Francesca Mazzia

Dipartimento di Matematica

Università di Bari

Interpolazione

Interpolazione

Problema dell'interpolazione:
dati i punti $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$
trovare una curva continua che passi per essi.

Interpolazione polinomiale :
dati i punti $(x_0, f_0), (x_1, f_1), \cdots, (x_n, f_n)$
trovare un polinomio di grado n tale che
$p(x_i) = f_i$, per $i = 0, 1, \cdots, n$

Sia P_n l'insieme di tutti i polinomi di grado minore od uguale ad n, n un numero positivo.

Questo insieme può essere generato dai polinomi elementari $1, x, x^2, \ldots, x^n$:

$$p(x) = \sum_{j=0}^{n} p_j x^j.$$

Al variare dei coefficienti p_j si ottengono tutti i polinomi dell'insieme.

Le funzioni $1, x, x^2, \ldots, x^n$ sono linearmente indipendenti.

Infatti il polinomio identicamente nullo si può ottenere solo prendendo tutti i coefficienti $p_j = 0$.

L'insieme di polinomi elementari $\{x^i\}_{i=0}^n$, forma una base di P_n, detta base delle potenze.

Siano (x_i, f_i), per $i = 0, 1, 2, \ldots, n$ delle coppie di punti nel piano cartesiano.

Supponiamo che i punti x_i, detti nodi, siano distinti, $x_i \neq x_j$ per $i \neq j$.

Vogliamo determinare un polinomio $p(x) \in P_n$ tale che

$$
\begin{aligned}
\sum_{j=0}^{n} p_j x_0^j &= f_0 \\
\sum_{j=0}^{n} p_j x_1^j &= f_1 \\
&\vdots \\
\sum_{j=0}^{n} p_j x_n^j &= f_n
\end{aligned}
$$

È questo il problema dell' interpolazione dei dati (x_i, f_i) mediante un polinomio, detto polinomio interpolante.

Introduciamo la matrice

$$
A = \begin{pmatrix}
1 & x_0 & x_0^2 & \dots & x_0^n \\
1 & x_1 & x_1^2 & \dots & x_1^n \\
1 & x_2 & x_2^2 & & x_2^n \\
\vdots & \vdots & & \ddots & \vdots \\
1 & x_n & x_n^2 & \dots & x_n^n
\end{pmatrix};
$$

il problema si pone nella forma:

$$A \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix},$$

e si riduce a risolvere un sistema lineare di dimensioni $n + 1$.

La matrice A, detta matrice di Vandermonde è non singolare essendo:

$$det(A) = \prod_{i>j} (x_i - x_j)$$

Quindi il sistema precedente ammette una unica soluzione e ciò permette di ricavare il polinomio interpolante.

La matrice di Vandermonde è spesso mal condizionata e, nel caso in cui i dati da interpolare siano molti, la soluzione del sistema precedente risulta molto costosa.

Il problema dell'interpolazione può essere risolto più efficientemente usando altre basi di P_n.

Se le quantità f_i sono i valori assunti da un polinomio $f \in P_n$ nei punti x_i, allora il polinomio $p(x)$ coincide con $f(x)$.

$f(x) - p(x)$ è un polinomio di P_n il quale si annulla negli $n + 1$ punti $x_0, x_1, \ldots, x_n$.

Ma un polinomio di grado minore o uguale ad n non può avere più di n radici a meno che non sia il polinomio nullo in tutti i punti, e quindi $f(x) - p(x) \equiv 0$.

Questo dimostra l'unicità del polinomio interpolante.

Interpolazione di Lagrange

Consideriamo i polinomi, tutti di grado n:

$$l_j(x) = \frac{\prod_{i=0, i\neq j}^{n} (x - x_i)}{\prod_{i=0, i\neq j}^{n} (x_j - x_i)} \qquad \text{per } j = 0, 1, \ldots, n.$$

Essi godono delle seguenti proprietà:

1) Per $j, k = 0, 1, \ldots, n$ risulta

$$l_j(x_k) = \delta_{jk} = \begin{cases} 1 & \text{per } j = k \\ 0 & \text{altrimenti.} \end{cases}$$

2) Sono linearmente indipendenti e quindi formano una base di P_n. Infatti da

$$\sum_{i=0}^{n} \alpha_i l_i(x) = 0$$

si ha, per $k = 0, 1, \ldots, n$

$$\sum_{i=0}^{n} \alpha_i l_i(x_k) = \alpha_k l_k(x_k) = 0$$

da cui si ricava che tutti i coefficienti α_k devono necessariamente essere nulli.

