Interpolazione polinomiale.

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

April 11, 2017

Introduzione

In questa lezione desideriamo introdurre dei metodi per determinare l'interpolante polinomiale di grado n, nei nodi a due a due distinti x_0, ..., x_n, di una funzione continua $f : [a, b] \to \mathbb{R}$, ovvero $p_n \in \mathbb{P}_n$ tale che

$$p_n(x_i) = f(x_i), i = 0, \ldots, n.$$

Mostreremo come farlo mediante

- la formulazione di Newton di p_n e la valutazione di p_n in un punto x mediante l'algoritmo di Horner;
- le routines Matlab `polyfit`, `polyval`.

Differenze divise

Siano

- $f : [a, b] \to \mathbb{R}$ una funzione continua;
- x_0, ..., x_n nodi a due a due distinti.

Dati $x_0^*, \ldots, x_k^* \in \{x_0, \ldots, x_n\}$, a due a due distinti, la quantità

$$f[x_0^*, \ldots, x_k^*] = \begin{cases} \frac{f[x_1^*,\ldots,x_k^*]-f[x_0^*,\ldots,x_{k-1}^*]}{x_k^*-x_0^*} & \text{se } k > 0 \\ f[x_0^*] = f(x_0) & \text{altrimenti,} \end{cases}$$

si chiamano differenze divise.

Differenze divise

Il polinomio p_n, descritto nella formulazione di Newton

$$p_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k](x - x_0) \cdot \ldots \cdot (x - x_{k-1})$$

è tale che

$$p_n(x_k) = f(x_k), \; k = 0, \ldots, n.$$

Risulta quindi rilevante calcolare

$$f[x_0, \ldots, x_k]$$

per $k = 0, \ldots, n$.

Differenze divise e polinomio interpolatore

Il polinomio p_n, descritto nella formulazione di Newton

$$p_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k](x - x_0) \cdot \ldots \cdot (x - x_{k-1})$$

è tale che

$$p_n(x_k) = f(x_k), \ k = 0, \ldots, n.$$

Differenze divise e polinomio interpolatore

Un pseudocodice è il seguente, dove $c_k = f[x_0, \ldots, x_k]$.

```
c(0) = f(x(0))
for i = 1, ... , n
   d = x(i) - x(i-1)
   u = c(i-1)
   for j = i - 2, ... , 0 step -1
      u = u * (x(i) - x(j)) + c(j)
      d = d * (x(i) - x(j))
   end for j
   c(i) = (f(x(i)) - u)/d
end for i
```

Differenze divise e polinomio interpolatore: esercizio

Si implementi un codice Matlab/Octave che calcola tali coefficienti $c(0), \dots, c(n)$, utilizzando quale intestazione:

```
function c = polnewton (x,y)
% POLNEWTON Calcola i coefficienti del polinomio interpolatore
% utilizzando la forma di Newton
%
% Uso:
% c = polnewton (x,y)
%
% Dati di ingresso:
% x vettore dei nodi
% y vettore dei valori della funzione da interpolare nei nodi
%
% Dati di uscita:
% c vettore colonna dei coefficienti ordinati per indici
% crescenti (c_0, c_1, ... )
```

Differenze divise e polinomio interpolatore: esercizio

Nota.

Si osservi che il precedente pseudocodice non calcola le differenze divise attraverso una tabella, ma calcola esclusivamente i coefficienti utili per la determinazione del polinomio interpolatore.

Differenze divise e polinomio interpolatore: esercizio

Alcune avvertenze:

- ricordare che in Matlab gli indici dei vettori cominciano da 1 e non da 0;
- ricordare che per avere uno step di -1 nel ciclo for, bisogna effettuare una chiamata del tipo:

  ```
  for j = i:-1:1
    ...
  end
  ```

Differenze divise e polinomio interpolatore: esercizio

Per valutare il polinomio interpolatore

$$p_n(x^*) = \sum_{k=0}^{n} c_k (x^* - x_0) \cdots (x^* - x_{k-1})$$

in un generico punto x^* si usa l'algoritmo di Horner (che richiede $2n$ addizioni e n moltiplicazioni). Un pseudocodice è il seguente

```
u = c(n)
for j = n - 1, ... , 0 step -1
   u = u * (x - x(j)) + c(j)
end for j
```

Differenze divise e polinomio interpolatore: esercizio

Si implementi un codice Matlab/Octave che valuta il polinomio p_n in un vettore di ascisse x^*, utilizzando quale intestazione:

```
function fxstar = horner (x,c,xstar)
% HORNER Calcola il valore del polinomio interpolatore in x^*
% utilizzando la forma di Newton e l'algoritmo di Horner
%
% Uso:
% fxstar = horner (x,c,xstar)
%
% Dati di ingresso:
% x vettore dei nodi
% c vettore dei coefficienti della forma di Newton
% ordinati per indici crescenti (c_0, c_1, ... )
% xstar valore in cui si vuole valutare il polinomio
%
% Dati di uscita:
% fxstar valore di P(x^*)
```

I comandi `polyfit` e `polyval`

Il comando Matlab `polyfit` viene utilizzato per determinare i coefficienti del polinomio interpolante le coppie (x_k, y_k) per $k = 0, \ldots, n$

```
>> help polyfit
 polyfit Fit polynomial to data.
    P = polyfit(X,Y,N) finds the coefficients of a polynomial
     P(X) of degree N that fits the data Y best in a
    least-squares sense. P is a row vector of length N+1
    containing the polynomial coefficients in descending
    powers, P(1)*X^N + P(2)*X^(N-1) +...+ P(N)*X + P(N+1).
```

Si noti che il vettore **p** codifica il polinomio interpolatore di grado n

$$p(x) = p_1 x^n + p_2 x^{n-2} + \ldots + p_{n+1}.$$

I comandi `polyfit` e `polyval`

Noto il vettore **p** è possibile valutare il polinomio associato $p(x) = \sum_{i=1}^{n+1} p_i x^{n-i+1}$ nelle ascisse **x** mediante `polyval`

```
>> help polyval
polyval Evaluate polynomial.
    Y = polyval(P,X) returns the value of a polynomial P
    evaluated at X. P is a vector of length N+1 whose
    elements are the coefficients of the
    polynomial in descending powers.

        Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)
```

Si noti che nel caso di più valori di ascisse da valutare x_i, si ha che $p(x_i) = y_i$, essendo x_i, y_i rispettivamente la i-sima componente dei vettori **x**, **y**.

Esercizio

Si adatti il pseudocodice

```
c(0) = f(x(0))
for i = 1, ... , n
   d = x(i) - x(i-1)
   u = c(i-1)
   for j = i - 2, ... , 0 step -1
      u = u * (x(i) - x(j)) + c(j)
      d = d * (x(i) - x(j))
   end for j
   c(i) = (f(x(i)) - u)/d
end for i
```

al caso in cui gli $n+1$ nodi in $[a,b]$ siano equispaziati con $a = x_0$, e $b = x_n$ in cui cioè $x_k = a + kh$, con $k = 0, \ldots, n$, $h = (b-a)/n$.