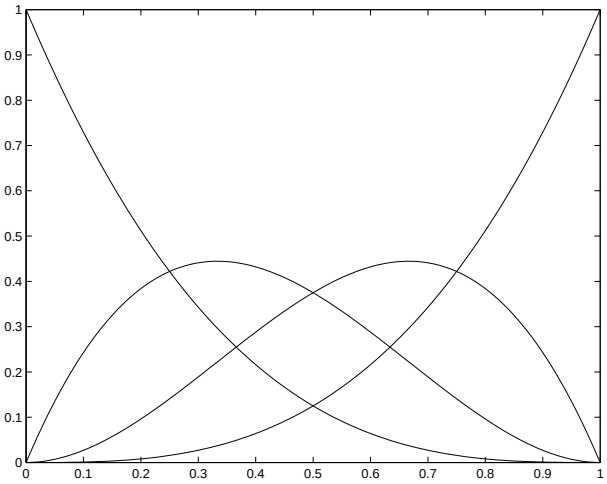
ARGOMENTI DEL CORSO INFORMATICA III

A.A. 2003/04



Interpolazione con Funzioni Polinomiali

Giulio Casciola

(ottobre 2003, rivisto febbraio 2004)

Indice

Capitolo 1

Funzioni Polinomiali nel Calcolo Numerico

Il problema dell'*approssimazione di funzioni*, cioè la determinazione di una funzione più semplice per rappresentarne una più complessa, selezionandola da una prefissata classe di funzioni a dimensione finita, è un problema di fondamentale importanza nella matematica applicata. Per meglio comprenderne l'importanza esaminiamo due situazioni classiche in cui si presenta tale problema.

- La funzione da approssimare è nota in forma analitica, ma è richiesta una sua approssimazione con una funzione più semplice per la sua valutazione numerica, tipicamente con un calcolatore, oppure per renderne possibili operazioni quali ad esempio la derivazione o l'integrazione;

- La funzione $f(x)$ da approssimare non è nota, ma di essa si conoscono alcuni valori y_i in corrispondenza di punti x_i, $i = 0, \ldots, n$ e si vogliono avere indicazioni sul comportamento della funzione in altri punti. In questo caso la funzione è nota in senso discreto, per esempio un insieme di dati sperimentali, e approssimare la $f(x)$ significa darne un modello rappresentativo.

In tal senso quello che vedremo in questa dispensa è un'introduzione alla modellazione. La Fig. 1.1 illustra un problema di interpolazione di punti nello spazio 3D con una funzione vettoriale (superficie in forma parametrica) che solitamente è il primo passo in un processo di ricostruzione e modellazione di forma.

Il problema può essere affrontato con differenti tecniche, che per essere adeguate devono tener conto della specifica situazione. In questo senso risulta importante la scelta della distanza con la quale si vuole approssimare

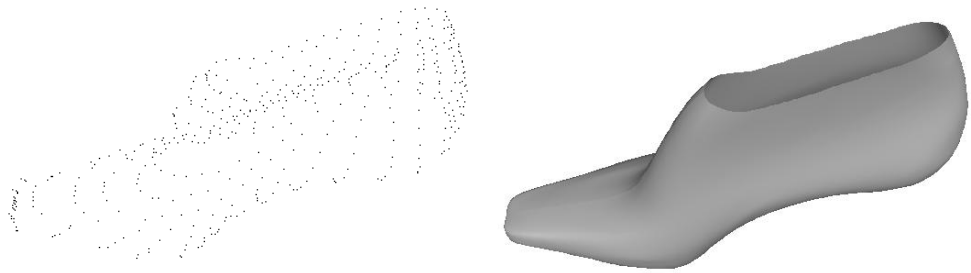


Figura 1.1: Interpolazione di punti 3D con una funzione vettoriale

e che misura l'errore, mentre in certe situazioni, piuttosto che ridurre la distanza, risulta importante conservare certe proprietà di forma dei dati. In questa parte esamineremo la tecnica dell'interpolazione (o collocazione), sulla quale si basano la maggior parte dei metodi numerici per il calcolo degli integrali definiti e per l'approssimazione delle equazioni differenziali, la tecnica dell'approssimazione nel senso dei minimi quadrati e, meno frequente nei testi numerici, l'approssimazione di forma. Per tutte queste tecniche ci limiteremo al caso polinomiale, ossia andremo a selezionare l'approssimazione desiderata dalla classe delle funzioni polinomiali a coefficienti reali e a variabile reale, nel semplice caso scalare. Questo capitolo, prima di introdurre le tecniche citate, vuole richiamare le nozioni più importanti sulle funzioni polinomiali e discuterne il loro utilizzo dal punto di vista numerico.

Definizione 1.1 (Polinomio) *Una funzione* $p : \mathbf{R} \to \mathbf{R}$ *definita da*

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i \tag{1.1}$$

dove n *è un intero non negativo e* $a_0, a_1, \ldots, a_n$ *sono numeri reali fissati è detto* polinomio. *In questa rappresentazione, se* $a_n \neq 0$*, si dice che* $p(x)$ *ha grado* n *(ordine* $n+1$*); se tutti i coefficienti* a_i *sono nulli, allora* $p(x)$ *è detto il polinomio nullo.*

Con $\mathbf{P}_n$ si denota l'insieme di tutti i plinomi p con grado non superiore ad n, insieme al polinomio nullo. Si può dimostrare che $\mathbf{P}_n$ è uno spazio vettoriale di dimensione $n+1$.

La principale ragione dell'importanza delle funzioni polinomiali nel calcolo numerico deriva dalle loro buone proprietà che qui richiamiamo brevemente:

- sono facilmente memorizzabili, e valutabili numericamente con un calcolatore;
- sono funzioni regolari;
- la derivata e l'antiderivata di un polinomio sono ancora polinomi i cui coefficienti sono determinati algoritmicamente (e quindi con un calcolatore);
- il numero di zeri di un polinomio di grado n non può essere superiore ad n;
- data una qualunque funzione continua su un intervallo $[a, b]$, esiste un polinomio che la approssima uniformemente qualunque sia l'ϵ prefissato
$$|f(x) - p(x)| < \epsilon \quad \forall x \in [a, b].$$

Teorema 1.1 (fondamentale dell'algebra) *Sia $p(x)$ un polinomio di grado $n \geq 1$. L'equazione algebrica di grado n, $p(x) = 0$ ha almeno una radice reale o complessa.*

Corollario 1.1 *Ogni equazione algebrica di grado n ha esattamente n radici reali o complesse, ciascuna con la sua molteplicità, cioè:*

$$p(x) = a_n(x - \alpha_1)^{m_1}(x - \alpha_2)^{m_2} \cdots (x - \alpha_k)^{m_k} \tag{1.2}$$

dove α_i, $i = 1, \ldots, k$ sono reali a due a due distinte e m_i, $i = 1, \ldots, k$ sono le relative molteplicità, tali che $m_1 + m_2 + \cdots + m_k = n$.

Teorema 1.2 *Siano $a(x)$ e $b(x)$ polinomi con $b(x) \neq 0$; allora esistono e sono unici i polinomi $q(x)$ ed $r(x)$ per cui*

$$a(x) = q(x)b(x) + r(x)$$

con $r(x) \equiv 0$ o $r(x)$ con grado minore di quello di $b(x)$.

1.1 Valutazione Numerica di un Polinomio

Assegnato un polinomio di grado n nella sua rappresentazione monomiale 1.1, il metodo più immediato per la sua valutazione in corrispondenza di un assegnato valore xv può essere descritto come segue:

```
s:=1
p:=a[0]
per k=1,..,n
  s:=s*xv
  p:=p+s*a[k]
p(xv):=p
```

Pertanto il calcolo di $p(x)$ richiede $2n$ moltiplicazioni ed n addizioni/sottrazioni.
Se si scrive il polinomio 1.1 nella seguente forma:

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots)) \tag{1.3}$$

si ricava il seguente metodo dovuto ad Horner:

```
p:=a[n]
per k=n-1,..,0
  p:=a[k]+xv*p
p(xv):=p
```

L'algoritmo di Horner richiede n moltiplicazioni ed n addizioni/sottrazioni e risulta quindi più rapido del precedente oltre che numericamente più stabile.

1.1.1 Lo schema di Horner e la regola di Ruffini

In questa sezione si vuol ricavare il metodo di Horner da un contesto più generale che sarà utile per ulteriori schemi di calcolo per polinomi. Richiamando il Teorema 1.2 sulla divisione di due polinomi, nel caso particolare in cui il polinomio divisore sia il binomio $(x - xv)$ per un assegnato valore reale xv, si ha che esistono unici i polinomi $q(x)$ ed $r(x)$ per cui

$$p(x) = q(x)(x - xv) + r(x)$$

e poiché $r(x)$ deve essere di grado inferiore a quello di $(x - xv)$ sarà una costante che indicheremo con r. Se si valuta $p(x)$ in xv si trova banalmente che $p(xv) \equiv r$. Quanto osservato viene formalizzato nel seguente teorema.

