Interpolazione e approssimazione di funzioni

Lucia Gastaldi

Dipartimento di Matematica,
http://dm.ing.unibs.it/gastaldi/

Laboratorio - 26 febbraio 2007

Outline

Interpolazione polinomiale

Problema

Dati $n+1$ punti $(x_i, f(x_i))$ per $i = 0, 1, \cdots, n$ si cerca una funzione approssimante $\tilde{f} : \mathbb{R} \to \mathbb{R}$ tale che

$$\tilde{f}(x_i) = f(x_i) \quad \text{per } i = 0, 1, \cdots, n.$$

L'esempio più semplice è il caso dei polinomi o dei polinomi trigonometrici.

Esistenza ed unicità del polinomio interpolatore

Teorema

Per ogni insieme di punti $(x_i, f(x_i))$ per $i = 0, 1, \cdots, n$, con gli x_i distinti tra loro, esiste un unico polinomio di grado n, che indicheremo con $\Pi_n f$, tale che

$$\Pi_n f(x_i) = f(x_i) \quad \text{per } i = 0, 1, \cdots, n.$$

$\Pi_n f$ è detto **polinomio interpolatore** dei valori $f(x_i)$ nei **nodi** x_i.

Errore di approssimazione

Teorema: stima dell'errore di interpolazione

Dati $n+1$ nodi di interpolazione x_i per $i = 0, 1, \cdots, n$. Sia f una funzione derivabile con continuità $n+1$ volte in un intervallo I contenente tutti i nodi di interpolazione e sia Π_n il polinomio interpolatore nei nodi x_i, allora per ogni $x \in I$, esiste un punto $\xi \in I$ tale che

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i).$$

Polinomi

Un polinomio di **grado** N, con N intero non negativo, è una funzione del tipo

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N = \sum_{i=0}^{N} a_i x^i$$

dove $a_i \in \mathbb{R}$ per $i = 0, 1, 2, \cdots, N$ sono i **coefficienti** del polinomio.
Il polinomio è individuato dai coefficienti che devono essere memorizzati in un vettore.

In MATLAB i coefficienti devono essere **ordinati** a partire da quello corrispondente al termine di grado **più elevato** fino a quello di grado zero.

I coefficienti nulli vanno esplicitati.

Ad esempio al polinomio $p(x) = 1 - 2x + 4x^3$ si associa il vettore `p=[4 0 -2 1]`.

Algoritmo di Horner–Ruffini

L'algoritmo di Horner–Ruffini permette di calcolare il valore di un polinomio in un punto ad un costo computazionale inferiore rispetto all'uso della formula

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{i=0}^{n} a_i x^i.$$

Consideriamo il polinomio

$$p(x) = 1 - 2x + 5x^2 + 4x^3;$$

questo si può scrivere anche nella forma seguente:

$$p(x) = 1 + x \cdot (-2 + x(5 + 4x)).$$

Contiamo le operazioni

- Nel primo caso: 6 (1+2+3) moltiplicazioni + 3 somme per ciascuna componente di x
- Nel secondo caso: 3 moltiplicazioni + 3 somme per ciascuna componente di x

Algoritmo di Horner–Ruffini

In generale il polinomio

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

può essere scritto nella forma di Horner:

$$p(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + x \cdot (\cdots (a_{n-1} + a_n x)))).$$

Numero di operazioni

- Nel primo caso: $n(n+1)/2$ $(1+2+\cdots+n)$ moltiplicazioni $+$ n somme per ciascuna componente di x

  Totale $\frac{n}{2}(n+3)N$ se N è il numero delle componenti di x

- Nel secondo caso: n moltiplicazioni $+$ n somme per ciascuna componente di x

  Totale $2nN$ se N è il numero delle componenti di x

polyval

La function polyval valuta il valore di un polinomio in una griglia di punti usando l'algoritmo di Horner_Ruffini.

```
y=polyval(p,x)
```

restituisce il vettore y contenente i valori di un polinomio di grado n calcolati nei punti x. Il vettore p di $n+1$ componenti deve contenere i coefficienti del polinomio corrispondenti alle potenze in ordine decrescente. Quindi per calcolare il valore del polinomio $p(x) = 1 + 2x - 4x^3$ nei punti x distribuiti in maniera equispaziata nell'intervallo $[a, b]$ si può usare la seguente sequenza di comandi:

```
>> x=linspace(-1,1,101);
>> p=[-4 0 2 1];
>> y=polyval(p,x);
```

polyfit

Per calcolare i coefficienti del polinomio interpolatore si usa il comando:

```
p=polyfit(xn,yn,N)
```

dove:

`xn`	vettore contenente i nodi;
`yn`	vettore contenente i valori della funzione;
`N`	grado del polinomio interpolatore;
`p`	vettore dei coefficienti del polinomio interpolatore.

