Avviso

12 marzo 2010

Questo mio lavoro è stato pubblicato per la prima volta sul n.2 della rivista Magazine di matematicamente.it nell'aprile 2007 con il titolo

Verso l'esame di stato

di Luigi Lecci

Ricerca degli zeri di una funzione

reperibile all'indirizzo
http://www.sergiosimone.it/public/upload/arearecords/201/Matematicamente.it%20Magazine%20n.02.pdf

Rendo ora disponibile lo stesso documento per gli utenti di www.matematicaescuola.it raccomandandolo sia agli studenti del triennio del Liceo Scientifico sia a coloro che si occupano di programmazione in ambito informatico.

Luigi Lecci

Applicazione dell'informatica allo studio delle funzioni

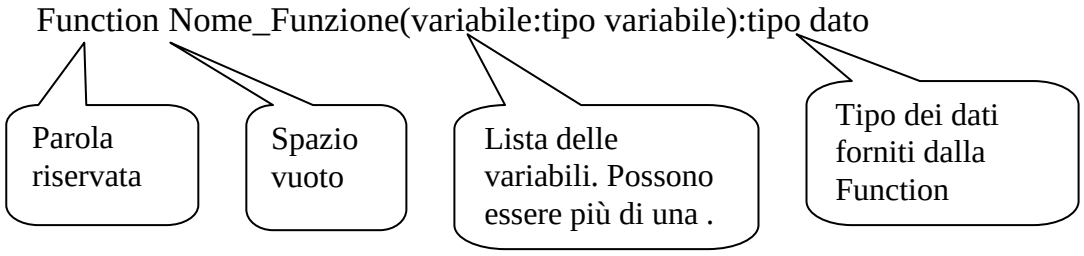
Ricerca degli zeri di una funzione

Introduzione

In analisi numerica spesso ci si imbatte nel problema di dover determinare un valore approssimato dello zero che una funzione continua ammette internamente ad un certo intervallo [a;b]. Illustriamo di seguito tre semplici programmi costruiti con lo scopo di guidare gli utenti poco esperti nella programmazione. In particolare, considerato che nelle classi delle scuole italiane dell'indirizzo PNI (Piano Nazionale per introduzione dell'Informatica)[(1)], il linguaggio che si insegna è il Turbo Pascal, il codice dei miniprogrammi è proprio in questo linguaggio.

Primo Obiettivo: familiarizzare con l'utilizzo di una Function

Il T.P. permette di organizzare il codice di un programma introducendo blocchi particolari definiti dall'utente in modo da rendere più elegante la struttura e la leggibilità del programma.
Una **Function** è un segmento di codice che l'utente scrive per risolvere un particolare problema. Nel nostro contesto utilizzeremo una particolare **Function** per gestire una funzione matematica della quale sia nota l'espressione analitica. La struttura di una Function è la seguente:



Ad esempio, se volessimo scrivere una Function con cui calcolare il cubo di un numero reale x ed alla quale volessimo assegnare come nome proprio Cubo potremmo scrivere il seguente codice in T.P.

```
Function Cubo(x:real):real;
 Begin
   Cubo:=x*x*x;
 End;
```

Il primo programma in Turbo Pascal (T.P.)

Il questo programma calcoliamo alcuni valori della funzione trascendente

$f(x) = 2^{-x} + x^2 - 1$

Premessa

Occorre precisare subito che in T.P. non esiste una funzione predefinita che permetta di calcolare la potenza ad esponente intero di un numero reale e neanche il valore della funzione esponenziale $f(x) = a^x$. Si può però aggirare l'ostacolo sfruttando la nota uguaglianza numerica

$$\forall x > 0,\ x = e^{\log x}, \qquad (1)$$

[(1)] ci riferiamo al Corso PNI del Liceo Scientifico

nella quale il logaritmo che compare ha come base il numero di Nepero e=2,7182818…. In T.P. il logaritmo in base e di un numero positivo x si calcola con la funzione predefinita ln(x).
Nell'espressione esponenziale indicata compare però la funzione esponenziale di base e.
Ebbene, in T.P. esiste la funzione predefinita $f(x)=e^x$ e la sua sintassi è la seguente

exp(x) (2)

In virtù della (1) il calcolo della potenza x^n può essere eseguito ponendo

$$x^n = \exp(n*\ln(x))$$

Analogamente, si può calcolare il valore della funzione esponenziale a^x, con a>0, osservando che risulta

$$a^x = e^{x\ln a} \quad (3)$$

e quindi che

$$a^x = \exp(x*\ln(a)) \quad (4)$$

Riportiamo di seguito il codice del programma, di nome Prova_fx nel quale è stata definita una funzione di nome F.

