Interpolazione e approssimazione polinomiale

prof. Michele Impedovo

La matematica è una parte della fisica.
La matematica è quella parte della fisica
dove gli esperimenti costano poco.
Vladimir Arnold

"Sull'insegnamento della matematica"
Russian Math. Surveys
traduzione italiana su Punti Critici II 3

1. Introduzione

I problemi che si vogliono qui affrontare sono due, ben distinti ma strettamente correlati:

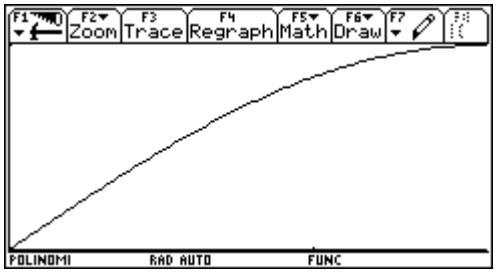
1) Dati n+1 punti $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ determinare un polinomio $p(x)$ di grado (al più) n tale che "soddisfi" i punti dati, cioè tale che

$$\begin{cases} p(x_0) = y_0 \\ p(x_1) = y_1 \\ \ldots \\ p(x_n) = y_n \end{cases}$$

2) Data una qualsiasi funzione $f(x)$ continua su un intervallo I=[a, b], determinare la "miglior" approssimazione polinomiale di grado n su I.

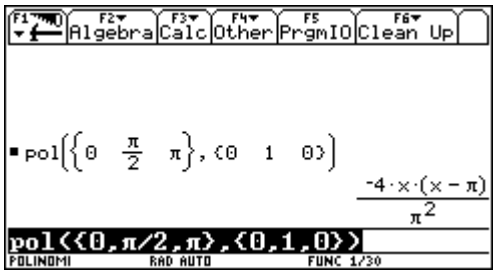
Lo strumento utilizzato è la calcolatrice simbolica TI-92, nell'ambito della sperimentazione LABCLASS della Direzione Generale Classica del MPI.

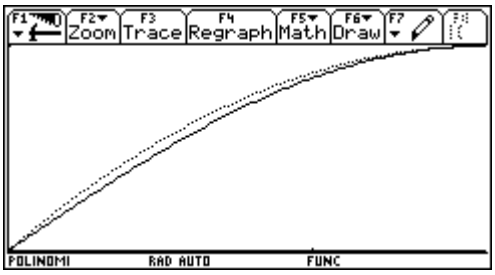
Il problema è nato un po' per gioco e un po' per sfida quando in classe (terza liceo scientifico), durante una lezione che aveva per oggetto il grafico di sin(x) nell'intervallo [0,π/2],



uno studente si era ostinato a sostenere che quella curva fosse un arco di parabola. Utilizzando un programma che avevamo sviluppato qualche tempo prima (il programma pol(xx,yy) prende in ingresso la lista delle ascisse e la lista delle ordinate di n+1 punti, e fornisce in uscita il polinomio di grado n) non è stato difficile mostrare allo studente che né la parabola per i punti

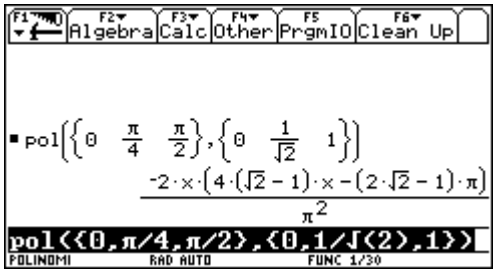
$$(0,0),\ \left(\frac{\pi}{2},1\right),\ (\pi,0)$$

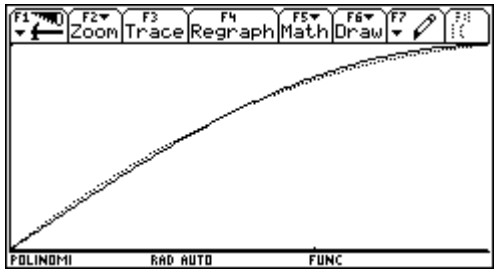




né la parabola per i punti

$$(0,0),\ \left(\frac{\pi}{4},\frac{1}{\sqrt{2}}\right),\ \left(\frac{\pi}{2},1\right)$$





coincidono con il grafico di sin(x).
Più in generale: nessuna funzione algebrica coincide con una funzione trascendente su un intervallo. È nato allora un problema più generale: se nessuna parabola può sovrapporsi al grafico di una funzione trascendente, qual è la funzione del tipo

$$x \to ax^2+bx+c$$

che approssima sin(x) su $[0,\pi/2]$ "meglio" di qualunque altra?

L'introduzione di strumenti informatici nell'insegnamento della matematica e in particolare l'utilizzo di CAS (*Computer Algebra System*) permette anche nell'insegnamento secondario di affrontare problemi, come questo, di una certa complessità; gli strumenti di programmazione offerti dalla TI-92 consentono di svolgere una attività di ricerca che favorisce il nascere di congetture, prove ed errori, verifiche e falsificazioni.
Quello che segue non vuole essere in nessun modo un saggio di matematica, ma solo un ingenuo e lacunoso resoconto dei risultati raggiunti con gli studenti (per esempio manca il tema della maggiorazione dell'errore). Sono risultati classici e ben noti agli specialisti; nella scuola secondaria tuttavia non vengono solitamente trattati. L'esigenza di aggiornare i curriculum di matematica (e di conseguenza le prove di valutazione) ci ha spinto a trattare questo problema nella prospettiva di proporre nuove forme di verifiche scritte, più aperte di quelle tradizionali, in cui sia lasciato un certo spazio alla congettura, alla ricerca e, perché no, alla creatività.

2. Interpolazione

Siano dati n+1 punti di coordinate (x_i, y_i), i=0, 1,…, n. Come dimostreremo tra breve, se le ascisse dei punti sono tutte distinte allora esiste ed è unico il polinomio $p_n(x)$ di grado n (al più n, nel seguito tralasceremo questa precisazione) che soddisfa i punti dati.
Si tratta di un problema classico, che ammette un gran numero di procedure risolutive.
Alcune di esse si prestano in modo particolare ad essere implementate in un programma; vogliamo occuparci di questo aspetto, quello più strettamente algoritmico. Si propongono tre metodi (e tre programmi), tutti relativamente semplici (cioè alla portata di uno studente di triennio) ma al contempo significativi per risolvere il problema. In ciascuno dei tre programmi vengono fornite in ingresso due liste (le ascisse e le ordinate dei punti) e in uscita si ottiene il polinomio richiesto.

2.1 Matrici e sistemi lineari

Determinare il polinomio di grado n che soddisfa n+1 punti (x_i, y_i), i=0, 1,…, n significa risolvere il sistema lineare

$$\begin{cases} a_n x_0^n + a_{n-1} x_0^{n-1} + \ldots + a_1 x_0 + a_0 = y_0 \\ a_n x_1^n + a_{n-1} x_1^{n-1} + \ldots + a_1 x_1 + a_0 = y_1 \\ \ldots \\ a_n x_n^n + a_{n-1} x_n^{n-1} + \ldots + a_1 x_n + a_0 = y_n \end{cases}$$

nelle incognite $a_0, a_1, \ldots, a_n$. In notazione matriciale:

$$\begin{bmatrix} x_0^n & \dots & x_0 & 1 \\ x_1^n & \dots & x_1 & 1 \\ & \ddots & & \\ x_n^n & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ \dots \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ \dots \\ y_{n-1} \\ y_n \end{bmatrix}$$

$$\mathbf{Ma} = \mathbf{b},$$

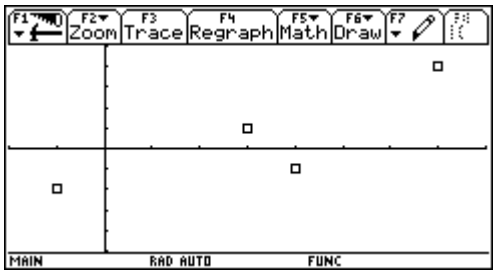
dove **M** è la matrice dei coefficienti, **a** è il vettore colonna delle incognite, **b** è il vettore colonna dei termini noti. Non è difficile dimostrare che il determinante della matrice **M** è uguale al prodotto di tutti i fattori $(x_h - x_k)$ con $h \neq k$; se le ascisse sono a due a due distinte allora tale determinante è sempre diverso da 0 e il sistema ammette una ed una sola soluzione: il polinomio richiesto esiste ed è unico.

La TI-92, come qualunque CAS, possiede una funzione predefinita, simult(M,b), che prende in ingresso una $n \times n$ matrice **M**, un $n \times 1$ vettore **b**, e fornisce in uscita il vettore **a** delle soluzioni. La calcolatrice viene qui usata come *black box*, lo studente sa già risolvere un sistema lineare con carta e penna.

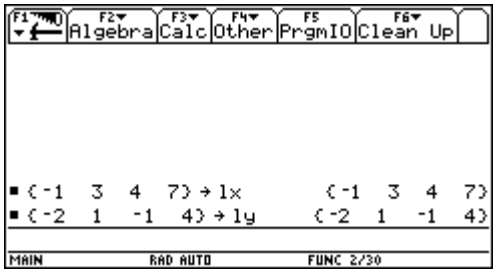
Illustriamo passo-passo il procedimento mediante un esempio: vogliamo calcolare il polinomio di terzo grado che soddisfi i quattro punti

$$(-1, -2), (3,1), (4,-1), (7,4).$$

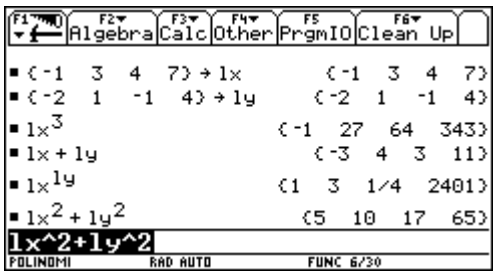
Eccoli rappresentati nel rettangolo di visualizzazione $[-2,8] \times [-5,5]$.



Definiamo innanzitutto le liste delle ascisse e delle ordinate, e memorizziamole nelle variabili lx e ly.



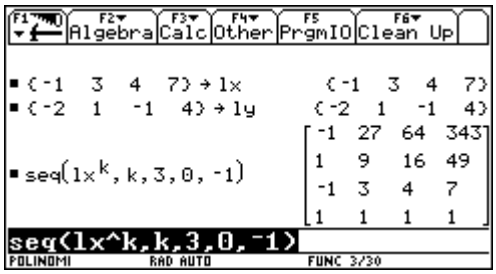
La TI-92 possiede una buona capacità di gestione delle liste: in particolare una operazione su una lista viene tradotta nell'operazione su ciascun elemento della lista. La figura seguente mostra qualche esempio.



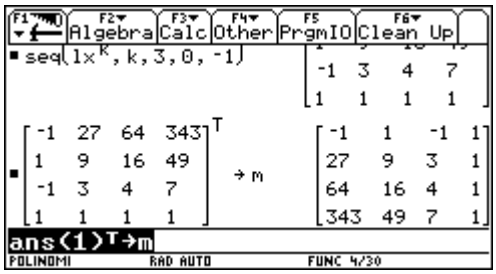
Costruiamo dunque la matrice dei coefficienti: la prima colonna è data da lx^3, la seconda colonna da lx^2, e così via. Il comando

seq(lx^k,k,3,0,–1)

(del tutto analogo al *vector* di DERIVE) costruisce, per righe, le successive potenze decrescenti delle ascisse.



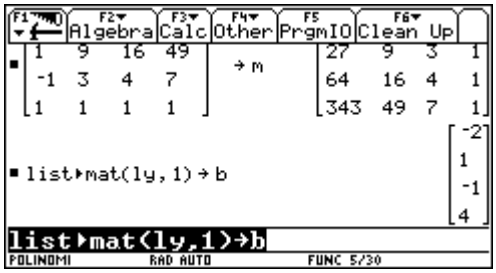
La matrice dei coefficienti **M** si ottiene scambiando le righe e le colonne, cioè è la *trasposta* della matrice precedente.



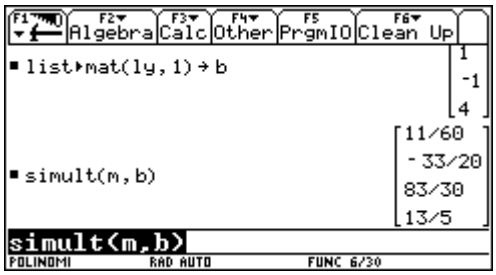
Poiché la TI-92 distingue tra liste e vettori (i vettori non sono altro che matrici a una riga, oppure una colonna), per costruire il vettore dei termini noti dobbiamo utilizzare il comando

list→mat(ly,1);

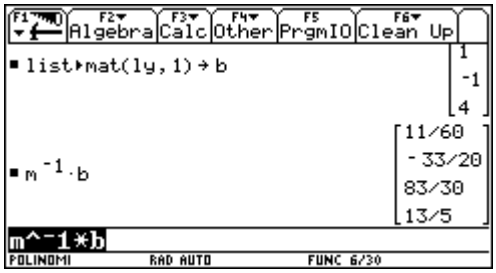
il parametro 1 indica il numero di elementi per riga.



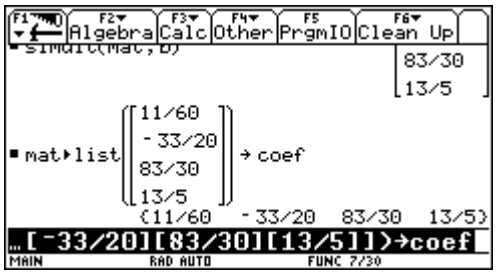
Finalmente risolviamo il sistema.



