Zeri di una funzione

Maxima è un sistema di calcolo algebrico comprendente un linguaggio di programmazione semplice dal punto di vista della sintassi ma efficace, mediante il quale è possibile descrivere algoritmi per risolvere problemi da quelli classici (di ricerca, ordinamento, etc.) a quelli di analisi numerica. Grazie alla sintassi semplificata del linguaggio, le istruzioni possono essere apprese in tempi ridotti e manipolate senza particolari difficoltà: in questo modo l'attenzione rimane focalizzata sulla risoluzione del problema oggetto di analisi senza disperdersi negli aspetti di carattere formale, ed è possibile conseguire risultati significativi con ragionevole impegno.

La programmazione consente di perseguire importanti obiettivi formativi, fra i quali:

- favorire lo sviluppo di capacità metacognitive;

- acquisire la capacità di lavorare in gruppo attraverso la realizzazione di progetti complessi, e collaborare tra pari nella correzione degli errori dei programmi;

- stimolare negli studenti spirito critico e curiosità intellettuale, permettendo loro di costruire in modo autonomo funzionalità che i vari ambienti rendono disponibili già predefinite.

E' fondamentale, ai fini di promuovere un apprendimento significativo, che la programmazione non venga trascurata, invertendo la tendenza ad abbandonarla che molti docenti hanno assunto, essenzialmente a causa della sproporzione tra lo sforzo prodigato e l'esiguità dei risultati conseguiti.

Vengono di seguito presentate alcune esperienze di programmazione, realizzate nell'ambito del triennio ad indirizzo Brocca scientifico tecnologico presso il Liceo scientifico "G. Ricci Curbastro" di Lugo(RA), che utilizzano moduli sviluppati mediante tale linguaggio, nell'intento di proporre Maxima ai docenti di Matematica come strumento per fare programmazione nell'ambito della disciplina "Matematica e Informatica".

Algoritmo delle divisioni successive per il calcolo del massimo comun divisore tra due interi a e b non nulli (classe terza)

Grafica a raster-scan: algoritmo di Bresenham per la generazione di linee (classe quarta)

Zeri di una funzione (classe quinta)

Interpolazione matematica (classe quinta)

Interpolazione statistica (classe quinta)

Metodo delle approssimazioni successive

Discende dal teorema del punto fisso di seguito enunciato.

Sia [a,b] un intervallo limitato e chiuso e sia $\Phi(x)$ una funzione definita su [a,b] che verifica le seguenti condizioni:

1. $\Phi(x)$ è continua su [a,b];
2. $\Phi(x)$ muta l'intervallo [a,b] in se', cioè $\Phi(x) \in [a,b] \ \forall \ x \in [a,b]$
3. $\Phi(x)$ è derivabile in [a,b] e per una opportuna costante $\lambda<1$ risulta $|\Phi'(x)|<\lambda$

Allora per ogni scelta di $x_0 \in [a,b]$ la successione definita da $x_n=\Phi(x_{n-1})$ converge all'unico punto fisso ξ della funzione $\Phi(x)$, cioè all'unica soluzione dell'equazione $\xi=\Phi(\xi)$.

Consideriamo la funzione

$$\Phi(x)=e^{-x}$$

```
(C1)plot2d([exp(-x),x],[x,-2,4],[y,-1,4]);
```

Su di un intervallo $[a,b]$ con $a>0$ la funzione verifica le condizioni del teorema.

Il punto fisso della funzione è l'intersezione con la retta $y=x$; attivando il comando ZOOM e cliccando col mouse in prossimità del punto è possibile dedurne le coordinate.