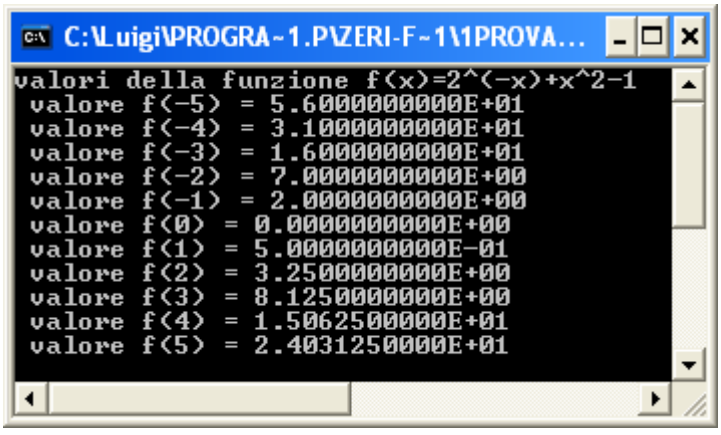
```
Program Prova_fx;
 Uses Crt;
 Var I:integer;
 Function F(x:real):real;
  Begin
   F:=exp(-x*ln(2))+x*x-1;
  End; (* chiude la Function *)
 Begin
  clrscr;
  Writeln('valori della funzione f(x)=2^(-x)+x^2-1');
  For I:=-5 to 5 do
  writeln(' valore f(',I,') =',f(I));
  readln;
 End.
```

Nome della funzione

Tipo del dato fornito in output

Tipo della variabile nella funzione

Il programma permette di calcolare i valori della funzione $f(x)=2^{-x}+x^2-1$ corrispondenti ai valori della variabile x che vanno da -5 a +5, con incremento +1.
L'aggiornamento della variabile è gestito dalla struttura ciclica **For …to…do….**
A lato riporto la schermata di esecuzione del programma.
Si noti che la funzione si annulla nel punto x=0. In verità, la funzione si annulla anche in un punto interno all'intervallo]0;1[.
Con i prossimi programmi indagheremo più a fondo la questione.



Il secondo programma

Il programma che segue è un po' più ricercato. Con esso si è realizzato l'obiettivo di visualizzare i valori che la funzione di riferimento assume in un intervallo [a;b] i cui estremi sono definiti

dall'utente. I valori della variabile in cui calcolare la funzione sono decisi dall'utente stesso. Infatti, all'utente si richiedono i due estremi dell'intervallo ed il valore dx dell'incremento da assegnare alla variabile per il calcolo del valore successivo. Il processo dell'aggiornamento della variabile ha luogo finché il valore della variabile non supera l'estremo destro dell'intervallo. Il primo valore calcolato è $f(a)$. Dunque si calcolano i valori

$$f(a),\ f(a+dx),\ f(a+2dx),\ldots\ldots,\ f(a+ndx)$$

e si procede finché risulta $a+ndx \le b$

Come si diceva, l'obiettivo di questo programma è permettere di osservare come variano i valori della funzione in esame quando la variabile *x* descrive (sia pure a salti) un certo intervallo. Questo metodo, detto degli incrementi finiti, permette di indagare se una funzione continua in un certo intervallo si annulla qualche volta. I risultati della ricerca dipendono molto dal senso critico dell'operatore. Egli dovrà essere attento a definire adeguatamente l'incremento da attribuire alla variabile e saper leggere le elaborazioni ottenute in output, se vuole ottenere informazioni significative .

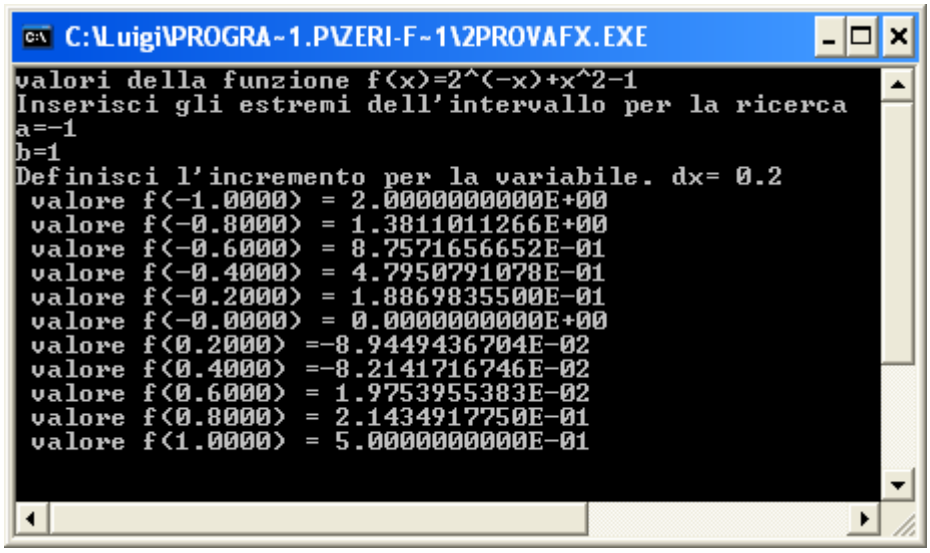
Osservazioni sul codice del programma

Il programma è strutturato con procedure. E' presente la **Procedure dati;** con cui si acquisiscono i dati che l'utente deve inserire e la **Procedure calcolo;** preposta al calcolo dei valori della funzione. All'interno di questa vi è la struttura ciclica **Repeat…Until…** che itera il calcolo finché il valore della variabile *x* non supera l'estremo destro dell'intervallo [a;b].