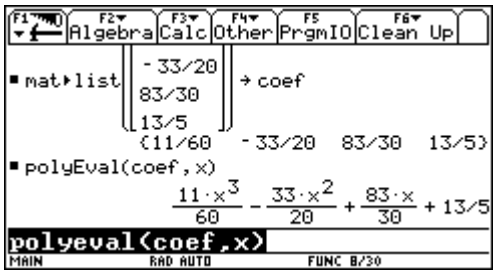
Si può mostrare agli studenti che si ottiene lo stesso risultato moltiplicando la matrice inversa di **M** per **b**.



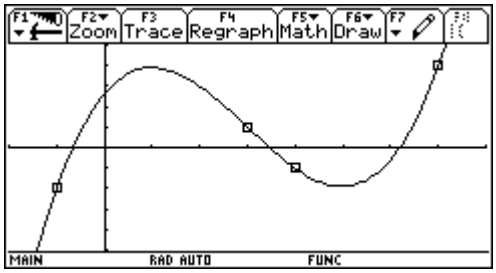
Ora trasformiamo i coefficienti in lista:



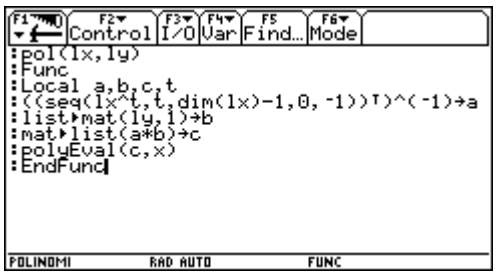
Calcoliamo infine il polinomio interpolatore mediante il potente comando polyeval(l,x) che prende in ingresso una lista di n+1 elementi e una variabile e fornisce in uscita il polinomio di grado n nella variabile data i cui coefficienti sono, ordinatamente, quelli della lista.

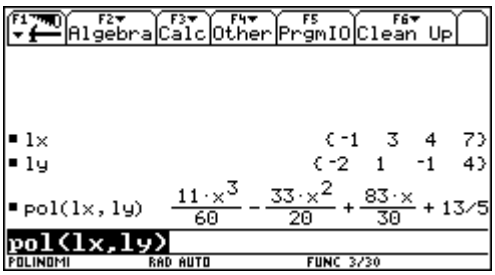


Verifichiamo graficamente che il polinomio $p(x)$ soddisfi le condizioni poste.



Il semplice programma pol(lx,ly), che implementa il procedimento ora visto, non fa che generalizzare i vari passaggi.





2.2 Il metodo di Lagrange

Un altro metodo per ottenere il polinomio interpolatore di n+1 punti, più efficiente del precedente dal punto di vista algoritmico, si deve a Lagrange (Joseph-Louis Lagrange 1736-1813). Siano (x_0, y_0), …, (x_n, y_n) gli n+1 punti dati, con l'ipotesi

se $h \neq k$ allora $x_h \neq x_k$.

Chiamiamo *polinomi di Lagrange* quei polinomi $L_k(x)$ di grado k, per $k = 0, 1, \ldots, n$ che valgono 0 in tutti gli x_i se $i \neq k$, e valgano 1 in x_k. Il generico polinomio di Lagrange ha quindi la forma

$$L_k(x) := a_k (x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)$$

dove a_k, affinché $L_k(x_k)=1$ deve valere

$$a_k = \frac{1}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}.$$

Dunque, per ogni $k = 0, 1, \ldots, n$ definiamo

$$L_k(x) := \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} = \prod_{i=0,\, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}.$$

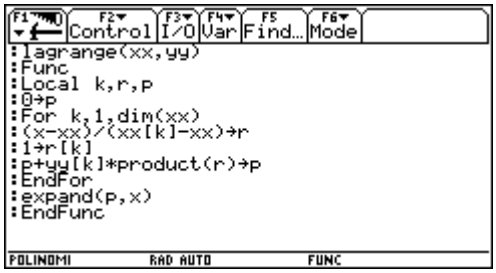
È immediato dimostrare che il polinomio di grado n

$$p(x) := \sum_{i=0}^{n} y_i L_i(x)$$

interpola gli n+1 punti dati. Infatti per ogni x_k l'unico termine non nullo della sommatoria è

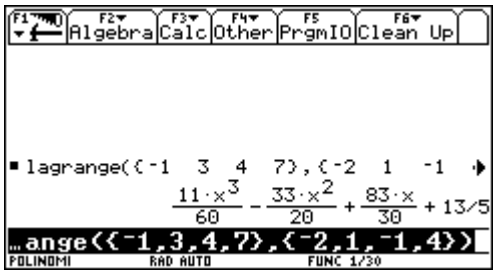
$$p(x_k) = y_k L_k(x_k) = y_k.$$

Costruiamo il programma lagrange(xx,yy) che, come il precedente, prende in ingresso le liste di ascisse e ordinate e fornisce in uscita il polinomio interpolatore.



Mettiamo alla prova il programma con l'esempio dei quattro punti già trattato:

(−1, −2), (3,1), (4,−1), (7,4).



2.3 Il metodo di Newton

È noto come "metodo di Newton" (Isaac Newton 1643-1727) l'algoritmo più diretto e semplice (anche con carta e penna!) per determinare il polinomio di grado n che interpola n+1 punti. Il pregio di questo algoritmo consiste nel fatto che se si aggiunge un nuovo punto (e quindi si aumenta di 1 il grado del polinomio) non occorre, come per gli altri metodi, ricominciare tutto da capo, è sufficiente sommare il nuovo contributo.

Siano $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ i punti dati. Il polinomio costante

$$p_0(x) = y_0$$

soddisfa il primo punto (vale y_0 in x_0) ma non il secondo; allora aggiungiamo a p_0 un nuovo termine, che valga 0 in x_0 (in modo da salvaguardare il risultato precedente), ottenendo un polinomio p_1:

$$p_1(x) = y_0 + t_1(x - x_0)$$

Imponendo $p_1(x_1) = y_1$ si determina t_1. Ora $p_1(x)$ soddisfa i primi due punti ma non il terzo. Dobbiamo aggiungere un termine che valga 0 sia in x_0 che in x_1:

$$p_2(x) = y_0 + t_1(x - x_0) + t_2(x - x_0)(x - x_1)$$

e imporre

$$p_2(x_2) = y_2$$

per determinare t_2, e così via fino al polinomio $p_n(x)$.
Riprendiamo ancora l'esempio dei quattro punti

$$(-1, -2),\ (3,1),\ (4,-1),\ (7,4).$$

Otteniamo successivamente:

$$p_0(x) = -2$$

$$p_1(x) = -2 + t_1(x+1)$$

$$p_1(3) = -2 + 4t_1 = 1 \quad \Rightarrow \quad t_1 = \frac{3}{4}$$

$$p_2(x) = -2 + \frac{3}{4}(x+1) + t_2(x+1)(x-3)$$

$$p_2(4) = -2 + \frac{15}{4} + 5t_2 = -1 \quad \Rightarrow \quad t_2 = -\frac{11}{20}$$

$$p_3(x) = -2 + \frac{3}{4}(x+1) - \frac{11}{20}(x+1)(x-3) + t_3(x+1)(x-3)(x-4)$$

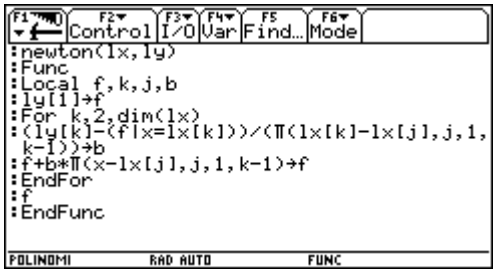
$$p_3(7) = -2 + 6 - \frac{88}{5} + 96t_3 = 4 \quad \Rightarrow \quad t_3 = \frac{11}{60}.$$

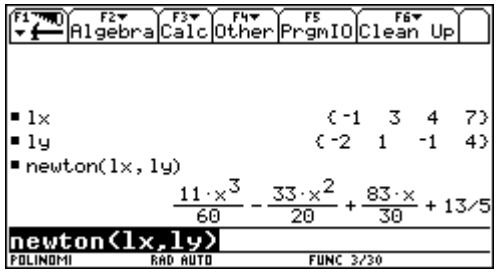
Quindi

$$p_3(x) = -2 + \frac{3}{4}(x+1) - \frac{11}{20}(x+1)(x-3) + \frac{11}{60}(x+1)(x-3)(x-4)$$

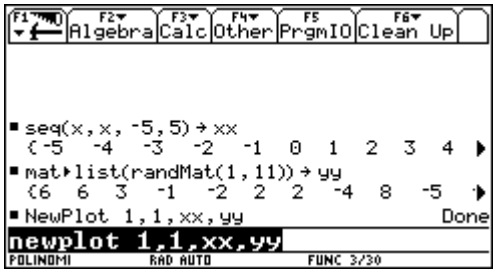
$$p_3(x) = \frac{11}{60}x^3 - \frac{33}{20}x^2 + \frac{83}{30}x + \frac{13}{5}.$$

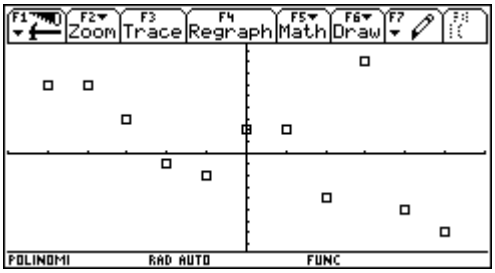
Ecco l'implementazione dell'algoritmo.

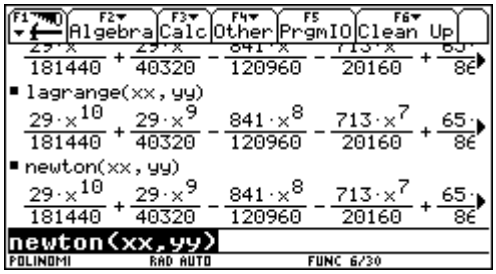


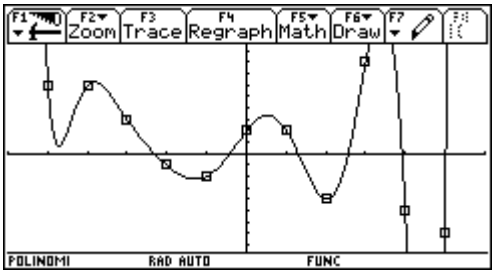


I tre algoritmi hanno diversa efficienza computazionale: quello di Newton è nettamente il migliore. Misuriamo con un cronometro i tempi di calcolo sulla TI-92 per interpolare 11 punti mediante un polinomio di grado 10.









pol(xx,yy): 23 s
lagrange(xx,yy): 17 s
newton(xx,yy): 8 s

2.4 Interpolazione di una funzione f(x) continua su [a, b]

Ora che sappiamo costruire polinomi per un certo numero di punti possiamo scegliere come ordinate dei punti i valori assunti da una qualunque funzione. Consideriamo una generica funzione continua su un intervallo $[a, b]$. Dividiamo (per esempio) l'intervallo $[a, b]$ in n parti uguali di lunghezza $\Delta x = \dfrac{b-a}{n}$ mediante i punti di suddivisione equispaziati

$$x_0 = a, \qquad x_1 = a+\Delta x, \qquad x_2 = a+2\Delta x, \qquad \ldots, \qquad x_n = a+n\Delta x = b.$$

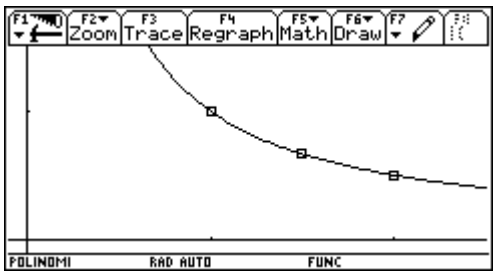
Il polinomio che soddisfa gli $n+1$ punti

$$\big(x_0, f(x_0)\big), \big(x_1, f(x_1)\big), \ldots, \big(x_n, f(x_n)\big)$$

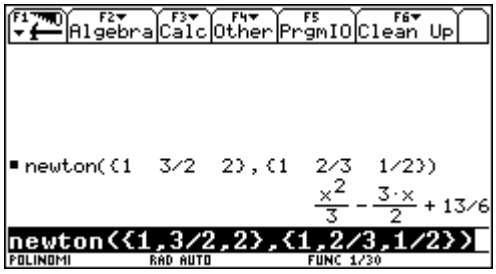
è un *polinomio interpolatore* (di grado n) di $f(x)$ su $[a, b]$.

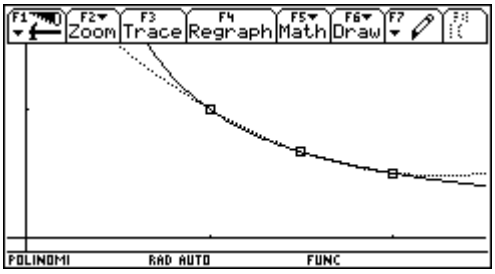
Per esempio, consideriamo la funzione $f(x)=1/x$ sull'intervallo [1, 2] e i 3 punti

$$(1, 1), \left(\frac{3}{2}, \frac{2}{3}\right), \left(2, \frac{1}{2}\right).$$



La parabola interpolatrice $p_2(x)$ per questi tre punti è quella tratteggiata nel grafico, e costituisce già (sull'intervallo dato!) una buona approssimazione.

