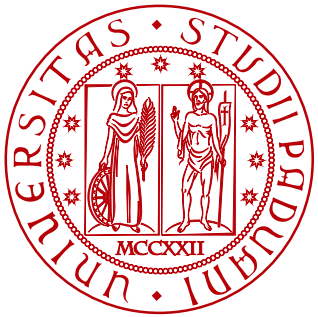


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Matematica

Laboratorio di Calcolo Numerico
Laboratorio 3: equazioni non lineari

Damiano Pasetto

E-mail: pasetto@math.unipd.it

Dispense: http://dispense.dmsa.unipd.it/putti/calcolo_ambientale/index.html

17 Marzo 2014

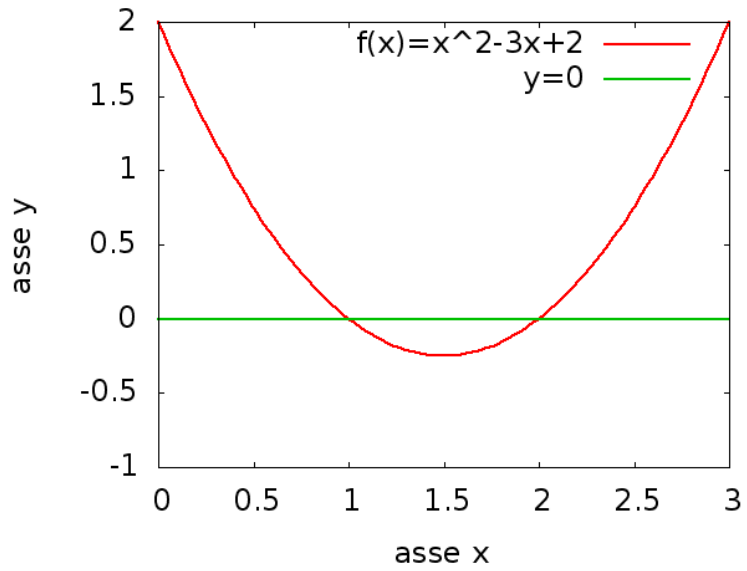
Equazioni non lineari

Problema

Vogliamo implementare un programma in fortran per risolvere la seguente equazione non lineare: trovare $\xi \in \mathbb{R}$ tale che ξ è uno zero della funzione $f(x)$:

$$f(\xi) = 0 \qquad (1)$$

Funzione test



$$f(x) = x^2 - 3x + 2$$

Soluzioni: $\xi_1 = 1$ e $\xi_2 = 2$.
Fissiamo $x > 1.5$ in modo che il problema ammetta una sola soluzione, $\xi = 2$.

Soluzione 1

Algoritmo di Picard

L'equazione non lineare si può riformulare con un problema di punto fisso equivalente, cioè avente la stessa soluzione: trovare $\xi \in \mathbb{R}$ tale che ξ è un punto fisso della funzione $g(x)$:

$$g(\xi) = \xi. \tag{2}$$

Sotto opportune condizioni, l'algoritmo di Picard costruisce una successione $x_0, x_1, x_2, \ldots$ che converge alla soluzione ξ: dato x_0,

$$x_{k+1} = g(x_k). \tag{3}$$

Condizioni di convergenza

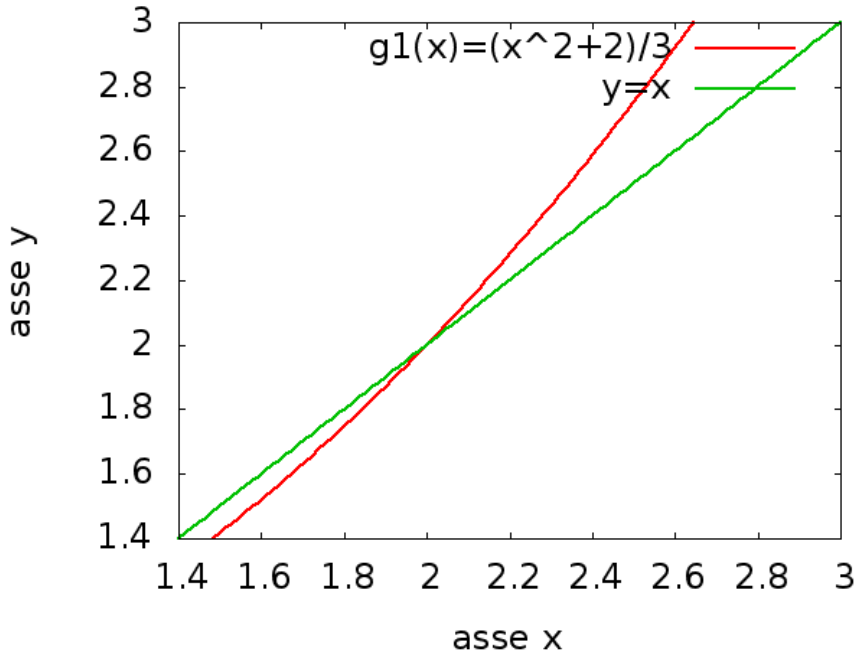
- $|g'(\xi)| < 1$
- x_0 sufficientemente vicina a ξ