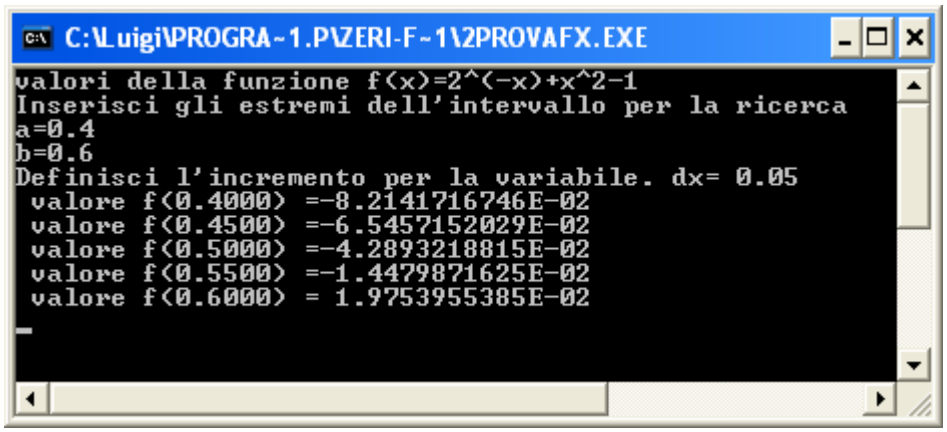
Di seguito riportiamo la codifica del programma e due videate ottenute con altrettante esecuzioni dello stesso.

```
Program Prova_fx;
 Uses Crt;
 Var a,b,x,dx:real;

 Procedure Dati;
  Begin
   Writeln('Inserisci gli estremi dell''intervallo per la ricerca');
   Write('a=');Readln(a);
   Write('b=');Readln(b);
   Write('Definisci l''incremento per la variabile. dx= ');Readln(dx);
  End;
 Function F(x:real):real;
  Begin
   F:=exp(-x*ln(2))+x*x-1;
  End; (* chiude la Function *)
 Procedure Calcolo;
  Begin
   x:=a;
   Repeat
    writeln(' valore f(',x:2:4,') =',f(x));
    x:=x+dx;
   Until (x>b);
  End; (* chiude la procedura Calcolo *)

  Begin (* Corpo principale *)
   clrscr;
   Writeln('valori della funzione f(x)=2^(-x)+x^2-1');
```

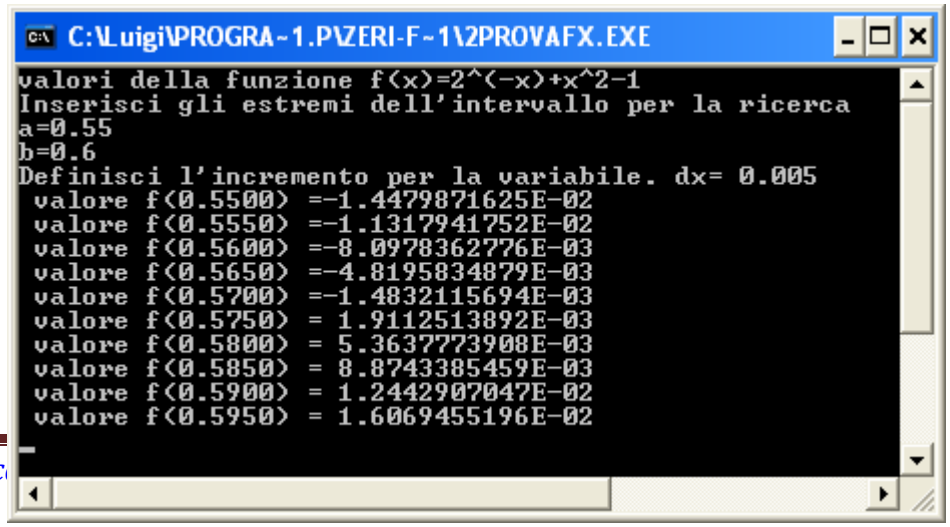
```
Dati;
Calcolo;
readln;
End.
```



Con questa prima esecuzione sono stati determinati valori assunti dalla funzione nell'intervallo [-1;1] procedendo con incremento dx=0.2 per la variabile. Dai risultati ottenuti si evince che $f(0.4)<0$ ed $f(0.6)>0$. Poiché la funzione in esame è continua, per il **teorema di esistenza degli zeri** deduciamo che internamente all'intervallo [0.4;0.6] essa si annulla almeno una volta. Detto **$x=\alpha$ il valore dello zero**, siamo interessati a cercarne un'**approssimazione**.Come fare?
Il programma ci aiuta.