Teorema 1.3 *Il resto della divisione del polinomio $p(x)$ per $(x - xv)$ è $p(xv)$.*

Dal teorema appena enunciato si deduce che un metodo per valutare un polinomio $p(x)$ in un punto xv consiste nel determinare il resto della divisione fra $p(x)$ e il binomio $(x - xv)$. A tal fine è ben nota la regola di Ruffini che viene applicata facendo uso del seguente schema di calcolo:

	a_n	a_{n-1}	...	...	a_2	a_1	a_0
xv		xvb_n	xvb_{n-1}	...	xvb_3	xvb_2	xvb_1
	b_n	b_{n-1}	...	...	b_2	b_1	r

In pratica, noti i coefficienti a_i ed xv, i b_i ed $r \equiv p(xv)$ vengono ricavati nel seguente modo:

$$\begin{array}{lcl} b_n & \leftarrow & a_n \\ b_{n-1} & \leftarrow & a_{n-1} + xvb_n \\ b_{n-2} & \leftarrow & a_{n-2} + xvb_{n-1} \\ \vdots & \vdots & \vdots \\ b_1 & \leftarrow & a_1 + xvb_2 \\ r & \leftarrow & a_0 + xvb_1 \end{array}$$

dove

$$q(x) = b_1 + b_2x + b_3x^2 + \ldots b_nx^{n-1} = \sum_{i=0}^{n-1} b_{i+1}x^i.$$

Come si può osservare, questo schema è esattamente quello di Horner visto; infatti se non interessa memorizzare i coefficienti del polinomio quoziente $q(x)$ e li si sostituisce con una variabile semplice p i due schemi coincidono.

Esempio 1.1 Sia dato

$$p(x) = 1 + x - 2x^2 + 3x^4$$

e si voglia calcolare $p(2)$. La regola di Ruffini è molto nota perché permette di procedere manualmente in modo semplice;

	3	0	-2	1	1
2		6	12	20	42
	3	6	10	21	$43 \equiv p(2)$

1.1.2 Valutazione Numerica della Derivata

Si vuole ora procedere alla valutazione numerica della derivata di un polinomio $p(x)$ in un assegnato punto xv, cioè si vuole valutare $p'(xv)$. La prima idea potrebbe essere quella di calcolare numericamente i coefficienti del polinomio derivato e valutarlo in xv con Horner; questo equivale a determinare l'espressione $p'(x) = a_1 + 2a_2x + 3a_3x^2 + \ldots na_nx^{n-1}$ e a procedere nel valutare un polinomio di grado $n-1$. Nel caso in cui oltre al valore della derivata in xv fosse necessario avere anche il valore del

polinomio, allora, in modo più economico, si può procedere nel seguente modo: per quanto detto nella sezione precedente è

$$p(x) = q(x)(x - xv) + p(xv),$$

derivando tale espressione si ha

$$p'(x) = q'(x)(x - xv) + q(x)$$

e valutandola in xv

$$p'(xv) = q(xv)$$

dove $q(x)$ ed i suoi coefficienti sono quelli che si ottengono applicando l'algoritmo di Ruffini-Horner per valutare $p(x)$ in xv.

Esempio 1.2 Sia dato

$$p(x) = 1 + x - 2x^2 + 3x^4$$

e si voglia calcolare $p(2)$ e $p'(2)$

	3	0	−2	1	1
2		6	12	20	42
	3	6	10	21	$43 \equiv p(2)$
2		6	24	68	
	3	12	34	$89 \equiv p'(2)$	

Il calcolo di $p(xv)$ e $p'(xv)$ può essere organizzato nel seguente modo:

```
p1:=0
p:=a[n]
per k=n-1,...,0
  p1:=p+xv*p1
  p:=a[k]+xv*p

p(xv):=p
p'(xv):=p1
```

Analogamente si possono calcolare anche le derivate di ordine superiore. Derivando l'espressione ottenuta per $p'(x)$ si ottiene:

$$p''(x) = q''(x)(x - xv) + 2q'(x)$$

e valutando questa espressione in xv

$$p''(xv) = 2q'(xv).$$

Allora per ottenere $p''(xv)$ è sufficiente calcolare $q'(xv)$ che possiamo ottenere dallo schema di Horner utilizzato per valutare $p'(x)$ in xv.
In generale se si indica la relazione fra polinomi dividendo, ed il suo polinomio quoziente, con un indice, cioè

$$p_j(x) = p_{j+1}(x - xv) + p_j(xv) \quad j = 0, \ldots, n-1$$

allora

- $p(x) \equiv p_0(x)$
- $p^{(j)}(xv) = j! p_j(xv)$
- i coefficienti di $p_j(xv)$ sono i valori determinati applicando il metodo di Horner al polinomio $p_{j-1}(x)$.

1.2 Valutazione di un Polinomio: Errore Inerente

Assegnato un polinomio ed un punto in cui valutarlo, l'errore inerente misura a piccole variazioni sui coefficienti, come varia, in senso relativo, il valore del polinomio. Più piccola è la variazione e migliore è il condizionamento del problema.

Esempio 1.3 Sia assegnato il polinomio

$$p(x) = a_0 + a_1 x = 100 - x$$

e lo si voglia valutare in punti $xv \in [100, 101]$. Si perturba il coefficiente a_1 dell'1% [1], per cui il polinomio perturbato risulta:

$$\tilde{p}(x) = 100 - (1 - \frac{1}{100})x = 100 - \frac{99}{100}x$$

Valutandoli in $x = 101$ si ha:

$$p(101) = -1 \qquad \tilde{p}(101) = 0.01$$

commettendo un errore relativo dato da:

$$\left| \frac{\tilde{p}(101) - p(101)}{p(101)} \right| = 101 \frac{1}{100}.$$

[1] cioè $|\frac{\tilde{a}_1 - a_1}{a_1}| = \frac{1}{100}$; segue che $\tilde{a}_1 = a_1 \pm \frac{1}{100} a_1 = (1 \pm \frac{1}{100}) a_1$

Risulta quindi che una perturbazione iniziale dell'1% porta una variazione sul risultato finale del 101%.
La causa di questa amplificazione dell'errore è evidente nella Fig. 1.2. In pratica il coefficiente a_1 rappresenta l'inclinazione della retta la quale, anche se alterata minimamente, comporta grossi errori per punti lontani dall'origine.
Lo stesso comportamento si ottiene perturbando entrambi i coefficienti o valutando in altri punti dell'intervallo.

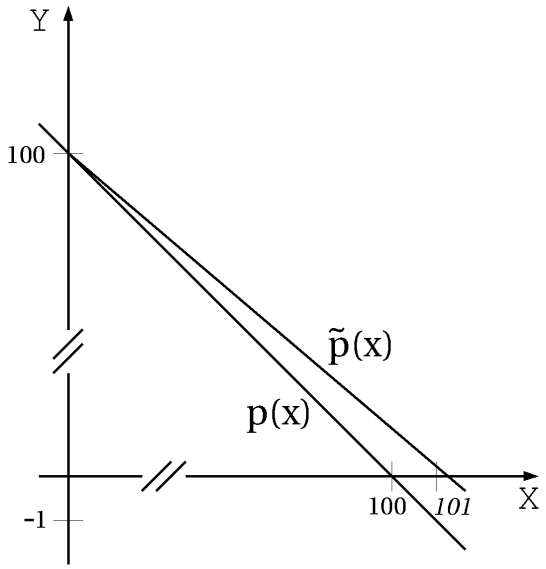


Figura 1.2: Amplificazione dell'errore

In generale il problema di valutare un polinomio dato nella base dei monomi in punti appartenenti ad un intervallo $[a, b]$ con a e b grandi e $a/b \simeq 1$ non risulta un problema ben condizionato, ossia si possono avere grossi errori inerenti.
Procediamo, per questo esempio, all'analisi dell'errore inerente mediante la stima precedentemente vista:

$$\epsilon_{in} \simeq \sum_{i=1}^{n} c_i \epsilon_i \tag{1.4}$$

dove

$$c_i = \frac{x_i}{f(x)} \frac{\partial f(x)}{\partial x_i} \tag{1.5}$$

con $\epsilon_i = \frac{\tilde{x}_i - x_i}{x_i}$, e $|\epsilon_i| < u$, per $i = 1, 2, \ldots, n$.
Nel caso specifico di $p(x) = a_0 + a_1 x$ sarà:

$$\epsilon_{in} \simeq c_1\epsilon_1 + c_2\epsilon_2 + c_3\epsilon_3$$
$$= \frac{a_0}{a_0 + a_1 x}\epsilon_1 + \frac{a_1 x}{a_0 + a_1 x}\epsilon_2 + \frac{x a_1}{a_0 + a_1 x}\epsilon_3$$

e nel caso $a_0 = 100$, $a_1 = -1$ e $x \in [100, 101]$ con, per esempio, $x = 101$ si ha:

$$= \frac{100}{-1}\epsilon_1 + \frac{-101}{-1}\epsilon_2 + \frac{-101}{-1}\epsilon_3$$

da qui si vede come una piccola perturbazione su uno dei coefficienti, per esempio a_1 dell'1% ($\epsilon_1 = 0$, $\epsilon_2 = 1/100$, $\epsilon_3 = 0$) porti ϵ_{in} ad assumere il valore $-101\frac{1}{100}$ e cioè 101 volte maggiore di quello iniziale.
Dall'analisi effettuata si evince che per qualunque punto x in $[100, 101]$ questo comportamento sarà inevitabile in quanto una lieve modifica dei coefficienti (ϵ_1 o $\epsilon_2 \neq 0$) viene grandemente amplificata (coefficienti c_1 e c_2). Si osservi inoltre che anche per $\epsilon_1 = \epsilon_2 = 0$ e $\epsilon_3 \neq 0$ si avrà un errore inerente grande in questo intervallo di valutazione.

