Laboratorio di Analisi Numerica
Lezione 8

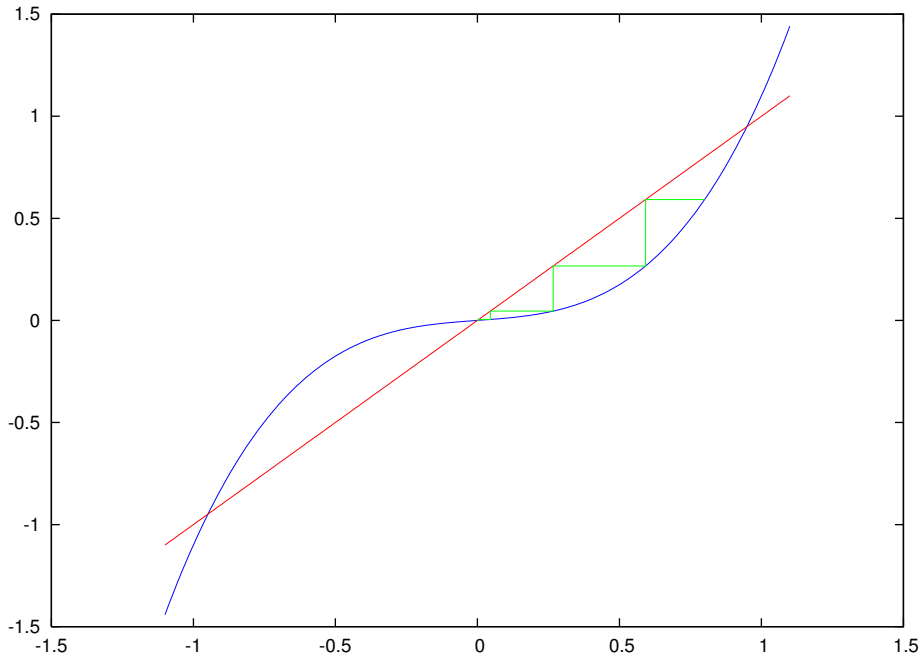
Gianna Del Corso <delcorso@di.unipi.it>
Federico Poloni <fpoloni@di.unipi.it>

20 Novembre 2012

Quantità di esercizi: in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesce a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché sono pensate per "tenere impegnati" per tutta la lezione anche quegli studenti che già hanno un solido background di programmazione.
Quindi fate gli esercizi che riuscite, partendo da quelli *non* segnati come facoltativi, e non preoccupatevi se non li finite tutti!

1 Disegnare il metodo del punto fisso

Oggi vogliamo cercare di disegnare grafici come il seguente che rappresentano il comportamento del metodo del punto fisso per una funzione arbitraria.



Le linee in blu e rosso sono i grafici di $y = g(x) = x^3 + 0.1x$ e $y = x$ rispettivamente. La linea verde rappresenta il comportamento del metodo del punto fisso partendo da $x_0 = 0.9$ per l'equazione non lineare $x = g(x)$. Partendo da $(x, g(x))$, che è l'estremo destro della spezzata verde, ad ogni passo prima ci spostiamo in *orizzontale* fino ad incontrare il grafico di $y = x$, poi in *verticale* fino ad incontrare di nuovo quello di $y = g(x)$.

Esercizio 1. Convincersi che le ascisse dei segmenti verticali rappresentano effettivamente le iterate successive $x_0, x_1 = g(x_0), x_2 = g(x_1), \ldots$ del metodo del punto fisso.

Esercizio 2. Scrivere una `function puntofisso(g,x0,k,int)` che disegna un grafico come il seguente, facendo k passi del metodo del punto fisso su $x = \mathrm{g}(x)$ partendo da `x0`, e disegna i tre grafici sull'intervallo `int`.

Per fare questo esercizio vi servirà un nuovo costrutto del linguaggio di Octave, vale a dire, il modo di passare funzioni come argomenti. Create innanzitutto un file `funzione.m` contenente la seguente funzione

```
function y=funzione(x)
  y=x.^3+0.1*x;
endfunction
```

Notate l'operatore `.^` — vi ricordate a cosa serve? Qui lo usiamo per fare in modo che `funzione(x)` funzioni anche quando `x` è un vettore (in questo senso: restituisce il vettore con elementi $y_i = funzione(x_i)$), che ci tornerà comodo in seguito.