E' sufficiente rilanciarlo, definire come intervallo di ricerca proprio [0.4;0.6] e fissando per l'incremento dx un valore più piccolo, per esempio dx=0.05. Lo facciamo.
A lato riportiamo la videata dell'esecuzione. Si osserva che lo zero della funzione è interno all'intervallo [0.55;0.60] perché la funzione assume agli estremi di quest'intervallo valori di segno diverso. Possiamo rilanciare il programma per affinare la ricerca. Questa volta porremo dx=0.005 .



A lato riportiamo la videata dell'elaborazione. Notiamo che lo zero cercato è interno all'intervallo [0.570:0;575]. La ricerca può continuare fino ad ottenere la precisione desiderata. Il metodo illustrato non presenta limitazioni nella ricerca. Bastano un po' di

attenzione nell'osservazione dei risultati che via via si ottengono e la pazienza di eseguire più volte il programma per **ottenere per lo zero una precisione prestabilita**,e dunque **conoscerlo con un errore inferiore ad un certo margine predefinito**. Con il successivo programma miglioreremo ulteriormente la ricerca. Cambieremo anche la strategia di ricerca.

Il terzo programma (sul metodo di ricerca dicotomico)

Con questo programma vogliamo alzare il livello nella qualità della ricerca.
Una volta acquisito che una determinata funzione continua presenta almeno uno zero internamente all'intervallo [*a*;*b*], vogliamo costruire un algoritmo che consenta al ricercatore di determinare un'approssimazione dello zero con un una precisione che egli stesso definirà. Detto x=α lo zero della funzione interno all'intervallo [*a*;*b*], l'operatore può fissare operativamente un margine superiore per lo scostamento tra α ed un valore approssimato che egli potrà determinare con l'esecuzione del programma. Il **margine di errore** è solitamente espresso da una potenza del 10 con esponente negativo. Ad esempio, si potrà richiedere di trovare un'approssimazione per lo zero con errore inferiore a 10^{-4}, vale a dire che il valore rappresentativo di α dovrà discostarsi da α per meno di un decimillesimo. Naturalmente l'operatore potrà essere più esigente e richiedere una precisione migliore. I risultati che potrà ottenere dipendono però anche dal software e dal processore matematico utilizzati. Non vogliamo addentrarci nella questione in questa sede. Ci basta aver segnalato il problema. Si tenga comunque presente che se gli estremi *a*, *b* sono razionali e lo zero della funzione è irrazionale, si potranno ottenere dello stesso solo approssimazioni che sono razionali (la semisomma di due numeri razionali è un numero razionale). Comunque, per quanto potenti possano essere le tecnologie in dotazione, i valori che i processori matematici forniscono possono avere solo un numero finito di cifre decimali e dunque anche se lo zero della funzione fosse un valore razionale periodico (semplice o misto), il processore fornirebbe solo un'approssimazione più o meno spinta. Evidentemente la questione si complica ulteriormente se il valore da determinare è irrazionale.

Per comprendere il funzionamento del programma si deve tener presente il teorema di esistenza degli zeri per una funzione continua che di seguito enunciamo.

Teorema

Sia $f:[a;b]\to R$ *una funzione reale di variabile reale continua nell'intevallo [a;b]. Se la funzione agli estremi dell'intervallo assume valori di segno discorde allora si annulla in almeno un punto interno all'intervallo.*

Osservato che l'ipotesi "la funzione agli estremi dell'intervallo assume valori di segno discorde " equivale all'affermazione che il prodotto dei valori assunti agli estremi è negativo, il teorema si può presentare nella seguente forma simbolica

$$\left(\begin{array}{l} f:[a;b]\to R \\ [a;b]\subseteq R, continua \\ f(a)\cdot f(b)<0 \end{array}\right) \Rightarrow \left(\exists c\in\,]a;b[\,:\, f(c)=0\right)$$

Il metodo di ricerca dicotomico

Il metodo implementato nel programma che segue è quello dicotomico, detto anche di bisezione. Di cosa si tratta?
Una volta acquisito che la funzione si annulla internamente all'intervallo [a;b], si potrebbe assumere come prima approssimazione dello zero x=α il valore del punto medio dell'intervallo

$$x_m=\frac{a+b}{2} \qquad (5)$$

Posto per brevità $b-a=l$, con la scelta fatta, se x_m non dovesse coincidere con α, il valore $|x_m-\alpha|$ rappresenta l'errore connesso alla scelta e risulta evidentemente