L'insieme $\{l_i(x)\}_{i=0}^n$ è detta base di Lagrange. Imponendo le condizioni di interpolazione si ha

$$\sum_{j=0}^{n} p_j l_j(x_0) = f_0$$
$$\sum_{j=0}^{n} p_j l_j(x_1) = f_1$$
$$\dots$$
$$\sum_{j=0}^{n} p_j l_j(x_n) = f_n$$

La matrice dei coefficienti è la matrice identica. Si ha $p_j = f_j$ e il polinomio interpolante risultante diventa:

$$p(x) = \sum_{j=0}^{n} f_j l_j(x).$$

Messo nella forma precedente il polinomio interpolante è detto polinomio di Lagrange.

Esempio: Consideriamo $n = 1$ (interpolazione lineare). In questo caso si ha

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}$$
$$l_1(x) = \frac{x - x_0}{x_1 - x_0}.$$

Il polinomio interpolante è una retta passante per i due punti x_0 e x_1.

Esempio Consideriamo $n = 2$ (interpolazione quadratica). In questo caso si ha

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$
$$l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$
$$l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Interpolazione di Newton

La formula di interpolazione di Lagrange è molto semplice nella forma, ma costosa se si vogliono aggiungere altri punti.
Le funzioni $l_i(x)$ dipendono da tutti i punti in cui si fa l'interpolazione. Se si vuol aggiungere un altro punto, tutte le funzioni di base cambiano.
Introduciamo un nuova base detta base di Newton

$$\begin{aligned}
&\phi_0(x) = 1,\\
&\phi_1(x) = (x - x_0),\\
&\ \vdots\\
&\phi_n(x) = (x - x_0)(x - x_1)\dots(x - x_{n-1}).
\end{aligned}$$

Ogni $p(x) \in P_n$ potrà scriversi nella forma:

$$p(x) = \sum_{j=0}^{n} a_j \phi_j(x).$$

Imponendo le condizioni di interpolazione si ha:

$$\begin{aligned}
&a_0 = f_0,\\
&a_0 + a_1\phi_1(x_1) = f_1,\\
&a_0 + a_1\phi_1(x_2) + a_2\phi_2(x_2) = f_2,\\
&\vdots\\
&a_0 + a_1\phi_1(x_n) + \ldots a_n\phi_n(x_n) = f_n.
\end{aligned}$$

Si ha quindi un sistema lineare la cui matrice dei coefficienti è triangolare inferiore:

$$N = \begin{pmatrix}
1 & 0 & 0 & \ldots & 0\\
1 & \phi_1(x_1) & 0 & & 0\\
1 & \phi_1(x_2) & \phi_2(x_2) & \ddots & \vdots\\
\vdots & \vdots & & \ddots & 0\\
1 & \phi_1(x_n) & \phi_2(x_n) & \ldots & \phi_n(x_n)
\end{pmatrix}.$$

I coefficienti del polinomio si ottengono risolvendo il sistema:

$$N \begin{pmatrix} a_0\\ a_1\\ \vdots\\ a_n \end{pmatrix} = \begin{pmatrix} f_0\\ f_1\\ \vdots\\ f_n \end{pmatrix}.$$

La matrice è non singolare e quindi i coefficienti sono unici.

I coefficienti a_i dipendono dai valori di $f_0, f_1, \ldots, f_{i-1}$ e $x_0, x_1, \ldots, x_{i-1}$.

Indichiamo la dipendenza con

$$a_i = f[x_0, x_1, \ldots, x_i].$$

Il polinomio interpolante assume la forma

$$p(x) = \sum_{i=0}^{n} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1} (x - x_j)$$

e viene chiamato polinomio di interpolazione di Newton.

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

vengono chiamate differenze divise.

Il sistema triangolare può essere risolto in $O(n^2)$ operazione per calcolare i coefficienti dell'interpolante di Newton. Sfortunatamente i coefficienti di questo sistema possono creare facilemente problemi di overflow e di underflow.

