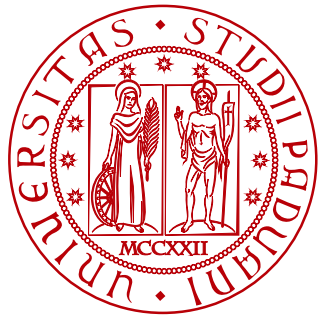


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Laboratorio di Calcolo Numerico
Laboratorio 4: Functions e metodo di Picard

Andrea Franceschini

E-mail: franceschini@dmsa.unipd.it

Dispense:
http://www.math.unipd.it/∼putti/teaching/calcolo_ambientale/index.html

31 Marzo 2015

Function in Fortran

Lo scopo di una funzione è quello di prendere in input un certo numero di valori, fare alcune operazioni con tali argomenti e quindi restituire un unico risultato.
Ci sono alcune funzioni già scritte in Fortran. Queste sono chiamate funzioni *intrinseche*. E.g.: $\cos(x)$, $\sin(x)$, etc.
La sintassi è la seguente (caso di risultato reale):

```
1: real*8 function funz1(arg1,arg2,arg3,...)
2: implicit none
3: integer :: arg1,...
4: real*8 :: arg2,arg3,...
5: Operazioni sugli argomenti per calcolare funz1
6: funz1 = ...
7: end function funz1
```

Function in Fortran (esempio) 1

Riprendiamo il *Programma 3* visto nella seconda lezione, ovvero:

```
 1: program eval3
 2: implicit none       ! Dichiarazione variabili
 3: integer :: n,i
 4: real*8 :: x,y,xmin,xmax,h
 5: n = 11
 6: xmin = -4.d0
 7: xmax = 6.d0
 8: ! "Omogeneizzazione" dell'operazione (n reale)
 9: h = (xmax - xmin) / float(n)
10: do i = 1,n+1
11:       x = xmin + (i-1) * h
12:       ! La funzione da valutare e' y = 4x^3-3x^2-3
13:       y = 4.d0 * x**3 - 3.d0 * x**2 - 3.d0
14:       write(*,*) 'La funzione vale y = ',y,'per x = ',x
15: end do
16: end program eval3
```

Function in Fortran (esempio) 2

Al posto della funzione scritta alla riga 13, utilizziamo la chiamata alla funzione.

```
 1: program eval3_funz
 2: implicit none       ! Dichiarazione variabili
 3: integer :: n,i
 4: real*8 :: x,y,xmin,xmax,h
 5: n = 11
 6: xmin = -4.d0
 7: xmax = 6.d0
 8: ! "Omogeneizzazione" dell'operazione (n reale)
 9: h = (xmax - xmin) / float(n)
10: do i = 1,n+1
11:       x = xmin + (i-1) * h
12:       ! La funzione da valutare e' y = 4x^3-3x^2-3
13:       y = funz1(x)
14:       write(*,*) 'La funzione vale y = ',y,'per x = ',x
15: end do
16: end program eval3_funz
```

Function in Fortran (esempio) 3

La funzione *funz1* è la seguente:

```
1: real*8 function funz1(x)
2: implicit none
3: real*8 :: x
4: funz1 = 4.d0 * x**3 - 3.d0 * x**2 - 3.d0
5: end function funz1
```

Function in Fortran (applicazione)

Scrivere un programma che valuti le seguenti funzioni in $n+1$ punti equispaziati nell'intervallo $I = [x_{min}, x_{max}]$:
$x_0 = x_{min}, x_1 = x_0 + h, \ldots, x_n = x_{max}$, con $h = (x_{max} - x_{min})/n$.
Considerare, per esempio, $x_{min} = -4$, $x_{max} = 6$ e $n = 11$.
Le funzioni da valutare sono:

$$f(x) = x^3 - 4x^2 + 4x - 14$$

e

$$f(x) = \cos(x) + 2\sin(x)$$

Lettura di dati in un file di INPUT

- Come visto per la stampa su file, utilizziamo il comando *open* per associare un file a un intero (identificativo Fortran del file):