$$|x_m-\alpha|<\frac{l}{2} \qquad (6)$$

Il metodo dicotomico ci guida per ottenere approssimazioni sempre migliori dello zero, cioè approssimazioni che si discostano sempre meno dal valore α. Come fare?
Avendo supposto che $f(x_m)\neq 0$, $f(x_m)$ sarà concorde con uno solo dei due valori $f(a)$, $f(b)$. Si può presentare dunque solo uno dei due seguenti casi:
1) se $f(x_m)$ è concorde con $f(a)$, allora lo zero previsto della funzione è interno all'intervallo $[x_m;b]$. In questo caso la ricerca continua "aggiornando" il valore dell'estremo sinistro dell'intervallo; si pone $a=x_m$.
2) se $f(x_m)$ è concorde con $f(b)$, allora lo zero previsto della funzione è interno all'intervallo $[a;x_m]$. In questo caso la ricerca prosegue "aggiornando" il valore dell'estremo destro dell'intervallo; si pone $b=x_m$.
Una volta stabilito in quale dei due casi ci si ritroverà, la ricerca proseguirà determinando il valore del punto medio del nuovo intervallo nel quale calcolare il valore della funzione e confrontandolo con i valori assunti dalla stessa negli estremi dell'intervallo corrente. E' evidente che dopo ogni aggiornamento dell'intervallo di ricerca migliora l'informazione. Così, assumendo per lo zero il valore:

del primo punto medio, l'errore è inferiore a $\frac{l}{2}$;

del secondo punto medio, l'errore è inferiore a $\frac{l}{2^2}$;

……

dell'n-esimo punto medio, l'errore è inferiore a $\frac{l}{2^n}$. (7)

Poiché sussiste il limite

$$\lim_{n\to+\infty}\frac{l}{2^n}=0$$

deduciamo che si può spingere la ricerca quanto occorre fino ad ottenere un valore approssimato dello zero della funzione che differisca dal valore effettivo per meno del valore positivo $\varepsilon=10^{-k}$, con k naturale positivo, comunque prefissato. La filosofia del metodo dicotomico è tutta qui!
Possiamo aggiungere di più. Una volta noto l'intervallo iniziale [a;b] in cui cade lo zero, e fissato il margine di errore, possiamo anche **prevedere il numero massimo di passaggi che saranno necessari per ottenere l'approssimazione dello zero con la precisione desiderata**. Volendo che l'errore sia inferiore a 10^{-k}, il numero n di passaggi necessari si determina risolvendo la disequazione seguente:

$$\frac{l}{2^n}<10^{-k}$$

che risolta nell'incognita n fornisce la seguente condizione

$$n>\log_2 l+k\log_2 10 \qquad (8)$$

Scrivendo la (8) utilizzando i logaritmi in base 10 la limitazione per n è

$$n>\frac{\log_{10} l+k}{\log_{10} 2}$$

Esempio

Abbiamo precisato che la funzione $f(x) = 2^{-x} + x^2 - 1$ ammette uno zero internamente all'intervallo [0;1]. In questo caso *l*=1-0=1, k=5. Si otterrà un'approssimazione dello zero previsto con un errore inferiore a 10^{-5} con un numero di passaggi

$n > \log_2 1 + 5\log_2 10 \quad \Leftrightarrow \quad n > 16,6$

Dunque basta eseguire *n*=17 passaggi.

Riportiamo ora il codice del programma che risolve il problema illustrato.