Deriviamo un algoritmo diverso, ma molto più stabile, utilizzando le differenze divise.
Dalla prima equazione del sistema si ricava che:

$$a_0 \equiv f[x_0] = f_0,$$

e dalla seconda

$$a_1 \equiv f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0} = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

Questa espressione è un caso speciale di una relazione più generale:

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}$$

Per dimostrare questa relazione poniamo:
p il polinomio interpolante

$$(x_0, f_0), (x_1, f_1), \ldots, (x_k, f_k)$$

q il polinomio interpolante

$$(x_0, f_0), (x_1, f_1), \ldots, (x_{k-1}, f_{k-1})$$

r il polinomio interpolante

$$(x_1, f_1), (x_2, f_2), \ldots, (x_k, f_k)$$

I coefficienti del termine di grado massimo per questi polinomi sono:
per p : $f[x_0, x_1, \ldots, x_k]$
per q : $f[x_0, x_1, \ldots, x_{k-1}]$
per r : $f[x_1, x_2, \ldots, x_k]$

Ora dimostriamo che

$$p(x) = q(x) + \frac{x - x_0}{x_k - x_0}(r(x) - q(x))$$

Per mostrare questo calcoliamo

$$q(x) + \frac{x - x_0}{x_i - x_0}(r(x) - q(x))$$

in x_i e proviamo che $p(x_i) = f_i$ per $i = 1, n$.
Il risultato segue dall'unicità del polinomio interpolante.

Per $x = x_0$

$$q(x_0) + \frac{x_0 - x_0}{x_k - x_0}(r(x_0) - q(x_0)) = f_0$$

poichè q interpola (x_0, f_0).
Per $x = x_i$ $(i = 1, \ldots, k-1)$

$$q(x_i) + \frac{x_i - x_0}{x_k - x_0}(r(x_i) - q(x_i)) = f_i + \frac{x_i - x_0}{x_k - x_0}(f_i - f_i) = f_i$$

Per $x = x_k$

$$q(x_k) + \frac{x_k - x_0}{x_k - x_0}(r(x_k) - q(x_k)) = r(x_k) = f_k$$

Segue che il coefficiente di x^k in p è :

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}$$

Quello che volevamo dimostrare.

Questa proprietà viene usata per ricavare ricorsivamente le differenze divise, in alternativa all'inversione della matrice N. Si procede secondo lo schema seguente:

$$\begin{array}{llllll}
f_0 & & & & \\
f_1 & f[x_0, x_1] & & & \\
f_2 & f[x_1, x_2] & f[x_0, x_1, x_2] & & \\
\vdots & \vdots & \vdots & & \ddots \\
f_n & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \ldots & f[x_0, \ldots, x_n]
\end{array}$$

La matrice matrice può essere costruita per colonne. Gli elementi diagonali sono i coefficienti del polinomio interpolante di Newton.

Nel caso in cui la funzione $f(x)$ sia derivabile nei nodi, allora le differenze divise sono definite anche se due argomenti coincidono. Infatti si ha, ad esempio:

$$f[x_0, x_0] = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$$

Il precedente risultato si può generalizzare arrivando alla conclusione che

Se la funzione $f(x)$ è derivabile n volte in un punto x, allora:

$$f[\underbrace{x, x, \ldots, x}_{n+1}] = \frac{f^{(n)}(x)}{n!}$$

Se la funzione f è derivabile m volte con derivata $m.ma$ continua, allora esiste un punto ξ appartenente al più piccolo segmento contenente i punti $x_0, x_1, \ldots, x_m$ tale che:

$$f[x_0, x_1, \ldots, x_m] = \frac{f^{(m)}(\xi)}{m!}$$

Esempio: Calcoliamo la tabella delle differenze divise relativa alla funzione
$f(x) = 2x^2 + 4x - 3$ nei punti $x_i = i - 2$, per $i = 0, 1, \ldots, 4$.

