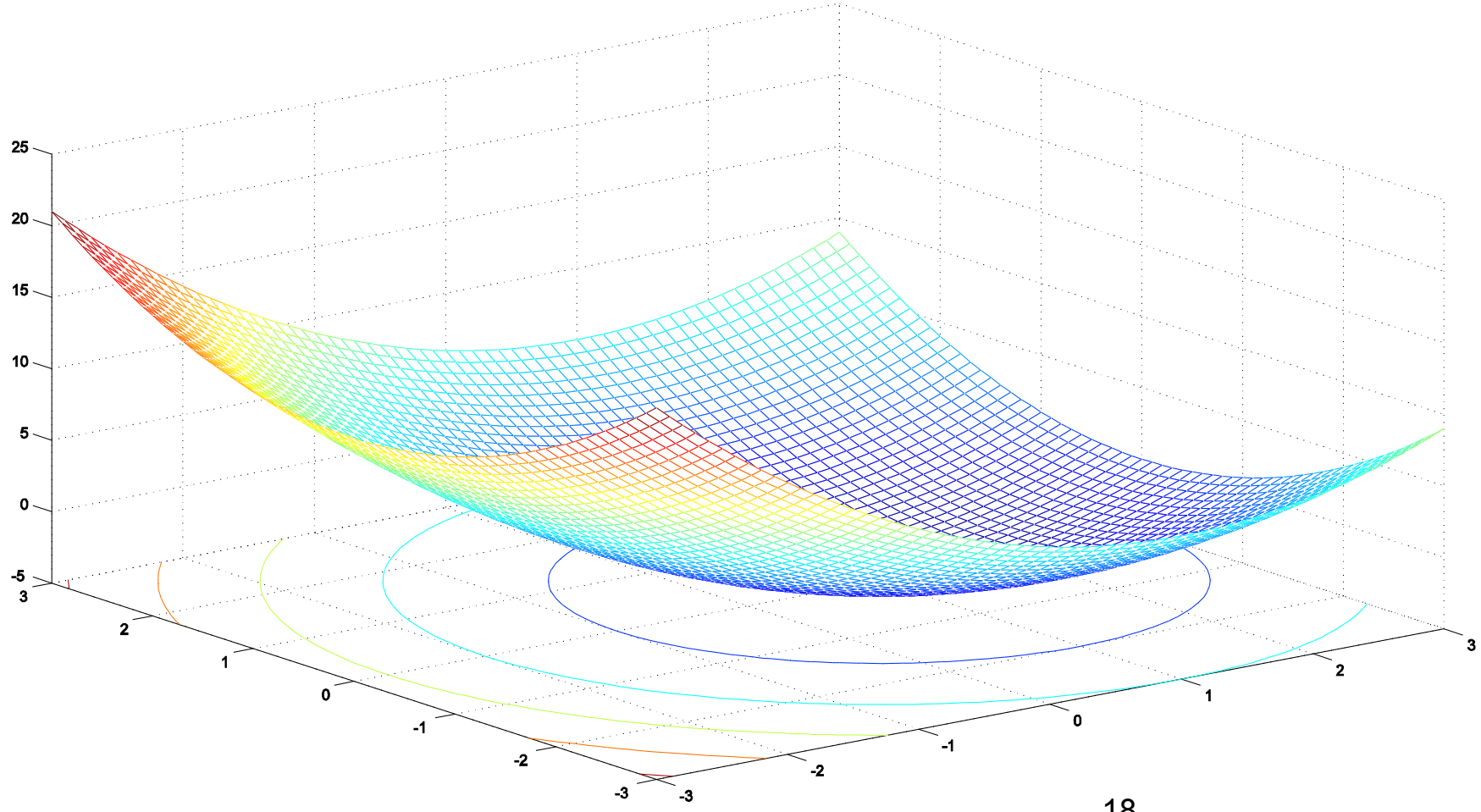
Grafici: esempi

Il comando **contour**(x,y,z) disegna le linee di livello, cioè le curve dei punti in cui la superficie assume un valore fissato costante



Grafici: esempi

meshc(x,y,z) *oppure* surfc(x,y,z) disegnano contemporaneamente la superficie e le linee di livello



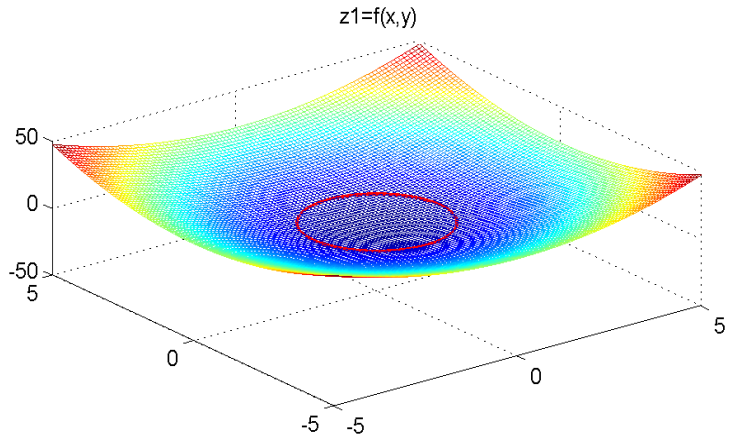
Localizzazione delle radici: rappresentazione grafica

