Analisi Numerica I
Approssimazione polinomiale

Ana Alonso

ana.alonso@unitn.it

15 novembre 2019

I polinomi

Un polinomio di grado n in Matlab si rapresenta mediante un vettore di $n+1$ componenti che contiene i coefficienti del polinomio ordinati da quello di grado n a quello di grado 0.

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

```
>> p = [3 -2 0 1 -5]
```

- `polyval(p,z)` calcola il valore di p in uno o più punti.
  ```
  >> z=[-1 0 1];
  >> pz = polyval(p,z)
  ```
- `roots(p)` calcola le radici di `p`.
- `polyder(p)` calcola i coefficienti del polinomio derivata di `p`.
  ```
  >> dp = polyder(p)
  ```
- `polyint(p)` calcola i coefficienti di una primitiva (quella che si annulla in $x = 0$) di `p`.
  ```
  >> q = polyint(p)
  ```

Esercizio

- Disegnare il grafico del polinomio $p(x) = 3x^4 - 2x^3 + x - 5$ nell'intervallo $[-1, 1]$.
- Calcolare
$$\int_{-1}^{1} p(x)\, dx$$

Il comando `polyfit`

- Se `x` e `y` sono due vettori di $n+1$ componenti, il comando `p=polyfit(x,y,n)` calcola il polinomio interpolatore dei dati $\{(x_i, y_i)\}_{i=0}^n$.
- Se $m \neq n$ il comando `p=polyfit(x,y,m)` calcola i coefficienti del polinomio di grado `m` che meglio approssima i dati $\{(x_i, y_i)\}_{i=0}^n$ nel senso dei minimi quadrati:

$$p \in \mathbb{P}_m \, : \sum_{i=0}^{n} (p(x_i) - y_i)^2 = \min_{q \in \mathbb{P}_m} \sum_{i=0}^{n} (q(x_i) - y_i)^2 \,.$$

Esercizio

Data la funzione $f(x) = \frac{x^3+1}{x+4}$ disegnare il grafico di $f(x)$ e

- del polinomio che interpola $f(x)$ nei nodi $x_0 = -1$, $x_1 = 0$, $x_2 = 1$,
- del polinomio che interpola $f(x)$ nei nodi $x_0 = -1$, $x_1 = -1/2$, $x_2 = 0$, $x_3 = 1/3$, $x_4 = 2/3$, $x_5 = 1$,

nell'intervallo $[-1, 1]$.

Esempio di Runge

Si consideri la funzione

$$f(x) = \frac{1}{1+x^2} \quad x \in I = [-5, 5] .$$

Scrivere un script di Matlab per disegnare:

- la funzione f,
- il grafico del polinomio $\Pi_N f$ che interpola f in $N+1$ punti equispaziati dell'intervallo I.

Si può osservare come

$$\max_{-5 \leq x \leq 5} |f(x) - \Pi_N f(x)| \longrightarrow \infty \quad \text{se } N \to \infty .$$

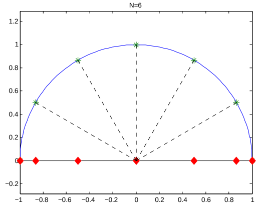
Esempio di Runge

```
fun=@(x) 1./(x.^2+1);
n=input('Grado del polinomio: ');
a=-5;b=5;
h=(b-a)/n;
x=[a:h:b];
fx=fun(x);
p=polyfit(x,fx,n);
xx=linspace(a,b);
pxx=polyval(p,xx);
fxx=fun(xx);
plot(xx,fxx,xx,pxx,x,fx,'*');
legend('1/(x^2+1)','Polinomio intepolatore','Dati');
```

Nodi di Chebishev

Nell'intervallo $[-1, 1]$ sono

$$\hat{x}_i = -\cos\left(\frac{i\pi}{N}\right) \qquad i = 0, \ldots, N\,.$$



E in un intervallo generico $[a, b]$

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i \qquad i = 0, \ldots, N\,.$$

Esempio di Runge

Si consideri di nuovo la funzione

$$f(x) = \frac{1}{1+x^2} \quad x \in I = [-5, 5] \,.$$

Scrivere un script di Matlab per disegnare:

- la funzione f,
- il grafico del polinomio $\widehat{\Pi}_N f$ che interpola f nei $N+1$ nodi di Chebishev dell'intervallo I.

Si può osservare come

$$\max_{-5 \le x \le 5} |f(x) - \widehat{\Pi}_N f(x)| \longrightarrow 0 \quad \text{se } N \to \infty \,.$$

Nodi di Chebishev

```
fun=@(x) 1./(x.^2+1);
n=input('Grado del polinomio: ');
a=-5; b=5;
x=-cos([0:n]*pi/n);
x=(a+b)/2+(b-a)/2*x;
fx=fun(x);
q=polyfit(x,fx,n);
xx=linspace(a,b);
qxx=polyval(q,xx);
fxx=fun(xx);
plot(xx,fxx,xx,qxx,x,fx,'* ');
legend('1/(x^2+1)','Polinomio interpolatore','Dati');
```

Splines

Se x e y sono due vettori di uguale lungheza

```
>> yy = spline(x,y,xx)
```

calcola il valore in xx della spline cubica "not-a-knot" (derivata terza continua in x_1 e x_{N-1}) e che interpola i dati $\{(x_i, y_i)\}_{i=0}^{N}$.