-2	-3			
-1	-5	-2		
0	-3	2	2	
1	3	6	2	0
2	13	10	2	0

Si ha che $\forall i \ f[x_i, x_{i+1}, x_{i+2}] = 2$
mentre $f[x_i, x_{i+1}, x_{i+2}, x_{i+3}] = 0$.
Le diagonali della tabella rappresentano i coefficienti delle seguenti diverse espressioni (in quanto utilizzano nodi diversi) del polinomio $f(x)$, ottenute mediante la base di Newton:

$$
\begin{aligned}
& f(x) = \\
& = f[-2] + f[-2,-1](x+2) + f[-2,-1,0](x+2)(x+1) \\
& = -3 - 2(x+2) + 2(x+2)(x+1) \\
& = f[-1] + f[-1,0](x+1) + f[-1,0,1](x+1)(x-0) \\
& = -5 + 2(x+1) + 2x(x+1) \\
& = f[0] + f[0,1](x-0) + f[0,1,2](x-0)(x+1) \\
& = -3 + 6x + 2x(x-1).
\end{aligned}
$$

Esempio: Calcoliamo la $\sqrt{11}$ utilizzando un polinomio di grado 2 ed uno di grado 4. Per ottenere i 2 polinomi occorrono rispettivamente 3 e 5 punti. Consideriamo i nodi $x_0 = 9$, $x_1 = 4$, $x_2 = 16$, $x_3 = 1$ e $x_4 = 25$ di cui è nota la radice quadrata. Per calcolare i coefficienti di entrambi i polinomi è sufficiente costruire la seguente tabella della funzione $\sqrt{x}$:

9	3				
4	2	1/5			
16	4	1/6	-1/210		
1	1	1/5	-1/90	1/1260	
25	5	1/6	-1/270	1/2835	-1/36288

da cui risulta:

$$p_2(x) = 3 + \tfrac{1}{5}(x-9) - \tfrac{1}{210}(x-4)(x-9),$$
$$p_4(x) = p_2(x) + \left(\tfrac{1}{1260} - \tfrac{1}{36288}(x-1)\right)(x-16)(x-4)(x-9).$$

Le approssimazioni ottenute sono:

$$p_2(11) = 3.333 \qquad \text{e} \qquad p_4(11) = 3.297$$

mentre $\sqrt{11} = 3.3166$.

Errore nella interpolazione

Finora abbiamo trattato le ordinate f_i come numeri arbitrari. Possiamo cambiare il punto di vista e assumere che $f_i = f(x_i)$, con f una funzione sufficientemente regolare.

Sia p il polinomio interpolante la f nei nodi $x_0, x_1, \ldots, x_n$.
Cerchiamo un'espressione per l'errore

$$e(t) = f(t) - p(t)$$

con t diverso dai nodi.

Sia $q(u)$ il polinomio di grado $n+1$ che interpola i nodi $x_0, x_1, \ldots, x_n$ e $x_{n+1} = t$. La forma di Newton per $q(u)$ è:

$$q(u) = \sum_{i=0}^{n+1} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1} (u - x_j)$$

$$q(u) = \sum_{i=0}^{n} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1} (u - x_j) +$$
$$f[x_0, x_1, \ldots, x_n, t] \prod_{j=0}^{n} (u - x_j)$$

Quindi

$$q(u) = p(u) + f[x_0, x_1, \ldots, x_n, t]\omega(u)$$

con

$$\omega(u) = \prod_{j=0}^{n} (u - x_j)$$

Per costruzione $q(t) = f(t)$, qundi:

$$f(t) = p(t) + f[x_0, x_1, \ldots, x_n, t]\omega(t)$$

o

$$e(t) = f(t) - p(t) = f[x_0, x_1, \ldots, x_n, t]\omega(t)$$

Consideriamo adesso la funzione

$$r(u) = f(u) - p(u) - f[x_0, x_1, \ldots, x_n, t]\omega(u)$$

con t fissato e u variabile. Poichè $p(x_i) = f(x_i)$ e $\omega(x_i) = 0$

$$r(x_i) = f(x_i) - p(x_i) - f[x_0, x_1, \ldots, x_n, t]\omega(x_i) = 0$$

inoltre

$$r(t) = f(t) - p(t) - f[x_0, x_1, \ldots, x_n, t]\omega(t) = 0$$

Se I è il più piccolo intervallo contenente $x_o, x_1, \ldots, x_n, t$ allora

$$r(u) \text{ ha almeno } n+2 \text{ zeri in } I$$

Per il Teorema di Rolle, fra ognuno di questi zeri vi è uno zero di $r'(u)$

$$r'(u) \text{ ha almeno } n+1 \text{ zeri in } I$$

e

$$r''(u) \text{ ha almeno } n \text{ zeri in } I$$

Continuando troviamo che

$$r^{(n+1)}(u) \text{ ha almeno 1 zero in } I$$

Sia ξ un zero di $r^{(n+1)}(u)$ in I. Allora poichè $p(u)$ è un polinomio di grado n, $p^{(n+1)}(u) = 0$, inoltre $\omega^{(n+1)}(u) = (n+1)!$, quindi

$$0 = r^{(n+1)}(\xi) = f^{(n+1)}(\xi) - f[x_0, \ldots . x_n, t](n+1)!$$

o

$$f[x_0, \ldots . x_n, t] = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Quindi se p è il polinomio interpolante le f nei nodi $x_0, \ldots, x_n$

$$e(x) = f(x) - p(x) = \omega(x)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Mentre il secondo termine dipende dalla funzione f e dalla sua regolarità, il primo termine dipende esclusivamente dalla scelta dei punti in cui si esegue l'interpolazione. Supponiamo che i punti siano tutti contenuti nell'intervallo $[a, b]$. Non è restrittivo supporre che $[a, b]$ sia $[-1, 1]$. In caso contrario possiamo usare la trasformazione $t = \frac{2x-b-a}{b-a}$. Poniamo ora:

$$\|f\|_\infty = \max_{-1<x<1} |f(x)|.$$

$\|\cdot\|_\infty$ è una norma e quindi soddisfa alle proprietà già viste per le norme di vettori, cioè:

N1) $\|f\|_\infty = 0$ se e solo se $f(x) = 0$,

N2) $\|\lambda f\|_\infty = |\lambda| \|f\|_\infty$,

N3) $\|fg\|_\infty \leq \|f\|_\infty \|g\|_\infty$.

Si ha

$$\|e\|_\infty \leq \|\omega\|_\infty \|f[x_0, x_1, \ldots, x_n, x]\|_\infty.$$

Si può dimostrare che esiste una scelta dei punti di interpolazione che minimizza la norma $\|\omega\|_\infty$.
Tale scelta richiede di prendere i seguenti nodi in $[-1, 1]$:

$$x_k = \cos \frac{(2k+1)\pi}{2(n+1)}, \qquad k = 0, 1, \ldots, n.$$

o in $[a, b]$

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{(2k+1)\pi}{2(n+1)}, \quad k = 0, \ldots, n.$$

Questi nodi sono detti nodi di Chebyshev. Quando i nodi da scegliere non sono fissati a priori, conviene prendere questi ultimi. Nell'esempio seguente la stessa funzione è interpolata su nodi equidistanti e su quelli di Chebyshev. Il polinomio interpolante con nodi equidistanti presenti oscillazioni molto più ampie, specialmente vicino agli estremi dell'intervallo, rispetto al polinomio ottenuto con i nodi di Chebyshev.

Spline

L'interpolazione polinomiale presenta degli aspetti negativi quando i nodi sono molti. I polinomi, essendo di grado elevato, presentano delle forti oscillazioni tra un nodo e l'altro. Supponiamo che i nodi $x_0 < x_1 < \ldots < x_n$ siano contenuti in un intervallo $[a, b]$. Sia $d \geq 1$. Definiamo *funzione spline* di ordine d una funzione $S_d(x)$ tale che:

1. $S_d(x)$ è un polinomio di grado d in ogni intervallo $[x_{i-1}, x_i]$, per $i = 1, 2, \ldots, n$;

2. $S_d^{(k)}(x)$, per $k = 0, 1, \ldots, d-1$ è continua in $[a, b]$.

La derivata S_d' è una funzione spline di ordine $d-1$ e l'integrale è una funzione spline di ordine $d+1$.