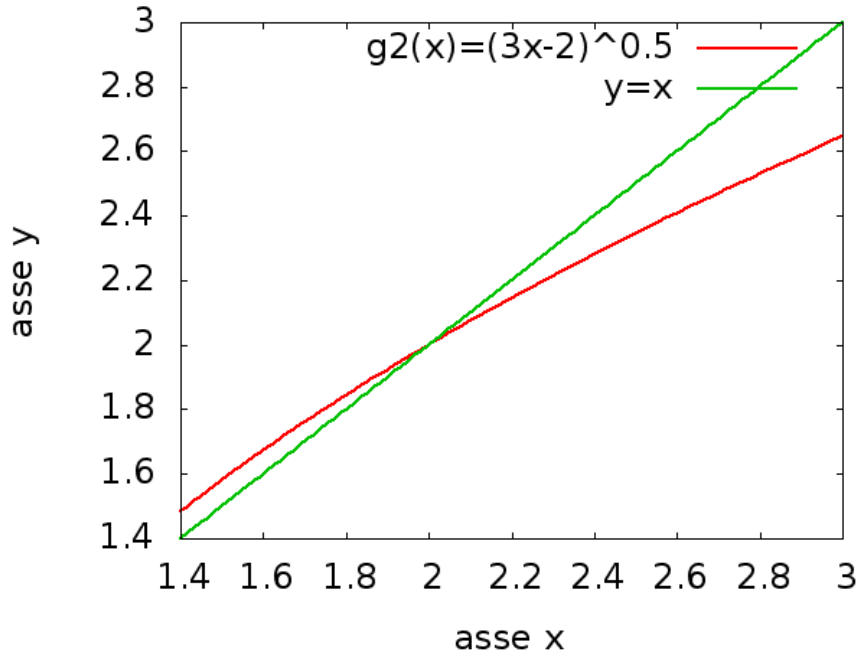
Per ottenere un problema di punto fisso equivalente all'equazione non lineare, isoliamo la variabile x nell'equazione $f(x) = 0$:

$$x^2 - 3x + 2 = 0 \quad \text{diventa}$$

$$x = (x^2 + 2)/3 \quad \text{oppure} \quad x = \sqrt{3x - 2}.$$

Possiamo scegliere $g_1(x) = (x^2 + 2)/3$ oppure $g_2(x) = \sqrt{3x - 2}$.





Soluzione 2

Algoritmo di Newton-Raphson

L'algoritmo di Newton-Raphson (o metodo delle tangenti) costruisce la seguente successione $x_0, x_1, x_2, \ldots$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{4}$$

Se x_0 è sufficientemente vicina a ξ, la successione converge a ξ.
Tale iterazione è analoga all'iterazione di Picard con

$$g(x) = x - \frac{f(x)}{f'(x)} \quad \Longrightarrow \quad g_3 = x - \frac{x^2 - 3x + 2}{2x - 3} = \frac{x^2 - 2}{2x - 3} \tag{5}$$

Criteri per la terminazione dell'algoritmo

- $|x_{k+1} - x_k| < \text{toll} \quad \longrightarrow \quad$ convergenza
- $k > k_{max} \quad \longrightarrow \quad$ non convergenza

Grafici delle funzioni: gnuplot

È sempre utile visualizzare graficamente le funzioni e i dati che si devono utilizzare. Il programma *gnuplot* permette di visualizzare i grafici delle funzioni direttamente dal terminale.

- `[studente@pc ~]$ mkdir laboratorio3`
- `[studente@pc ~]$ cd laboratorio3`
- `[studente@pc laboratorio3]$ gnuplot`
- `gnuplot>`
  NOTA: all'interno del programma *gnuplot* non possiamo utilizzare i comandi Linux del terminale.
- Grafico della funzione $y = x$:
  `gnuplot> plot x`
- Grafico delle funzioni $y = x^2 - 3x + 2$ e $y = 0$:
  `gnuplot> plot x**2-3*x+2, 0`

Comandi di gnuplot 1

- Guida ai comandi di gnuplot: *help* seguito dal comando:

```
gnuplot> help plot
gnuplot> help style
```

- Cambiare la legenda: *title* seguito dalla legenda tra apici:

```
gnuplot> plot x**2-3*x+2 title 'y=f(x)', 0 title 'y=0'
```

- Limiti dell'asse x, [xmin:xmax]

```
gnuplot> plot [0:3] x**2-3*x+2, 0
```

- Limiti dell'asse x e y, [xmin:xmax] [ymin:ymax]

```
gnuplot> plot [0:3] [-1:2] x**2-3*x+2, 0
```

- Nomi degli assi x e y: *set xlabel* seguito dal nome:

```
gnuplot> set xlabel 'asse x'
gnuplot> set ylabel 'asse y'
gnuplot> replot
```

- Uscire da gnuplot: *quit*

```
gnuplot> quit
```

Comandi di gnuplot 2

- Salvare un grafico in un file con formato *png*:

```
gnuplot> set terminal 'png'
gnuplot> set output 'nomefile.png'
gnuplot> replot
```

  NOTA: questa sequenza di comandi non visualizza il grafico sullo schermo, ma salva direttamente il grafico nel file *nomefile.png*.