Osservazione

- Se `N=length(xn)-1` allora si costruisce il polinomio interpolatore di grado N.
- Se `N<length(xn)-1` allora si costruisce il polinomio di grado N nel senso della migliore approssimazione ai **minimi quadrati**.

Funzione di Runge

Si consideri la **funzione di Runge** $f(x) = \dfrac{1}{1+25x^2}, \quad x \in [-1, 1]$.

Esercizio

- Interpolare con polinomi di grado `n=2:2:12`, la funzione data, usando $n+1$ punti equispaziati nell'intervallo $[-1, 1]$.
- Confrontare il grafico di ciascun polinomio interpolatore con quello della funzione data.
- Calcolare per ciascun valore di n l'errore commesso ossia

$$E_n = \max_{a \leq x \leq b} |f(x) - \Pi_n f(x)|$$

  Costruire un vettore contenente gli errori ottenuti per ciascun valore di n e riportare gli errori in un grafico in scala semilogaritmica `semilogy(n,E)`.

Traccia per la risoluzione dell'esercizio

1. Assegnare la funzione.
2. Assegnare un vettore che contiene i valori di n.
3. Costruire il vettore `xx` dei punti per valutare tutti i polinomi.
4. Valutare la funzione in `xx` (risultato `yy`).
5. Per ogni valore di n (`for i=1:length(n)`) eseguire la seguente sequenza:
   - Costruire il vettore `x` dei nodi con il comando `linspace`.
   - Valutare la funzione nei nodi.
   - Trovare i coefficienti del polinomio con il comando `polyfit`.
   - Valutare il polinomio nei punti `xx` con il comando `polyval` (risultato `py`).
   - Plottare la funzione e il polinomio di grado n (inserire una pausa `pause`).
   - Calcolare l'errore

     ```
     E(i)=norm(yy-py,inf)/norm(yy,inf).
     ```
6. Plottare l'errore con il comando `semilogy(n,E)`.

Interpolazione di Chebyshev

Il fenomeno di Runge può essere evitato utilizzando opportune distribuzioni di nodi.
Nell'intervallo $[a, b]$ consideriamo i nodi x_i dati da:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i \quad \text{con } \hat{x}_i = -\cos(\frac{\pi i}{n}), \; i = 0, \ldots, n$$
$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\tilde{x}_i \quad \text{con } \tilde{x}_i = -\cos(\frac{2i+1}{n+1}\frac{\pi}{2}), \; i = 0, \ldots, n$$

I punti $\hat{x}_i$ e $\tilde{x}_i$ si dicono **nodi di Chebyshev**.

Teorema di Bernstein

Sia $f : [a, b] \to \mathbb{R}$ una funzione di classe $\mathbf{C}^1$. Sia Π_n il polinomio interpolatore di grado n costruito usando i nodi di Chebyshev. Allora

$$\lim_{n\to\infty} \|f - \Pi_n f\|_\infty = 0.$$

Function `chebyshev`

La function `chebyshev` calcola il valore del polinomio interpolatore usando il seguente comando

```
yc=chebyshev(f,a,b,xx,n)
```

dove

`f`	nome della funzione che si vuole interpolare;
`a,b`	estremi dell'intervallo;
`xx`	punti in cui si vuole valutare il polinomio;
`n`	grado del polinomio;
`yc`	valore del polinomio nei punti `xx`.

Traccia

- costruzione dei nodi di Chebyshev sull'intervallo $[-1, 1]$: `xhat=-cos((0:n)*pi/n)`;
- nodi di Chebyshev sull'intervallo $[a, b]$: `xc=(a+b)/2+(b-a)*xhat/2`;
- valutazione della funzione nei nodi: `yc=f(xc)`;
- coefficienti del polinomio: `pc=polyfit(xc,yc,n)`;
- valutazione del polinomio: `yc=polyval(p,xx)`.