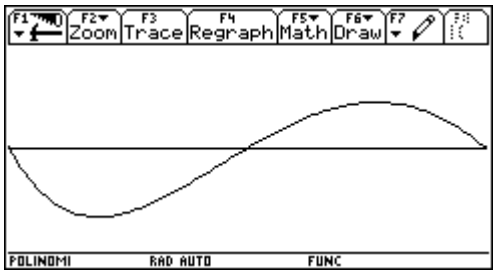




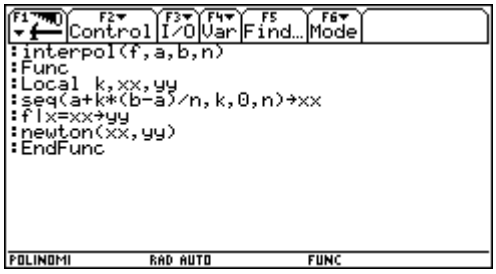
Possiamo valutare meglio la bontà dell'approssimazione osservando il grafico della *curva d'errore*

$$f(x) - p_2(x)$$

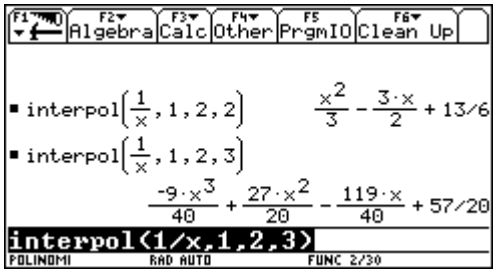
nel rettangolo $[1, 2]\times[-0.02, 0.02]$.

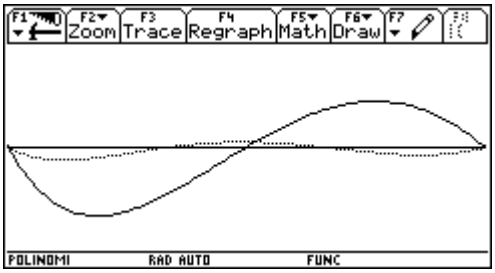


Utilizzando uno dei programmi già costruiti per la determinazione del polinomio per n+1 punti (per esempio newton) risulta facile implementare un programma che prenda in ingresso una funzione $f(x)$, gli estremi un intervallo $[a, b]$, un numero naturale n e fornisca in uscita il polinomio di grado n che interpola $f(x)$ sugli n+1 punti equispaziati compresi tra a e b.



Calcoliamo il polinomio interpolatore di terzo grado $p_3(x)$ di $1/x$ su $[1, 2]$ e confrontiamo $f(x) - p_2(x)$ con $f(x) - p_3(x)$ nello stesso rettangolo $[-1, 2]\times[-0.02, 0.02]$.



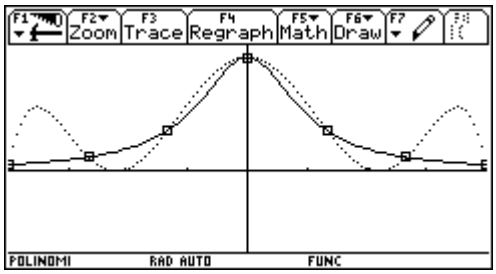


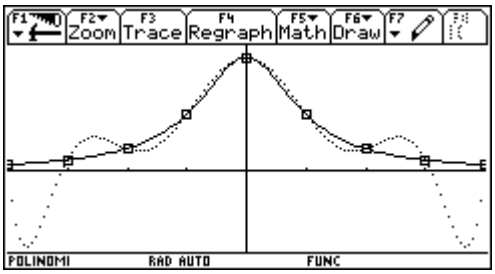
L'approssimazione di p_3, è visibilmente migliore di quella di p_2. Si potrebbe congetturare che al crescere di n il polinomio p_n si avvicina indefinitamente a f. Tuttavia questo non è vero in generale: esiste il classico *controesempio di Runge* (Carle Runge, 1856-1927)

$$f(x) := \frac{1}{1+x^2}$$

per il quale il polinomio interpolatore, all'aumentare del grado n (e quindi dei punti in comune con $f(x)$) presenta un comportamento patologico, e anziché avvicinarsi a f se ne allontana indefinitamente. Ecco il grafico di $\frac{1}{1+x^2}$ nell'intervallo $[-4, 4]$ confrontato prima con con p_6 e poi

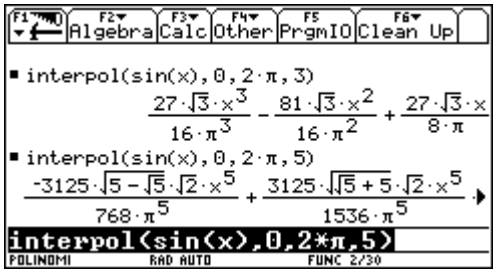
con p_8 (i grafici dei polinomi sono tratteggiati). Il rettangolo di visualizzazione è $[-4, 4]\times[-0.7, 1.1]$.

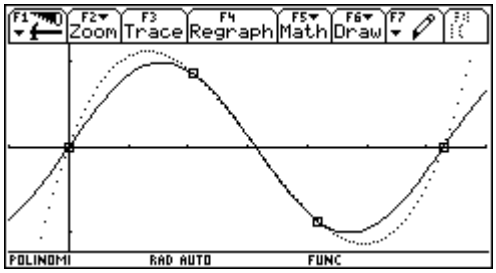


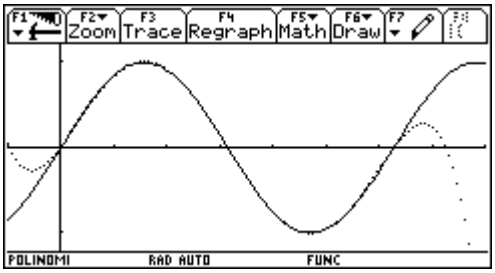


L'approssimazione migliora vicino a 0 ma peggiora agli estremi dell'intervallo; tale comportamento è amplificato al crescere di n. Questo è dovuto al fatto che $\frac{1}{1+x^2}$ è una funzione continua in **R**, ma in **C** presenta due singolarità nei punti $-i$ e i, entrambi interni al cerchio di raggio 4.

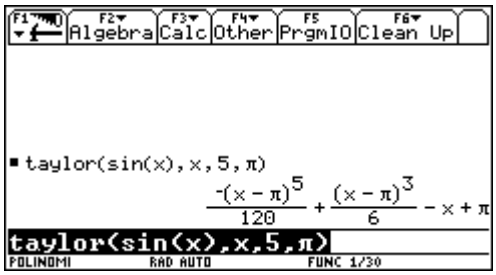
Vediamo un altro esempio: la funzione $f(x)$:=sin(x) nell'intervallo $[0, 2\pi]$. Calcoliamo i polinomi interpolatori di terzo e quinto grado.

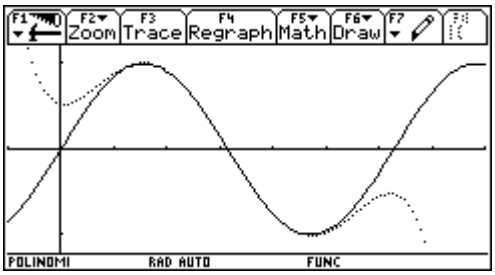




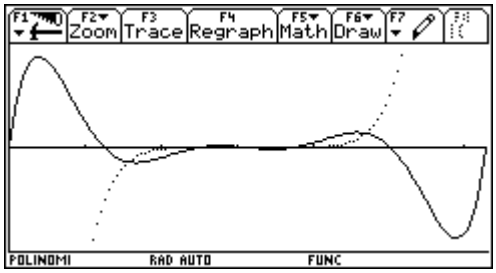


Il polinomio di quinto grado $p_5(x)$ è molto vicino a sin(x) in $[0, 2\pi]$.
Potrebbe essere interessante confrontare p_5 con T_5, il polinomio di Taylor di ugual grado con centro in π.





Il polinomio T_5 si allontana visibilmente da sin(x) agli estremi dell'intervallo $[0, 2\pi]$. Possiamo confrontare le curve d'errore sin(x)$-p_5(x)$ e sin(x)$-T_5(x)$ (linea tratteggiata). Il rettangolo di visualizzazione è $[0, 2\pi]\times[-0.03, 0.03]$.



L'aderenza di T_5 è migliore di p_5 soltanto vicino al centro π.

3. Approssimazione

Affrontiamo ora il secondo problema: abbiamo una funzione f continua su $[a, b]$, vogliamo costruire un polinomio che l'approssimi al meglio.
Vale la pena di citare subito due risultati fondamentali, che danno senso e struttura al problema.

3.1 I teoremi di Weierstrass e Bernstein

Teorema di Weierstrass (Karl Weierstrass 1815-1897). Data una funzione $f(x)$, continua su un intervallo $[a, b]$, per ogni $\varepsilon>0$ esiste un polinomio $p(x)$ tale che

$$\left| f(x) - p(x) \right| < \varepsilon$$

per ogni $x \in [a, b]$.

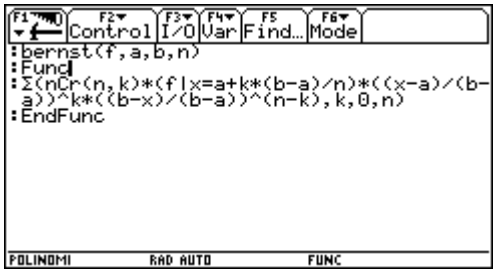
Questo teorema è potentissimo: la sola ipotesi di continuità garantisce l'esistenza di un polinomio che è vicino a f quanto si vuole.
Il secondo risultato è complementare al primo: si tratta di un teorema costruttivo, che esibisce una successione di polinomi che converge a f, e la cui dimostrazione è quindi implicitamente una dimostrazione del teorema di Weierstrass.

Teorema di Bernstein (Sergi Natanovich Bernstein 1880-1968). La successione di polinomi (*polinomi di Bernstein*)

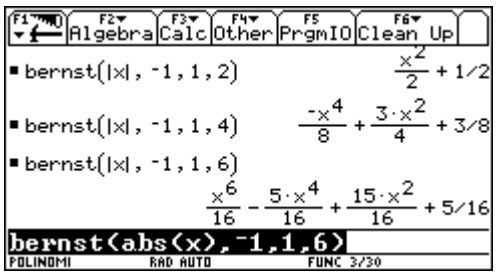
$$B_n(x) := \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) x^k (1-x)^{n-k}$$

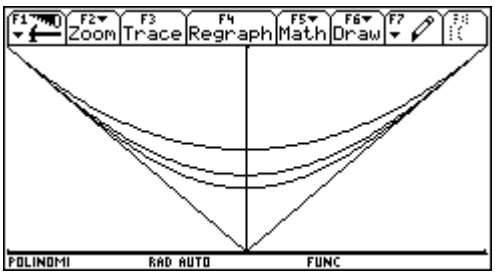
converge a f su $[a, b]$, nel senso che per ogni $\varepsilon>0$ esiste un naturale N tale che per ogni $n>N$ e per ogni $x \in [a, b]$ risulta $\left| f(x) - B_n(x) \right| < \varepsilon$.
Per la dimostrazione del teorema di Bernstein e del teorema di Weierstrass si veda ad esempio [RA], § 2.1.

Il programma bernstein(f,a,b,n) fornisce il polinomio di grado n della successione.



Vediamo tale programma alle prese con una funzione continua ma non derivabile su [−1, 1]: $f(x) := |x|$.





L'importanza teorica del teorema di Bernstein è evidente; a differenza dei polinomi di Taylor (f deve essere n volte differenziabile) è qui sufficiente la continuità di f. Mediante i polinomi di Bernstein è quindi possibile approssimare funzioni non derivabili. Tuttavia i polinomi di Bernstein non hanno grande efficienza computazionale, poiché in generale convergono lentamente a $f(x)$.

3.2 La distanza tra due funzioni

Torniamo alla domanda iniziale: qual è il "miglior" polinomio $p(x)$ di grado n che approssima una data funzione $f(x)$ continua su un intervallo $[a,b]$? Naturalmente dobbiamo dare la definizione di "miglior polinomio", e abbiamo diverse possibilità (addirittura infinite), a seconda di quale strumento adottiamo per misurare la "distanza" tra due funzioni. Scelta una distanza, il miglior polinomio di grado n che approssima $f(x)$ in $[a, b]$ è il polinomio $p(x)$ (esiste, è unico?) tale che la distanza tra $f(x)$ e $p(x)$ sia minima tra tutti i polinomi di grado n.
In effetti quelle che definiremo sono delle vere e proprie distanze tra elementi dello spazio vettoriale S delle funzioni continue su $[a, b]$, nel senso che per ogni $f, g, h \in S$ risultano soddisfatte le proprietà

1. $d(f, g) = d(g, f)$
2. $d(f, g) = 0$ se e solo se $f=g$
3. $d(f, g) \le d(f, h) + d(h, g)$

Un modo molto naturale di procedere potrebbe essere quello di misurare la distanza tra due funzioni $f(x)$ e $p(x)$ su $[a,b]$ mediante l'integrale del valore assoluto della differenza:

$$d_1(f,p) := \int_a^b |f(x) - p(x)|\,dx$$

Allora il polinomio "più vicino" a f sarà quello che (ammesso che esista, che sia unico), tra tutti i polinomi di grado n, minimizza $d_1(f,p)$.

Un altro modo di misurare la distanza tra due funzioni su un intervallo $[a, b]$ è quello di trasportare nel continuo il metodo *dei minimi quadrati*, cioè considerare, al posto della somma dei quadrati degli scarti da n punti (x_i, y_i), l'integrale del quadrato di $f(x)-p(x)$ su $[a, b]$

$$d_2(f,p) := \sqrt{\int_a^b (f(x) - p(x))^2\,dx}\,,$$

e scegliere come miglior polinomio di grado n quello che minimizza tale integrale.