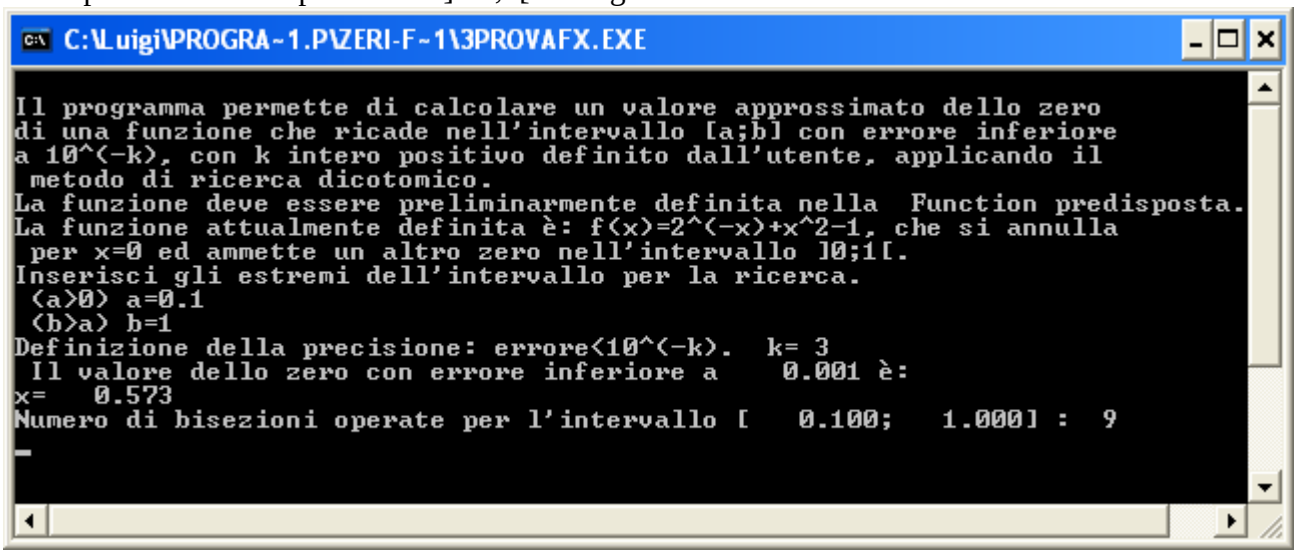
```
Program Prova_fx;
 Uses Crt;
 Var a,b,x,dx   :real;
    k         :integer;
 Procedure Avviso;
  Begin
   Clrscr;writeln;
   writeln('Il programma permette di calcolare un valore approssimato dello zero ');
   writeln('di una funzione che ricade nell''intervallo [a;b] con errore inferiore');
   writeln('a 10^(-k), con k intero positivo definito dall''utente, applicando il');
   writeln(' metodo di ricerca dicotomico.');
   write('La funzione deve essere preliminarmente definita nella ');
   writeln(' Function predisposta.');
   writeln('La funzione attualmente definita è: f(x)=2^(-x)+x^2-1, che si annulla ');
   writeln(' per x=0 ed ammette un altro zero nell''intervallo ]0;1[.');
  End;
 Procedure Dati;
  Begin
   Writeln('Inserisci gli estremi dell''intervallo per la ricerca.');
   Write(' (a>0) a=');Readln(a);
   Write(' (b>a) b=');Readln(b);
   Write('Definizione della precisione: errore<10^(-k).  k= ');Readln(k);
  End;
 Function F(x:real):real;
  Begin
   F:=exp(-x*ln(2))+x*x-1; (* Qui va definita la struttura della funzione. *)
  End; (* chiude la Function *)
 Function Controllo(a,b:real):Boolean;
  (* La Function controlla se la funzione f(x) agli estremi dell'intervallo *)
  (* [a;b] definito assume segno discorde. In caso negativo la ricerca non si effettua.*)
  Begin
    Controllo:=True;
    If (F(a)*F(b)>0) Or (F(a)*F(b)=0) then
      Begin
        Textcolor(2+blink);
        writeln('Nell''intervallo [',a:8:k,';',b:8:k,'] la funzione non verifica le ipotesi del');
        writeln(' teorema sull''esistenza degli zeri per una funzione continua. ');
        write(' Il metodo dicotomico non è applicabile.');
        Controllo:=False;
      End;
  End; (*chiude la function controllo *)
```

```
Procedure Itera_Calcolo;
 Var x1,x2,xm,error     :real;    (* variabili locali per il ciclo *)
    N_Elab            :integer; (* variabile per contare il numero delle bisezioni dell'intervallo *)
 Begin
  x1:=a;x2:=b;
  error:=(b-a)/2;N_Elab:=0;
  While (f(x1)*f(x2)<0) and (error>exp(-k*ln(10))) do
   Begin
    xm:=(x1+x2)/2;
    N_Elab:=N_Elab+1;
    If f(x1)*f(xm)<0 then x2:=xm
                     Else x1:=xm;
    error:=(x2-x1)/2;
   End;
   Writeln(' Il valore dello zero con errore inferiore a ',exp(-k*ln(10)):8:k,' è:');
   Writeln('x=',xm:8:k);
   Write('Numero di bisezioni operate per l''intervallo [',a:8:k,';',b:8:k,'] :');
   Writeln(N_Elab:3);
 End; (* chiude la procedura Itera_Calcolo *)

 Begin (* Corpo principale del programma *)
  Avviso;
  Dati;
  If Controllo(a,b) then Itera_Calcolo;
  readln;
 End.
```

Riportiamo ora le videate di tre elaborazioni del programma con cui si ottengono approssimazioni sempre migliori. Per ogni elaborazione sceglieremo un intervallo ed un margine di errore diversi, affinando la ricerca.

Prima elaborazione

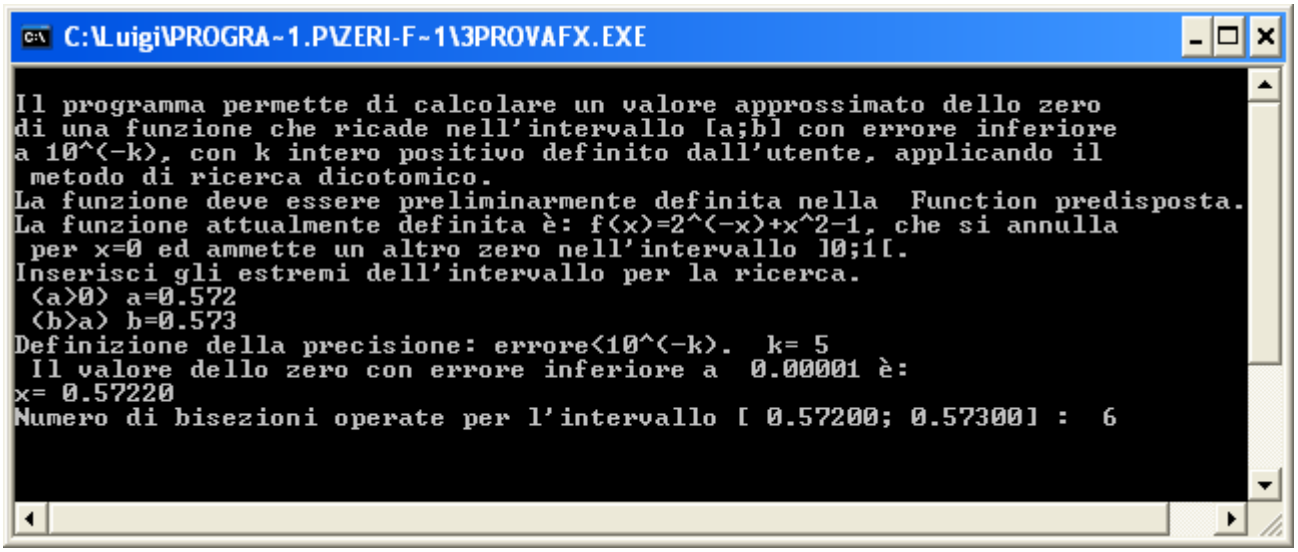
Come primo intervallo prendiamo]0.1;1[e margine di errore minore di 10^{-3}.