Generalizzando l'analisi fatta in questo esempio alla valutazione di un polinomio $p(x)$ nella base monomiale, l'errore inerente si può presentare in due componenti:

$$\epsilon_{in1} = \sum_{i=0}^{n} \frac{a_i}{p(x)}\frac{\partial p(x)}{\partial a_i}\epsilon_i = \sum_{i=0}^{n} \frac{a_i x^i}{p(x)}\epsilon_i \quad \text{con} \quad \epsilon_i = \frac{\tilde{a}_i - a_i}{a_i}$$

e

$$\epsilon_{in2} = \frac{x}{p(x)}\frac{\partial p(x)}{\partial x}\epsilon_x = \frac{x p'(x)}{p(x)}\epsilon_x \quad \text{con} \quad \epsilon_x = \frac{\tilde{x} - x}{x}.$$

Si può quindi desumere che:

- ϵ_{in1} dipende da $p(x)$ e dai valori $a_i x^i$. Questi termini sono in parte controllabili riformulando il problema (la sua rappresentazione);
- ϵ_{in2} dipende unicamente dal polinomio in esame $p(x)$ e dalla sua derivata, per cui è un errore intrinseco al valore stesso di $p(x)$ e non dipende dalla sua rappresentazione.
  In particolare si ha una maggiorazione dell'errore relativo per $p(x) \to 0$ o per $p'(x) \to \infty$.

Nel seguito vogliamo esaminare se è possibile agire sull'espressione di $p(x)$ al fine di ridurre l'errore inerente. Si introduce il concetto di *condizionamento di una base polinomiale* per indicare quale rappresentazione polinomiale sia affetta da minor errore inerente allorquando se ne usi una

sua combinazione lineare su un intervallo per risolvere un problema che può andare dalla valutazione alla determinazione dei suoi zeri. Si osservi che, come introdotto anche nell'esempio, lavorando numericamente resta sempre definito l'intervallo di lavoro o di interesse e questa informazione può essere sfruttata per individuare una base meglio condizionata.

1.3 Polinomi nella base di Bernstein

Con polinomio in forma di Bernstein si intende una espressione del tipo

$$p(x) = \sum_{i=0}^{n} b_i B_{i,n}(x) \quad \text{con} \quad x \in [a, b]$$

dove $\{B_{i,n}(x)\}$ sono le funzioni base di Bernstein definite da

$$B_{i,n}(x) = \binom{n}{i} \frac{(b-x)^{n-i}(x-a)^i}{(b-a)^n} \quad i = 0, \ldots, n$$

e $b_0, b_1, \ldots, b_n \in \mathbf{R}$ sono i coefficienti nella base di Berstein.

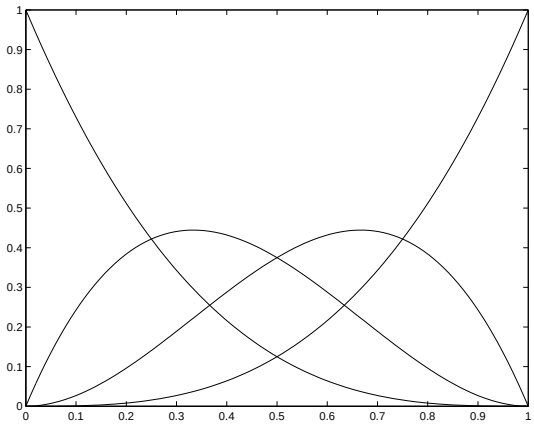


Figura 1.3: Polinomi base di Bernstein di grado 3.

Esempio 1.4 Riprendiamo l'esempio del polinomio precedente, ma nella base di Bernstein, sarà:

$$B_{0,1}(x) = 101 - x \qquad B_{1,1}(x) = x - 100$$

da cui

$$p(x) = 100 - x = b_0 B_{0,1}(x) + b_1 B_{1,1}$$

$$= 0(101 - x) - 1(x - 100) \quad \text{ossia} \quad b_0 = 0 \quad b_1 = -1.$$

Rieseguendo l'analisi sull'errore inerente si ha:

$$\epsilon_{in} = c_1\epsilon_1 + c_2\epsilon_2 + c_3\epsilon_3 =$$

$$\frac{b_0}{p(x)}(101 - x)\epsilon_1 + \frac{b_1}{p(x)}(x - 100)\epsilon_2 + \frac{xp'(x)}{p(x)}\epsilon_3$$

e perturbando solo b_1 (cioè $\epsilon_1 = \epsilon_3 = 0$, ma $\epsilon_2 \neq 0$), valutando in un qualunque punto in $[100, 101]$ il coefficiente moltiplicatore di ϵ_2 sarà al massimo dell'ordine dell'unità.
Procediamo numericamente; si perturba il coefficiente b_1 dell'1%, per cui il polinomio perturbato risulta:

$$\tilde{p}(x) = -(1 - \frac{1}{100})(x - 100) = -\frac{99}{100}(x - 100)$$

Valutandoli in $x = 101$ si ha:

$$p(101) = -1 \qquad \tilde{p}(101) = -0.99$$

commettendo un errore relativo dato da:

$$\left|\frac{\tilde{p}(101) - p(101)}{p(101)}\right| = \frac{1}{100}$$

Risulta quindi che una perturbazione iniziale dell'1% porta ad una variazione sul risultato finale della stessa entità.
Come già fatto notare anche nell'analisi precedente, se $\epsilon_3 \neq 0$, $\frac{xp'(x)}{p(x)}$ potrebbe essere grande indipendentemente dalla base di rappresentazione. Vediamo un esempio numerico.
Perturbiamo x dell'1%; se $x = 101$ sarà $\tilde{x} = \frac{99}{100}101 = 99.99$; allora:

$$p(101) = -1 \qquad p(99.99) = 0.01$$

commettendo un errore relativo dato da:

$$\left|\frac{p(101) - p(99.99)}{p(101)}\right| = 101\frac{1}{100};$$

si può fare qualcosa?

Si ricorda che i polinomi godono della proprietà di essere invarianti per traslazione e scala dell'intervallo di definizione o cambio di variabile.

Ciò significa che dato un polinomio $p(x)$ in un intervallo $[a, b]$ e un intervallo traslato e scalato di questo, per esempio $[0, 1]$, è possibile definire il mapping dal primo al secondo e viceversa

$$x \in [a, b] \rightarrow t \in [0, 1]$$

$$t = \frac{x - a}{b - a} \quad \text{o viceversa} \quad x = a + t(b - a) \tag{1.6}$$

ed effettuando un cambio di variabile determinare un polinomio $q(t)$ che per ogni $t \in [0, 1]$ assume gli stessi valori di $p(x)$ per la x corrispondente secondo il mapping definito. Un problema nell'applicare tale cambio di variabile consiste nel dover determinare numericamente i coefficienti del polinomio $q(t)$ rischiando di commettere degli errori di approssimazione. I polinomi di Bernstein godono della proprietà che nel cambio di variabile i coefficienti non cambiano. Vediamolo nel dettaglio.

$$B_{i,n}(x) = B_{i,n}(a+t(b-a)) = \binom{n}{i} \frac{[b - a - t(b - a)]^{n-i}[a + t(b - a) - a]^i}{(b - a)^n} =$$

$$= \binom{n}{i} \frac{[(b - a)(1 - t)]^{n-i}[t(b - a)]^i}{(b - a)^n} =$$

$$\binom{n}{i}(1 - t)^{n-i}t^i = B_{i,n}(t) \tag{1.7}$$

Riassumendo, avremo

$$p(x) = \sum_{i=0}^{n} b_i B_{i,n}(x) \quad x \in [a, b]$$

ed effettuando il cambio di variabile otteniamo

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \quad t \in [0, 1]$$

che dice che un cambio di variabile, per un polinomio nella base di Bernstein, non comporta alcun errore o costo computazionale sui coefficienti perché restano i medesimi.

Tornando all'errore inerente nella valutazione ed in particolare al nostro esempio numerico si ha:

$$p(x) = b_0(101 - x) + b_1(x - 100) \quad x \in [100, 101]$$

$$p(t) = b_0(1 - t) + b_1(t - 0) \quad t \in [0, 1]$$

e la stima dell'errore inerente

$$\epsilon_{in} = \frac{b_0}{p(t)}(1-t)\epsilon_1 + \frac{b_1}{p(t)}(t-0)\epsilon_2 + \frac{tp'(t)}{p(t)}\epsilon_3$$

dove i primi due termini restano uguali in valore, mentre il terzo pur restando $p(t)$ lo stesso, potenzialmente può avere $tp'(t)$ inferiore; questo é sicuramente vero per t che assume valori in $[0,1]$ e se si richiama che vale la relazione $p'(t) = (b-a)p'(x)$, verrà ridotto anche $p'(t)$ se $(b-a) < 1$. Tornando all'esempio numerico si ha:

$$p(t) = -t \quad t \in [0,1]$$

e poiché in questo caso $b-a = 1$ vale $p'(t) = p'(x)$. Peturbiamo t dell'1%, allora $\tilde{t} = \frac{99}{100}t$; se $t = 1$ sarà $\tilde{t} = 0.99$, allora

$$p(1) = -1 \qquad p(0.99) = -0.99$$

commettendo un errore relativo dato da:

$$\left|\frac{p(1) - p(0.99)}{p(1)}\right| = \frac{1}{100}.$$

Se $b-a > 1$, si può suddividere l'intervallo di interesse in più intervalli tutti di ampiezza minore di 1 (vedi sezione 1.3.5).
Si noti che in $[0,1]$ i polinomi di Bernstein sono più semplici da calcolare (vedi la relazione 1.7 che li definisce) e la valutazione del polinomio risulta in genere più stabile dal punto di vista numerico.
Riassumendo, se si vuole valutare $p(xv)$, si determina $tv := \frac{\bar{x}v - a}{(b-a)}$, si calcola $p(tv)$ con $p(t)$ polinomio di Bernstein in $[0,1]$, e sarà $p(xv) \equiv p(tv)$.
Da ora in poi useremo sempre i polinomi di Bernstein in $[0,1]$.