Una terza scelta per definire la distanza tra due funzioni è quella già vista nei teoremi di Weierstrass e Bernstein: si chiede che $f(x)-p(x)$, per ogni $x \in [a, b]$, non superi, in valore assoluto, un certo errore. Si definisce così la distanza tra f e p nel seguente modo:

$$d_{\max}(f,p) := \max_{x\in[a,b]}\left\{\left|f(x)-p(x)\right|\right\}$$

È interessante notare che le tre distanze così definite sono tutte casi particolari di una stessa distanza definita in funzione di un parametro α:

$$d_\alpha(f,p) := \left(\int_a^b \left|f(x)-p(x)\right|^\alpha dx\right)^{\frac{1}{\alpha}}$$

Con α=1 si ottiene d_1, con α=2 si ottiene d_2. La distanza $d_{\max}$ si ottiene per α tendente a $+\infty$. È facile dimostrare infatti che

$$\lim_{\alpha\to+\infty} d_\alpha(f,p) = \lim_{\alpha\to+\infty}\left(\int_a^b \left|f(x)-p(x)\right|^\alpha dx\right)^\alpha = \max_{x\in[a,b]}\left\{\left|f(x)-p(x)\right|\right\}.$$

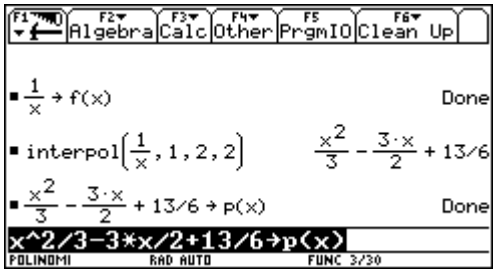
Per questo motivo la distanza $d_{\max}$ viene spesso indicata con il simbolo d_∞.
Riassumendo:

Distanza d_1: $\displaystyle\int_a^b \left|f(x)-p(x)\right|dx$

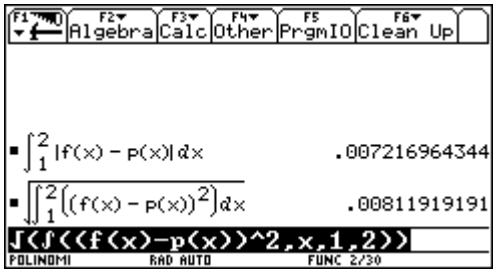
Distanza d_2: $\displaystyle\sqrt{\int_a^b \left(f(x)-p(x)\right)^2 dx}$

Distanza d_∞: $\displaystyle\max_{x\in[a,b]}\left\{\left|f(x)-p(x)\right|\right\}$

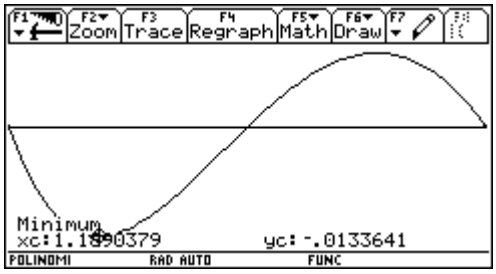
Come esempio calcoliamo le tre distanze della funzione $f(x)$:=$1/x$ dal polinomio interpolatore di secondo grado $p(x)$:=$\frac{1}{3}x^2-\frac{3}{2}x+\frac{13}{6}$ che abbiamo precedentemente calcolato.



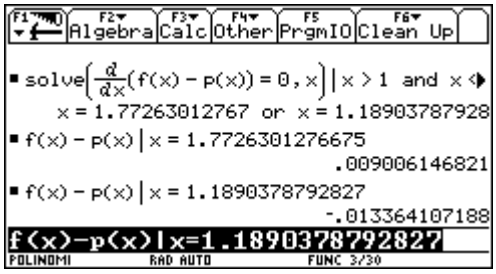
I primi due integrali non offrono difficoltà.



In realtà il primo integrale, la cui funzione integranda non è derivabile in 3/2, è ostico per qualsiasi sistema di calcolo simbolico. La TI-92 passa direttamente in modalità approssimata. Per la terza distanza osserviamo ancora il grafico di $f(x)$–$p(x)$ in $[1, 2]$. Il massimo di $\left|f(x)-p(x)\right|$ è nel minimo relativo di $f(x)$–$p(x)$, circa x=1.189.



Confermiamo questa analisi svolta direttamente sul grafico cercando gli zeri della derivata prima di $f(x)-p(x)$ in [1, 2].



In definitiva risulta

$$d_1(f, p) \approx 0.0072$$
$$d_2(f, p) \approx 0.0081$$
$$d_\infty(f, p) \approx 0.0134.$$

Siamo pronti per affrontare il problema centrale: calcolare, per ciascuna delle tre distanze, il miglior polinomio di grado fissato n, cioè quello (se esiste e se è unico) che, tra tutti i polinomi di grado n, minimizza la distanza da $f(x)$.

3.3 Il miglior polinomio secondo la distanza d_1

Data una funzione $f(x)$ continua su $[a, b]$, cerchiamo il polinomio $p_n(x)$ di grado n che minimizza la distanza

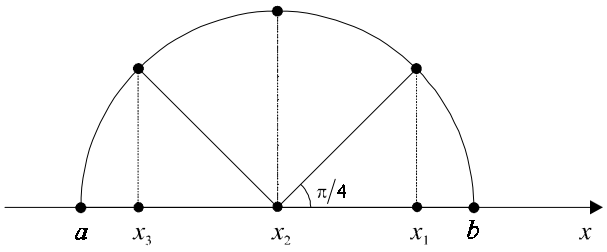
$$d_1(f, p) := \int_a^b |f(x) - p(x)|\,dx .$$

Si dimostra (vedi [RI], § 3.1) che, per ogni n, tale polinomio esiste ed è unico. Sotto certe ipotesi di regolarità per la funzione f (per esempio l'esistenza della derivata $n+1$-esima) il polinomio $p_n(x)$ si può calcolare con un semplice algoritmo. Vale infatti il seguente teorema, dovuto a Chebychev (Pafnuty Lvovich Chebyshev, 1821-1894).

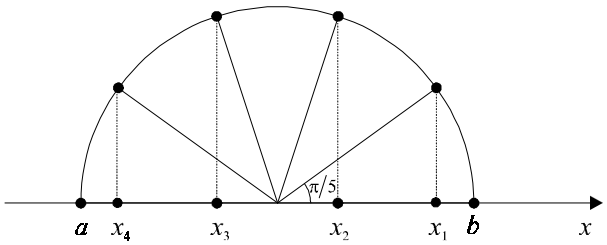
Teorema. Il polinomio $p_n(x)$ di grado n che meglio approssima $f(x)$ su $[a, b]$ è il polinomio che interpola $f(x)$ negli $n+1$ punti di ascissa

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(k\frac{\pi}{n+2}\right), \qquad k = 1, 2, \ldots, n+1.$$

In sostanza: si immagini di costruire la semicirconferenza di diametro $[a, b]$ (dunque $(a+b)/2$ è il centro e $(b-a)/2$ è il raggio), e di dividerla in $n+2$ archi uguali mediante $n+1$ punti di suddivisione. La proiezione dei punti di suddivisione sull'asse x ci dà le ascisse dei punti di interpolazione. Nelle figure seguenti sono mostrati tali punti nei casi $n=2$ e $n=3$.

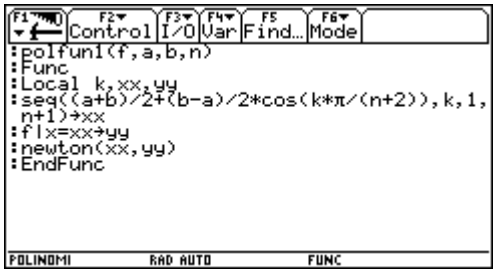


$n = 2$

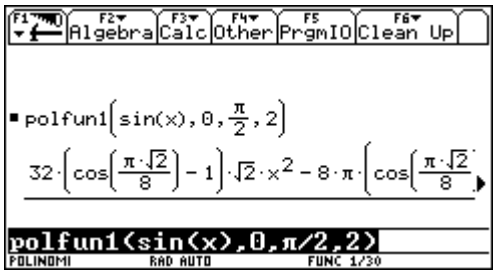


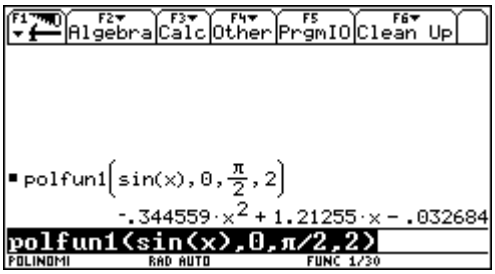
$n = 3$

Non è difficile ora implementare il programma polfun1(f,a,b,n) che calcola il miglior polinomio $p_2(x)$ che approssima f sull'intervallo $[a, b]$.

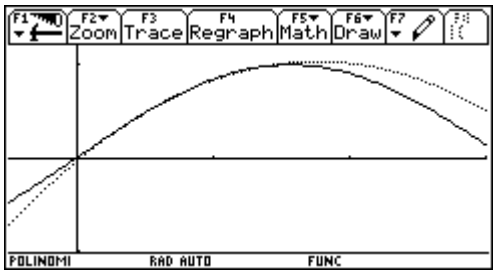


Applichiamo il programma a $\sin(x)$ su $[0, \pi/2]$. Otteniamo $p_2(x)$ in forma simbolica e poi in forma approssimata.

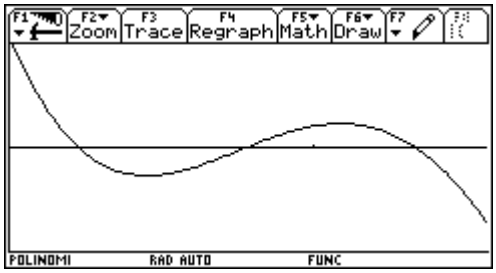




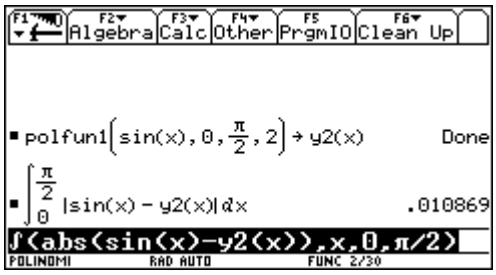
I due grafici sull'intervallo $[0, \pi/2]$ sono quasi sovrapposti: già con il polinomio di secondo grado (tratteggiato nella figura seguente) si ottiene una buona approssimazione.



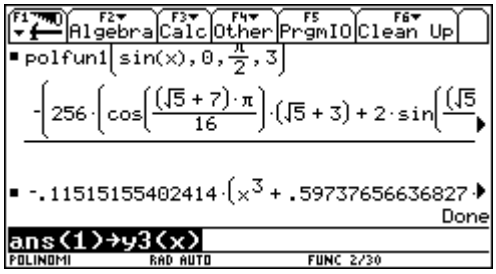
Forse è più istruttivo osservare la curva d'errore $\sin(x)–p_2(x)$ nel rettangolo $[0, \pi/2]\times[-0.03, 0.03]$.



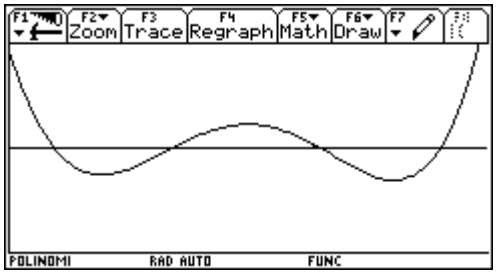
Calcoliamo ora la distanza d_1 tra $\sin(x)$ e $p_2(x)$.

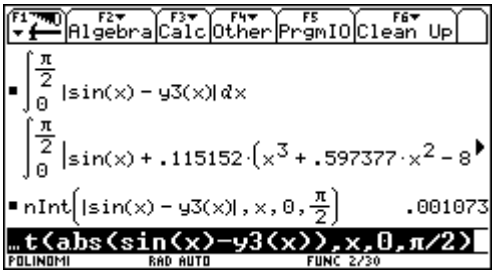


Tale distanza (circa 0.01087) è minore rispetto a quella che si ottiene per qualunque altro polinomio di secondo grado. Per avvicinarci di più a $\sin(x)$ dobbiamo salire di grado. Calcoliamo $p_3(x)$.

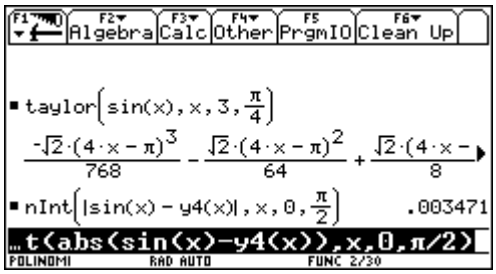


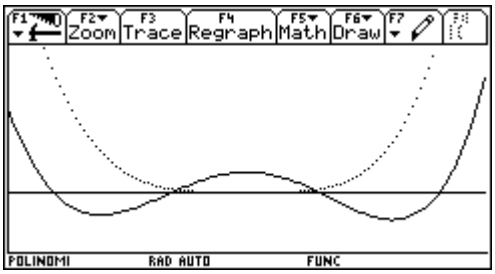
Ecco il grafico di $\sin(x)-p_3(x)$ nel rettangolo $[0, \pi/2]\times[-0.003, 0.003]$ e il calcolo della distanza.





L'approssimazione è migliorata di circa un ordine di grandezza.
Possiamo confrontare p_3 con il polinomio T_3 di terzo grado di Taylor, con centro in $\pi/4$, punto medio dell'intervallo.





