LABORATORIO DI ANALISI NUMERICA

Laurea Magistrale in Statistica e Informatica

Esercitazione sull'interpolazione e approssimazione polinomiale

Prof. Stefano De Marchi

Padova, 12 novembre 2009

1 Alcuni utili comandi Matlab

- 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`.

2 Polinomio d'interpolazione in forma di Newton

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

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

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 $i = 1$, $\quad m_0(x) = 1$. 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 x, noti il vettore dei nodi `nodi` 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=(x-nodi(i))*p+b(i);
 end
```

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

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

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

Esercizi proposti

1. 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.

2. 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?

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

   (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 $f(x)$?