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

- Con il comando *read*, il programma legge il contenuto del file e lo memorizza nella variabile da noi indicata. Una volta letta la riga, il Fortran è pronto per leggere la successiva. Non si può leggere due volte la stessa riga.
- La sintassi è:

```
read(ounit,*) var1, var2, var3,...
```

- Infine, si chiude il file:

```
close(ounit)
```

- È molto raro leggere in formato.
- NOTA: non si può leggere E scrivere dallo/nello stesso file.

Gestione dei file in Fortran

Modificare il programma appena scritto in modo che i valori x_{min}, x_{max} e n siano letti da file di input.

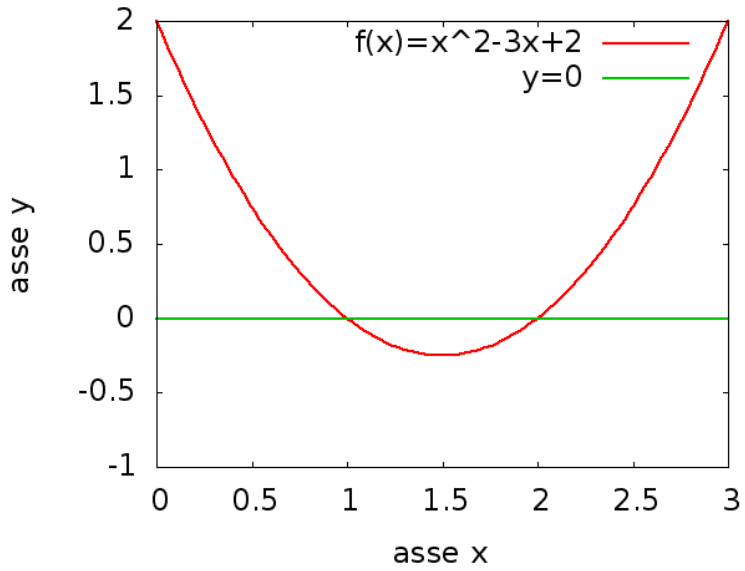
Equazioni non lineari

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 $f(x)$:

$$f(\xi) = 0$$

Esempio



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

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 $g(x)$: $g(\xi) = \xi$.
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, $x_{k+1} = g(x_k)$.