La distanza $d_1(\sin(x), T_3)$ è più che tripla rispetto alla distanza $d_1(\sin(x), p_3)$. I grafici di $\sin(x)-p_3(x)$ e $\sin(x)-T_3(x)$ (tratteggiato) nel rettangolo $[0, \pi/2]\times[-0.002\times0.005]$ mettono in luce ancora una volta il fatto che il polinomio di Taylor, che ha carattere locale, è più efficiente di p_3 vicino a $\pi/4$ ma è molto meno efficiente di p_3 sull'intervallo $[0, \pi/2]$.

3.4 Il miglior polinomio secondo la distanza d_2

Vogliamo ora calcolare il polinomio $p_n(x)$ di grado n che minimizza la distanza

$$d_2(x) := \sqrt{\int_a^b \big(f(x) - p(x)\big)^2 \, dx} \, .$$

Tale distanza tra funzioni trasporta nel continuo il classico procedimento discreto dei minimi quadrati. Anche in questo caso un teorema ci assicura che il polinomio di grado n che minimizza la distanza d_2 da f esiste ed è unico. Per determinarlo possiamo operare in due modi distinti, che portano allo stesso risultato; vogliamo illustrare i due procedimenti sull'esempio

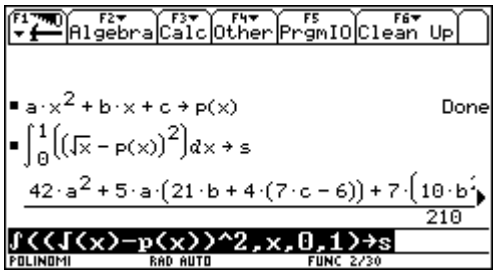
$$f(x) := \sqrt{x} \text{ in } [0, 1].$$

3.4.1 Utilizzare direttamente la definizione

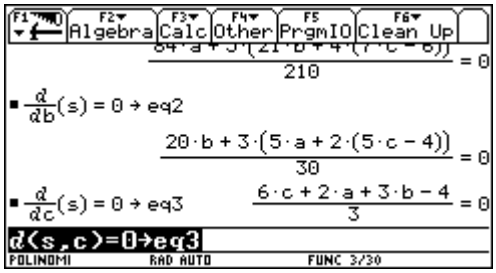
Si deve minimizzare la funzione

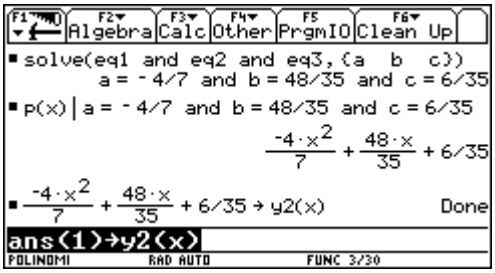
$$S(a, b, c) := \int_0^1 \left(\sqrt{x} - ax^2 - bx - c\right)^2 dx \, .$$

Calcoliamo l'espressione di S e memorizziamola in s.



Per determinare il minimo di S, calcoliamo le tre derivate parziali rispetto a a, b, c e memorizziamo nelle variabili eq1, eq2, eq3 le relative equazioni che le annullano. Risolviamo infine il sistema (lineare, dato che S è quadratico in a, b, c) delle tre equazioni.

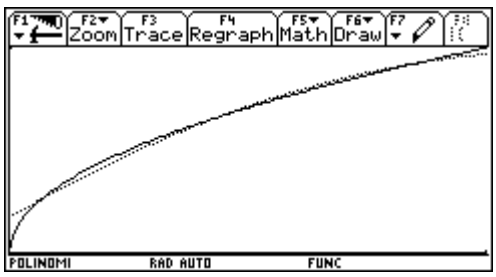


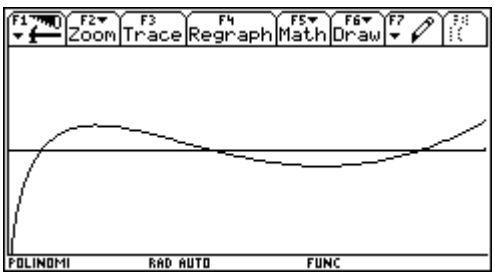


Si ottiene il polinomio

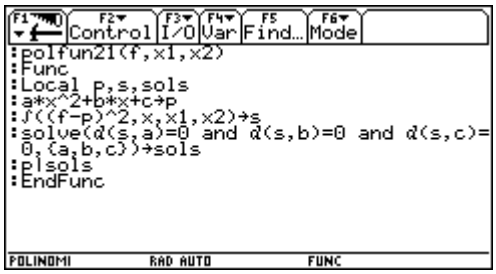
$$p_2(x) := -\frac{4}{7}x^2 + \frac{48}{35}x + \frac{6}{35} \, .$$

Ecco i grafici di f e p_2 (nel rettangolo $[0, 1]\times[0, 1]$)e di $f-p_2$ (nel rettangolo $[0, 1]\times[-0.1, 0.1]$.

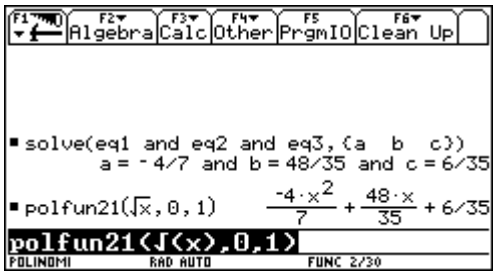




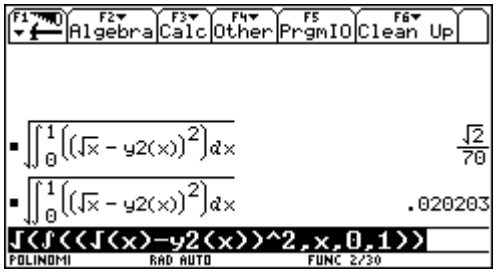
Come si vede c'è un errore non trascurabile in 0, dove la pendenza di f tende a ∞.
È facilmente implementabile un programma polfun21 che calcoli $p_2(x)$.



Verifichiamo la correttezza del programma.

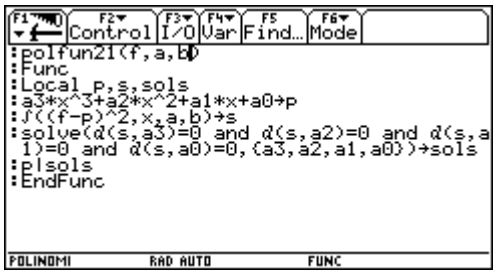


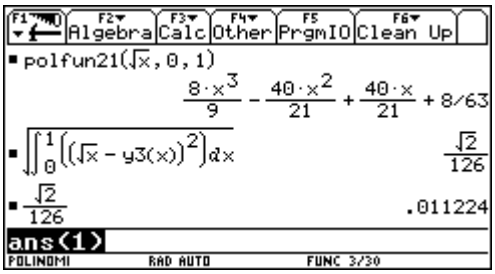
Calcoliamo la distanza d_2 di $p_2(x)$ da $\sqrt{x}$.



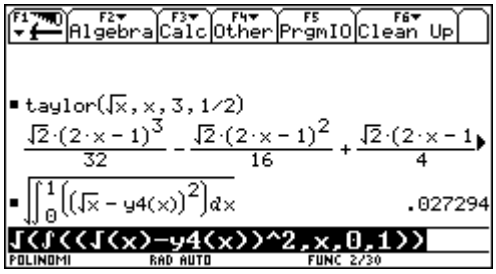
Risulta $d_2(f, p_2) \approx 0.02$.

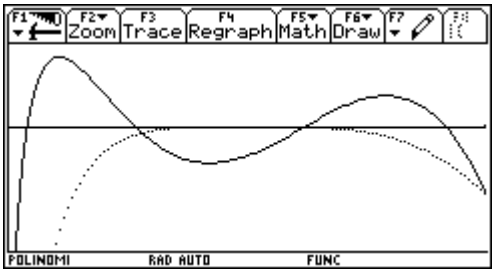
Il CAS della TI-92 non è sofisticato come quello di Maple o di Mathematica; in particolare è assai laborioso (ma non impossibile) definire un numero di variabili (i coefficienti del polinomio) che sia funzione di un parametro dato in ingresso (il grado). Quindi non è semplice costruire un programma per calcolare i polinomi di qualsiasi grado; d'altra parte se si vuole calcolare $p_n(x)$ si può modificare facilmente il programma polfun21, come illustrano le figure seguenti, in cui viene calcolato $p_3(x)$.





Risulta $d_2(f, p_3) \approx 0.01$: la distanza è quasi dimezzata.
Possiamo anche in questo caso confrontare $p_3(x)$ con $T_3(x)$ (tratteggiato), il polinomio di Taylor con centro in 1/2, punto medio dell'intervallo. I grafici di $\sqrt{x} - p_3$ e $\sqrt{x} - T_3$ sono tracciati nel rettangolo $[0, 1]\times[-0.03\times0.02]$.





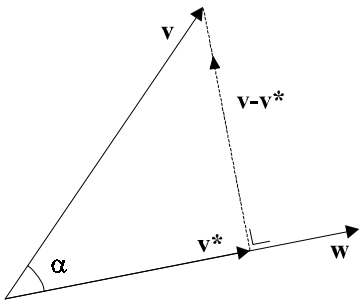
La distanza $d_2(\sqrt{x}\,, T_3)$ è ben maggiore della distanza $d_2(\sqrt{x}\,, p_3)$.

3.4.2 La teoria dei polinomi ortogonali

È possibile trattare le funzioni come vettori? Che senso ha parlare di funzioni "perpendicolari"?
Partiamo da un esempio significativo. Lo spazio tridimensionale euclideo ammette come modello lo spazio vettoriale $\mathbf{R}^3$ dotato di prodotto scalare: sono definite le operazioni di somma $\mathbf{v}+\mathbf{w}$ di due vettori, il multiplo reale $k\mathbf{v}$ di un vettore e il prodotto scalare $\mathbf{v}\cdot\mathbf{w}$ di due vettori. Se $\mathbf{v}=[x_1, y_1, z_1]$, $\mathbf{w}=[x_2, y_2, z_2]$ allora

1) $\mathbf{v}+\mathbf{w} := [x_1+x_2, y_1+y_2, z_1+z_2]$
2) $k\mathbf{v} := [kx_1, ky_1, kz_1]$
3) $\mathbf{v}\cdot\mathbf{w} := x_1x_2+y_1y_2+z_1z_2$

Le operazioni godono delle note proprietà.
Due vettori $\mathbf{v}$ e $\mathbf{w}$ sono paralleli se esiste $k\in\mathbf{R}$ tale che $\mathbf{v}=k\mathbf{w}$.
Due vettori $\mathbf{v}$ e $\mathbf{w}$ sono perpendicolari se $\mathbf{v}\cdot\mathbf{w} = 0$.
La lunghezza (norma) di un vettore può essere definita mediante il prodotto scalare: $\|\mathbf{v}\| = \sqrt{\mathbf{v}\cdot\mathbf{v}}$.
L'angolo convesso α tra due vettori non nulli è univocamente determinato e risulta $\mathbf{v}\cdot\mathbf{w} = \|\mathbf{v}\|\,\|\mathbf{w}\|\cos(\alpha)$.



La proiezione ortogonale $\mathbf{v}^*$ di un vettore $\mathbf{v}$ su un vettore $\mathbf{w}$ (cioè su un sottospazio di dimensione 1) si ottiene nel seguente modo:

$$\mathbf{v}^* = \frac{\|\mathbf{v}\|\cos(\alpha)}{\|\mathbf{w}\|}\mathbf{w} = \frac{\|\mathbf{v}\|\,\|\mathbf{w}\|\cos(\alpha)}{\|\mathbf{w}\|\,\|\mathbf{w}\|}\mathbf{w} = \frac{\mathbf{v}\cdot\mathbf{w}}{\mathbf{w}\cdot\mathbf{w}}\mathbf{w}.$$

Per esempio la proiezione ortogonale di $\mathbf{v}=[1, 2, 3]$ su $\mathbf{w}=[-2, 5, 1]$ è il vettore

$$\mathbf{v}^* = \frac{[1,2,3]\cdot[-2,5,1]}{[-2,5,1]\cdot[-2,5,1]}\mathbf{w} = \frac{1}{2}\mathbf{w} = \left[-1, \frac{5}{2}, \frac{1}{2}\right].$$

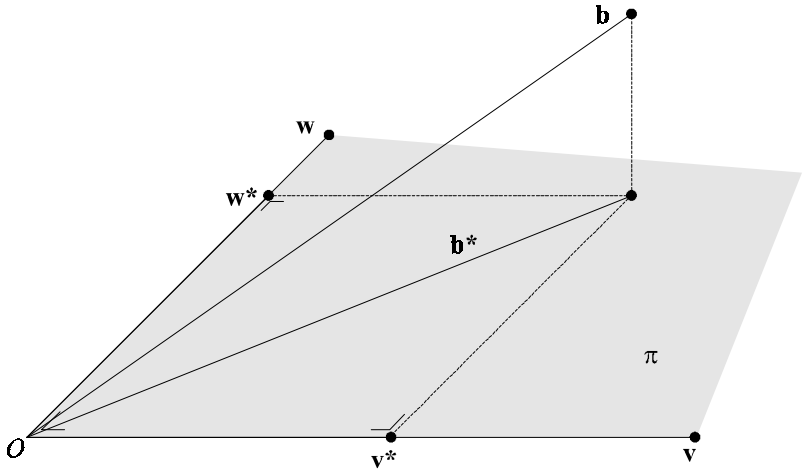
Il vettore $\mathbf{v}^*$ è il vettore parallelo a $\mathbf{w}$ "più vicino" a $\mathbf{v}$, nel senso che

$$\|\mathbf{v}-\mathbf{v}^*\| \le \|\mathbf{v}-\mathbf{x}\| \quad \text{per ogni } \mathbf{x} \text{ parallelo a } \mathbf{w}.$$