Il valor approssimato per lo zero è x=0.573, affetto da un errore inferiore a 10^{-3}. Sono stati necessari 9 passaggi, cioè l'intervallo iniziale di ricerca è stato diviso in due parti uguali per nove volte.

Seconda elaborazione

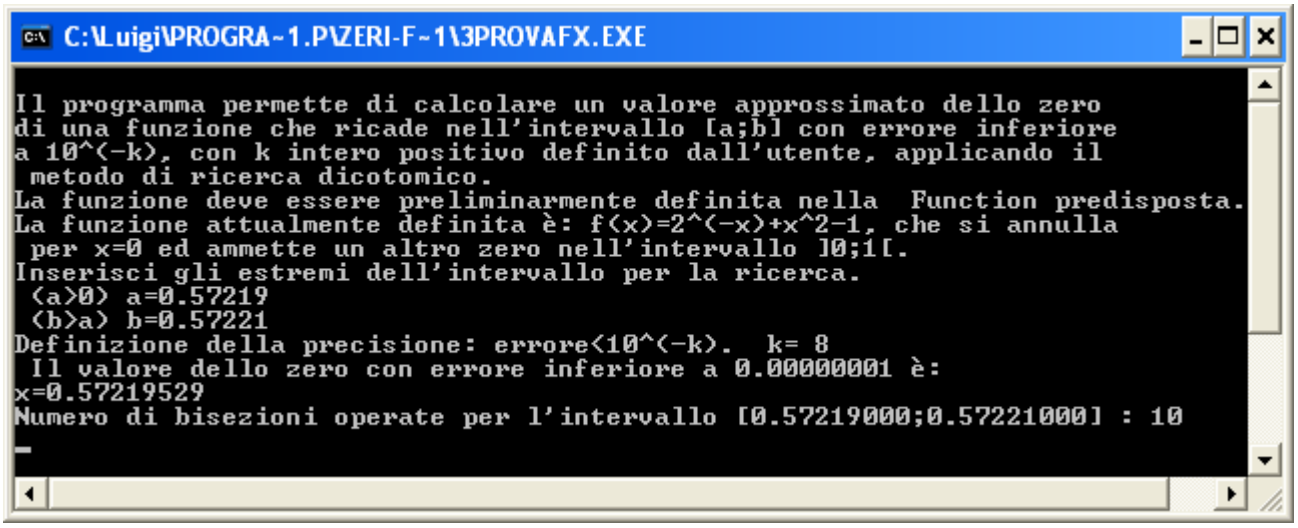
Dall'elaborazione precedente abbiamo l'informazione che lo zero cercato appartiene all'intervallo di estremi
0.573-0.001=0.572 e 0.573+0.001=0.574
Scegliamo dunque questo come secondo intervallo di ricerca e come margine superiore per l'errore 10^{-5}. Segue la videata dell'elaborazione.



L'approssimazione trovata è x=0.57220 con un errore <0.00001. Sono stati necessari 6 passaggi. Dunque lo zero x=α verifica la doppia disuguaglianza

$$0.57219<\alpha<0.57221.$$

Eseguiamo la nuova elaborazione prendendo come intervallo di partenza [0.57219;0.57221] e come margine superiore per l'errore 10^{-8}. Segue la videata dell'elaborazione.



Con dieci elaborazioni è stata ottenuta per lo zero della funzione l'approssimazione x=0.57219529, affetta da errore<10^{-8}.