Spline cubiche

Sia $d = 3$, cioè considereremo le spline cubiche $S_3(x)$. Imponiamo le condizioni di interpolazione:

$$\begin{aligned} S_3(x_0) &= f_0 \\ S_3(x_1) &= f_1 \\ &\vdots \\ S_3(x_n) &= f_n. \end{aligned}$$

Essendo $S_3(x)$ un polinomio di grado 3 in ogni intervallo, poniamo:

$$S_3(x) = a_i + b_i x + c_i x^2 + d_i x^3 \quad x \in [x_{i-1}, x_i] \qquad i = 1, 2, \ldots, n$$

i $4n$ coefficienti sono da determinare. Le condizioni di interpolazione forniscono $n+1$ condizioni. Bisogna ancora imporre le condizioni di continuità, le quali forniscono:

$$\begin{aligned} S_3(x_i^+) &= S_3(x_i^-) \\ S_3'(x_i^+) &= S_3'(x_i^-) \\ S_3''(x_i^+) &= S_3''(x_i^-) \end{aligned} \qquad i = 1, 2, \ldots, n-1.$$

con $g(x_i^+)$ e $g(x_i^-)$ indichiamo rispettivamente il limite destro e sinistro della funzione $g(x)$.
Queste sono altre $3n-3$ condizioni. In totale abbiamo $4n-2$ condizioni. Per arrivare a $4n$ ne rimangono ancora due che vengono scelte a piacere.

Ponendo $S_3''(x_0) = S_3''(x_n) = 0$,
si ottengono le *spline naturali*.

Le condizioni precedenti sono sufficienti a ricavare i coefficienti a_i, b_i, c_i, d_i. Si potrebbero usare tali condizioni per ricavare un sistema avente come incognite a_i, b_i, c_i, d_i e risolverlo.

Un metodo efficiente si ottiene prendendo come incognite i valori $M_i = S_3''(x_i)$.

$S_3''(x)$ è una spline del primo ordine e quindi una funzione lineare a tratti, ponendo $h_i = x_i - x_{i-1}$, si ha:

$$S_3''(x) = \frac{(x - x_{i-1})M_i + (x_i - x)M_{i-1}}{h_i} \quad x \in [x_{i-1}, x_i].$$

Integrando due volte nell'intervallo $[x_{i-1}, x_i]$, si ha:

$$S_3'(x) = \frac{(x - x_{i-1})^2 M_i - (x_i - x)^2 M_{i-1}}{2h_i} + C_i$$

e

$$S_3(x) = \frac{(x - x_{i-1})^3 M_i + (x_i - x)^3 M_{i-1}}{6h_i} + \\ + C_i(x - x_{i-1}) + D_i$$

Imponendo le condizioni di interpolazione, si ottiene, per $i = 1, \ldots, n$

$$\frac{(x_i - x_{i-1})^3 M_{i-1}}{6h_i} + D_i = f_{i-1}$$
$$\frac{(x_i - x_{i-1})^3 M_i}{6h_i} + C_i(x_i - x_{i-1}) + D_i = f_i$$

da cui si ricava:

$$D_i = f_{i-1} - \frac{h_i^2}{6} M_{i-1}, \quad C_i = \frac{f_i - f_{i-1}}{h_i} - h_i \frac{M_i - M_{i-1}}{6}.$$

Sostituendo nella $S'(x)$ si ottiene, per $x \in [x_{i-1}, x_i]$:

$$S_3'(x) = \frac{(x - x_{i-1})^2 M_i - (x_i - x)^2 M_{i-1}}{2h_i} + \frac{f_i - f_{i-1}}{h_i} - h_i \frac{M_i - M_{i-1}}{6}$$

Imponendo la continuità di S_3' in ogni punto x_i, per $i = 1, 2, \ldots, n-1$ si ha:

$$h_i M_{i-1} + 2(h_i + h_{i+1}) M_i + h_{i+1} M_{i+1} = 6 \left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right).$$

Tenendo conto che $M_0 = M_n = 0$, si ottiene un sistema di equazioni lineari nelle $n-1$ incognite $M_1, \ldots, M_{n-1}$, la cui matrice dei coefficienti è

$$\begin{pmatrix} 2(h_1 + h_2) & h_2 & & \\ h_2 & 2(h_2 + h_3) & \ddots & \\ & \ddots & \ddots & h_{n-1} \\ & & h_{n-1} & 2(h_{n-1} + h_n) \end{pmatrix}.$$

La matrice è tridiagonale, simmetrica e diagonale dominante. La soluzione del sistema non richiede l'uso del pivot.

