LABORATORIO DI CALCOLO NUMERICO

Laurea in Statistica e Informatica

Esercitazione sull'interpolazione e approssimazione polinomiale

Prof. Stefano De Marchi

Padova, 17 novembre 2010

1 Comando `repmat`

Il comando, sostanzialmente serve per fare copie di una matrice . Vediamo un esempio che ci chiarisce la cosa. `repmat`

```
>> repmat([1,2;3,4],2,2)

ans =

     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
```

2 Polinomio d'interpolazione in forma di Lagrange

Dati $n+1$ coppie $\{x_i, y_i\}$, $i = 1, \ldots, n+1$, il polinomio d'interpolazione di grado n in forma di Lagrange si scrive come

$$p_n(x) = \sum_{i=1}^{n+1} l_i(x) y_i \tag{1}$$

dove l_i è un polinomio elementare di Lagrange di grado n definito come

$$l_i(x) = \prod_{i=1, i\neq j}^{n+1} \frac{x - x_j}{x_i - x_j} \, .$$

Osserviamo che la (2) può interpretarsi anche come un prodotto scalare tra i vettori $\mathbf{y} = (y_1, \ldots, y_{n+1})^T$ e $\mathbf{l} = (l_1(x), \ldots, l_{n+1}(x))^T$.

Come appena osservato, nella valutazione del polinomio d'interpolazione $p_n(x)$ su un insieme di *punti target*, $\bar{x}$, che sono in generale diversi dai punti d'interpolazione x_i ed in numero maggiore (si pensi al plot del polinomo p_n o ad una stima dell'errore d'interpolazione), sarà opportuno disporre di una funzione che consenta di valutare il l'i-esimo polinomio elementare di Lagrange l_i nel vettore $\bar{x}$. Per far questo, tramite il comando `repmat`, possiamo avvalerci della funzione

```
function l = lagrange(i,x,xbar)
%--------------------------
% i=indice del polinomio
% x=nodi d'interpolazione
% xbar= punti di valutazione
%       (vettore colonna!)
%
% l=vettore dell'iesimo pol.
%   di Lagrange su xbar
%--------------------------
n = length(x); m = length(xbar);

l = prod(repmat(xbar,1,n-1)-repmat(x([1:i-1,i+1:n]),m,1),2)/...
prod(x(i)-x([1:i-1,i+1:n]));
```

Nota bene. Una volta costruiti gli $n+1$ vettori colonna **l**, li si assemblano in una matrice, sia essa L, e con il prodotto **p=L*y** si conoscerà il valore del polinomio p su tutti i punti target.

2.1 Punti di Chebyshev e di Chebyshev-Lobatto

I *punti di Chebyshev* sono gli zeri dei polinomi di Chebyshev di prima specie, appartengono all'intervallo $[-1, 1]$ e sono così definiti:

$$x_i^{(C)} = \cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i = 1, \dots n\,.$$

Quelli di *Chebsyshev-Lobatto* includono anche gli estremi dell'intervallo e sono definiti

$$x_i^{(CL)} = \cos\left(\frac{(i-1)\pi}{(n-1)}\right), \quad i = 1, \dots n\,.$$

3 Polinomio d'interpolazione in forma di Newton

Dati $n+1$ coppie $\{x_i, y_i\}$, $\;i = 1, \dots, n+1$, il polinomio d'interpolazione di grado n in *forma di Newton* si scrive compattamente come

$$p_n(x) = \sum_{i=1}^{n+1} b_i m_{i-1}(x) \tag{2}$$

dove ciascuna funzione m_{i-1} è un polinomio di grado $i-1$ definito

$$m_{i-1}(x) = (x - x_1) \cdots (x - x_{i-1})$$

e per $i = 1$, $\;m_0(x) = 1$. I coefficienti b_i altro non sono che le *differenze divise di ordine* $i-1$. Nel caso in cui $y_i = f(x_i)$ allora $b_i := f[x_1, ..., x_i]$.