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

restituisce la spline cubica "not-a-knot" che interpola i dati $\{(x_i, y_i)\}_{i=0}^{N}$ come funzione polinomiale a tratti.

Per calcolare il valore della spline cosi costruita in xx si usa il comando

```
>> yy = ppval(s,xx)
```

Esercizio

Per i dati contenuti nella tabella

x_i	0	0.5	1.6	2.9	3.4	4.8	5.2	5.7	6.0
y_i	2.20	2.13	3.32	5.21	5.10	7.05	7.00	8.00	8.25

disegnare il grafico della retta di migliore approssimazione nel senso dei minimi quadrati, del polinomio interpolatore e della spline cubica interpolatoria "not-a-knot".

Forma di Newton del polinomio interpolatore

Dati $\{(x_i, y_i)\}_{i=0}^{N} = \{(x_i, f(x_i))\}_{i=0}^{N}$ sia $\Pi_m f(x)$ il polinomio di grado m che interpola $\{(x_i, y_i)\}_{i=0}^{m}$

$$\Pi_N f(x) = \Pi_{N-1} f(x) + q_N(x)$$

- $q_N(x)$ è un polinomio di grado N.
- $q_N(x_i) = 0$ per $i = 0, 1, \ldots, N-1$.

Quindi

$$q_N(x) = \alpha(x - x_0)(x - x_1) \ldots (x - x_{N-1}) \,.$$

- α è il coefficiente di grado N nel polinomio $\Pi_N f(x)$.
- Si chiama N-esima differenza divisa di Newton e si indica

$$f[x_0, x_1, \ldots, x_N]$$

Forma di Newton del polinomio interpolatore

Chiaramente

$$f[x_0, x_1, \ldots, x_N] = \frac{f(x_N) - \Pi_{N-1} f(x_N)}{(x_N - x_0)(x_N - x_1)\ldots(x_x - x_{N-1})}$$

ma si usa una formula ricorsiva più efficiente.

$$f[x_0, x_1, \ldots, x_N] = \frac{f[x_1, \ldots, x_N] - f[x_0, x_1, \ldots, x_{N-1}]}{x_N - x_0}\,.$$

(Questo perché

$$\Pi_{0,1,\ldots,N} f(x) = \frac{x - x_0}{x_N - x_0} \Pi_{1,\ldots,N} f(x) + \frac{x - x_N}{x_0 - x_N} \Pi_{0,\ldots,N-1} f(x)\,.)$$

Forma di Newton del polinomio interpolatore

Abbiamo visto che

$$\Pi_N f(x) = f[x_0, x_1, \ldots, x_N](x - x_0)(x - x_1)\ldots(x - x_{N-1}) + \Pi_{N-1} f(x)$$

ma questa formula vale anche per $\Pi_{N-1} f(x)$ quindi

$$\begin{aligned} \Pi_N f(x) &= f[x_0, x_1, \ldots, x_N](x - x_0)(x - x_1)\ldots(x - x_{N-1}) \\ &\quad + f[x_0, x_1, \ldots, x_{N-1}](x - x_0)(x - x_1)\ldots(x - x_{N-2}) \\ &\quad + \cdots + f[x_0, x_1](x - x_0) + f[x_0] \end{aligned}$$

e $f[x_0] = y_0$.

Tabella delle differenze divise

$$
\begin{array}{c|llll}
x_0 & f[x_0] & & & \\
x_1 & f[x_1] & f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} & & \\
x_2 & f[x_2] & f[x_1,x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1} & f[x_0,x_1,x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} & \\
x_3 & f[x_3] & f[x_2,x_3] = \frac{f[x_3]-f[x_2]}{x_3-x_2} & f[x_1,x_2,x_3] = \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1} & f[x_0,x_1,x_2,x_3]
\end{array}
$$

Forma di Newton del polinomio interpolatore

$$\Pi_{0,1,2,3}f(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$
$$+(x - x_0)(x - x_1)(x - x_2)f[x_0, x_1, x_2, x_3]$$

$$= f[x_0] + (x - x_0)\big(f[x_0, x_1] + (x - x_1)\big[f[x_0, x_1, x_2] + (x - x_2)f[x_0, x_1, x_2, x_3]\big]\big)$$

Algoritmo di Horner

$$c_1 = f[x_0], \quad c_2 = f[x_0, x_1], \quad c_3 = f[x_0, x_1, x_2], \quad c_4 = f[x_0, x_1, x_2, x_3].$$

$$\begin{array}{l} Pz = c_4 \\ fori = 3 : -1 : 1 \\ \qquad Pz = c_i + (z - x_{i-1})\, Pz \\ end \end{array}$$

Differenze divise

La seguente funzione calcola il valore in **z** del polinomio che interpola i dati $\{(x_i, y_i)\}_{i=1}^{n}$ usando la forma di Newton del polinomio interpolatore e l'algoritmo di Horner.

```
function Pz=interpol(x,y,z)
n=length(x);
A=zeros(n,n);
% Costruzione della tabella
A(:,1)=y;
for j=2:n
  A(j:n,j)=(A(j:n,j-1)-A((j:n)-1,j-1))./(x(j:n)-x((j:n)-(j-1)))';
end
% Horner
Pz=A(n,n);
for k=n-1:-1:1
  Pz=A(k,k)+(z-x(k)).*Pz;
end
```