- Per tornare alla visualizzazione dei grafici a schermo:

```
gnuplot> set terminal 'wxt'
```

Esercizio

- Visualizzare le seguenti funzioni in un grafico:

$$y = g_1(x); \quad y = g_2(x); \quad y = g_3(x), \quad y = x.$$

- Salvare il grafico in un file con formato *png*.

Algoritmo di Picard - Lettura dati in INPUT 1

L'algoritmo di Picard dipende dai seguenti dati di *INPUT*:

- `itmax`: massimo numero di iterazioni (*integer*).
- `x0`: soluzione iniziale (*real*8*).
- `toll`: tolleranza per stabilire quando l'algoritmo é arrivato a convergenza (*real*8*).

Siccome vorremo eseguire il programma per diversi valori di questi dati, conviene leggere i dati da un file di testo. Creiamo il file di input *input.dat* con gedit:

```
1 | 100     itmax
2 | 3.0     x0
3 | 1.0e-6  toll
```

Algoritmo di Picard - Lettura dati in INPUT 2

- Comando fortran per l'apertura del file `nomefile`:

  ```
  open(iunit, file='nomefile')
  ```

  Il comando `open` associa il numero intero `iunit` per identificare il file `nomefile` all'interno del programma.
- Comando fortran per la lettura della variabile `nomevar` dall'unità `iunit`:

  ```
  read(iunit,*) nomevar
  ```

  Il comando `read` legge il file corrispondente ad `iunit` ed associa il primo numero del file alla variabile `nomevar`. La lettura successiva comincia dalla linea successiva.
- Comando fortran per la chiusura del file `nomefile`:

  ```
  close(iunit)
  ```

Esempio: salviamo il seguente file sorgente con il nome *picard.f*

```
1   ␣␣␣␣␣␣program picard
2   ␣␣␣␣␣␣implicit none
3   ␣␣␣␣␣␣integer itmax
4   ␣␣␣␣␣␣real*8 x0, toll
5
6   ␣␣␣␣␣␣open(1, file = 'input.dat')
7   ␣␣␣␣␣␣read(1,*) itmax
8   ␣␣␣␣␣␣read(1,*) x0
9   ␣␣␣␣␣␣read(1,*) toll
10
11  ␣␣␣␣␣␣write(*,*) 'INPUT'
12  ␣␣␣␣␣␣write(*,*) 'iterazioni massime: ',itmax
13  ␣␣␣␣␣␣write(*,*) 'condizione iniziale: ',x0
14  ␣␣␣␣␣␣write(*,*) 'tolleranza: ',toll
15  ␣␣␣␣␣␣close(1)
16  ␣␣␣␣␣␣stop
17  ␣␣␣␣␣␣end
```

Esercizio

- Completare il programma picard.f con l'implementazione dell'algoritmo di Picard per la funzione $g_1(x)$.
- Eseguire il programma con i seguenti dati iniziali:
  - $itmax = 100$, x_0=4.0, $toll = 1.0e-8$
  - $itmax = 100$, x_0=2.5, $toll = 1.0e-6$
- L'algoritmo di Picard converge allo zero di $f(x)$? In caso affermativo, in quante iterazioni?
- Modificare il programma utilizzando la funzione g_2 (e poi g_3) al posto di g_1. Con quale funzione l'algoritmo di Picard converge con meno iterazioni?

Traccia del programma: modificare le parti in rosso

```
1  ......program picard
2  ......implicit none
3  ......integer itmax, ...
4  ......real*8 x0, toll, ...
5
6  ......open(1, file = 'input.dat')
7  ......read(1,*) itmax
8  ......read(1,*) x0
9  ......read(1,*) toll
10
11 ......write(*,*) 'INPUT'
12 ......write(*,*) 'itmax: ',itmax
13 ......write(*,*) 'x0: ',x0
14 ......write(*,*) 'toll: ',toll
```

```
15 ......xk = ...
16 ......iter = ...
17 ......scarto = ...
18 ......do while(condizione
                  di terminazione)
19 ......    iter = iter +1
20 ......    xkp1 = g1(xk)
21 ......    scarto = |xkp1-xk|
22 ......    xk = ...
23 ......    write(*,*) ...
24 ......end do
25 ......close(1)
26 ......stop
27 ......end
```