1 Teorema degli zeri

Teorema 1.1 *Sia* $f : [a, b] \to \mathbb{R}$ *continua e sia* $f(a)f(b) < 0$, *allora esiste* $x_0 \in (a, b)$ *tale che* $f(x_0) = 0$.

Per fissare le idee faremo la dimostrazione nel caso

$$f(a) > 0,\ f(b) < 0,$$

lo studente ripeta la dimostrazione nel caso $f(a) < 0,\ f(b) > 0$.

La dimostrazione si propone di "costruire " la soluzione come l'unico punto comune a una successione di intervalli contenuti uno nell'altro. Il procedimento é basato sulla seguente procedura di iterazione, detta *algoritmo di bisezione*, che permette di costruire, a partire da un intervallo che soddisfi le ipotesi del teorema, un sottointervallo di lunghezza metá che soddisfa ancora alle ipotesi del teorema.
Iterazione. Sia $I = [\alpha, \beta]$ un intervallo che soddisfa alle ipotesi del teorema e sia $m = (\alpha + \beta)/2$ il suo punto medio, scegliamo un nuovo intervallo J in base al valore di $f(m)$

$$\begin{array}{lll} \text{se} & f(m) = 0 & \text{abbiamo trovato la soluzione} \\ \text{se} & f(m) > 0 & \text{si pone} \quad J = [m, \beta] \\ \text{se} & f(m) < 0 & \text{si pone} \quad J = [\alpha, m]. \end{array}$$

La procedura di iterazione ci permette di generare una successione decrescente di intervalli nel modo seguente.

- Partiamo dall'intervallo $I = I_0 = [a, b]$ e generiamo un nuovo intervallo $I_1 = [a_1, b_1]$ secondo la procedura di iterazione.
- Se non abbiamo trovato la soluzione, ripetiamo la procedura per trovare $I_2 = [a_2, b_2]$ a partire da I_1.
- Procedendo in maniera analoga, se l'algoritmo non si arresta, costruiamo una successione decrescente di intervalli

$$I_1 \supset I_2 \supset \cdots I_n \supset \cdots$$

di lunghezza data da

$$|b_n - a_n| = |b - a|/2^n. \tag{1}$$

La successione $\{a_n\}$ e' crescente e superiormente limitata (ogni b_n e' una limitazione superiore), analogamente la successione $\{b_n\}$ e' decrescente e inferiormente limitata. Posto

$$x_0 = \sup\{a_n, n \in \mathbb{N}\}, \quad x_1 = \inf\{a_n, n \in \mathbb{N}\}$$

otteniamo che $a_n \to x_0,\ b_n \to x_1$, ma $(b_n - a_n) = (b-a)/2^n \to 0$ e quindi dalle proprietá dei limiti segue che

$$x_0 = x_1.$$

Individuato x_0 come candidato, dobbiamo dimostrare che é effettivamente una soluzione dell'equazione. In questo interviene in maniera essenziale l'ipotesi di continuitá di f attraverso il teorema della permanenza del segno. Infatti, per la procedura che abbiamo seguito, in ogni *delta*-intorno di x_0 ci sono sia punti a_n su cui la funzione é positiva, sia punti b_n su cui la funzione é negativa, quindi l'unica possibilitá di non contraddire il teorema della permanenza del segno é che sia

$$f(x_0) = 0.$$

2 Calcolo della soluzione

Per il calcolo della soluzione possiamo ovviamente fare solo un numero finito di passi, ma questo numero puó essere scelto in dipendenza dell'approssimazione voluta. Infatti ogni punto di I_n dista da x_0 meno della lunghezza dell'intervallo stesso, quindi se vogliamo calcolare la soluzione con un'approssimazione $\varepsilon > 0$, basterá fermarsi dopo aver determinato un intervallo di lunghezza minore di ϵ. Poiché la lunghezza dell'intervallo I_n é $(b-a)/2^n$, basterá fissare n in maniera tale che $(b-a)/2^n < \varepsilon$, cioé

$$n > \log_2 \frac{b-a}{\varepsilon}$$

e scegliere come soluzione b_n (approssimazione per eccesso) oppure a_n (approssimazione per difetto). Se scegliamo come soluzione $m_n = (a_n + b_n)/2$, non sappiamo se sia per eccesso o per difetto, ma la sua distanza da ogni punto dell'intervallo é sicuramente minore di $\varepsilon/2$ e quindi m_n é un'approssimazione migliore delle precedenti.

