Zeri di funzione

Ricerca grafica
Abbiamo già visto la function *plotapp.m* ; per localizzare con più precisione gli zeri di una funzione, possiamo anche farne il grafico ed utilizzare il comando

```
>> zoom on
```

che consente di zoomare attorno ad un punto del grafico mediante un click del mouse (anche più volte).

Localizzazione degli zeri
Il metodo di scansione consente di individuare il numero degli zeri e gli intervalli dove cadono con una precisone voluta, mediante lo studio del cambio di segno della funzione. Può essere usato prima di un vero e proprio metodo per la ricerca degli zeri o anche direttamente. È chiaro che è un metodo costoso, perché richiede un gran numero di valutazioni della funzione. Inoltre alcuni zeri potrebbero sfuggirci.
Usiamo per esempio la function *scan.m* :

```
function [ X ] = scan(fun,a,b,step)
% Questo programma individua gli intervalli dove cambia di segno una
% funzione continua fun nell'intervallo [a,b] con passo step
% e li salva nelle colonne dell'array X
z=a:step:b; l=length(z);
y=feval(fun,z);
val=y(1:l-1).*y(2:l)<0;
I=find(val);
X=[z(I);z(I+1)];
end
```

Provare con la funzione già esaminata (f(x)=e^x-x^2(sin x +1)):

```
>> X=scan('f',-3,3,0.2)
>> X=scan('f',-3,3,2)  % che succede in questo caso?
```

Esercizio.
Calcolare con *scan.m* tutti gli zeri della funzione f(x)=x(sin(2x+1)) nell'intervallo [-10,10] con 2 cifre esatte.
[Sono 13: -9.92, -8.35, -6,78, -5,21, -3,64, -2,07, 0, 1.07, 2.64, 4.21, 5.78, 7.35, 8.92]

Bisezione
Un algoritmo da tutti conosciuto è il metodo di bisezione. Eccone un esempio. Facciamo l'analisi del listato seguente:

```
function [b,steps]=bisect(fun,x,tol)
% trova gli zeri di una funzione col metodo di bisezione
% Imput: fun è una stringa che rinvia al nome di un funzione
% Matlab di una variabile reale, x è un valore iniziale nel cui intorno si
% cerca lo zero, tol la tolleranza richiesta(se manca si assume
% tol=eps).
% Output: b contiene il valore dello zero trovato,
% steps, se c'è, fornisce la traccia dei passi effettuati

% Inizializzazione
if nargin<3, tol=eps; end
trace=(nargout==2);
if x~=0, dx=x/20; else, dx=1/20; end
a=x-dx; fa=feval(fun,a);
b=x+dx; fb=feval(fun,b);

% Ricerca del cambio di segno
while (fa>0) == (fb>0)
    dx=2.0*dx;
    a=x-dx; fa=feval(fun,a);
    if (fa>0) ~= (fb>0), break, end
    b=x+dx; fb=feval(fun,b);
end
%format short e
if trace, steps=[a fa; b fb]; end

% Main loop
while abs(b-a)>2.0*tol*max(abs(b),1.0)
    c=a+0.5*(b-a); fc=feval(fun,c);
    if trace, steps=[steps; [c fc]]; end
    if (fb>0) == (fc>0)
        b=c; fb=fc;
    else
        a=c; fa=fc;
    end
end
```

Attenzione all'uso di *nargin* per la scelta della tolleranza. Applicare agli esempi precedenti, ricordando che l'output *steps* è facoltativo.

<u>Esercizio</u>.
Scrivere un programma che implementa il <u>metodo regula falsi</u> per la ricerca degli zeri di una funzione continua. Dati in input due punti *a* e *b* (tali che $f(a)*f(b)<0$) e una tolleranza *tol*, il programma calcola ad ogni passo lo zero *c* della retta secante che unisce $(a,f(a))$ e $(b,f(b))$, e lo sostituisce ad uno degli estremi (come per bisezione). Itera finché $|f(c)|<tol$ oppure si superano le 100 iterazioni (alla fine ne stampa il numero).
Non è necessario seguire la falsariga del programma precedente. Per i più esperti: provare ad aggiungere una parte grafica in cui ad ogni iterazione, premendo un tasto, nella finestra grafica viene disegnata la retta secante.

La funzione *fzero* di Matlab.
Matlab ha al suo interno una funzione specifica per la ricerca degli zeri di una funzione continua, il cui algoritmo combina i metodi di bisezione, secante e interpolazione quadratica inversa (si veda l'help). Questa la sintassi:

```
>> X=fzero(fun,x0)
```

dove *fun* è come al solito una funzione richiamata attraverso una stringa, e *x0* è o un punto nelle cui vicinanze cercare lo zero, oppure un vettore *[a,b]* che indica l'intervallo in cui cercarlo (in tal caso dà errore se il segno non cambia agli estremi). Non funziona per zeri di tangenza.
Varianti:

```
>> [X,Fval]=fzero(fun,x0)  % restituisce il valore di fun(X)
>> X=fzero(fun,x0,optimset('disp','iter')) % mostra i dettagli
```

Un menu per la scelta della funzione
In molti programmi è utile poter scegliere quale funzione usare a partire da un menu. Questo potrebbe essere il listato della nuova function

```
function fv=fscelta(x,scelta)
switch(scelta)
    case 1% a capo
        fv=x-cos(x);
    case 2
        fv=x.^2;
    case 3
        fv=x.^3+x.^2+x+1;
end
end
```

Chiaramente nel programma chiamante sarà mostrato un menu che consente di determinare il parametro *scelta*.

Il metodo del punto fisso
Troviamo lo zero della parabola col metodo del punto fisso: uno zero di f(x) è un punto fisso della funzione g(x)=x-f(x). Ricordate le condizioni per cui l'iterazione x_{n+1}=g(x_{n}) converga?

```
xold=input('dammi xo= ')
x=[];%Dichiarazione matrice vuota
x=[x;xold];% Inserisco in x una nuova riga contenente xold
n=1; d=1;
while (d > 0.00001 & n < 100)
    n = n+1;
    xnew=xold -fscelta(xold,2);
    d = abs(xnew-xold); xold = xnew; x=[x;xold]; end
fprintf('radice trovata f(%4.10f)=%1.10f\n',xold,fscelta(xold,2))
```