1.3.1 Errore algoritmico: un esempio famoso

Della valutazione numerica di un polinomio abbiamo attentamente analizzato le cause che possono portare ad un errore inerente grande. Anche se solo con un esempio si vuole evidenziare l'errore algoritmico dovuto a propagazione di errori e mostrare che l'arrotondamento può rovinare una computazione ordinaria. A tal fine considereremo un polinomio nella base monomiale con coefficienti esattamente rappresentabili (errore di rappresentazione nullo) e utilizzeremo il metodo di Horner per valutare tale polinomio in punti di un intervallo che siano numeri finiti (errore di rappresentazione nullo sui punti di valutazione). Grazie poi al fatto che tale polinomio ammette una rappresentazione esatta anche nella base di Bernstein, potremo utilizzare un metodo alternativo per la sua valutazione e confrontare le risposte dei due metodi.

Esempio 1.5 *Si consideri il polinomio*

$$p(x) = 1 - 6x + 15x^2 - 20x^3 + 15x^4 - 6x^5 + x^6$$

e lo si valuti, usando la precisione **basic double**, *nell'intervallo* $[a, b] = [0.99609375, 1.00390625]$ *di estremi numeri finiti rappresentabili in base 2 con meno di 8 cifre, usando un passo* $h = 0.000244140625$ *per un totale di* 33 *punti rappresentabili in base 2 tutti al più con 12 cifre. Lo stesso polinomio ammette la seguente rappresentazione nella base di Bernstein:*

$$p(x) = \begin{cases} (1-x)^6 & x \in [0,1] \\ (x-1)^6 & x \in [1,2] \end{cases}$$

La Fig. 1.4 mostra i grafici delle valutazioni effettuate ed evidenzia errori algoritmici nell'utilizzo del metodo di Horner del polinomio in forma monomiale. La Fig.1.5 mostra i grafici di analoghe valutazioni, ma in 201

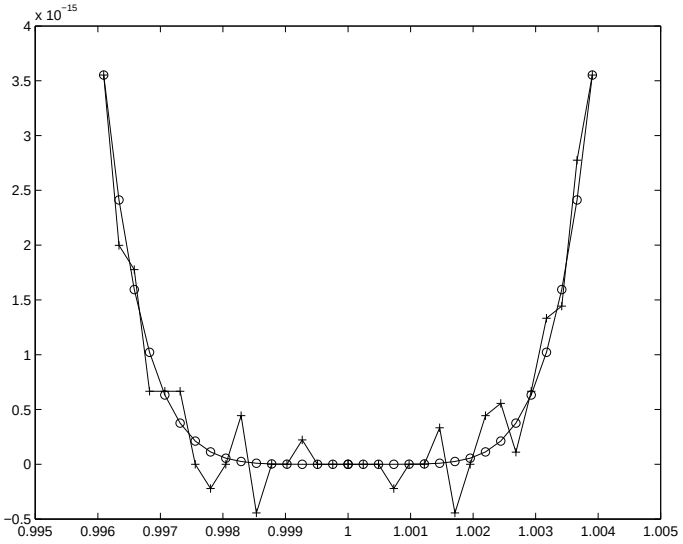


Figura 1.4: Grafici delle valutazioni effettuate in 33 punti.

punti equidistanti dell'intervallo $[a, b] = [0.995, 1.005]$*; ancora maggiormente si nota l'instabilità della valutazione in oggetto.*

Osservazione 1.1 *Le basi di rappresentazione polinomiale sono infinite e numerose sono quelle di interesse numerico, nel senso che a seconda del problema, si può individuare la base più idonea per trattarlo; fra queste ricordiamo la base delle potenze traslate con un centro, la forma di Newton*

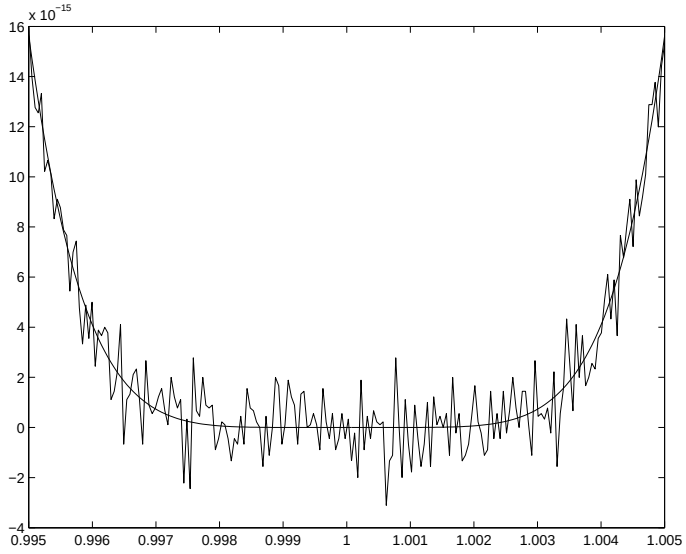


Figura 1.5: Grafici delle valutazioni effettuate in 201 punti.

(o base delle potenze traslate con n centri), la base dei polinomi cardinali o di Lagrange ed altre ancora. Forse la più eclettica per le sue svariate buone proprietà e ottime qualità numeriche è proprio la base di Bernstein, che abbiamo deciso di sponsorizzare in questa trattazione.

1.3.2 Proprietà dei polinomi di Bernstein

I polinomi base di Bernstein $B_{i,n}(t)$ godono di alcune interessanti proprietà:

- $B_{i,n}(t) \geq 0 \quad i = 0, \ldots, n \quad \forall t \in [0,1]$;
- Partizione dell'unità:

$$\sum_{i=0}^{n} B_{i,n}(t) = 1 \quad \forall t \in [0,1]$$

- Per le proprietà precedenti, $p(t)$ espresso nella base di Bernstein è una combinazione convessa dei b_i, da cui segue

$$\min_i \{b_i\} \leq p(t) \leq \max_i \{b_i\} \quad \forall t \in [0,1].$$

  Si richiama che una combinazione lineare si dice affine se la somma dei coefficienti è 1 ($\sum B_{i,n}(t) = 1$), se poi questi sono non negativi ($0 \leq B_{i,n}(t) \leq 1$) allora si dice combinazione convessa.

- Variazione di segno dei coefficienti:
  un polinomio espresso nella base di Bernstein ha un numero di variazioni di segno minore o uguale al numero di variazione di segno del vettore dei suoi coefficienti (valori o coefficienti nulli non vengono considerati).

1.3.3 Formula ricorrente

I polinomi base di Bernstein di grado n, sono legati ai polinomi base di Bernstein di grado $n-1$, dalla seguente formula ricorrente

$$B_{i,n}(t) = tB_{i-1,n-1}(t) + (1-t)B_{i,n-1}(t)$$

con $B_{0,0}(t) = 1$ e $B_{i,n}(t) = 0 \quad \forall i \notin [0,n]$.

Viene riportato il codice Matlab che implementa la formula ricorrente, sui polinomi base di Bernstein in [0,1], per la valutazione in un punto o in un array di punti.

```
function bs=bernst(n,t)
% calcola i polinomi di Bernstein in [0,1] in un punto t;
% se t e' un vettore di punti torna una matrice di valori;
% n  --> grado del polinomio
% x  --> valore di un punto o vettore di punti
% bs <-- matrice dei polinomi di bernstein nei punti
m=length(t);
np1=n+1;
bs=zeros(m,np1);
for ii=1:m
 l=np1;
 bs(ii,l)=1.0;
 d1=t(ii);
 d2=1.0-t(ii);
 for i=1:n
   temp=0.0;
   for j=l:np1
     beta=bs(ii,j);
     bs(ii,j-1)=d2.*beta+temp;
     temp=d1.*beta;
   end
   bs(ii,np1)=temp;
   l=l-1;
  end
end
```

Osservazione 1.2 *Mediante tale algoritmo la valutazione dei polinomi base di Bernstein in un punto costa* $\frac{n(n+1)}{2}$ *addizioni/sottrazioni ed* $n(n+1)$ *moltiplicazioni.*

Esempio 1.6 *Polinomi base di Bernstein di grado 3 ottenuti mediante formula ricorrente:*

$$\begin{array}{llll} B_{0,0}(t) = 1 & & & \\ B_{0,1}(t) = (1-t) & B_{1,1}(t) = t & & \\ B_{0,2}(t) = (1-t)^2 & B_{1,2}(t) = 2t(1-t) & B_{2,2}(t) = t^2 & \\ B_{0,3}(t) = (1-t)^3 & B_{1,3}(t) = 3t(1-t)^2 & B_{2,3}(t) = 3t^2(1-t) & B_{3,3}(t) = t^3 \end{array}$$

1.3.4 Valutazione via algoritmo di de Casteljau

Per valutare un polinomio nella base di Bernstein si può utilizzare la formula ricorrente sulle funzioni base per valutarle nel punto desiderato, quindi effettuare la combinazione convessa

$$p(xv) = \sum_{i=0}^{n} b_i B_{i,n}(xv).$$

Alternativamente si può usare l'algoritmo di de Casteljau sui coefficienti del polinomio, che deriva dall'applicare ripetutamente la formula ricorrente vista.