Una volta creata questa funzione, potremo chiamare `puntofisso` con la sintassi `puntofisso(@funzione,0.9,10,-1.1:0.01:1.1)`, che fa 10 passi del metodo e disegna il grafico utilizzando $[-1.1, 1.1]$ come intervallo delle ascisse utilizzando punti spaziati di 0.01 l'uno all'interno di esso.

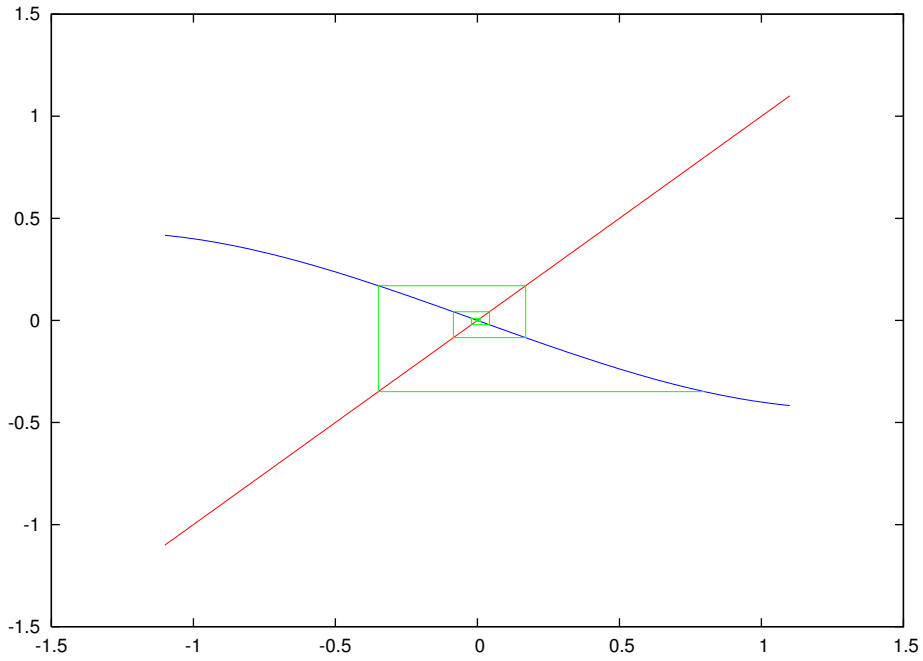
Notate la sintassi `@funzione`, che è quella che si usa per passare una funzione come argomento ad un'altra funzione. All'interno di `puntofisso.m` potete ora invocare la funzione utilizzando la sintassi `g(x)` (se avete chiamato `g` il primo parametro, come suggerito sopra.

La struttura della funzione quindi sarà qualcosa di questo tipo:

```
function puntofisso(g,x0,k,int)
%visualizza k passi del metodo del punto fisso
%per risolvere x=g(x)
%plotta nell'intervallo "int"

clearplot;
hold on; %per disegnare piu' linee sullo stesso grafico
plot(int,g(int)); % per questo abbiamo usato .^ !

%... disegna anche y=x ...

%calcola i punti della spezzata verde
xs=...
ys=...

% disegna la spezzata verde
plot(xs,ys,'g'); %codici colore: g=verde, r=rosso, b=blu
```

2 Altri esperimenti

Esercizio 3. Provate altri punti iniziali e altre funzioni! In particolare, trovatene una per cui la spezzata assume una forma a spirale, come nell'esempio seguente.



