Il metodo di Newton per sistemi di eq. non lineari

Algoritmo (`newtonsys.m`):

Input: `f, Jf, x0, tol, kmax`

Output: `z, k, res, ERR`.

k=0; err=tol+1; ERR=[ ]; x0 colonna ;

while $k < kmax$ *and* $err \geq tol$ **do**

valuto `A=Jf(x0);`

valuto `b=-f(x0);`

risolvo `A y=b` con il comando \;

aggiorno la soluzione: `xnew=x0+y;`

calcolo l'errore `err=norm(y), ERR=[ERR; err];`

aggiorno `k`: `k=k+1;`

aggiorno `x0`: `x0=xnew;`

end

salvo la soluzione in `z=x0;`

calcolo il residuo `res = norm(f(z),inf)`

$(\|\mathbf{v}\|_\infty = \max_{i=1,\ldots,n} |v_i|)$;

ESERCIZIO

(essisnl)
Calcolare numericamente le 4 radici del sistema non lineare

$$\begin{cases} x^2 - y^2 = 1 \\ x^2 + y^2 - 2x = 3 \end{cases} \Leftrightarrow \begin{cases} x_1^2 - x_2^2 = 1 \\ x_1^2 + x_2^2 - 2x_1 = 3 \end{cases} \quad (1)$$

con il metodo di Newton. Rappresentare graficamente su un solo grafico le storie di convergenza (ovvero gli errori $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$ al variare di $k = 0, ..., k_{max}$) e commentare i risultati ottenuti.
Scegliere questi tre dati inziali `x0=[1;1]`, `x0=[2;-1]`, `x0=[-2;-2]`.

Svolgimento

1. Rappresentazione grafica delle funzioni $f_1(x_1, x_2)$ e $f_2(x_1, x_2)$ per la localizzazione delle radici
2. Risoluzione numerica

Rappresentazione grafica per la localizzazione delle radici

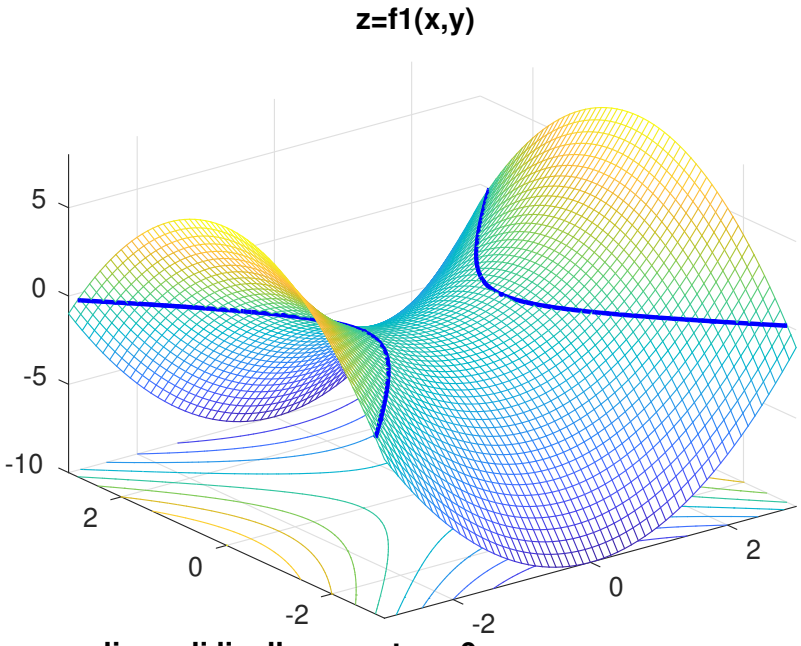
Disegno le superfici $z = f_1(x_1, x_2)$ e $z = f_2(x_1, x_2)$ e le contour a quota $z = 0$:

