Laurea Magistrale in Informatica
METODI NUMERICI
Laboratorio 2 – Zeri di funzioni

Esercizio 1

es1.m

```
clear all
phi=@(t) cos(t);
[x,nit,INC]=puntofisso_0(phi,1,10);
x
figure(1); clf
semilogy(1:10 , abs(INC) , 'b.-');
hold on
[x,nit,INC]=puntofisso_0(phi,1.2,10);
x
semilogy(1:10 , abs(INC) , 'k.-');
[x,nit,INC]=puntofisso_0(phi,0.5,10);
semilogy(1:10 , abs(INC) , 'g.-');
x
figure(2); clf
phider=@(t) -sin(t);
fplot(phider , [0,2],'b');
hold on
plot(x,phider(x),'b*')
grid on

figure(1);
phitilde=@(t) (t.^2+2)./(2*t);
[x,nit,INC]=puntofisso_0(phitilde,1,10);
x
semilogy(1:10 , abs(INC) , 'b.--');
[x,nit,INC]=puntofisso_0(phitilde,1.2,10);
x
semilogy(1:10 , abs(INC) , 'k.--');
[x,nit,INC]=puntofisso_0(phitilde,0.5,10);
x
semilogy(1:10 , abs(INC) , 'g.--');

phitildeder=@(t) (2*t*2.*t-(t.^2+2)*2)./(4*t.^2);
figure(2);
fplot(phitildeder , [0.7,1.7],'r');
plot(x,phitildeder(x),'ro')

figure(1);
xlabel('iterazioni');
ylabel('incremento');
legend('caso 1, x0=1','caso 1, x0=1.2','caso 1, x0=0.5',...
       'caso 2, x0=1','caso 2, x0=1.2','caso 2, x0=0.5')

figure(2);
```

```
xlim([0,1.7])
legend('@phider','phider(limite)',...
       '@phitildeder','phitildeder(limite)',...
       'location','southeast')
c@FancyVerbLinee','southeast')
```

```
>> es1
x =  0.74424
x =  0.74764
x =  0.73501
x =  1.4142
warning: axis: omitting non-positive data in log plot
x =  1.4142
warning: axis: omitting non-positive data in log plot
x =  1.4142
warning: axis: omitting non-positive data in log plot
```

Nel primo caso le iterazioni convergono ad un valore di circa $0.73 \div 0.74$ (Nota: l'incremento alla decima iterazione è ancora dell'ordine di 10^{-2} e quindi è normale che la seconda cifra decimale sia ancora incerta). Per tutti i valori d'innesco l'incremento ha diversi valori iniziali, ma poi descresce nello stesso modo (tutte le 3 linee nel grafico sono parallele).

Nel secondo caso, l'incremento descresce molto più rapidamente: le tre curve non sono più rettilinee come nel caso precedente. Siamo di fronte ad un metodo di ordine superiore al primo.

Infatti i grafici di $\phi'(x) = -sin(x)$ e di $\tilde{\phi}'(x) = \frac{4t^2-2(t^2+2)}{4t^2}$ mostrati nella figura 2 indicano che $\phi'(\lim_{k\to\infty} x_k) \neq 0$, mentre $\tilde{\phi}'(\lim_{k\to\infty} x_k) \simeq 0$.

Per verificare che il metodo definito da $\tilde{\phi}$ sia del secondo ordine, prima caratterizziamo il limite $\tilde{\alpha}$ della successione: esso soddisfa l'equazione

$$\tilde{\alpha} = \tilde{\phi}(\tilde{\alpha}) = \frac{\tilde{\alpha}^2 + 2}{2\tilde{\alpha}}$$

che semplificata diventa $\tilde{\alpha}^2 = 2$ e quindi $\tilde{\alpha} = \sqrt{2}$. A questo punto calcoliamo

$$\tilde{\phi}'(\tilde{\alpha}) = \tilde{\phi}'(\sqrt{2}) = \frac{4 \cdot 2 - 2(2+2)}{4 \cdot 2} = 0$$

e con questo abbiamo dimostrato che il metodo è (almeno) del secondo ordine. Per verificare se sia o meno del terzo ordine, calcoliamo

$$\tilde{\phi}''(x) = \frac{d}{dx}\frac{4t^2 - 2(t^2+2)}{4t^2} = \frac{d}{dx}\frac{2t^2-2}{4t^2} = \frac{4t(4t^2) - (2t^2-2)(8t)}{16t^4}$$

e successivamente

$$\tilde{\phi}''(\sqrt{2}) = \frac{4\sqrt{2}(4 \cdot 2) - (2 \cdot 2 - 2)(8\sqrt{2})}{16 \cdot 2^2} = \frac{\sqrt{2}}{4} \neq 0$$

col che abbiamo verificato che il metodo è esattamente del secondo ordine.