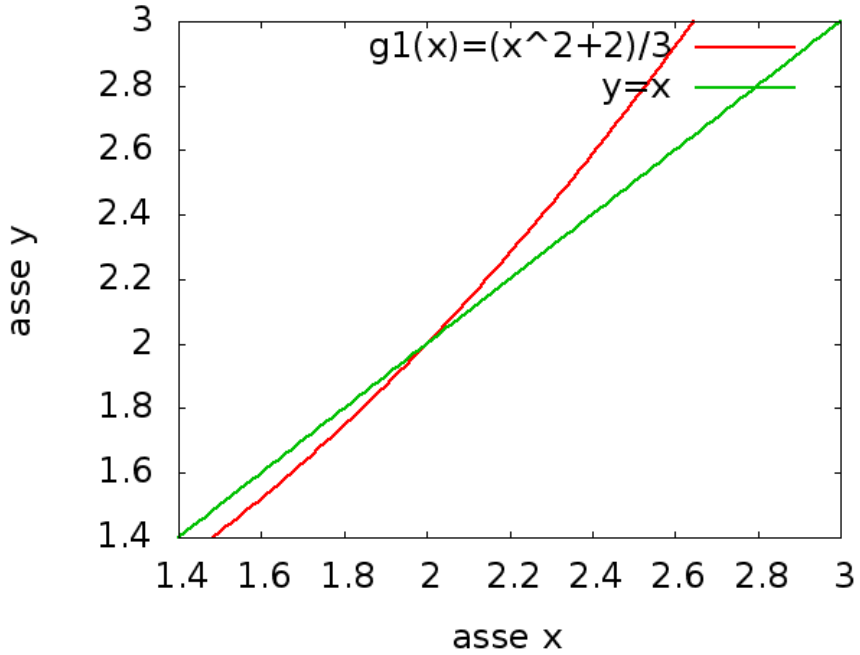
Condizioni di convergenza

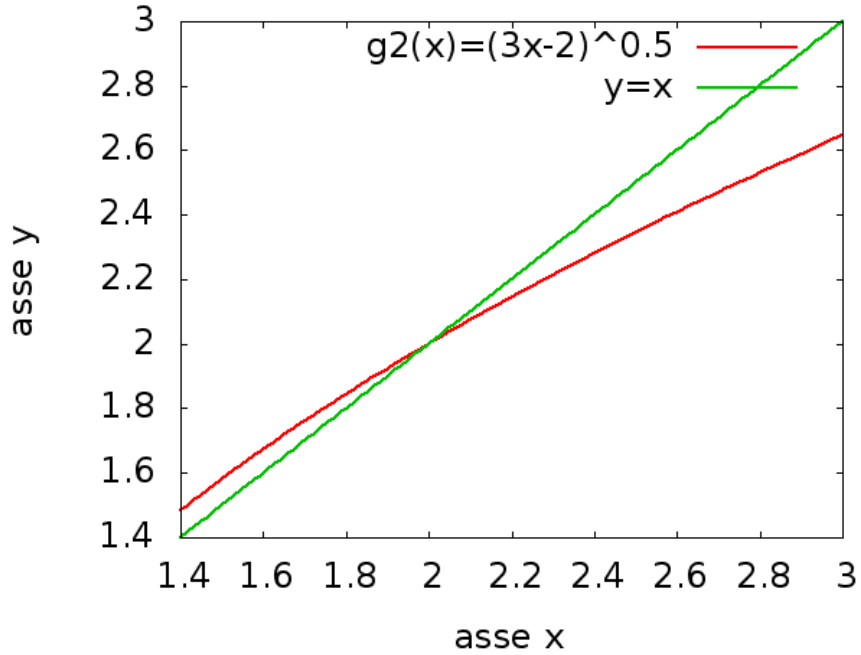
- $|g'(\xi)| < 1$
- x_0 sufficientemente vicina a ξ

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

$$x^2 - 3x + 2 = 0$$

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





Nuova funzione $g_3(x)$

Un'altra possibile formulazione dello stesso problema è:

$$y = \frac{x^2 - 2}{2x - 3}$$

Rappresentare tale funzione con *gnuplot*. Valutare visivamente il criterio di convergenza.
Qualche osservazione?

Traccia del programma: modificare le parti in rosso

```
 1: program picard
 2: implicit none
 3: (dichiarazione variabili)
 4: open(10,file = 'input_picard.dat')
 5: open(11,file = 'scarti_picard.dat')
 6: (leggere itmax,x0,toll)
 7: (inizializzare xk, iter e scarto)
 8: do while (condizione)
 9:       iter = iter + 1
10: ! Utilizzo f1, f2 o f3
11:       xkp1 = funzg1(xk)
12:       scarto = |xkp1-xk|
13:       xk = ...
14:       write(11,'(i6,e15.7)') iter,scarto
15: end do
16: close(10)
17: close(11)
18: end program picard
19:
20: ! - - - - - - - - - - - - - - - - - - - - - -
21: real*8 function funzg1(x)
22: real*8 ...
23: funzg1 = ...
24: end function funzg1
25:
26: ! - - - - - - - - - - - - - - - - - - - - - -
27: (funzione funzg2)
28:
29: ! - - - - - - - - - - - - - - - - - - - - - -
30: (funzione funzg3)
```