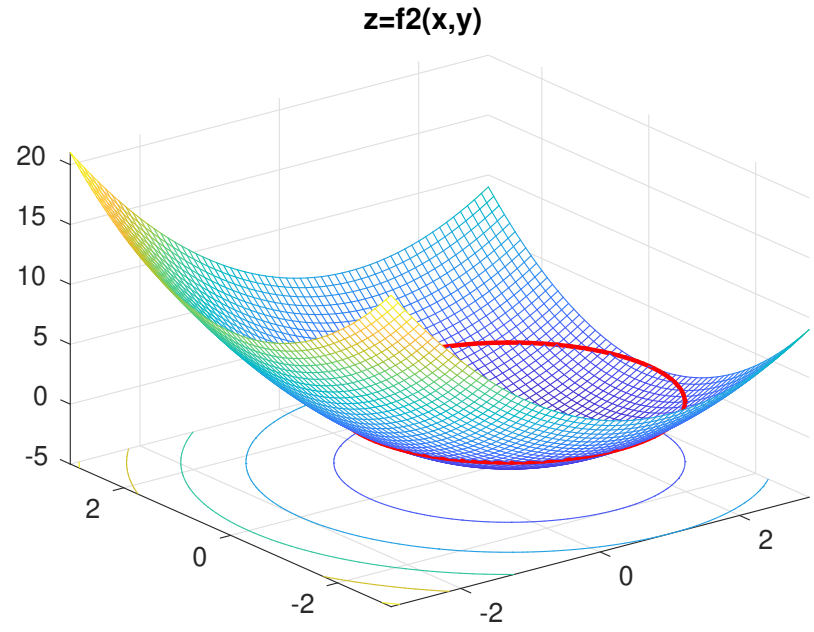
```
[x1,x2]=meshgrid(-3:.1:3);
f1=@(x1,x2)x1.^2-x2.^2-1;
f2=@(x1,x2)x1.^2+x2.^2-2*x1-3;
z1=f1(x1,x2);    z2=f2(x1,x2);
figure(1);clf
% divido la figura in 2 x 2 subplot,
subplot(2,2,1)  % disegno sul subplot 1 (alto sx)
meshc(x1,x2,z1); title('z=f1(x_1,x_2)')
xlabel('x_1'); ylabel('x_2')
hold on
grid on
contour(x1,x2,z1,[0 0],'b','linewidth',2);
```

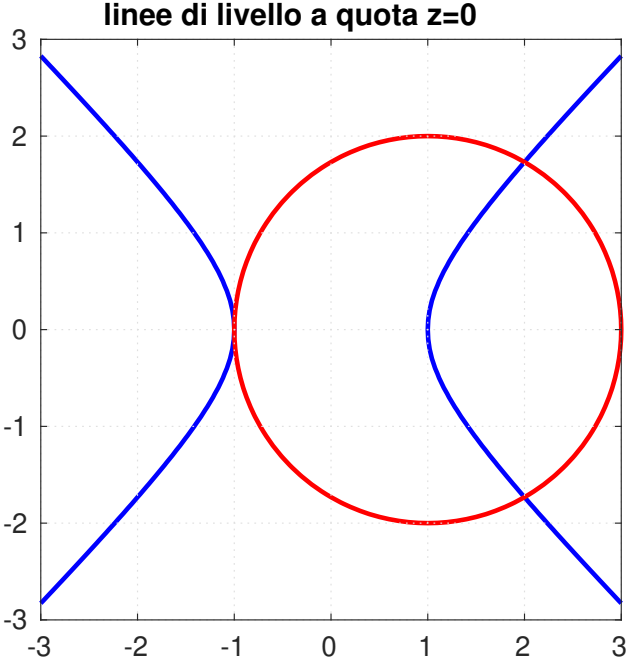
Continuo lo script

```
subplot(2,2,2) % disegno sul subplot 1 (alto dx)
meshc(x1,x2,z2); title('z=f2(x_1,x_2)')
xlabel('x_1'); ylabel('x_2')
hold on
contour(x1,x2,z2,[0 0],'r','linewidth',2);
grid on
```

```
subplot(2,2,3) % disegno sul subplot 3 (basso sx)
contour(x1,x2,z1,[0 0],'b','linewidth',2);
hold on
contour(x1,x2,z2,[0 0],'r','linewidth',2);
grid on
```







Ho una radice doppia (ho la tangenza fra le due curve) e due radici semplici.

Definisco f e Jf per newtonsys.m

f e Jf devono essere function handle che dipendono dal vettore x.
Prima, per disegnare le superifici ho utilizzato le variabili x1 e x2.
Ora per risolvere il sistema, devo usare una sola variabile vettoriale x di due componenti: x(1) e x(2).

```
f=@(x)[f1(x(1),x(2));    % <-- f_1(x_1,x_2)
       f2(x(1),x(2))];   % <-- f_2(x_1,x_2)
```

Jf è un function handle matrice 2×2

```
Jf=@(x)[2*x(1),    -2*x(2);   % df_1/dx_1 , df_1/dx_2
        2*x(1)-2,  2*x(2)];    % df_2/dx_1 , df_2/dx_2
```