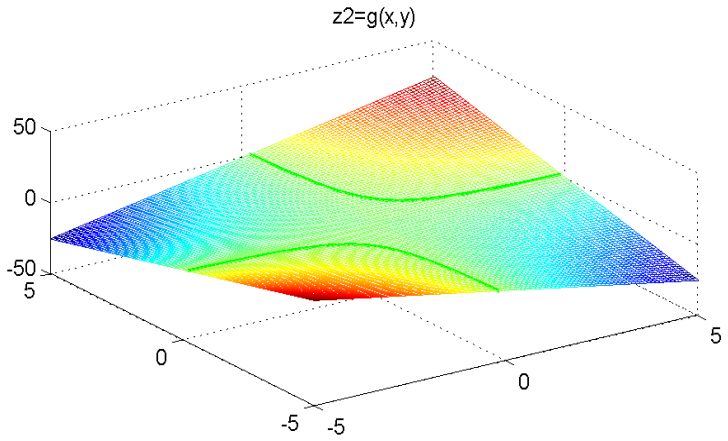
Esempio: $f(x; y) = x^2+y^2-3$, $g(x,y) = xy-1$

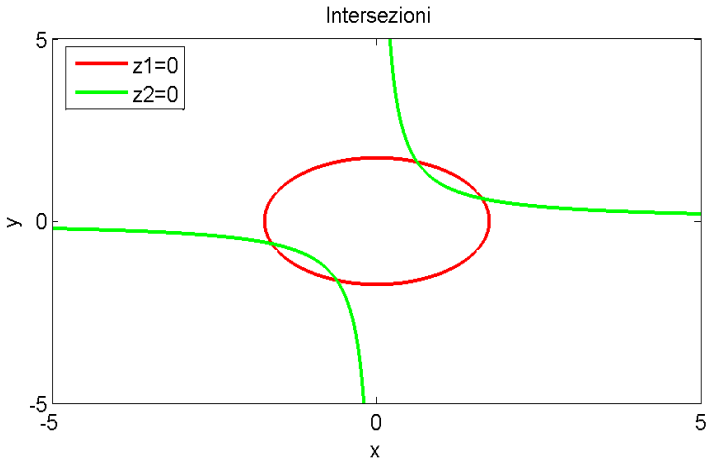
Per localizzare le radici del sistema si disegnano le superfici $z1 = f(x, y)$ e $z2 = g(x, y)$ e le curve di livello $f(x, y) = 0$ e $g(x, y) = 0$.

```
[x,y]=meshgrid(-3:.1:3);
z1=x.^2+y.^2-3;
z2=x.*y-1;
figure,
subplot(2,2,1), mesh(x,y,z1), title('z1=f(x,y)')
hold on,
contour(x,y,z1,[0 0],'r','linewidth',2);  % linee di livello f(x,y) = 0
subplot(2,2,2), mesh(x,y,z2), title('z2=g(x,y)')
hold on
contour(x,y,z2,[0 0],'g','linewidth',2); % linee di livello g(x,y) = 0
subplot(2,2,3), contour(x,y,z1,[0 0],'r','linewidth',2);
hold on, contour(x,y,z2,[0 0],'g','linewidth',2);
xlabel('x'), ylabel('y'), legend('z1=0','z2=0',2)
title('Intersezioni')
```

Esempio: $f(x; y) = x^2+y^2-3$, $g(x,y) = xy-1$







Codice Matlab

```
% script per la soluzione di un sistema non lineare di due equazioni
% in due incognite

%format long;

% funzioni del sistema e derivate parziali rispetto alle incognite
f = @(x,y)(x.^2+y.^2-3);

fx = @(x,y)(2*x);
fy = @(x,y)(2*y);


g = @(x,y)(x.*y-1);

gx = @(x,y)(y);
gy = @(x,y)(x);
```

```
% punto iniziale
xn = 1/2; yn = 3/2;

% inizializzazione variabili
iter = 0;
errx = 10; erry = 10;
errf = abs(f(xn,yn));    errg = abs(g(xn,yn));

[iter xn errx yn erry errf errg]

% iterazioni del metodo di Newton
for iter= 1:3,        % oppure ciclo while
   xv = xn;   yv = yn;
   Jn = [fx(xv,yv),fy(xv,yv); gx(xv,yv),gy(xv,yv)]; % matrice jacobiana
   [iter det(Jn)]
   Bn = [f(xv,yv);g(xv,yv)];
   Vn = Jn\Bn
   xn = xv-Vn(1);,   yn = yv-Vn(2);
   errx = abs(xn-xv);    erry = abs(yn-yv);
   errf = abs(f(xn,yn));    errg = abs(g(xn,yn));
   [iter xn errx yn erry errf errg]
end
```

```
ans =

         0    0.5000   10.0000    1.5000   10.0000    0.5000    0.2500

ans =

     1    -4

Vn =

   -0.1250
   -0.1250

ans =

    1.0000    0.6250    0.1250    1.6250    0.1250    0.0312    0.0156
```

```
ans =

    2.0000   -4.5000

Vn =

    0.0069
    0.0069

ans =

    2.0000    0.6181    0.0069    1.6181    0.0069    0.0001    0.0000
```

```
    3.0000   -4.4722
```

Vn =

```
   1.0e-04 *

    0.2157
    0.2157
```

ans =

```
    3.0000    0.6180    0.0000    1.6180    0.0000    0.0000    0.0000
```