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \stackrel{\text{def}}{=} \sum_{i=0}^{n} b_i t B_{i-1,n-1}(t) + \sum_{i=0}^{n} b_i (1-t) B_{i,n-1}(t) =$$

poichè $B_{-1,n-1} = B_{n,n-1} = 0$

$$= \sum_{i=0}^{n-1} b_{i+1} t B_{i,n-1}(t) + \sum_{i=0}^{n-1} b_i (1-t) B_{i,n-1}(t) =$$

raccogliendo

$$= \sum_{i=0}^{n-1} [b_{i+1} t + b_i (1-t)] B_{i,n-1}(t) =$$

$$= \sum_{i=0}^{n-1} b_i^{[1]} B_{i,n-1}(t) =$$

riapplicando la formula ricorrente più volte, alla fine si ottiene:

$$\sum_{i=0}^{0} b_0^{[n]} B_{0,0}(t) = b_0^{[n]}.$$

Si ha quindi il seguente algoritmo:

$$b_i^{[k]} = tb_{i+1}^{[k-1]} + (1-t)b_i^{[k-1]} \tag{1.8}$$

con $k = 1, \ldots, n$, $i = 0, \ldots, n-k$ e $p(t) := b_0^{[n]}$; in modo esplicito si ha il seguente schema triangolare:

$$\begin{array}{lllll} b_0^{[0]} & b_1^{[0]} & b_2^{[0]} & \ldots & b_{n-1}^{[0]} \quad b_n^{[0]} \\ b_0^{[1]} & b_1^{[1]} & b_2^{[1]} & \ldots & b_{n-1}^{[1]} \\ \ldots & & & \ldots & \\ b_0^{[n-2]} & b_1^{[n-2]} & b_2^{[n-2]} & & \\ b_0^{[n-1]} & b_1^{[n-1]} & & & \\ b_0^{[n]} & & & & \end{array} \tag{1.9}$$

Osservazione 1.3 *Mediante tale algoritmo la valutazione di un polinomio nella base di Bernstein costa* $\frac{n(n+1)}{2}$ *addizioni/sottrazioni ed* $n(n+1)$ *moltiplicazioni.*

```
function y=decast(c,t)
% calcola il valore di un polinomio nella base di Bernstein
% definito in [0,1] dai coefficienti c nei punti t mediante
% l'algoritmo di de Casteljau;
% c --> coefficienti del polinomio
% t --> vettore dei punti
% y <-- vettore dei valori del polinomio nei punti
np1=length(c);
n=np1-1;
m=length(t);
for k=1:m
 w=c;
 d1=t(k);
 d2=1.0-t(k);
 for j=1:n
   for i=i:np1-j
     w(i)=d1.*w(i+1)+d2.*w(i);
   end
 end
y(k)=w(1);
end
```

Osservazione 1.4 *Il valore di un polinomio di Bernstein agli estremi è banalmente dato dai suoi coefficienti* b_0 *e* b_n*, cioè:*

$$p(0) := b_0 \quad e \quad p(1) := b_n.$$

1.3.5 Suddivisione

Una utile applicazione dell'algoritmo di de Casteljau è quella di poter essere utilizzato per suddividere un polinomio $p(t)$ in un punto t_c, ottenendo i suoi coefficienti nelle basi di Bernstein per gli intervalli $[0, t_c]$ e $[t_c, 1]$; tali coefficienti risultano essere rispettivamente:

$$b_0^{[0]}, b_0^{[1]}, \ldots, b_0^{[n]} \quad \text{e} \quad b_0^{[n]}, b_1^{[n-1]}, \ldots, b_n^{[0]} \tag{1.10}$$

che corrispondono ai lati verticale e diagonale dello schema triangolare 1.9. Si noti che, grazie all'invarianza per traslazione e scala vista, questi restano gli stessi anche quando i due intervalli vengono mappati in $[0, 1]$ mediante le trasformazioni

$$\frac{t}{t_c} \equiv u \in [0,1] \quad \text{e} \quad \frac{t - t_c}{1 - t_c} \equiv v \in [0,1].$$

Per dimostrare quanto asserito, notiamo che le quantità $b_i^{[k]}$ generate mediante la formula ricorrente 1.8 possono essere espresse in termini dei coefficienti iniziali b_i nella base di Bernstein nella forma:

$$b_i^{[k]} = \sum_{j=0}^{k} b_{i+j} B_{j,k}(t_c) \quad i = 0, \ldots, k$$

che è facilmente verificata per come sono stati costruiti. I coefficienti 1.10 possono allora essere scritti come:

$$b_0^{[k]} = \sum_{j=0}^{k} b_j B_{j,k}(t_c) \quad \text{e} \quad b_k^{[n-k]} = \sum_{j=0}^{n-k} b_{k+j} B_{j,n-k}(t_c) \quad k = 0, \ldots, n. \tag{1.11}$$

Si noti che la suddivisione descritta altro non è che un cambio di base da Bernstein in $[0, 1]$ a Bernstein in $[0, t_c]$ e $[t_c, 1]$ rispettivamente. A questo proposito si richiama il seguente risultato.

Teorema 1.4 *Sia $\{B_{j,n}(t)\}$ la base di Bernstein in $[0, 1]$, allora le basi di Bernstein in $[0, t_c]$ e $[t_c, 1]$, che indicheremo rispettivamente con $\{\bar{B}_{j,n}(t)\}$ e $\{\tilde{B}_{j,n}(t)\}$, sono date da:*

$$[B_{0,n}, B_{1,n}, \ldots, B_{n,n}] = [\bar{B}_{0,n}, \bar{B}_{1,n}, \ldots, \bar{B}_{n,n}]\Gamma$$

$$[B_{0,n}, B_{1,n}, \ldots, B_{n,n}] = [\tilde{B}_{0,n}, \tilde{B}_{1,n}, \ldots, \tilde{B}_{n,n}]\Lambda$$

con

$$\Gamma = \begin{pmatrix} B_{0,0}(t_c) & 0 & \ldots & 0 \\ B_{0,1}(t_c) & B_{1,1}(t_c) & \ldots & 0 \\ \ldots & & & \\ B_{0,n}(t_c) & B_{1,n}(t_c) & \ldots & B_{n,n}(t_c) \end{pmatrix}$$

e

$$\Lambda = \begin{pmatrix} B_{0,n}(t_c) & B_{1,n}(t_c) & \dots & B_{n,n}(t_c) \\ \dots & & & \\ 0 & \dots & B_{0,1}(t_c) & B_{1,1}(t_c) \\ 0 & \dots & 0 & B_{0,0}(t_c) \end{pmatrix}$$

dim.
Se sostituiamo $t = t_c u$, $1 - t = (1 - u) + (1 - t_c)u$ e $t = t_c(1 - v) + v$, $1 - t = (1 - t_c)(1 - v)$ nella definizione 1.7 della funzione base $B_{j,n}(t)$ $t \in [0, 1]$, si ottiene il seguente sviluppo in termini delle basi $\bar{B}_{i,n}(u)$ e $\tilde{B}_{i,n}(v)$ su $u \in [0, 1]$ e $v \in [0, 1]$ rispettivamente.

$$B_{j,n}(t) = \binom{n}{j}(1 - t)^{n-j}t^j =$$

$$= \binom{n}{j}[(1 - u) + (1 - t_c)u]^{n-j}[t_c u]^j$$

e per lo sviluppo della potenza di un binomio

$$= \binom{n}{j}\sum_{k=0}^{n-j}\binom{n-j}{k}(1 - u)^{n-j-k}(1 - t_c)^k u^k t_c^j u^j$$

shiftando gli indici della sommatoria si ha

$$= \binom{n}{j}\sum_{k=j}^{n}\binom{n-j}{k-j}(1 - u)^{n-k}(1 - t_c)^{k-j}u^{k-j}t_c^j u^j$$

e osservando che

$$\binom{n}{j}\binom{n-j}{k-j} \equiv \binom{n}{k}\binom{k}{j}$$

si ha

$$= \sum_{k=j}^{n}\binom{k}{j}(1 - t_c)^{k-j}t_c^j\binom{n}{k}(1 - u)^{n-k}u^k$$

$$= \sum_{k=j}^{n} B_{j,k}(t_c)\bar{B}_{k,n}(u)$$

che dimostra il primo cambio di base; per il secondo si procede analogamente.

È ora banale osservare che quanto si ottiene con l'algoritmo di de Casteljau applicato in un punto t_c e che si è riscritto nella forma 1.11, corrisponde proprio ai coefficienti della rappresentazione dello stesso polinomio,

ma nelle basi di Bernstein in $[0, t_c]$ e $[t_c, 1]$; infatti da

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) = [B_{0,n}(t), B_{1,n}(t), \ldots, B_{n,n}(t)] \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_n \end{bmatrix}$$

applicando il cambio di base, per esempio in $[0, t_c]$ si ha

$$= [\bar{B}_{0,n}(u), \bar{B}_{1,n}(u), \ldots, \bar{B}_{n,n}(u)]\Gamma \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_n \end{bmatrix}$$

dove

$$\Gamma \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_n \end{bmatrix}$$

equivale proprio alla rappresentazione della prima delle 1.11; analogamente per la seconda.