La funzione Matlab che calcola il vettore delle differenze divise `b` è la seguente:

```
function d = DiffDiv(nodi,valori)
%------------------------------
% nodi, valori: vettori
%
% b = vettore differenze divise
%------------------------------
n = length(nodi);
b=valori;
for i = 1:n
  for j = 1:i-1
    b(i) = (b(i)-b(j))/(nodi(i)-nodi(j));
  end
end
```

Per calcolare il polinomio p_n in un punto z, noti il vettore dei nodi `x` e quello delle differenze divise `b`, si può utilizzare lo *schema di Hörner* come segue:

```
p=b(n);
 for i=n-1:-1:1
   p=(z-x(i))*p+b(i);
 end
```

Un'alternativa potrebbe essere quella di costruirsi il vettore `m` con $m_i = m_i(z)$ (sopra definiti), ovvero

$$\texttt{m} = [1, z - x_1, m_2(z - x_2), \ldots, m_{n-1}(z - x_n)]$$

per cui alla fine il polinomio in z si otterrà come segue `p=b'*m`.

4 Comandi `polyfit`, `polyval` e `spline`

- Del comando `polyfit` consideriamo i seguenti due casi

  ```
  p = polyfit(x,y,n) [p,s,mu] = polyfit(x,y,n)
  ```

  Vediamoli in dettaglio.

  1. `p=polyfit(x,y,n)`. Restituisce nel vettore `p` i coefficienti del polinomio di grado $n \leq$ `length`(`x`) che approssima, nel senso dei minimi quadrati i dati memorizzati in `y`.
  2. `[p,s,mu]=polyfit(x,y,n)`. In questo caso si determinano i coefficienti del polinomio approssimante non in x ma in
  $$\hat{x} = \frac{x - \mu_1}{\mu_2}$$
  dove $\mu_1 =$ `mean`(x) (media dei valori di `x`) e $\mu_2 =$ `std`(x) (la deviazione standard dei valori di `x`).

- Il comando `y=polyval(p,x)`. Consente di valutare in x un polinomio di coefficienti `p` (vettore), ovvero
$$y = p_1 x^n + p_2 x^{n-1} + \ldots + p_{n+1}\,.$$
Se il vettore `p` proviene dalla chiamata `[p,s,mu]=polyfit(p,x)` allora si userà `y=polyval(p,x,[],mu)`.

- Il comando `spline`, consente di determinare la spline cubica interpolante. La chiamata

  `s=spline(x,y,xx)`

  determina la spline cubica `s` sulla suddivisione `x` e valori `y`, nell'insieme di punti target `xx`. Il vettore `s` ha la stessa lunghezza di `xx`.

Esercizi proposti

1. Si determini il polinomio d'interpolazione di gradi $n = 5, \ldots, 10$, in forma di Lagrange, della funzione di Runge
$$g(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5]$$
su nodi equispaziati. Si facciano anche i plot della funzione e del polinomio interpolante.

2. Si determini il polinomio d'interpolazione di grado 10, in forma di Newton, della funzione di Runge
$$g(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1]$$
su nodi di Chebyshev.

3. Si consideri la *funzione errore*,
$$\texttt{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt\,,$$
in Matlab `erf`, su un insieme di punti equispaziati `x=(-5.0:0.1:5.0)'`. Si determinino i coefficienti del polinomio approssimante calcolati con `polyfit` di gradi variabili da 4 a 10. Usare `polyval` per valutare il polinomio approssimante. Perchè il fitting non funziona?

4. Si consideri la funzione $f(x) = \sin(x) + \sin(5x), \quad x \in [0, 2\pi]$.

   (i) Costruire la spline cubica interpolante.

   (ii) Si costruisca anche il polinomio approssimante di grado 8 sul sottinsieme di punti equispaziati `x=0:0.1:3.0` ottenuto con `polyval` nella modalità richiesta al punto precedente. Se ne calcoli anche la norma 2 dell'errore.

   Quale delle modalità (i) e (ii) approssima meglio $f(x)$?

Tempo: 2 ore.