Esercizio

Esercizio

Si consideri la funzione di Runge. Per `n=2:2:12`, confrontare il polinomio interpolatore di grado n costruito usando $n+1$ nodi equispaziati nell'intervallo $[-1,1]$ con quello di Chebyshev di grado n.
Riportare i grafici della funzione di Runge e dei due polinomi di grado n in una stessa figura.
Calcolare anche i corrispondenti errori e riportarli in un grafico in scala semilogaritmica.

Funzioni MATLAB per la manipolazione di polinomi

Funzione	**Significato**
poly	Costruisce un polinomio con assegnate radici
polyval	Valuta un polinomio su una griglia di punti x
polyvalm	Valuta un polinomio con una matrice come argomento
residue	Ritorna lo sviluppo parziale in frazioni del rapporto fra due polinomi
polyfit	Approssimazione di dati con polinomi
polyder	Derivata di un polinomio
conv	Moltiplicazione di polinomi
deconv	Divisione di polinomi
roots	Calcolo delle radici di un polinomio

Interpolazione a tratti

Dato un'intervallo $I = [a, b]$, si introduce una partizione mediante un numero finito di punti

$$a = x_0 < x_1 < \cdots < x_m = b;$$

$I_k = [x_{k-1}, x_k]$, $k = 1, \cdots, m$ indica il k-esimo sottointervallo.

Definizione

Si definisce **polinomio a tratti** una funzione $g : [a, b] \to \mathbb{R}$ tale che

$$g(x) = p_n(x) \quad \forall x \in I_k,$$

essendo $p_n(x)$ un polinomio di grado n.

Interpolazione lineare a tratti

Sia $f : [a, b] \to \mathbb{R}$ una funzione sufficientemente regolare.

Problema

costruire un polinomio lineare a tratti che interpoli la funzione f nei nodi x_i, $i = 0, \ldots, n$.

Consideriamo la partizione dell'intervallo $[x_0, x_n]$ data dai nodi x_i. Quindi su ciascun intervallino $[x_{i-1}, x_i]$ $i = 1, \ldots, n$ il polinomio interpolatore a tratti è

$$g(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

Stima dell'errore di approssimazione

Sia $H = (b-a)/n$ ed f una funzione continua insieme alle sue derivate prima e seconda. Sia g il polinomio lineare a tratti definito prima.
Per ogni $i = 1, \cdots, n$ esiste un punto $\eta_i \in [x_{i-1}, x_i]$ tale che

$$f(x) - g(x) = \frac{f''(\eta_i)}{2}(x - x_{i-1})(x - x_i) \quad \text{per } x \in [x_{i-1}, x_i],$$

da cui segue la seguente maggiorazione:

$$\max_{1 \le i \le n} \max_{x_{i-1} \le x \le x_i} |f(x) - g(x)| \le \max_{1 \le i \le n} \frac{(x_i - x_{i-1})^2}{2} \max_{x_{i-1} \le x \le x_i} |f''(x)| \le \frac{H^2}{8} \max_{a \le x \le b} |f''(x)|.$$

Interpolazione a tratti

Function **interp1**

`yy=interp1(x,y,xx,metodo)`

`x, y` specificano le coordinate dei punti di interpolazione.
`xx` sono i punti in cui si vuole valutare il valore interpolato.
`metodo` è una stringa di caratteri che specifica il metodo da utilizzare:

- `metodo='nearest'` si sceglie il valore nel nodo di interpolazione più vicino;
- `metodo='linear'` interpolazione lineare a tratti;
- `metodo='spline'` interpolazione con spline cubica;

Stima dell'errore

Esercizio

Usare l'interpolazione lineare a tratti per approssimare le funzioni $f_1(x) = \sin x$ in $[-\pi, \pi]$ e la funzione di Runge. Calcolare l'errore relativo in funzione del numero di suddivisioni usate.

Suggerimenti

- Suddividere l'intervallo assegnato in `n=2:2:200` parti.
- Valutare l'errore nei punti `xx=linspace(a,b,1001)` essendo $[a, b]$ l'intervallo di interesse.
- Per verificare l'ordine di convergenza plottare l'errore in scala bilogaritmica `loglog(n,E)`. Ricordando che la stima dell'errore è

  $$E(H) \approx CH^2 \quad \text{essendo } H = (b-a)/n,$$

  calcolando il logaritmo ad entrambi i membri, si ottiene:

  $$\log E(h) \approx \log C + 2 \log h.$$