Definire gli altri input per newtonsys.m e chiamare newtonsys.m

Fisso un dato iniziale $\mathbf{x}^{(0)} = [2; 2]$

```
x0=[1;1]; % per la radice in alto a destra
tol=1e-8; kmax=50;
[z1,k1,res1,ERR1]=newtonsys(f,Jf,x0,tol,kmax)
```

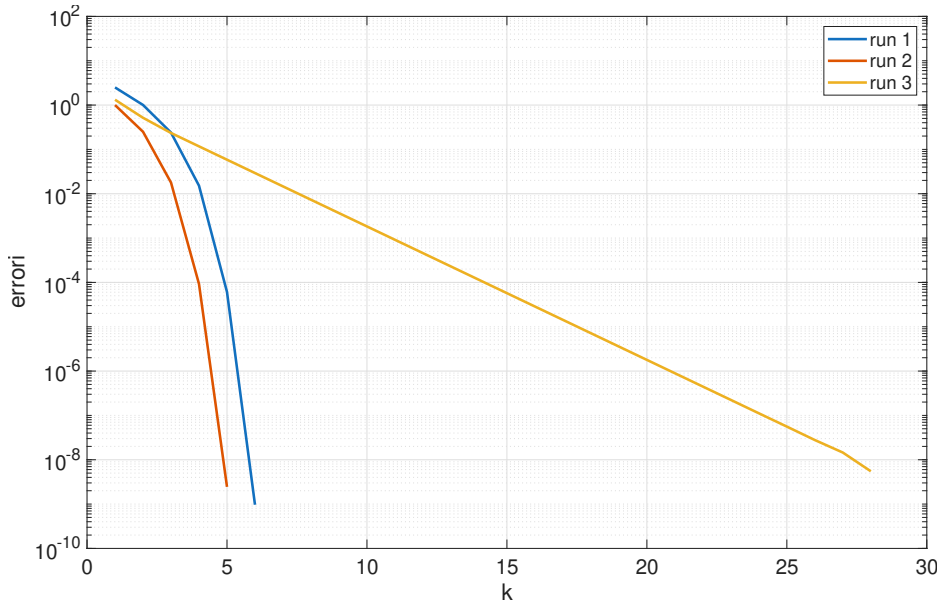
Cambio dato iniziale per trovare le altre radici.

```
x0=[2;-1]; % per la radice in basso a destra
```

```
x0=[-2;-2]; % per la radice doppia a sinistra
```

Storie di convergenza.

```
semilogy((1:k1),Err1,'Displayname','run 1')
hold on
legend('-dynamiclegend')
semilogy((1:k2),Err2,'Displayname','run 2')
semilogy((1:k3),Err3,'Displayname','run 3')
grid on
xlabel('k')
ylabel('errori')
```



La curva gialla mostra una convergenza lineare. Le altre due curve convergenza quadratica. Infatti `alpha=(-1,0)` è radice multipla, le altre due sono semplici.

Il metodo di Broyden

E‘ una possibile generalizzazione del metodo delle secanti per risolvere sistemi di equazioni non lineari.
Non richiede il calcolo della matrice Jacobiana, ma costruisce ad ogni k una matrice B_k che approssima in maniera opportuna $J_F(\mathbf{x}^{(k)})$.
Richiede una matrice iniziale B_0, una scelta comune consiste nel prendere $B_0 = I$ (matrice identità).
Scaricare `broyden.m` dal direttorio delle function matlab `paola-gervasio.unibs.it/CS/matlab`

La sintassi di chiamata è:

```
[zero,k,res,ERR]=broyden(fun,B0,x0,tol,kmax,pflag)
```

Risolvere il problema precedente con il metodo di Broyden.

Altri esempi di sistemi non lineari

$$\begin{cases} 4x_1^2 + x_2^2 = 4 \\ x_1 + x_2 = sin(x_1 - x_2). \end{cases}$$

$$\begin{cases} -\frac{1}{81}\cos(x_1) + \frac{1}{9}x_2^2 + \frac{1}{3}\sin(x_3) = x_1 \\ \frac{1}{3}\sin(x_1) + \frac{1}{3}\cos(x_3) = x_2 \\ -\frac{1}{9}\cos(x_1) + \frac{1}{3}x_2 + \frac{1}{6}\sin(x_3) = x_3. \end{cases}$$

$$\begin{cases} x_1^2 + x_2^2 - 2x_1 - 2x_2 + 1 = 0 \\ x_1 + x_2 - 2x_1x_2 = 0 \end{cases}$$