Nota: i warning sono causati dal fatto che 0 non può essere rappresentato in scala logaritmica: infatti per k sufficientemente grande, nel secondo metodo abbiamo che $x_{k+1} = x_k$ (nel senso dei numeri macchina) e quindi l'incremento è nullo. Lo potete verificare stampando INC:

```
>> format short e
>> INC
INC =
 Columns 1 through 7:
   1.7500e+00  -6.8056e-01  -1.4755e-01  -7.6561e-03  -2.0723e-05  -1.5184e-10   0.0000e+00
 Columns 8 through 10:
   0.0000e+00   0.0000e+00   0.0000e+00
```

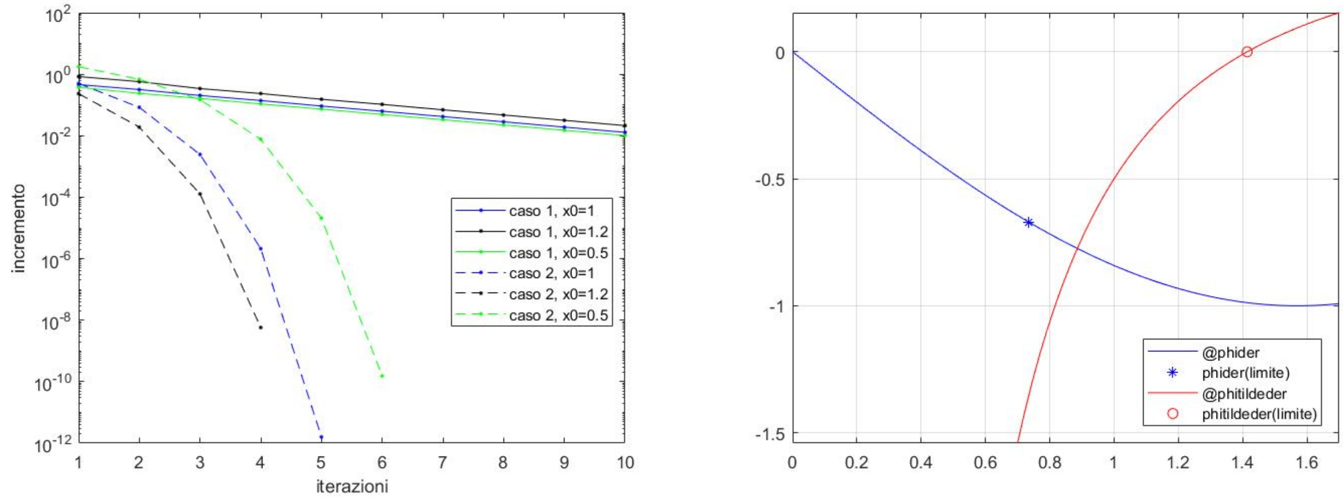

Figure 1: Esercizio 1. A sinistra: incremento al variare delle iterazioni. A destra: funzioni derivate e valore della derivata nel punto.

Esercizi 2,3: introdurre un criterio d'arresto e rendere L opzionale

puntofisso.m

```
function [x,nit,INC,L] = puntofisso(phi,x0,toll,L,maxIt)
% function [x,nit,res] = puntofisso(phi,x0,toll,L,maxIt)
%Input:
% * phi : funzione di cui cercare il punto fisso
% * x0  : valore d'innesco
% * toll: tolleranza
% * L   : valore da usare nella stima d'errore
%         (stimato dal metodo se vuoto o non viene passato)
% * maxIt: numero massimo di iterazioni
%         (default 1000)
%Output:
% * x  : ultima iterata calcolata
% * nit: numero di iterazioni
% * INC: elenco degli incrementi

if (nargin<5)
  maxIt=1000;
end
if (nargin<4) || (isempty(L))
  stimaL = true;
  L=1-eps; %il max valore per un metodo convergente...
else
  stimaL = false;
end
nit=0;
INC=[]; %vettore vuoto
stima=2*toll;
if (stimaL)
  x1=x0; %iterazione precedente a quella salvata in x0
end
while (stima>toll) && (nit<maxIt)
```

```
32    x=phi(x0);
33    inc=x-x0;
34    stima = abs(inc)/(1-L);
35    if (stimaL)&&(nit>1)
36      L = min(abs((x-x0)/(x1-x0)) , 1-eps);
37      x1=x0;
38    end
39    x0=x;
40    if (nargout>=3)
41      INC(end+1) = inc;
42    end
43    nit=nit+1;
44  end
```

Nota: `stima` viene inizializzato a `2*toll` in modo che la condizione di riga 31 sia falsa ed il metodo entri almeno una volta nel ciclo `while`; dopo la prima iterazione sarà disponibile una stima d'errore vera calcolata a riga 34.