Si noti che ogni suddivisione di un polinomio raddoppia il numero di coefficienti di Bernstein richiesti per rappresentarlo sull'intero intervallo originale. Applicando un processo iterativo di suddivisione uniforme i valori dei coefficienti su ciascun sottointervallo convergono ai valori del polinomio. Questa osservazione è alla base della procedura per generare una approssimazione lineare a tratti di un polinomio di Bernstein che a sua volta è alla base di numerose applicazioni.

Si noti inoltre che una suddivisione, dal punto di vista del costo computazionale, altro non è che una valutazione in cui si memorizzano alcuni coefficienti intermedi. L'idea di valutare/suddividere in un punto dell'intervallo, quindi utilizzare i coefficienti della suddivisione per valutare/suddividere ulteriormente, fa sì che il condizionamento per ogni valutazione succesiva migliori. Questo è dovuto semplicemente al fatto che ogni sottointervallo, una volta operata una suddivisione, viene rimappato in $[0, 1]$ per la successiva valutazione/suddivisione.

1.3.6 Derivata

La derivata di un polinomio $p(t)$ nella base di Bernstein sarà espressa da:

$$p'(t) = \sum_{i=0}^{n} b_i B'_{i,n}(t)$$

dove

$$B'_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t)).$$

Sostituendo si ottiene:

$$p'(t) = n\sum_{i=0}^{n} b_i[B_{i-1,n-1}(t) - B_{i,n-1}(t)] =$$

$$= n\sum_{i=0}^{n} b_i B_{i-1,n-1}(t) - n\sum_{i=0}^{n} b_i B_{i,n-1}(t) =$$

$$= n\sum_{i=0}^{n-1} b_{i+1} B_{i,n-1}(t) - n\sum_{i=0}^{n-1} b_i B_{i,n-1}(t) =$$

$$= \sum_{i=0}^{n-1} d_i B_{i,n-1}(t)$$

con $d_i = n(b_{i+1} - b_i) \quad i = 0, \ldots, n-1$ che sono i coefficienti del polinomio derivata prima, di grado $n-1$, nella base di Bernstein.

Osservazione 1.5 *I valori della derivata di un polinomio di Bernstein agli estremi sono banalmente dati dai coefficienti d_0 e d_{n-1}, cioè:*

$$p'(0) := d_0 \quad e \quad p'(1) := d_{n-1}.$$

Osservazione 1.6 *La derivata di un polinomio di Bernstein di grado n e definito su $[a, b]$ è data da:*

$$p'(x) = \frac{1}{b-a} p'(t)$$

con $p'(t)$ la derivata di $p(t)$ in $[0, 1]$.

Osservazione 1.7 *Nel caso la situazione richiedesse contemporaneamente il valore ed il valore della derivata prima in un punto $\bar{t}$, si può procedere nel seguente modo: si applichi l'algoritmo di de Casteljau per valutare il polinomio ottenendo $p(\bar{t}) = b_0^{[n]}$, quindi la derivata prima sarà data da $p'(\bar{t}) = n(b_0^{[n]} - b_0^{[n-1]}) \equiv n(b_1^{[n-1]} - b_0^{[n]})$; questo deriva dal fatto che l'algoritmo di de Casteljau nel valutare in $\bar{t}$ il polinomio ne fornisce una rappresentazione nella base di Bernstein negli intervalli $[0, \bar{t}]$ e $[\bar{t}, 1]$ e negli estremi dell'intevallo di definizione il valore ed il valore della derivata sono noti in modo banale.*

```
function [y,yd]=decast_der(c,t)
% calcola il valore ed il valore della derivata di un polinomio
% nella base di Bernstein definito in [0,1] dai coefficienti c
% nei punti t mediante l'algoritmo di de Casteljau;
% c --> coefficienti del polinomio
% t --> vettore dei punti
% y <-- vettore dei valori del polinomio nei punti
% yd <-- vettore dei valori di derivata del polinomio nei punti
np1=length(c);
n=np1-1;
m=length(t);
for k=1:m
 w=c;
 d1=t(k);
 d2=1.0-t(k);
 for j=1:n
   for i=i:np1-j
     w(i)=d1.*w(i+1)+d2.*w(i);
   end
 end
y(k)=w(1);
yd(k)=n(w(2)-w(1))/d2;
end
```

1.3.7 Antiderivata e integrazione

Sia $p(t) = \sum_{i=0}^{n-1} d_i B_{i,n-1}(t)$ un polinomio di grado $n-1$ definito in $[0,1]$ nella base di Bernstein. Una sua antiderivata o primitiva sarà un polinomio di grado n che nella base di Bernstein può essere espressa come

$$p^{-1}(t) = \sum_{i=0}^{n} b_i B_{i,n}(t)$$

con

$$b_{i+1} := \frac{d_i}{n} + b_i \quad i = 0, \ldots, n-1$$

avendo fissato un valore per b_0 (per esempio $b_0 := 0$).
La scelta arbitraria del valore iniziale b_0 è data dal fatto che esistono più primitive tutte differenti a meno di una costante additiva.

Volendo procedere a determinare l'integrale definito di un polinomio $p(t)$ si avrà:

$$\int_0^1 p(t)dt = [p^{-1}(t)]_0^1 = b_n - b_0 = b_n$$

se, come suggerito prima, si sceglie $b_0 = 0$. In particolare sarà poi:

$$b_n = \frac{d_{n-1}}{n} + b_{n-1} == \frac{d_{n-1}}{n} + \frac{d_{n-2}}{n} + b_{n-2} = \ldots = \frac{1}{n}\sum_{i=0}^{n-1} d_i.$$

Osservazione 1.8 *L'integrale definito di ogni polinomio di Bernstein di grado n in* $[0,1]$ *è la costante* $\frac{1}{n+1}$.
Infatti $B_{i,n}(t)$ *è un polinomio nella base di Bernstein di coefficienti*

$$d_j = \begin{cases} 0 & j \neq i \\ 1 & j = i \end{cases}$$

così che

$$\int_0^1 B_{i,n}(t)dt = \frac{1}{n+1}\sum_{j=0}^{n} d_j = \frac{1}{n+1}.$$

Osservazione 1.9 *L'integrale definito di ogni polinomio base di Bernstein di grado n in* $[a,b]$ *è la costante* $\frac{b-a}{n+1}$.
Questo è banalmente vero pensando di calcolare

$$\int_a^b B_{i,n}(x)dx$$

effettuando un cambio di variabile da $[a,b]$ *a* $[0,1]$*; infatti sarà:*

$$\int_a^b B_{i,n}(x)dx = (b-a)\int_0^1 B_{i,n}(t)dt = \frac{b-a}{n+1}$$

1.4 Un'applicazione: le curve di Bézier

Bézier, negli anni '60, fu l'ideatore di uno dei primi sistemi CAD, UNISURF [Bez70], usato dalla casa automobilistica francese Renault. La teoria delle curve di Bézier, su cui era basato UNISURF, nasce dal rappresentare un polinomio nella base di Bernstein.
Una curva piana di Bézier $\mathbf{C}(t)$ di grado n può essere definita a partire da un insieme di punti di controllo $P_i \in \mathbf{E}^2 \quad i = 0, \ldots, n$ ed è data da

$$\mathbf{C}(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad t \in [0,1].$$

Si consideri per esempio una curva di grado 3. La Fig. 1.6 a sinistra mostra una tale curva e i punti di controllo associati P_0, P_1, P_2 e P_3. Si può vedere semplicemente che i punti estremi della curva coincidono con

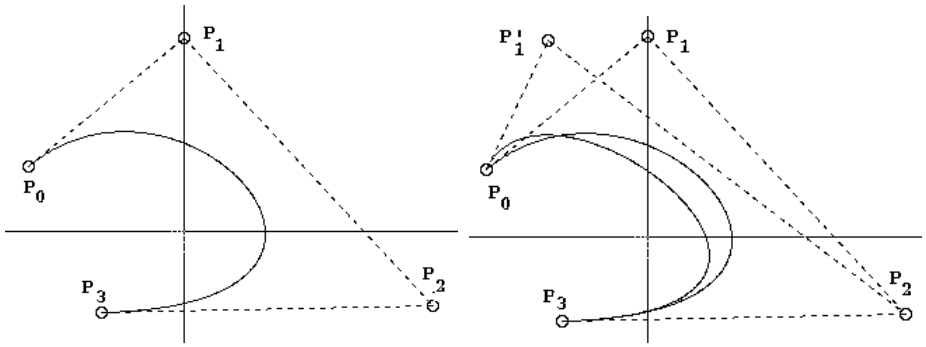


Figura 1.6: Esempio di curva di Bézier; caso $n = 3$ a sinistra; modifica di forma mediante spostamento di un punto di controllo a destra.

P_0 e P_3 e che i punti di controllo non sono punti della curva.
Spostando i punti di controllo si influenza la forma della curva; la Fig. 1.6 a destra mostra l'effetto di spostare il punto P_1.
E' facile pensare ad un software interattivo che permetta ad un utente di muovere un punto di controllo e quindi deformare la curva in una maniera che risulti subito intuitiva. Questa semplice possibilità offerta dai punti di controllo è il fondamento dell'importanza e utilità pratica delle curve di Bézier.

Le curve di Bézier sono utilizzate in molti sistemi CAD, sistemi di disegno e software o linguaggi grafici come Adobe Illustrator, Postscript, OpenGL, Xfig e tanti altri.