Queste considerazioni suggeriscono il seguente schema per un programma di calcolo.

Schema per il calcolo di una soluzione approssimata x_0 dell'equazione $f(x) = 0$ nell'intervallo $[a, b]$ in cui f è continua e $f(a) > 0,\ f(b) < 0$.

Nel programma dovremo immettere dei dati, che verifichino le ipotesi del teorema,
Input:

f	funzione
a	primo estremo dell'intervallo
b	secondo estremo dell'intervallo
ε	approssimazione voluta

per ottenere un risultato,
Output:

x_0	soluzione approssimata

Si noti che non e' noto se la soluzione trovata e' per difetto o per eccesso. Lo studente determini uno schema per calcolare con approssimazione per difetto (eccesso) ϵ la soluzione.

Lo schema è essenzialmente basato sulla procedura di iterazione descritta precedentemente e può essere così descritto
Inizio

Se $b - a < 2\,\varepsilon$ poni $x_0 := (a+b)/2$ e Stampa x_0

Altrimenti

se $f((a+b)/2) = 0$ poni $x_0 := (a+b)/2$ e Stampa x_0
se $f((a+b)/2) > 0$ poni $a := (a+b)/2 \quad b := b$ e vai a Inizio
se $f((a+b)/2) < 0$ poni $a := a \quad b = (a+b)/2$ e vai a Inizio

Il programma si fermerà necessariamente dopo un numero finito di passi, quando la metà della lunghezza dell'intervallo risulterà minore dell'errore prescelto.

3 Esempio

Vogliamo trovare una soluzione dell'equazione

$$f(x) = x + e^x = 0.$$

Per prima cosa notiamo che sia $x \mapsto x$ che $x \mapsto e^x$ sono funzioni continue e crescenti, ne segue che f é continua e crescente, quindi, se la soluzione esiste é unica. Abbiamo inoltre

$$f(0) = 1, \quad f(-1) = -1 + 1/e < 0$$

e possiamo quindi dedurre che l'equazione ha una sola soluzione che appartiene all'intervallo $(-1, 0)$.

Se si usa l'algoritmo di bisezione per calcolare una soluzione approssimata, per avere l'errore $\epsilon < 1/100$ si devono fare un numero di passi

$$n > log_2 50 \text{ ad esempio } n = 6.$$

Come si vede i passi sono molti per ottenere solo l'approssimazione di 1/100, in pratica una sola cifra decimale giusta.

Lo studente provi a fare alcuni passi dell'algoritmo di bisezione, calcolando soluzione approssimata e approssimazione e controlli i risultati ottenuti con quelli ottenuti con il primo dei due programmi di seguito riportati.

Il primo e' un programma che mostra il passo fondamentale della procedura di bisezione , si immettono: la funzione f e gli estremi x,y dell'intervallo e si ottiene: $c = (a + b)/2$ = la soluzione approssimata (punto medio), $[a, b]$ = il nuovo intervallo, $e = (a - b)/2$ = l'approssimazione. Il programma e' leggermente diverso da quello suggerito negli appunti, cercare di capire su cosa e' basato.

bisezbase := **proc**(*f*::*procedure*, *x*, *y*)
local *a*, *b*, *c*, *e*;
a := *x* ;
b := *y* ;
if 0 ≤ evalf(*f*(*a*) × *f*(*b*)) **then** ERROR(‘*la funzione non cambia segno*‘) **fi** ;
c := 1/2 × *a* + 1/2 × *b* ;
e := 1/2 × *b* − 1/2 × *a* ;
if evalf(*f*(*a*) × *f*(*c*)) < 0 **then** *b* := *c* ; *c* := 1/2 × *a* + 1/2 × *b* ; *e* := 1/2 × *b* − 1/2 × *a*
elif evalf(*f*(*a*) × *f*(*c* = 0) **then** RETURN(‘*soluzione_esatta*’ = *c*)
else *a* := *c* ; *c* := 1/2 × *a* + 1/2 × *b* ; *e* := 1/2 × *b* − 1/2 × *a*
fi;
RETURN(*c*, [*a*, *b*], *e*)
end