La proiezione ortogonale di un vettore $\mathbf{b}$ su un piano (cioè su un sottospazio di dimensione 2, generato da due vettori non paralleli; tale coppia di vettori è una base del sottospazio) è più complessa, ma tutto si semplifica se conosciamo una base ortogonale del sottospazio, cioè una

coppia di vettori **v** e **w** tra loro ortogonali: $\mathbf{v}\cdot\mathbf{w} = 0$. In questo caso la proiezione **b*** di **b** sul piano generato da **v** e **w** è la somma delle proiezioni ortogonali:

$$\mathbf{b}^* = \frac{\mathbf{b}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v} + \frac{\mathbf{b}\cdot\mathbf{w}}{\mathbf{w}\cdot\mathbf{w}}\mathbf{w}$$



Per esempio, calcoliamo la proiezione ortogonale **b*** del vettore **b**=[1, 2, 3] sul piano generato da **v**=[2, 5, −1] e **w**=[−2, 1, 1] (i vettori **v** e **w** sono ortogonali).

$$\begin{aligned}\mathbf{b}^* &= \frac{[1,2,3]\cdot[2,5,-1]}{[2,5,-1]\cdot[2,5,-1]}\mathbf{v} + \frac{[1,2,3]\cdot[-2,1,1]}{[-2,1,1]\cdot[-2,1,1]}\mathbf{w} \\ &= \frac{9}{30}\mathbf{v} + \frac{3}{6}\mathbf{w} \\ &= \frac{3}{10}[2,5,-1] + \frac{1}{2}[-2,1,1] \\ &= \left[-\frac{2}{5}, 2, \frac{1}{5}\right]\end{aligned}$$

Il vettore **b*** è il vettore del piano generato dalle combinazioni lineari di **v** e **w** "più vicino" a **b**, nel senso che

$$\|\mathbf{b} - \mathbf{b}^*\| \le \|\mathbf{b} - \mathbf{x}\| \quad \text{per ogni } \mathbf{x} = a\mathbf{v} + b\mathbf{w}.$$

Passiamo ora a parlare degli stessi concetti in termini di funzioni.
L'insieme delle funzioni continue su $[a, b]$ costituisce uno spazio vettoriale reale $C[a, b]$ (di dimensione infinita) rispetto alle operazioni

$(f_1+f_2)(x) := f_1(x)+f_2(x)$
$(kf)(x) := kf(x)$

Infatti se f_1 e f_2 sono continue su $[a, b]$ sono continue anche f_1+f_2 e kf.
Nello spazio $C[a, b]$ possiamo considerare il sottospazio P_2 dei polinomi di grado minore o uguale a 2:

$$P_2 := \{ax^2+bx+c: a,b,c\in \mathbf{R}\}$$

(più in generale possiamo considerare P_n, il sottospazio dei polinomi di grado minore o uguale a n: per semplicità limitiamoci a n=2, la generalizzazione sarà immediata).
Il sottospazio P_2 ha dimensione finita uguale a 3: una base è per esempio

$$\{1, x, x^2\},$$

nel senso che qualunque polinomio ax^2+bx+c di P_2 si ottiene come combinazione lineare dei vettori della base.
Ora in $C[a, b]$ introduciamo un prodotto scalare, che è l'analogo (discreto→continuo) del prodotto scalare in $\mathbf{R}^3$; ad una somma di prodotti si sostituisce l'integrale del prodotto:

$$f_1 \cdot f_2 := \int_a^b f_1(x) f_2(x)\, dx$$

È semplice verificare che sono verificate le proprietà del prodotto scalare:

1) se $f \neq \mathbf{0}$ allora $f \cdot f > 0$
2) $f_1 \cdot f_2 = f_2 \cdot f_1$
3) $(f_1 + f_2) \cdot f_3 = (f_1 \cdot f_3) + (f_2 \cdot f_3)$
4) $(kf_1) \cdot f_2 = k\,(f_1 \cdot f_2)$

La possibilità di definire un prodotto scalare permette in modo naturale di definire la *norma* di una funzione e la *distanza* tra due funzioni, che coincide con la distanza d_2:

$$\| f \| := \sqrt{\int_a^b f(x)^2\, dx}$$

$$d(f_1, f_2) = \| f_1 - f_2 \| = \sqrt{\int_a^b \left(f_1(x) - f_2(x)\right)^2 dx}$$

Ora la situazione è questa: abbiamo un vettore-funzione di $C[a, b]$, la funzione $f(x)$, e abbiamo il sottospazio P_2 delle funzioni polinomiali di secondo grado. Cerchiamo, in P_2, il vettore-polinomio "più vicino" a f, cioè quel vettore p_2 tale che

$$\| f - p_2 \| \leq \| f - p \| \quad \text{per ogni } p \in P_2.$$

È bellissimo pensare ai polinomi come se fossero vettori geometrici! Bene: il polinomio p_2 non è altro che la proiezione ortogonale di f su P_2. Tutto quello che ci serve è una base ortogonale $\{b_0, b_1, b_2\}$ di P_2. Una volta trovata quella sarà immediato calcolare p_2:

$$p_2 = \frac{f \cdot b_0}{b_0 \cdot b_0} b_0 + \frac{f \cdot b_1}{b_1 \cdot b_1} b_1 + \frac{f \cdot b_2}{b_2 \cdot b_2} b_2 = \sum_{i=0}^{2} \frac{f \cdot b_i}{b_i \cdot b_i} b_i$$

La base $\{1, x, x^2\}$ non è ortogonale; infatti il prodotto scalare tra 1 e x (per esempio) non è nullo:

$$1 \cdot x = \int_a^b 1x\, dx = \frac{b^2 - a^2}{2}.$$

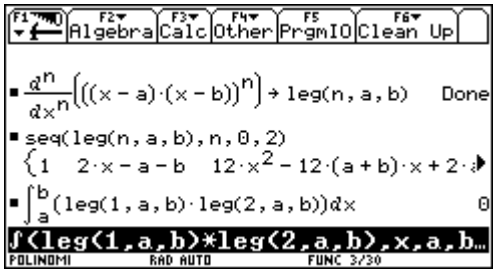
Tutto ciò che dobbiamo fare ora è procurarci una base ortogonale di P_2. A questo scopo chiameremo in causa i polinomi di Legendre (Adrien-Marie Legendre, 1752-1832), vedi per esempio [RI] § 2.4, [FL] § 5.3.

Il polinomio di Legendre di grado n relativo all'intervallo $[a, b]$ è così definito:

$$L(n) := \frac{d^n}{dx^n}\left((x-a)(x-b)\right)^n.$$

La caratteristica dei polinomi di Legendre che a noi interessa è che essi sono a due a due ortogonali rispetto al prodotto scalare definito precedentemente. Più precisamente: se $h \neq k$ allora

$$L(h) \cdot L(k) = \int_a^b L(h) L(k)\, dx = 0.$$



I polinomi di Legendre di gradi 0, 1, 2 sono rispettivamente

$L(0) = 1$ $\qquad$ $L(1) = 2x-a-b$ $\qquad$ $L(2) = 12x^2-12(a+b)x+2a^2+8ab+2b^2$.

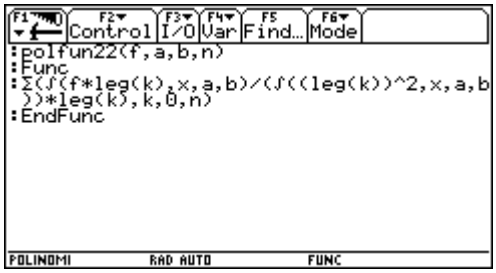
Abbiamo ora tutti gli strumenti che ci servono: la proiezione ortogonale di f su P_2 è il polinomio

$$p_2(x) = \sum_{i=0}^{2} \frac{f \cdot L(i)}{L(i) \cdot L(i)} L(i) = \sum_{i=0}^{2} \frac{\int_a^b f L(i)\,dx}{\int_a^b L(i)^2\,dx} L(i)$$

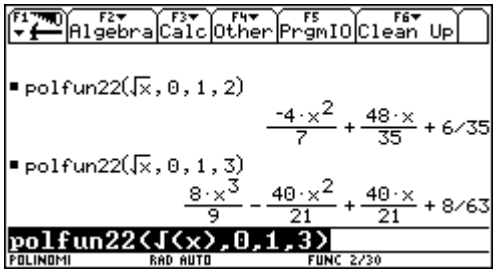
Tutti i discorsi svolti per P_2 possono essere immediatamente generalizzati a P_n:

$$p_n(x) := \sum_{i=0}^{n} \frac{f \cdot L(i)}{L(i) \cdot L(i)} L(i) = \sum_{i=0}^{n} \frac{\int_a^b f L(i)\,dx}{\int_a^b L(i)^2\,dx} L(i).$$

Ecco finalmente il semplicissimo (una riga soltanto!) programma polfun22(f,a,b,n) che calcola $p_n(x)$ per $f(x)$ nell'intervallo $[a, b]$.



Verifichiamo la correttezza del programma sull'esempio già svolto: $f(x):=\sqrt{x}$ in $[0, 1]$.



Ritroviamo i risultati $p_2(x)$ e $p_3(x)$ già ottenuti.

3.4.3 Minimi quadrati: confronto tra discreto e continuo

È interessante osservare che, sotto opportune ipotesi (largamente soddisfatte dalle funzioni continue più comuni), il discreto … tende al continuo! Precisamente: la parabola di regressione che interpola $f(x)$ su m punti equispaziati $\left(x_i, f\left(x_i\right)\right)$, cioè la parabola che minimizza la somma

$$\sum_{i=1}^{m} \left(f\left(x_i\right) - ax_i^2 - bx_i - c\right)^2,$$

al crescere di m tende alla funzione polinomiale $p_2(x)$, cioè la parabola che minimizza l'integrale

$$\int_a^b \left(f\left(x\right) - ax^2 - bx - c\right)^2 dx.$$

La TI-92, in ambiente Data Matrix Editor, calcola la miglior funzione quadratica di un set di punti. Apriamo un Data, mettiamo nella colonna c1 solo il numero m. Nell'intestazione di colonna c2 inseriamo il comando

seq(k/c1[1],k,0,c1[1])

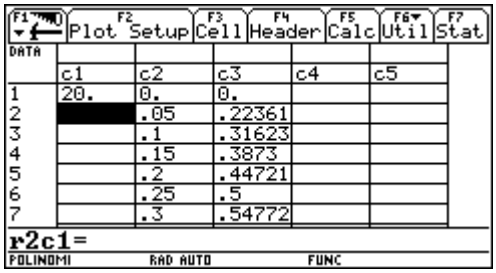
che crea la lista delle ascisse; nell'intestazione di colonna c2 inseriamo il comando

$$\sqrt{c2}$$

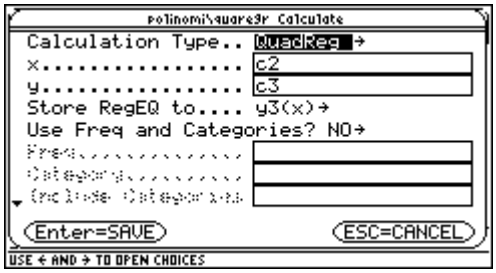
che crea la lista delle ordinate. Per esempio, con c1[1]=20 si ottengono le liste delle ascisse e ordinate di 21 punti.

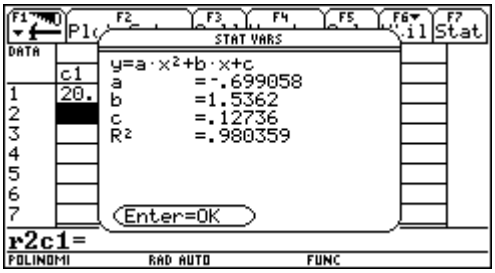
$$c2:=\left\{0,\frac{1}{20},\frac{2}{20},\ldots,\frac{19}{20},1\right\}$$

$$c3:=\left\{0,\sqrt{\frac{1}{20}},\sqrt{\frac{2}{20}},\ldots,\sqrt{\frac{19}{20}},1\right\}$$



Il menù F5, Calc, QuadReg ci permette di calcolare rapidamente il polinomio dei minimi quadrati di secondo grado relativo ai 21 punti elencati in c2 e in c3.

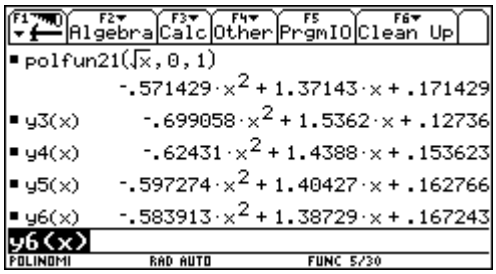




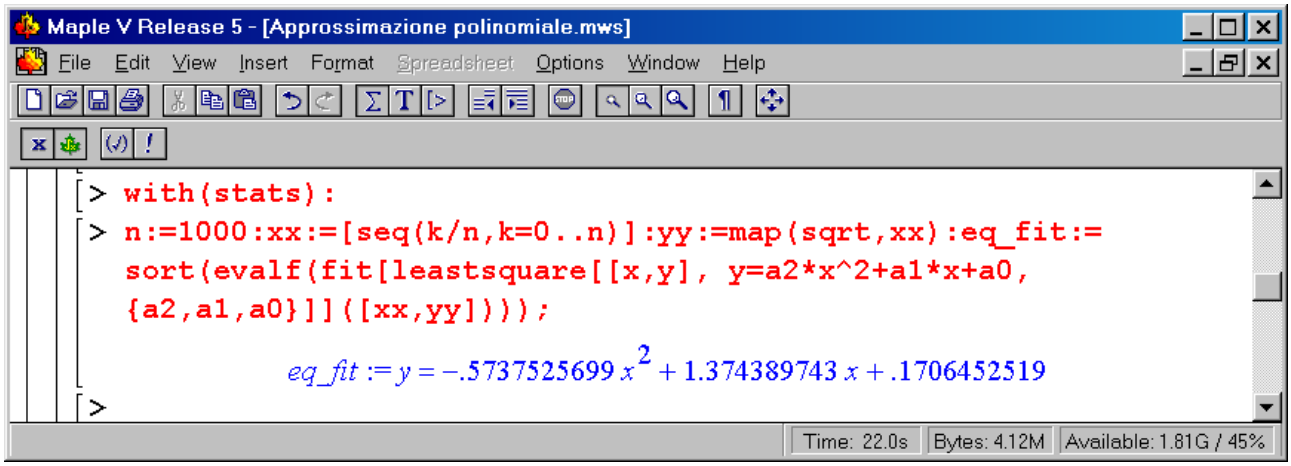
È il polinomio

$$-0.699x^2+1.54x+0.127.$$

Ora aumentiamo il numero di punti. In c1[1] inseriamo 50, poi 100, poi 200. Seppure lentamente, i polinomi che otteniamo convergono a $p_2(x)$ già trovato.



