LABORATORIO DI CALCOLO NUMERICO

Laurea in Statistica e Informatica

Esercitazione su Zeri di Funzione

Prof. Stefano De Marchi

Padova, December 13, 2010

Alcune utili istruzioni Matlab

- La variabile predefinita `varargin` consente di specificare un numero variabile di paramentri in una funzione. Ad esempio, se definiamo una funzione

```
function myplot(x,varargin)
plot(x,varargin{:});
return
```

  La potremo successivamente chiamare come segue

```
myplot(sin(0:.1:1),'color',[.5 .7 .3],'linestyle',':');
```

  Per sapere poi quanti sono i parametri di input, esiste anche la variabile

```
nargin
```

  Pertanto, potremo fare questo controllo e modificare `myplot` come segue:

```
function myplot(x,varargin)
if nargin==0
    error('bad number of parameters')
    return
elseif nargin==1
plot(x)
else
    plot(x,varargin{:})
end
return
```

- Analogamente esistono

```
varargout,
nargout
```

  con ovvio significato.

Esercizi proposti

1. Data la funzione $f(x) = \mathtt{e}^x - 4x^2$, le cui radici sono $\xi_1 \in (-1, 0)$, $\xi_2 \in (0, 1)$ e $\xi_3 \in (4, 4.5)$. Determinare ξ_1, ξ_2 con il metodo di Newton e ξ_3 con il metodo iterativo $x_{i+1} = \log(4x_i^2)$ usando `tol=1.e-6` e `maxiter=10`.

   Nel caso di Newton si plottino anche x_k, $k = 1, 2, 3, 4$, con simboli differenti, e le rette tangenti in questi punti.

   Il metodo iterativo per calcolare ξ_3 converge per ogni x_0 iniziale?

2. Si consideri la funzione
$$f(x) = x^2 - c\,, \quad c \geq 0 \tag{1}$$
   della quale consideriamo due funzioni d'iterazione:

   (a)
$$g_1(x) = x - \frac{x^2 - c}{2x}\,.$$

   (b)
$$g_2(x) = x - \frac{x^2 - c}{2x} - \frac{\left(x - \frac{x^2-c}{2x}\right)^2 - c}{2x}\,.$$

   Studiare le proprietà di convergenza dei due metodi. Provare in particolare che il metodo (b) è del *terzo ordine* per $c > 0$ e del *primo ordine* per $c = 0$. In quest'ultimo caso usare il metodo Δ^2 di Aitken per accelerarne la convergenza.

3. Data la funzione $f(x) = x^3 - 8x^2 + \frac{85}{4}x - \frac{75}{4}$, $x \in I = [2, 3.5]$.

   (a) Si determini la radice di modulo più grande di $f(x) = 0$ con lo schema iterativo:
$$x_{k+1} = y - \frac{f(y)}{f'(x_k)}$$
   con y ottenuta con l'iterazione di Newton.

   (b) Si determini poi la radice di modulo minimo e la stima della sua molteplicità con lo schema di Newton per radici multiple.

$\Diamond\Diamond$

Tempo massimo: 2 ore.

Due utili funzioni

Ecco una possibile implementazione del metodo di Newton che determina anche la molteplicità della radice.

```
function[x,iter,stimaerr,m]=NewtonRadiciMult(f,fder,x0,tol,maxit,varargin)
% ------------------------------------------------
% Function che determina la radice di f e
% la molteplicita' con il metodo di Newton
% ------------------------------------------------
% inputs  f: funzione; fder: derivata di f
%         x0: valore iniziale
% output  x: radice; iter: iterazioni eseguite
%         stimaerr: stima errore; m: molteplicita'
%
%-------------------------------------------------
m0 = 1; x = x0; iter = 1;

stimaerr= -feval(f,x,varargin{:})/feval(fder,x,varargin{:});

x = x+stimaerr; x1 = x; iter = iter+1;

stimaerr = -feval(f,x,varargin{:})/feval(fder,x,varargin{:});

m = m0+1;

while (abs(stimaerr) > tol) & (iter < maxit) & (abs(m-m0)> m/100)
  m0 = m;
  x = x+stimaerr;
  iter = iter+1;
  stimaerr = -feval(f,x,varargin{:})/feval(fder,x,varargin{:});
  m = (x1-x0)/(2*x1-x-x0);
  x0 = x1;
  x1 = x;
end

stimaerr = stimaerr*m;

while (abs(stimaerr) > tol) & (iter < maxit)
  x = x+stimaerr;
  iter = iter+1;
  stimaerr = -m*feval(f,x,varargin{:})/feval(fder,x,varargin{:});
end
 stimaerr = abs(stimaerr);
  if (stimaerr > tol)
    warning('Impossibile raggiungere la tolleranza richiesta')
    warning('entro il numero massimo di iterazioni consentito.')
  end
```

```
function [xv,f,k]=Met_NewtonModificato(x0,kmax,tol,fun,dfun,molt)
%-------------------------------------------------------------
% Metodo di Newton Modificato per trovare zeri multipli
% di funzioni
%
% Inputs: x0, valore da cui iniziare a iterare,
%         kmax, max numero iterazioni
%         tol, tolleranza
%         fun, stringa della funzione
%         dfun, stringa della derivata della funzione
%         molt, molteplicit della radice
%
% Outputs:
%         xv, vettore delle iterate
%         f, vettore valori delle approssimazioni
%         k, numero iterazioni
%         m, vettore della approssimazioni della molteplicit
%-------------------------------------------------------------

xv=x0;
k=0;
err=tol+1;
while(k <= kmax) & (err > tol)
    k=k+1;
    x=xv(k);
    f(k)=fun(x);
    df=dfun(x);
    if df==1, disp('Arresto per annullamento dfun ');
        return;
    end;

    x=x-molt*f(k)/df;
    xv=[xv;x];
    f=[f;fun(x)];
    err=abs(xv(k+1)-xv(k));
end
    return
```