Stiamo studiando nell'intervallo $[-1, 1]$ la funzione

$$f := x \to e^x + x$$

Il primo passo sara'

$$\frac{-1}{2},\ [-1,\ 0],\ \frac{1}{2}$$

Gli ulteriori dieci passi sono

$$sol := \frac{-3}{4},\ [-1,\ \frac{-1}{2}],\ \frac{1}{4}$$
$$sol := \frac{-5}{8},\ [\frac{-3}{4},\ \frac{-1}{2}],\ \frac{1}{8}$$
$$sol := \frac{-9}{16},\ [\frac{-5}{8},\ \frac{-1}{2}],\ \frac{1}{16}$$
$$sol := \frac{-19}{32},\ [\frac{-5}{8},\ \frac{-9}{16}],\ \frac{1}{32}$$
$$sol := \frac{-37}{64},\ [\frac{-19}{32},\ \frac{-9}{16}],\ \frac{1}{64}$$
$$sol := \frac{-73}{128},\ [\frac{-37}{64},\ \frac{-9}{16}],\ \frac{1}{128}$$
$$sol := \frac{-145}{256},\ [\frac{-73}{128},\ \frac{-9}{16}],\ \frac{1}{256}$$

$$sol := \frac{-291}{512}, [\frac{-73}{128}, \frac{-145}{256}], \frac{1}{512}$$
$$sol := \frac{-581}{1024}, [\frac{-291}{512}, \frac{-145}{256}], \frac{1}{1024}$$
$$sol := \frac{-1161}{2048}, [\frac{-581}{1024}, \frac{-145}{256}], \frac{1}{2048}$$

Riportiamo un programma che applica la procedura di bisezione per ottenere la soluzione con una desiderata approssimazione. Si immettono: la funzione f, gli estremi x,y dell'intervallo, l'approssimazione desiderata z; l' input 'name' calcola il numero di iterazioni che il programma deve fare per ottenere l'approssimazione desiderata. Si ottiene come output la soluzione approssimata (punto medio) e il numero di iterazioni.

bisezione := **proc**(*f*::*procedure*, *x*, *y*, *z*::*positive*, *iter*::*name*)
local *a*, *b*, *c*, *e*;
if $0 <$ evalf$(f(x) \times f(y))$ **then** ERROR(‘*la funzione non cambia segno*‘)
else $a := \min(x, y)$; $b := \max(x, y)$; $c := 1/2 \times a + 1/2 \times b$; $e := 1/2 \times b - 1/2 \times a$
fi;
while $z < e$ **do**
if evalf$(f(a) \times f(c)) < 0$ **then** $b := c$; $c := 1/2 \times a + 1/2 \times b$; $e := 1/2 \times b - 1/2 \times a$
elif evalf$(f(a) \times f(c) = 0)$ **then** RETURN(’*soluzione_esatta*’ $= c$)
else $a := c$; $c := 1/2 \times a + 1/2 \times b$; $e := 1/2 \times b - 1/2 \times a$
fi
od;
if $4 <$ nargs **then** *iter* := ceil(solve(abs$(y - x)/2^n = z$)) **fi** ;
c
end

Immettendo come dati la funzione

$$f := x \to e^x + x,$$

l'intervallo $[-1, 1]$, e l'approssimazione $z = 10^{-8}$, otteniamo come soluzione approssimata

$$-0.5671432842$$

e come numero di iterazioni necessarie

$$28$$

Si noti che il numero di iterazioni dipende solo dalla lunghezza dell'intervallo e dall'approssimazione scelta e non dalla funzione, invatti il "valore" di iter (numero di iterazioni) dipende solo da x, y, z. Il comando ceil significa il piu' piccolo intero piu' grande o uguale ad un numero reale, cioe'

$$ceil(a) = [a] + 1.$$