Nota: la variabile `x1` viene definita ed aggiornata solo se è necessario stimare `L`; la stima di `L` viene calcolata solo dalla seconda iterazione in avanti (riga 35) perché la formula di di riga 36 richiede di aver già calcolato 2 iterazioni del metodo (altrimenti `x0=x1` e divideremmo per 0).

Nota: poiché per la convergenza deve essere $L < 1$, si usa `1-eps` come valore di default (riga 21) e come valore massimo (riga 36).

Esercizio 4: test finali

Punto 1

es4_1.m

```
clear all, close all
phi=@(t) cos(t);
TOLL=10.^(0:-1:-10);
%oppure TOLL=logspace(0,-10,11);
IT=[];
for toll=TOLL
  [x,nit]=puntofisso(phi,1,toll);
  IT(end+1)=nit;
end
semilogx(TOLL,IT,'o-')
xlabel 'tolleranza'
ylabel 'iterazioni'
```

Punti 2 e 3 Un grafico (figura 1) delle funzioni $y = -0.5\cos(x)e^{-x}$ e $y = x$ nell'intervallo $[-2, 0]$ mostra che c'è una soluzione nell'intervallo $[-1, -0.5]$. Il metodo di punto fisso con $\phi(x) = -0.5\cos(x)e^{-x}$, innnescato con $x_0 = -1$, converge alla soluzione $x = -0.775$ in 4 iterazioni.

La funzione $-0.5\cos(x)e^{-x}$ presenta infinite oscillazioni sempre più ampie per $x \to \infty$, quindi devono esistere infinite soluzioni dell'equazione proposta. Tuttavia innescare il metodo con altri valori di x_0 lo fa convergere ancora alla stessa soluzione oppure divergere.

Nella figura 2 si mostra, in scala logaritmica, la funzione $|\phi(x) - x|$ e $|\phi'(x)|$. Il grafico mostra chiaramente che nell'intervallo $[-2, 0]$ uno zero di $\phi(x) - x$ è localizzato nello stesso punto di uno zero di $\phi'(x)$. Il metodo inquesta regione quindi converge (verosimilmente con ordine 2) alla soluzione trovata prima ($x \simeq -0.775$). In corrispondenza degli altri zeri della funzione $\phi(x) - x$ invece $|\phi'| > 1$ e pertanto il metodo di punto fisso non può convergere a queste soluzioni. (per $x \simeq -5$, $|\phi'| \simeq 10^2$; per $x \simeq -8$, $|\phi'| \simeq 10^3$, etc)

es4_2.m

```
clear all, close all
phi=@(t) -0.5*cos(t).*exp(-t);
figure(1);
fplot(phi , [-2,0],'b');
hold on
fplot(@(x) x, [-2,0] ,'k--');
legend('y=phi(x)','y=x')

[x,nit]=puntofisso(phi,-1,1e-3)



phider=@(t) 0.5*(sin(t)+cos(t)).*exp(-t)
figure(2)
tt=linspace(-10,2,100000);
semilogy(tt,abs(phi(tt)-tt), tt,abs(phider(tt)),tt,1+0*tt)
grid on
xlabel 'x'
legend('abs(phi(x)-x)','derivata di phi')
c@FancyVerbLineex)-x)','derivata di phi')
```

Punto 4 Si richiede di applicare il metodo di Newton per approssimare gli zeri della funzione

$$f(x) = \phi(x) - x = -0.5\cos(x)e^{-x} - x$$

la cui derivata prima è

$$f'(x) = \phi(x) - x = 0.5(\sin(x) + \cos(x))e^{-x} - 1$$

Il seguente script mostra che il metodo di Newton converge alla soluzione $x = -0.775$ in 3 iterazioni quando innescato con $x_0 = -1$ e che è in grado di convergere anche alle altre soluzioni dell'equazione considerata quando innescato con opportuni valori d'innesco:

es4_1.m

```
clear all, close all
f=@(t) -0.5*cos(t).*exp(-t)-t;
fder=@(t) 0.5*(sin(t)+cos(t)).*exp(-t)-1;

[x,nit]=puntofisso(@(t) t-f(t)./fder(t),-1,1e-3 ,0)
[x,nit]=puntofisso(@(t) t-f(t)./fder(t),-5,1e-3 ,0)
[x,nit]=puntofisso(@(t) t-f(t)./fder(t),-8,1e-3 ,0)
[x,nit]=puntofisso(@(t) t-f(t)./fder(t),-10,1e-3,0)
```

Command prompt

```
>> es4_3
x = -0.77536
nit =  3
x = -4.7920
nit =  3
x = -7.8479
nit =  3
```

```
x = -10.996
nit =  7
```

Nota: poiché sappiamo che il metodo di Newton è almeno di secondo ordine, abbiamo impostato $L = 0$ dalla riga di comando quando abbiamo chiamato la funzione `puntofisso`.