La convergenza è molto lenta: con 1001 punti otteniamo un polinomio (calcolato con Maple) i cui coefficienti hanno soltanto le prime due cifre decimali esatte.



Tale lentezza di convergenza ci permette di concludere che approssimare $p_2(x)$ con tale metodo non è efficace.

3.3 Il miglior polinomio secondo la distanza d_∞

Vogliamo calcolare il polinomio $p_n(x)$ di grado n che minimizza la distanza

$$d_\infty(f,p) := \max_{x\in[a,b]}\left\{\left|f(x)-p(x)\right|\right\}.$$

Poiché si tratta di minimizzare un massimo la distanza d_∞ è anche chiamata distanza *minimax*, o approssimazione *minimax* (o anche *approssimazione uniforme*).
Questa distanza realizza un obiettivo molto importante: una volta determinato $p_n(x)$ per un certo grado n e misurata la distanza $d_\infty(f,p_n)=\varepsilon$, siamo certi che per qualunque x compreso tra a e b risulti

$$\left|f(x)-p_n(x)\right| \le \varepsilon.$$

Perché questa proprietà è importante? Si pensi all'implementazione di una funzione trascendente, per esempio e^x, su una calcolatrice scientifica; supponiamo (ipotesi semplificatrice ma non lontana dalla realtà) che la calcolatrice sappia calcolare solo somme e prodotti, e quindi sappia calcolare agevolmente il valore di un polinomio in un punto. La calcolatrice visualizza k cifre decimali; se riusciamo a trovare un polinomio p tale che

$$d_\infty(f,p) < 0.5\cdot 10^{-k}$$

su un dato intervallo $[a, b]$ allora possiamo, a tutti gli effetti, sostituire il polinomio p alla funzione f. La restrizione all'intervallo $[a, b]$ non è limitativa; la conoscenza dei valori di f su un intervallo è spesso sufficiente a calcolare f ovunque. Supponiamo infatti di voler approssimare e^x per un certo $x\in\mathbf{R}$. Nell'ipotesi che la calcolatrice lavori in base 2, passiamo al logaritmo in base 2 di e^x.

$$\log_2\left(e^x\right) = m+d,$$

dove m è la parte intera e d la parte decimale: $0\le d<1$. Risulta

$$2^{\log_2\left(e^x\right)} = e^x = 2^{m+d} = 2^m\cdot 2^d.$$

Ora l'unico numero che dobbiamo calcolare è 2^d, dato che il prodotto per 2^m produce, nella rappresentazione in base 2, solo uno *shift*, cioè lo spostamento del punto decimale. Poiché d è compreso tra 0 e 1, 2^d è compreso tra 1 e 2: tutto ciò che ci serve è la conoscenza di e^x per i valori che esso assume tra 1 e 2, cioè per

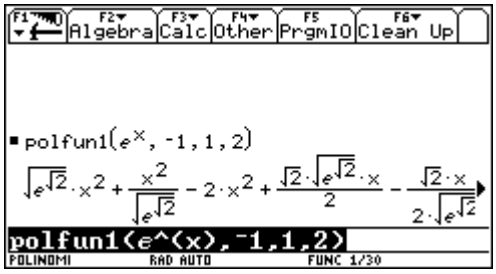
$$\ln(1) \le x < \ln(2)$$
$$0 \le x < 0.6931\ldots$$

In definitiva conoscere i valori di e^x (o meglio opportune approssimazioni di e^x) nell'intervallo $[0, 0.7]$, oppure equivalentemente nell'intervallo $[-0.35, 0.35]$ che si ottiene dal precedente mediante una trasformazione lineare (cioè una moltiplicazione e una addizione) è sufficiente a calcolare e^x su tutto $\mathbf{R}$ (vedi [RA] § 7.4).

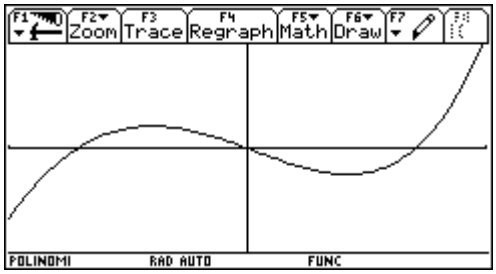
Un teorema, ancora dovuto a Chebychev, ci assicura che, fissato n, il polinomio p^* di grado n che minimizza

$$d_\infty(f,p) := \max_{x\in[a,b]}\{|f(x)-p(x)|\}$$

esiste ed è unico. Tuttavia non sono molte le funzioni per le quali tale polinomio può essere calcolato in forma simbolica. Esistono piuttosto diversi algoritmi iterativi che, a partire da un polinomio che interpola f su n+2 punti $x_0, x_1, \ldots, x_{n+1}$, costruiscono una successione di polinomi che converge a p^*.
Vogliamo illustrare uno di questi algoritmi, noto come *algoritmo di Remez* (Evgeny Yakovlevich Remez, 1896-1975). Mostriamo sull'esempio $f(x):=e^x$ su $[-1, 1]$ come costruire p^* tra i polinomi di grado 2. In seguito non sarà difficile generalizzare.
Consideriamo come polinomio iniziale p_0 quello che interpola $f(x)$ negli n+2 *punti di Chebychev* che abbiamo già incontrato a proposito della distanza d_1; utilizziamo quindi il programma polfun1.



Valutiamo la curva d'errore e^x-p_0.



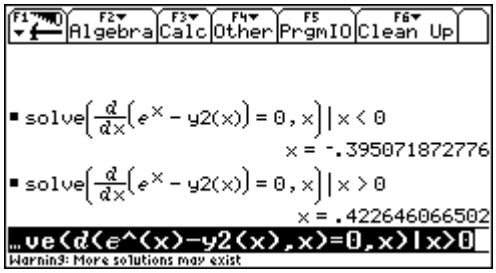
Come si vede la curva d'errore ammette 4 (in generale n+2) estremi:

$$x_0=a, x_1, x_2, x_3=b.$$

Sempre Chebychev ha dimostrato che la curva d'errore del polinomio che minimizza d_∞ deve essere fatta in modo tale che gli n+2 estremi abbiano, con segni alterni, lo stesso valore assoluto. Detto in altri termini: la curva d'errore deve oscillare uniformemente tra $-\varepsilon$ e $+\varepsilon$, dove ε è proprio il massimo di $|f(x)-p_0(x)|$, e quindi

$$\varepsilon = d_\infty(f, p_0).$$

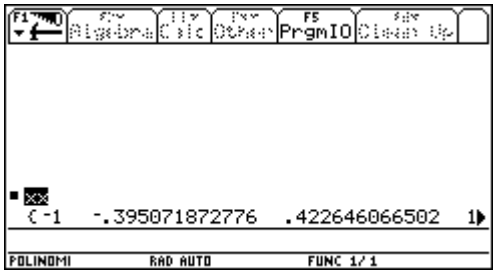
Questo palesemente non avviene per il nostro p_0; poiché f è derivabile, cerchiamo gli estremi della curva d'errore annullando la derivata.



Risulta

$$x_0=-1,\ x_1=-0.395\ldots,\ x_2=0.422\ldots,\ x_3=1.$$

Memorizziamo nella lista xx tali punti.



L'algoritmo di Remez, la cui difficoltà computazionale consiste proprio nella ricerca dei massimi della curva d'errore, cerca il polinomio di secondo grado tale che la curva d'errore ammetta, nei punti x_k, a segni alterni lo stesso valore incognito ε:

$$f(x_k) - p(x_k) = (-1)^k \varepsilon$$

cioè

$$p(x_k) + (-1)^k \varepsilon = f(x_k)$$

$$a_2 x_k^2 + a_1 x_k + a_0 + (-1)^k \varepsilon = f(x_k)$$

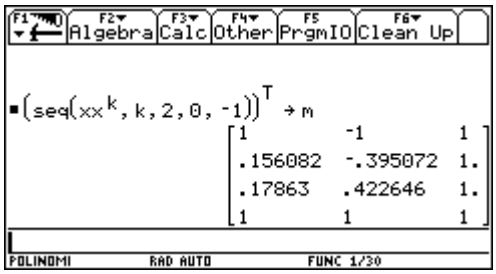
Si tratta di un sistema di quattro equazioni (quattro punti) in quattro incognite (a_2, a_1, a_0, ε).

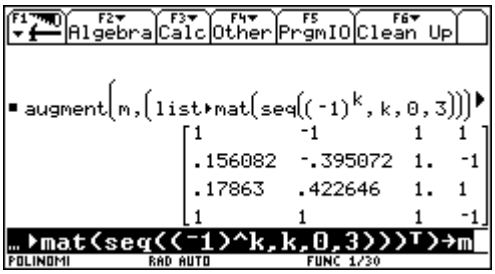
$$\begin{cases} a_2 x_0^2 + a_1 x_0 + a_0 + \varepsilon = f(x_0) \\ a_2 x_1^2 + a_1 x_1 + a_0 - \varepsilon = f(x_1) \\ a_2 x_2^2 + a_1 x_2 + a_0 + \varepsilon = f(x_2) \\ a_2 x_3^2 + a_1 x_3 + a_0 - \varepsilon = f(x_3) \end{cases}.$$

In notazione matriciale:

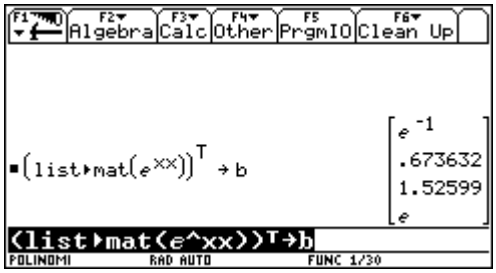
$$\begin{bmatrix} x_0^2 & x_0 & 1 & 1 \\ x_1^2 & x_1 & 1 & -1 \\ x_2^2 & x_2 & 1 & 1 \\ x_3^2 & x_3 & 1 & -1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \\ \varepsilon \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \end{bmatrix}$$

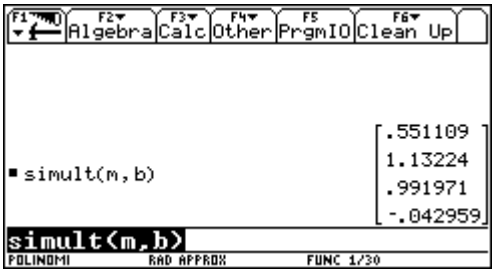
Nel nostro caso:



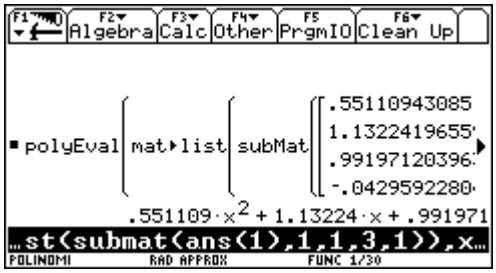


Risolviamo il sistema.

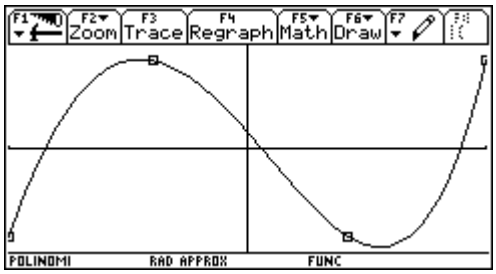




I primi tre elementi sono i coefficienti del polinomio p_1, l'ultimo elemento (in valore assoluto) è ε.



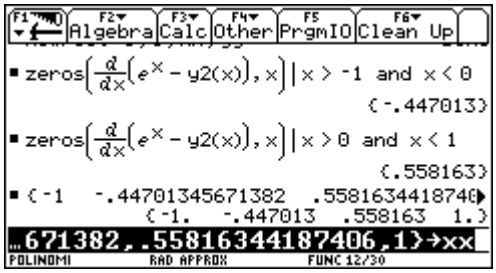
Abbiamo ottenuto un nuovo polinomio p_1 tale che $f(x)–p_1(x)$ che ammette lo stesso valore ε a segni alterni nei quattro punti x_k. Se tali punti fossero effettivamente gli estremi della curva d'errore allora avremmo trovato il polinomio che minimizza la distanza d_∞ da e^x su $[a, b]$.