Capitolo 2

Interpolazione

Siano assegnate $n+1$ osservazioni in corispondenza di $n+1$ punti distinti, cioè siano assegnati (x_i, y_i) $i = 0, \ldots, n$. Nel caso generale viene individuato uno spazio a dimensione finita Φ di funzioni reali e una loro base di rappresentazione $\phi_0, \phi_1, \ldots, \phi_n$ così che ogni elemento $\phi \in \Phi$ sia definito da una loro combinazione lineare, cioè

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

Allora il problema di interpolare i dati assegnati con una funzione $\phi \in \Phi$ consiste nel determinare la funzione ϕ^* (od anche i coefficienti $a_0^*, a_1^*, \ldots, a_n^*$ che la definiscono) tale che

$$\phi^*(x_i) = y_i \quad i = 0, \ldots n.$$

Spesso, in pratica, questo problema viene applicato per determinare una approssimazione su un intervallo di una data funzione; in questo caso viene assegnata una funzione continua $f(x)$, $x \in [a, b]$ e la si vuole approssimare con una funzione, in genere più semplice, $\phi \in \Phi$, ma in modo abbastanza fedele così che la funzione errore

$$R(x) = |f(x) - \phi(x)| < \epsilon \quad x \in [a, b]$$

con ϵ una costante sufficientemente piccola legata all'applicazione.

Nel caso generale qui descritto, la scelta dello spazio Φ è solitamente dettata dal fatto che esista sempre una soluzione unica al problema, dalla regolarità desiderata per l'interpolante e dalla particolarità dei dati.

In queste dispense si tratterà solamente il caso dell'interpolazione polinomiale che garantisce sempre l'esistenza e l'unicità della soluzione, una buona regolarità ed una buona ricostruzione in casi particolari.

2.1 Forma Monomiale

Teorema 2.1 *Siano dati $n+1$ punti (x_i, y_i), $i = 0, \ldots, n$ con x_i distinti, allora esiste ed è unico il polinomio $p \in \mathbf{P}_n$ che verifica le condizioni*

$$p(x_i) = y_i \quad i = 0, \ldots, n$$

dim. Si consideri $p(x)$ nella forma monomiale

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n$$

e si impongano le $n+1$ condizioni di interpolazione

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 \ldots + a_n x_0^n & = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 \ldots + a_n x_1^n & = y_1 \\ \ldots & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 \ldots + a_n x_n^n & = y_n \end{cases}$$

Si tratta di un sistema lineare di $n+1$ equazioni in $n+1$ incognite; se tale sistema ammette una ed una sola soluzione, allora tale sarà il polinomio di grado $\leq n$. In forma matriciale il sistema si presenterà come:

$$V\mathbf{a} = \mathbf{y} \tag{2.1}$$

con

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \ldots & & & & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

dove V è nota come matrice di Vandermonde; il determinante di tale matrice ha la seguente espressione analitica:

$$detV = \prod_{i,j=0, j>i}^{n} (x_j - x_i)$$

che nelle ipotesi che i punti x_i siano distinti risulta sempre non nullo. Segue che il sistema lineare ha un'unica soluzione e quindi $p(x)$ esiste ed è unico.

La dimostrazione del Teorema fornisce anche un metodo per procedere alla determinazione del polinomio interpolante risolvendo il sistema lineare 2.1; purtroppo questo non risulta numericamente stabile a causa del mal condizionamento del problema di determinare la soluzione del sistema lineare con matrice dei coefficienti la matrice di Vandermonde.

2.2 Forma di Bernstein

Nella sezione precedente si è dimostrata l'esistenza ed unicità del polinomio interpolante nella base monomiale, ma il polinomio interpolante può essere rappresentato in una qualunque altra base. Procediamo quindi alla sua determinazione mediante soluzione di un sistema lineare, ma a partire dalla base di Bernstein. Siano dati $n+1$ punti (x_i, y_i), $i = 0, \ldots, n$ con x_i distinti in $[a, b]$ e si vuole determinare $p(x)$ nella base di Bernstein tale che

$$p(x_i) = y_i \quad i = 0, \ldots, n$$

Siano (t_i, y_i), $i = 0, \ldots, n$ i punti traslati e scalati in $[0, 1]$ mediante la 1.6 e sia

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \quad \text{con} \quad t \in [0, 1]$$

la forma polinomiale che vogliamo interpoli i valori assegnati. Imponendo le condizioni di interpolazione si ottiene il seguente sistema lineare:

$$B\mathbf{b} = \mathbf{y}$$

con

$$B = \begin{pmatrix} B_{0,n}(t_0) & B_{1,n}(t_0) & \ldots & B_{n,n}(t_0) \\ B_{0,n}(t_1) & B_{1,n}(t_1) & \ldots & B_{n,n}(t_1) \\ \ldots & & & \\ B_{0,n}(t_n) & B_{1,n}(t_n) & \ldots & B_{n,n}(t_n) \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Il vettore $\mathbf{b}$, soluzione del sistema lineare sopra dato fornirà sia il polinomio interpolante in $[0, 1]$ che in $[a, b]$. La soluzione di un sistema lineare $n \times n$ costa $O(n^3/3)$ operazioni floating point.

Si noti che la matrice dei polinomi base di Bernstein nei punti risulta meglio condizionata che non la matrice di Vandermonde, come si può osservare nella Tab. 2.2 in cui vengono mostrati gli indici di condizionamento delle matrici calcolate su punti equidistanti nell'intervallo di definizione mediante polinomi di Bernstein e base delle potenze.

2.3 Forma di Newton

Come accennato nell'introduzione a questo capitolo, a seconda del problema che si vuole risolvere, si può determinare una base più idonea delle altre

grado n	Bernst.	Potenze
3	2.27	1.41E02
4	4.52	9.04E02
5	9.65	3.78E03
6	21.60	2.50E04
7	49.98	1.30E05
8	118.38	8.10E05
9	285.34	5.08E06
10	697.14	3.04E07

Tabella 2.1: Indici di condizionamento delle matrici ottenute valutando i polinomi di Bernstein e la base delle potenze in punti equidistanti.

sia dal punto di vista del condizionamento, ma anche che permetta di usare un metodo più efficiente oltre che stabile. Nel caso dell'interpolazione polinomiale sicuramente la base più idonea è quella di Newton.

Siano dati i punti x_i, $i = 0, \ldots, n$ distinti, allora le funzioni

$$1, (x - x_0), (x - x_0)(x - x_1), \ldots, (x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

formano una base polinomiale per lo spazio $\mathbf{P}_n$ detta base di Newton. Assegnati (x_i, y_i), $i = 0, \ldots, n$, si voglia determinare il polinomio interpolante nella base di Newton, cioè si voglia determinare il polinomio $p(x)$ nella forma

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}).$$

Imponendo le condizioni di interpolazione si ottiene il sistema lineare:

$$N\mathbf{c} = \mathbf{y}$$

con

$$N = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & (x_1 - x_0) & 0 & \ldots & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \ldots & 0 \\ \vdots & & & & \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \ldots & (x_n - x_0) \cdots (x_n - x_{n-1}) \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

che risulta facilmente risolubile per sostituzione in avanti con una complessità di $O(n^2)$. Tale soluzione è ottimale da ogni punto di vista, inoltre è applicabile una variante dell'algoritmo di Horner per la sua valutazione. Purtroppo se il polinomio così determinato deve essere sottoposto ad ulteriori elaborazioni come per esempio a derivazione o integrazione, allora tale forma non è più raccomandabile in quanto non sono noti algoritmi efficienti per procedere.

2.4 Forma di Lagrange

Con forma di Lagrange si intende un polinomio nella base delle funzioni cardinali o polinomi elementari di Lagrange. Siano dati i punti x_i, $i = 0, \ldots, n$ distinti, allora le funzioni elementari di Lagrange sono i polinomi $L_i(x)$ di grado n che soddisfano le seguenti condizioni:

$$L_i(x_j) = \begin{cases} 1 & \text{se} \quad i = j \\ 0 & \text{se} \quad i \neq j \end{cases}$$

Questa base risulta la più semplice in cui risolvere un problema di interpolazione; assegnati (x_i, y_i), $i = 0, \ldots, n$, il problema è risolto banalmente da

$$p(x) = \sum_{i=0}^{n} y_i L_i(x)$$

infatti per le condizioni che definiscono i polinomi $L_i(x)$, banalmente in ogni punto x_i l'espressione polinomiale data varrà y_i.
Dal punto di vista computazionale determinare tale polinomio non costa nulla essendo i coefficienti del polinomio in tale base proprio i valori y_i assegnati. Già la valutazione del polinomio interpolante comporta la determinazione, ma soprattutto la valutazione dei polinomi $L_i(x)$. Per il Teorema 1.2 i polinomi $L_i(x)$ avendo come radici i punti x_j con $j = 0, \ldots, n$ e $j \neq i$ saranno della forma

$$L_i(x) = c_i(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)$$

con c_i una costante da determinare in modo che $L_i(x_i) = 1$; allora

$$c_i = \frac{1}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

da cui in definitiva

$$L_i(x) = \frac{\prod_{j=0 j \neq i}^{n} (x - x_j)}{\prod_{j=0 j \neq i}^{n} (x_i - x_j)} \quad i = 0, \ldots, n$$

La valutazione di questo polinomio interpolante può essere resa competitiva con il polinomio di interpolazione nella forma di Newton, ma come quello risulta difficoltosa la valutazione della derivata. Come si vedrà nel capitolo dedicato all'integrazione numerica di funzioni, tale rappresentazione risulterà molto importante.