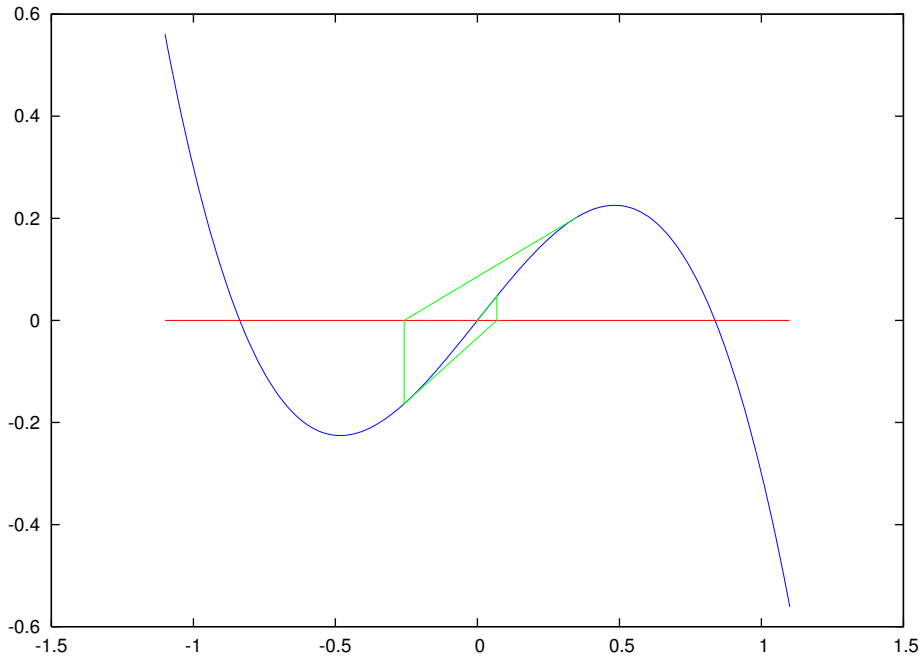
Octave supporta una sintassi alternativa per dichiarare funzioni "brevi" da passare come argomento senza metterle in un file esterno: per esempio, senza creare `funzione.m`, potevamo usare nell'esempio precedente la sintassi

```
octave:18> puntofisso(@(x)(x.^3+0.1*x), 0.9,10,-1.1:0.01:1.1)
```

Dentro la prima coppia di parentesi vanno i parametri della funzione, dentro la seconda l'espressione da valutare per ottenere il risultato.

3 Metodo di Newton

Vogliamo ora disegnare dei grafici come quello seguente



La linea spezzata verde questa volta parte da un punto $(x, f(x))$ (l'estremo più a destra), "segue" la tangente fino ad incontrare l'asse x in rosso, poi prosegue verticalmente fino ad incontrare di nuovo il grafico della funzione in blu.

Esercizio 4 (facoltativo). Convincersi che le ascisse dei segmenti verticali sono le iterate x_i del metodo di Newton, $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.

C'è un ostacolo aggiuntivo per provare a implementare e disegnare il metodo di Newton nello stesso modo: Octave non sa fare le derivate! (almeno senza installare alcuni suoi componenti aggiuntivi...)

Per questo, dovremo passare contemporaneamente due parametri che calcolano la funzione $f(x)$ e la sua derivata in un punto x, per esempio

```
octave:47> plotnewton(@(x)(-x.^3+0.7*x),@(x)(-3*x.^2+0.7),0.35,4,-1.1:0.01:1.1)
```

Esercizio 5 (facoltativo). Scrivere una funzione `plotnewton(f,df,x0,k,int)` che disegna k passi del metodo di Newton come qui sopra.

4 Altri esercizi

Esercizio 6 (facoltativo). Trovare una funzione $g(x)$ e un valore iniziale x_0 tali che sia il metodo del punto fisso che il metodo di Newton su $x - g(x) = 0$ convergono allo stesso zero della funzione. In che modo potete verificare sperimentalmente qual è l'ordine di convergenza dei due metodi? Cosa potete calcolare o disegnare ad ogni passo per verificarlo?

Esercizio 7 (facoltativo). Scrivere una funzione `plotnewtonpoly(p,x0,k,int)` che prende un vettore contenente i coefficienti di un polinomio (esempio: $\begin{bmatrix}1 & 2 & 3\end{bmatrix}$ per $x^2 + 2x + 3$) e si comporta come `plotnewton` su quel polinomio. Notate che stavolta siete in grado di calcolare automaticamente la derivata...