Il MATLAB dispone di una funzione $spline$ che determina la funzione spline cubica, le condizione aggiuntive sono diverse da quelle usate per determinare la spline cubica naturale, e viene denotata .

Accetta come dati di input:

$xnodi$	vettore contenente i nodi,
$fnodi$	vettore contenente i valori nei nodi,
x	vettore con i punti in cui si vuole calcolare il valore della funzione spline.

Dà come risultato:

y	vettore contenente il valore assunto nei punti in x dalla funzione spline.

La funzione è richiamabile tramite l'istruzione

```
y = spline(xnodi, fnodi, x)
```

Se costruiamo la spline interpolante $S_1(x)$ date le coppie $(x_0, f_0), \ldots, (x_n, f_n)$, in ogni intervallo $[x_i, x_{i+1}]$, la spline è definita da

$$S_1(x) = \frac{(x_{i+1} - x)f_i + (x - x_i)f_{i+1}}{x_{i+1} - x_i}$$

quando $f_i = f(x_i)$ con f continua con derivata prima e seconda continua, dalla formula per l'errore dell'interpolazione

$$\|S_1(x) - f(x)\|_\infty \leq \|(x - x_i)(x - x_{i+1})\|_\infty \frac{\|f''(x)\|_\infty}{2}$$

quindi se $h = \max_i(x_{i+1} - x_i)$ si ha

$$\|S_1(x) - f(x)\|_\infty \leq O(h^2)$$

Per le spline cubiche vale il seguente teorema:

Teorema. Sia $S_3(x)$ la spline cubica naturale associata ai dati $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$, se $f(x) \in C^4([a, b])$ ad esiste una costante γ tale che $h/h_i \leq \gamma \leq \infty$ per $h \to 0$

$$\|f^{(p)}(x) - S_3^{(p)}(x)\|_\infty \leq O(h^{4-p}), \qquad p = 0, 1, 2, 3$$

Teorema. Tra tutte le funzioni $f(x) \in C^2([a,b])$ tali che $f(x_i) = f_i, f''(x_0) = 0, f''(x_n) = 0$, la spline cubica naturale è la funzione che minimizza l'integrale

$$E(f) = \int_{x_0}^{x_n} (f''(x))^2 dx$$

$E(f)$ rappresenta una misura approssimata della curvatura totale dell $f(x)$ nell'intervallo $[a,b]$.

Approssimazione ai minimi quadrati

Finora abbiamo affrontato il problema di costruire un polinomio che passa per punti prefissati. Se i valori assegnati sono affetti da errori, come in genere capita se tali valori sono ottenuti da osservazioni sperimentali, allora non ha più molto senso "costringere" il polinomio ad assumere tali valori. In questo caso è più significativo richiedere che $p(x)$ sia "vicino" ai valori f_i senza che $p(x_i) = f_i$ nei punti x_i.

Sia $\phi_0(x), \phi_1(x), \ldots, \phi_n(x)$ una base di P_n e siano (x_i, f_i), per $i = 0, \ldots, m$, $m + 1 > n$ coppie di punti assegnati. Cercheremo il polinomio $p(x)$ tale che la quantità

$$F = \sum_{i=0}^{m} (p(x_i) - f_i)^2$$

risulti minima. Il polinomio $p(x)$ così ottenuto approssima i dati nel senso dei *minimi quadrati*.

Posto

$$p(x) = \sum_{j=0}^{n} a_j \phi_j(x),$$

si ha

$$F(a_0, a_1, \ldots, a_n) = \sum_{i=0}^{m} \left(\sum_{j=0}^{n} a_j \phi_j(x_i) - f_i \right)^2 .$$

Per ricavare i coefficienti che minimizzano la precedente funzione bisogna annullare le derivate parziali. Si pone:

$$\frac{\partial F}{\partial a_k} = 2 \sum_{i=0}^{m} \left(\sum_{j=0}^{n} a_j \phi_j(x_i) - f_i \right) \phi_k(x_i) = 0,$$

da cui si ottiene un sistema lineare nelle incognite a_j :

$$\sum_{j=0}^{n} a_j \sum_{i=0}^{m} \phi_j(x_i)\phi_k(x_i) = \sum_{i=0}^{m} f_i \phi_k(x_i), \quad k = 0, 1, \ldots, m$$