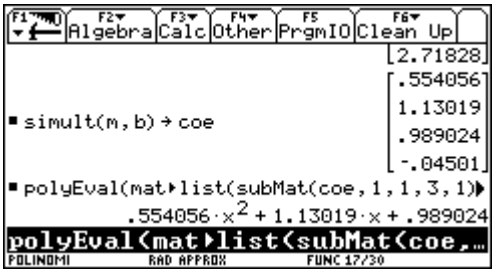


Tuttavia questi quattro punti non sono gli estremi della curva d'errore. Iteriamo allora il procedimento: cerchiamo ora gli estremi x_0, x_1, x_2, x_3 di $f(x)–p_1(x)$ e otterremo quattro nuovi punti x_0, x_1, x_2, x_3 (sempre con $x_0= -1$ e $x_3= 1$) a partire dai quali impostiamo di nuovo il sistema

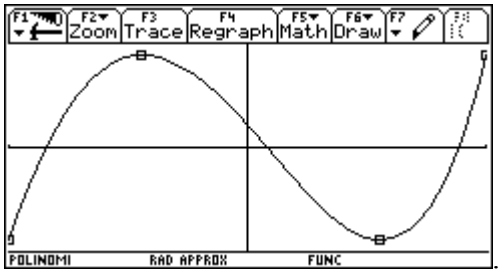
$$\begin{cases} a_2x_0^2 + a_1x_0 + a_0 + \varepsilon = f(x_0) \\ a_2x_1^2 + a_1x_1 + a_0 - \varepsilon = f(x_1) \\ a_2x_2^2 + a_1x_2 + a_0 + \varepsilon = f(x_2) \\ a_2x_3^2 + a_1x_3 + a_0 - \varepsilon = f(x_3) \end{cases}$$

la cui soluzione ci fornisce i coefficienti di un nuovo polinomio p_2 tale che $f(x)–p_2(x)$ ammette lo stesso valore ε (a segni alterni) nei punti x_k; tali punti sono sempre più vicini agli estremi della curva d'errore, e il procedimento converge a p^*, il miglior polinomio d'approssimazione *minimax* di e^x su $[a, b]$.





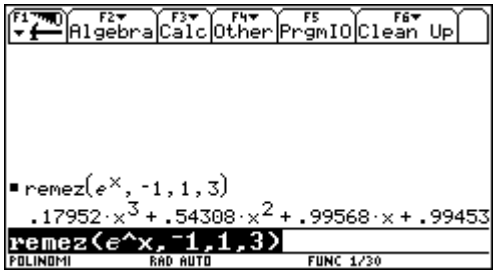
Ecco la curva d'errore $e^x–p_2(x)$.

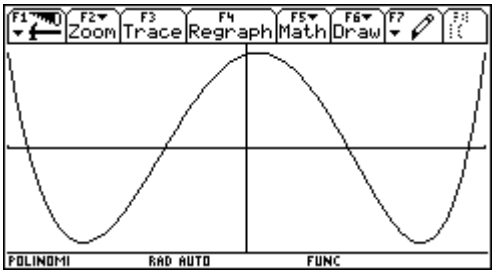


Il programma remez(f,a,b,n) automatizza e generalizza (almeno per funzioni derivabili, per le quali la ricerca dei massimi può essere svolta mediante derivazione) il procedimento illustrato.

```
remez(f,a,b,n)
Func
Local  k,i,j,sol,so,pti,m,v,p,p1,temp,tol
approx(seq((a+b)/2-(b-a)/2*cos(k*./(n+1)),k,0,n+1))→pti
newMat(n+2,n+2)→m
For  i,1,n+2
1→m[i,n+1]
EndFor
For  i,1,n+2
(-1)^(i+1)→m[i,n+2]
EndFor
0→tol
While  tol=0
For  i,1,n+2
For  j,1,n
pti[i]^(n+1-j)→m[i,j]
EndFor
EndFor
listÑmat(seq(f|x=pti[i],i,1,n+2),1)→v
simult(m,v)→so
matÑlist(subMat(so,1,1,n+1,1))→sol
polyEval(sol,x)→p
∫(f-p,x)→p1
zeros(p1,x)|x>a and x<b→temp
If dim(temp)<n Then
newList(n+2)→temp
For  i,2,n+1
nSolve(p1=0,x)|x>pti[i]-(b-a)/(n+1) and x<pti[i]+(b-a)/(n+1)→temp[i]
If temp[i]="No solution found" Then
(temp[i-1]+(b-a)/(n+1))/2→temp[i]
EndIf
EndFor
temp→pti:a→pti[1]:b→pti[n+2]
Else
temp→pti:augment({a},pti)→pti:augment(pti,{b})→pti
EndIf
If abs(so[n+2])<tol Then
abs(so[n+2]-tol)→tol
Else
Exit
EndIf
EndWhile
p
EndFunc
```

Ecco il programma remez all'opera per approssimare e^x su [−1, 1] con un polinomio di terzo grado, e la relativa curva d'errore nel rettangolo [−1, 1]×[−0.006, 0.006]





Resta da osservare che un software di calcolo potente e aggiornato come MAPLE possiede una funzione predefinita minimax che fornisce (ma non in forma simbolica!) il polinomio di miglior approssimazione uniforme: tale funzione utilizza (tuttora) l'algoritmo di Remez, come è chiaramente indicato nell'*help*.

Maple V Release 5 - [numapprox,minimax]

File Edit View Window Help

Mathematics... | Finding Roots, Factorization, e | Optimization... | minimax
New User's Tour | General Information... | Roots... | minimize

numapprox[minimax] - **minimax rational approximation**

Calling Sequence:

minimax(f, x=a..b, [m,n], w, 'maxerror')

minimax(f, a..b, [m,n], w, 'maxerror')

Parameters:

`f` - procedure or expression representing the function

`x` - variable name appearing in f, if f is an expression

`a, b` - numerical values specifying the interval of approximation

`m` - integer specifying the desired degree of the numerator

`n` - integer specifying the desired degree of the denominator

`w` - (optional) procedure or expression specifying the weight function, default 1

`maxerror` - (optional) a name which will be assigned the minimax norm

Description:

- This procedure computes the best minimax rational approximation of degree (m,n) for a given real function f(x) on the interval [a,b] with respect to the positive weight function w(x), via Remez algorithm.
- Specifically, it computes the rational expression p(x)/q(x) such that

```
        max       w(x) * abs( f(x) - p(x)/q(x) )                (1)
    x in [a,b]
```

is minimized over all rational expressions p(x)/q(x) with numerator of degree <= m and denominator of degree <= n .

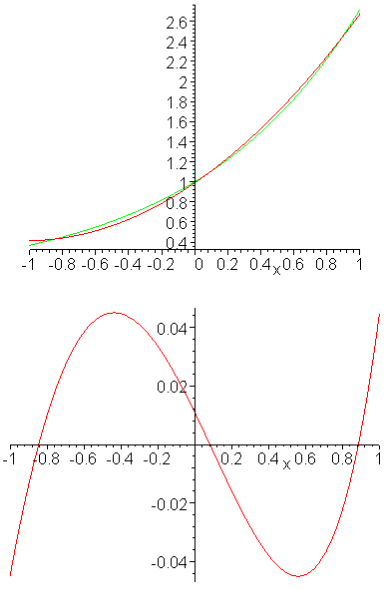
Maple V Release 5 - [Untitled (1)]

File Edit View Insert Format Spreadsheet Options Window Help

```
> p:=expand(minimax(exp(x), x=-1..1, 2));
```

$$p := .9890365552 + 1.130258690\,x + .5540440796\,x^2$$

```
> plot({exp(x),p},x=-1..1);plot({exp(x)-p},x=-1..1);
```



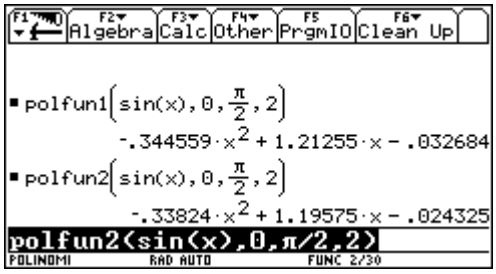

```
>
```

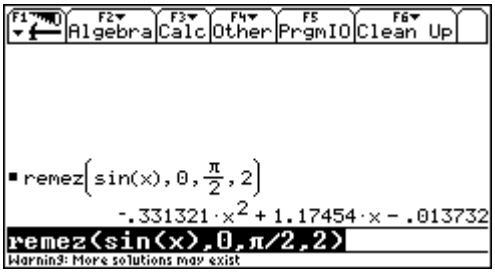
3.4 Confronti

Possiamo concludere le considerazioni svolte confrontando, per un grado fissato, i diversi polinomi che minimizzano le rispettive distanze.

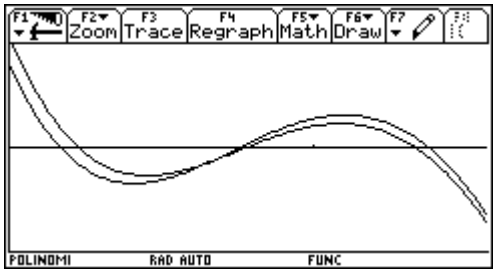
Possiamo finalmente rispondere alla domanda iniziale: qual è il "miglior" polinomio di secondo grado che approssima sin(x) su [0, π/2]?

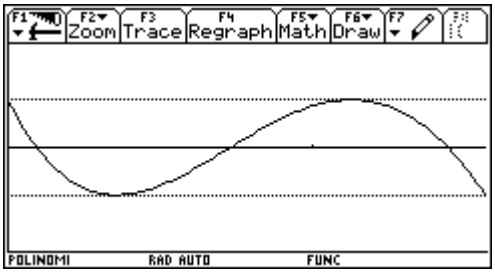
Chiamiamo p_1, p_2, p_∞ rispettivamente i polinomi migliori rispettivamente secondo le distanze d_1, d_2, d_∞.





Osserviamo le tre curve d'errore nel rettangolo [0, π/2]×[−0.03, 0.03].





Come si vede le curve sin(x)–p_1 e sin(x)–p_2 (nel grafico di sinistra) oscillano in modo non uniforme in [0, π/2], e hanno valori sensibilmente più elevati agli estremi dell'intervallo. Invece sin(x)–p_∞ (a destra) oscilla in modo uniforme: i suoi valori estremi (quindi la distanza d_∞ tra sin(x) e p_∞ su [0, π/2]) valgono tutti circa 0.0138.

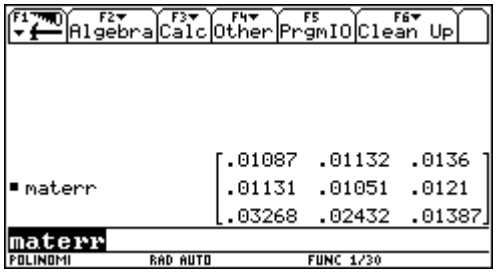
Per ciascun polinomio possiamo calcolare tre distanze diverse da sin(x) su [0, π/2]: d_1, d_2, d_∞.

Abbiamo così a che fare con 3^2=9 distanze, che possiamo disporre secondo la *matrice degli errori*:

$$\begin{bmatrix} d_1(f,p_1) & d_1(f,p_2) & d_1(f,p_\infty) \\ d_2(f,p_1) & d_2(f,p_2) & d_2(f,p_\infty) \\ d_\infty(f,p_1) & d_\infty(f,p_2) & d_\infty(f,p_\infty) \end{bmatrix}$$

$$\begin{bmatrix} \int_0^{\pi/2} |\sin(x)-p_1(x)|\,dx & \int_0^{\pi/2} |\sin(x)-p_2(x)|\,dx & \int_0^{\pi/2} |\sin(x)-p_\infty(x)|\,dx \\ \int_0^{\pi/2} (\sin(x)-p_1(x))^2\,dx & \int_0^{\pi/2} (\sin(x)-p_2(x))^2\,dx & \int_0^{\pi/2} (\sin(x)-p_\infty(x))^2\,dx \\ \max_{x\in[0,\pi/2]} \{|\sin(x)-p_1(x)|\} & \max_{x\in[0,\pi/2]} \{|\sin(x)-p_2(x)|\} & \max_{x\in[0,\pi/2]} \{|\sin(x)-p_\infty(x)|\} \end{bmatrix}$$

Eseguiamo il calcolo (molto laborioso!) e otteniamo la seguente matrice.



Il risultato è soddisfacente: ci aspettiamo infatti che il miglior polinomio per la distanza d_1 sia p_1, per la distanza d_2 sia p_2, per la distanza d_∞ sia p_∞; come si vede gli elementi della diagonale principale sono i valori minimi di ciascuna riga.

Conclusioni

Siamo partiti da un problema apparentemente semplice e siamo arrivati abbastanza lontano, applicando strumenti matematici in un contesto significativo. L'esplorazione, la ricerca, la congettura sono state protagoniste indiscusse del cammino. L'idea di algoritmo, che in qualche modo era già in testa allo studente che cercava di interpolare $\sin(x)$ su tre punti, è stata al tempo stesso guida e strumento; in qualche modo il senso di soddisfazione più volte provato è dovuto al fatto che l'oggetto che stiamo cercando è inchiodato una volta per tutte da un algoritmo che ci permette, dopo tanta fatica, di liberare la mente: polfun1(sin(x),0,π/2,2) è una nuova funzione, che abbiamo costruito noi, che possiamo usare d'ora in poi per nuovi problemi.

Bibliografia

[RI] Theodore Rivlin, *An introduction to the approximation of functions*, Dover 1969
[L] David Lay, *Linear algebra*, Addison Wesly 1994
[RA] Anthony Ralston, *A first course in numerical analysys*, McGraw-Hill 1965
[FL] J.M. Ferrard, H. Lemberg, *Mathématiques concrètes, illustrées par la TI-92 et la TI-89*
[B] E. Barbeau, *Polynomials*, Springer 1989
[QSS] A. Quarteroni, R. Sacco, F. Saleri, *Matematica numerica*, Springer 1998
[I] Michele Impedovo, *Matematica: computer algebra e insegnamento*, Springer 1999