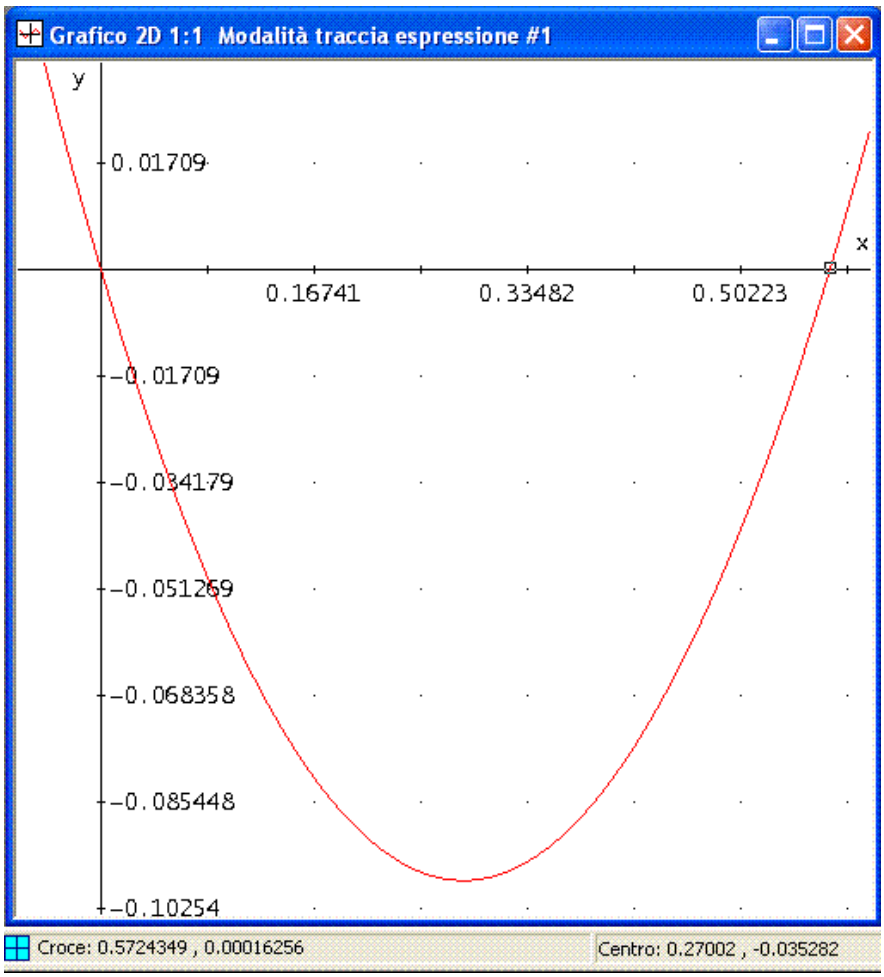
Curiosità

Se si lancia il programma scegliendo come intervallo di ricerca [0.1;1] e 10^{-8} come margine superiore per l'errore sono necessarie 26 bisezioni e si trova lo stesso valore per l'approssimazione: x=0.57219529.

Scegliendo per la ricerca l'intervallo [0.1;0.9] e margine di errore 10^{-9} si ottiene il valore approssimato x=0.5721952787 dopo 29 bisezioni.

Cosa Offre Derive?

Da diversi anni nelle scuole superiori italiane, ma anche nelle università, si utilizzano sempre più frequentemente programmi di grafica applicativi come Derive. Senza dubbio l'interfaccia grafica è stimolante e gratifica immediatamente lo studioso con grafici eleganti. Ma se si vuole sviluppare una ricerca più approfondita l'affascinante mondo grafico spesso non è sufficiente. Qui vedremo una differenza sostanziale tra la qualità dell'informazione che si può estrarre operando con un linguaggio di programmazione e quella che si può ricavare dall'osservazione di un'elaborazione grafica.
A completamento del lavoro sviluppato riportiamo la rappresentazione grafica ottenuta con Derive, V.6, della funzione che abbiamo analizzato.
Inizialmente abbiamo rappresentato la funzione su un intervallo piuttosto esteso contenente l'intervallo [0;1] per osservare quale valutazione dello zero si riesce a rilevare con l'applicazione in modalità traccia. Successivamente, per avere approssimazioni migliori dello zero della funzione, abbiamo operato degli ingrandimenti (zoom) impostando via via intervalli di rappresentazione di dimensione sempre più piccola.



L'immagine riportata a lato è quella che permette di "vedere" ancora abbastanza bene la forma del diagramma della funzione. Si possono leggere chiaramente le coordinate del puntatore, collocato apparentemente sullo "zero" della funzione che interessa a noi. Si noti che l'ordinata del punto non è zero, ma 0.00016256 e la sua ascissa vale 0.5724349.
Confrontando il valore dell'ascissa suddetta con il valore dell'approssimazione determinato con la terza esecuzione del programma, cioè x=0.57219529, possiamo notare che in quello di Derive solo le prime tre cifre decimali sono esatte.
Certamente si può migliorare il valore dell'approssimazione ottenibile con l'applicazione Derive, ma occorre restringere notevolmente l'ampiezza dell'intervallo in cui visualizzare la curva e a livello grafico non ne vale proprio la pena.
Concludiamo questa riflessione affermando che le elaborazioni numeriche ottenibili con un linguaggio di programmazione sono senza dubbio di qualità superiore rispetto alle informazioni ottenibili da elaborazioni grafiche significative.