Definiamo un sottoprogramma avente come parametri di ingresso: la funzione, la variabile x da considerare, l'iterato iniziale x_0 , la precisione eps a meno della quale determinare l'approssimazione della soluzione, il numero massimo di iterazioni da eseguire che assicura lam, terminazione del ciclo iterativo anche nel caso in cui il metodo non converga. Vengono utilizzate le variabili locali k, x_n , xprec; L'istruzione RETURN determina l'uscita dal sottoprogramma.

```
(C2)  apprsucc(funz,var,x0,eps,numiter):=block([xn,xprec,k],  define(f(var),funz),  k:0,
      xn:0, do (k:k+1, xprec:xn, xn:f(xprec), if(abs(xprec-xn)¡eps or k¿numiter) then
      return(xn) ))$
```

```
(C2)h:exp(-x);
```

```
(D3) %E^-x
```

```
(C4)setcheck:[xn]$
```

```
(C5)apprsucc(h,x,0.0,0.01,10);
```

```
 xn set to 0.0
xn set to 1.0
xn set to 0.36787944117144
xn set to 0.69220062755535
xn set to 0.50047350056364
xn set to 0.6062435350856
xn set to 0.54539578597503
```

xn set to 0.57961233550338
xn set to 0.56011546136109
xn set to 0.57114311508018
xn set to 0.56487934739105

(D5) 0.56487934739105

Il valore di xn ad ogni iterazione viene visualizzato due volte perchè compare in due assegnazioni

```
(C6)kill(all);
```

(D0) DONE

Modifichiamo il sottoprogramma affinchè stampi anche il numero di iterazioni eseguite:

```
(C1) apprsucc(funz,var,x0,eps,numiter):=block([xn,xprec,k], define(f(var),funz), k:0, xn:0,
     do ( k:k+1, xprec:xn, xn:f(xprec), if(abs(xprec-xn)¡eps or k¿numiter) then
      (print("xn=",xn," in ",k,"iterazioni"),return(fine) ) ) )$
```

```
(C5) h:exp(-x);
```

(D2) $\%E^{-x}$

```
(C3) setcheck:[xn]$
```

```
(C4) apprsucc(h,x,0.0,0.001,100);
```

xn set to 0.0

xn set to 1.0

xn set to 0.36787944117144

xn set to 0.69220062755535

xn set to 0.50047350056364

xn set to 0.6062435350856

xn set to 0.54539578597503

xn set to 0.57961233550338

xn set to 0.56011546136109

xn set to 0.57114311508018

xn set to 0.56487934739105

xn= 0.56487934739105 in 14 iterazioni

(D4) fine

Metodo di bisezione

```
(C1) segno(x):=block([], if x=0 then return(0) else (if x¿0 then return(1) else return(-1) ) )$

(C2) bisez(funz,var,a,b,eps,numiter):=block([fa,fb,fc,c,l,k], define(f(var),funz), fa:f(a), fb:f(b), l:abs(b-a), k:0, do ( k:k+1, l:l/2.0, c:ev(a+(b-a)/2.0,float), fc:f(c), if (fc=0.0 or l¡eps or k¿numiter) then (print("c=",c," in ",k," iterazioni"), return(fine)) else (if (segno(fa)*segno(fc)¿0) then (a:c, fa:fc) else (b:c, fb:fc) ) ))$

(C3)h:x*x-78.8;

(D3) x^2 - 78.8

(C4)bisez(h,x,6.0,12.0,0.0001,100);

c=8.876936408468858 in 101 iterazioni

(D4) fine
```

Bibliografia

1) G.Casadei, M.Lacchini, *Il Problem Solving algoritmico*, Atti didamatica 2004, Omniacom, Ferrara, 2004

2) D.Bini, M.Capovani, O.Menchi, *Metodi numerici per l'algebra lineare* , Zanichelli, Bologna, 1988

3) S. Harrington, *Computer Graphics*, McGraw Hill, 1983

4) F.Fontanella, A.Pasquali, *Calcolo numerico* , Pitagora, Bologna, 1984

5) I.Galligani, *Elementi di analisi numerica*, Zanichelli, Bologna, 1987

6) A.Paoluzzi, *Informatica grafica*, La Nuova Italia Scientifica, 1988

7) D.F. Rogers, *Procedural Elements for Computer Graphics*, McGraw Hill, USA, 1985, ed.italiana Tecniche Nuove, Milano, 1988

8) J. Stoer, *Einfuhrung in die Numerische Mathematik*, Springer Verlag, Berlin-Heidelberg-New York,1972, ed. italiana Zanichelli, Bologna, 1974