Poniamo

$$(\phi_j, \phi_k) = \sum_{i=0}^{m} \phi_j(x_i)\phi_k(x_i),$$
$$(f, \phi_k) = \sum_{i=0}^{m} f_i \phi_k(x_i).$$

Il sistema precedente diventa:

$$\sum_{j=0}^{n} (\phi_j, \phi_k) a_j = (f, \phi_k) \qquad \text{per } k = 0, 1, \ldots, n.$$

La matrice dei coefficienti è:

$$G = \begin{pmatrix} (\phi_0, \phi_0) & (\phi_1, \phi_0) & \cdots & (\phi_n, \phi_0) \\ (\phi_1, \phi_0) & (\phi_1, \phi_1) & \cdots & (\phi_n, \phi_1) \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_n, \phi_0) & (\phi_n, \phi_1) & \cdots & (\phi_n, \phi_n) \end{pmatrix}.$$

G è simmetrica, è anche definita positiva. Considerato che G coincide con l'Hessiano di F, ciò dimostra che la F ha realmente un punto di minimo.

Prendiamo $n = 0$ e $\phi_0(x) = 1$. Vogliamo quindi trovare una costante che approssimi i dati $f_0, f_1, \ldots, f_m$. Si ha:

$$(\phi_0, \phi_0) = m + 1, \quad \text{e} \quad (f, \phi_0) = \sum_{i=0}^{m} f_i.$$

Il sistema si riduce all'unica equazione:

$$(m+1)\ a = \sum_{i=0}^{m} f_i$$

da cui si ricava:

$$a = \frac{1}{m+1} \sum_{i=0}^{m} f_i,$$

cioè la media aritmetica dei valori f_i.

Se ci costruiamo il vettore $\mathbf{f} = (f_0, f_1, \ldots, f_m)^T$ e il vettore $\mathbf{p} = (p(x_0), p(x_1), \ldots, p(x_m))$, contenente i valori che assume il polinomio di approssimazione ai minimi quadrati nei punti x_i, possiamo definire i seguenti parametri:
la deviazione standard dei residui (indice della "dispersione" intorno al valore medio)

$$\sigma_{res} = \frac{\|\mathbf{f} - \mathbf{p}\|_2}{\sqrt{m-n}};$$

i residui scalati

$$\frac{\mathbf{f} - \mathbf{p}}{\sigma_{res}}.$$

Per essere accettabile il 95 % del residuo scalato dovrebbe variare nell'intervallo [-2,2].
Un indice facilmente calcolabile che misura quanto il modello riesca a seguire l'andamento dei dati è noto come R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(f_i - p(x_i))^2}{\sum_{i=1}^{m}(f_i - \bar{\mathbf{f}})}$$

con $\bar{\mathbf{f}}$ il valore medio delle f_i. Quando $R^2 \approx 1$ il modello sta seguendo bene l'andamento dei dati. Quando $R^2 \approx 0$ il modello è meno significativo rispetto a scegliere come approssimazione dei dati la loro media aritmetica.

Il MATLAB ha una funzione $polyfit$ che fornisce i coefficienti a_i del polinomio nella forma

$$p(x) = \sum_{i=0}^{n} a_i x^i.$$

I dati di input di questa funzione sono:

x	vettore contenente un insieme di punti,
f	vettore contenente il valore assunto nei punti di x,
n	grado del polinomio approssimatore.

risultato:

a	vettore contenente i coefficienti a_j del polinomio.

La funzione è richiamabile tramite:

```
a = polyfit(x, f, n)
```

Il risultato di $polyfit$ viene in genere utilizzato dalla funzione MATLAB $polyval$. Questa funzione fornisce il valore del polinomio interpolante in un insieme di punti.

Ha come dati di input:

a	vettore contenente i coefficienti a_j del polinomio,
x	vettore contenente i punti in cui si vuole calcolare il valore del polinomio.

risultato:

y	vettore contenente il valore del polinomio nei punti in x.

È richiamabile tramite:

```
y = polyval(a, x1)
```