2.5 Errore nell'interpolazione polinomiale

Assegnate le $n+1$ osservazioni y_i, $i = 0, \ldots, n$ in corrispondenza dei punti distinti x_i, si è visto come costruire il polinomio interpolante di grado minimo $p(x)$. Se i valori y_i altro non sono che i valori di una funzione $f(x)$ nei punti x_i, cioè $y_i \equiv f(x_i)$, $i = 0, \ldots, n$ con f definita in $[a, b]$, ha senso chiedersi quanto sia grande l'errore di interpolazione

$$R(x) = f(x) - p(x)$$

che si commette in un punto $\bar{x} \in [a, b]$ diverso dai punti di interpolazione x_i. Per dare una risposta a tale domanda è necessario fare alcune ipotesi di regolarità sulla funzione $f(x)$; infatti se $f(x)$ non è almeno continua, l'errore fra i punti di interpolazione può essere qualunque, anche grandissimo. Si dimostra il seguente teorema.

Teorema 2.2 *Sia $f(x) \in C^{(n+1)}_{[a,b]}$, e sia $\bar{x}$ un punto qualsiasi in $[a, b]$. Allora esiste un punto ξ (dipendente da $\bar{x}$) interno ad $[a, b]$ per cui*

$$f(\bar{x}) = p(\bar{x}) + \frac{w(\bar{x})}{(n+1)!} f^{(n+1)}(\xi)$$

con $w(\bar{x}) = (\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)$.

Corollario 2.1 *Posto $M_{(n+1)} = \max_{x \in [a,b]} |f^{(n+1)}(x)|$, un limite superiore all'errore di interpolazione $R(x) = f(x) - p(x)$ è dato da*

$$|R(x)| \leq \frac{|w(x)|}{(n+1)!} M_{(n+1)}$$

Esempio 2.1 *Sia $f(x) = e^x$; per ogni $x \in [a, b]$ si ha $M_{(n+1)} = e^b$ e per ogni scelta dei punti x_i, $|w(x)| \leq (b-a)^{(n+1)}$, da cui*

$$\max_{x \in [a,b]} |R(x)| \leq \frac{(b-a)^{(n+1)}}{(n+1)!} e^b.$$

In questo caso si ricava

$$\lim_{n\to\infty}\{\max_{x\in[a,b]}|R(x)|\}=0,$$

cioè il polinomio interpolante $p(x)$ converge a $f(x)$ uniformemente su $[a,b]$ quando il numero n dei punti di interpolazione tende all'infinito.

Osservazione 2.1 *È opportuno rilevare che, anche se la funzione $f(x)\in C^{\infty}_{[a,b]}$, la successione $p_n(\bar{x})$ dei valori assunti dai polinomi di interpolazione di grado n in un punto $\bar{x}\in[a,b]$ può non convergere ad $f(x)$, per n che tende all'infinito.*

Esempio 2.2 *Per il polinomio di interpolazione in n punti equidistanti della funzione di Runge*

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

si può dimostrare che al crescere di n la successione dei polinomi di interpolazione sui punti

$$x_i=\frac{10}{n}i-5$$

non converge puntualmente ad $f(x)$ e che i corrispondenti resti diventano in modulo arbitrariamente grandi in punti dell'intervallo $[-5,5]$.
La Tab. 2.2 mostra l'errore massimo in valore assoluto fra la funzione di Runge e i polinomi interpolanti. In particolare l'errore è grande vicino agli estremi ed aumenta con n (vedi Fig.2.1).

Risultato 2.1 *Se oltre a fare delle ipotesi sulla regolarità della funzione (almeno $C^1_{[a,b]}$), si utilizzano opportune distribuzioni dei punti di interpolazione (come per esempio gli zeri del polinomio di Chebishev di grado $n+1$[1]), si può dimostrare la convergenza del polnomio interpolante $p_n(x)$ alla $f(x)$ per $x\in[a,b]$ quando $n\to\infty$ (si veda Fig.2.2 e la si confronti con Fig.2.1).*

Nonostante l'ultimo risultato, in generale, non si può essere sicuri che scegliendo opportuni punti di interpolazione ed un polinomio di interpolazione di grado alto, si otterrà una buona approssimazione di una certa

[1] $x_i=cos(\frac{(2i+1)\pi}{2(n+1)})$, $i=0,\ldots,n$ sono gli $n+1$ zeri reali del polinomio di Chebishev di grado $n+1$ definito in $[-1,1]$; se i punti di interpolazione sono in $[a,b]$, si applichi un mapping degli zeri da $[-1,1]$ in $[a,b]$.

$n+1$ punti	$\max_x \lvert R(x)\rvert$	$n+1$ punti	$\max_x \lvert R(x)\rvert$
11	1.92	12	0.55
21	58.59	22	17.29
31	2277.70	32	665.64

Tabella 2.2: Errore Massimo per l'interpolazione polinomiale della funzione di Runge su punti equidistanti.

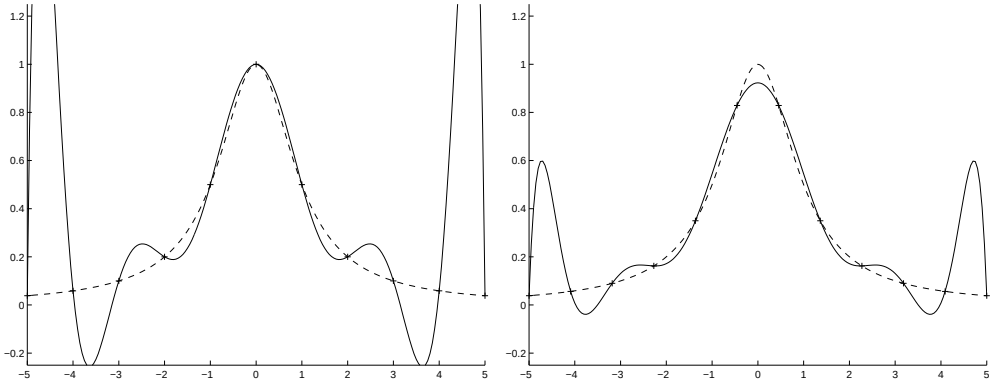


Figura 2.1: Interpolazione polinomiale della funzione di Runge su punti equidistanti; a sinistra 11 punti, a destra 12.

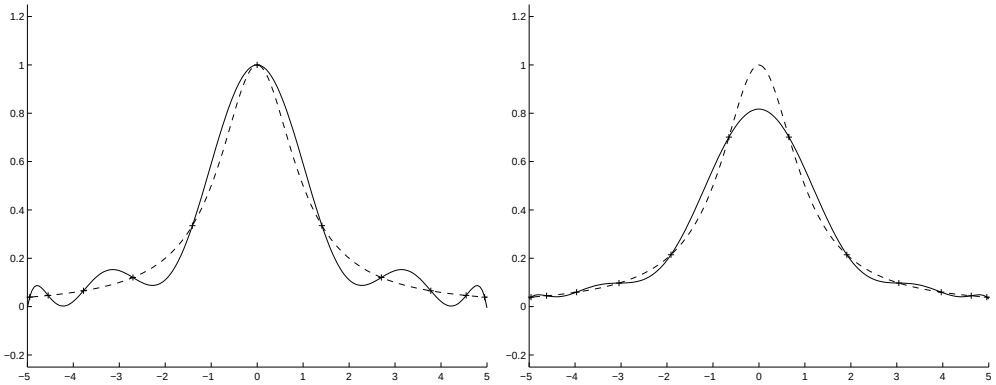


Figura 2.2: Interpolazione polinomiale della funzione di Runge su punti zeri di Chebishev; a sinistra 11 punti, a destra 12.

funzione. Si può anche vedere che i polinomi non sono sufficientemente flessibili ad approssimare funzioni che hanno differenti andamenti in differenti parte dell'intervallo di definizione. Questo è il caso di molte funzioni

$n+1$ punti	$\max_x \|R(x)\|$	$n+1$ punti	$\max_x \|R(x)\|$
11	0.1089	12	0.1828
21	0.0153	22	0.0253
31	0.0021	32	0.0035

Tabella 2.3: Errore Massimo per l'interpolazione polinomiale della funzione di Runge su punti zeri di Chebyshev.

che descrivono un fenomeno fisico o la forma di un oggetto. Oltre a questo si sottolinea il pericolo, dal punto di vista numerico, di lavorare con polinomi di grado alto. Numericamente parlando si dovrebbero usare solo polinomi di grado basso. Queste considerazioni sono alla base dell'interpolazione con polinomi a tratti che consiste nell'approssimare una funzione con differenti polinomi di grado basso in differenti parti dell'intervallo di interesse.

2.6 Interpolazione polinomiale a tratti

Con interpolazione polinomiale a tratti si intende l'interpolazione di un set di dati su un intervallo con più polinomi ciascuno dei quali definito in un sottointervallo dell'intervallo dato. In particolare siano assegnate $m+1$ osservazioni in corispondenza di $m+1$ punti distinti e ordinati, cioè siano assegnati (x_i, y_i) $i = 0, \ldots, m$ con $x_i < x_{i+1}$. Allora un intepolante poliniomiale a tratti consiste in m polinomi $p_i(x)$, $i = 0, \ldots, m-1$ di grado n, definiti sugli intervalli $[x_i, x_{i+1}]$ con le seguenti proprietà:

- $$p_{i-1}(x_i) \equiv p_i(x_i) = y_i \quad i = 1, \ldots, m-1$$
- fissato $k \leq n-1$, non negativo, deve valere
$$p_{i-1}^{(\ell)}(x_i) \equiv p_i^{(\ell)}(x_i) \quad \ell = 1, \ldots, k \quad i = 1, \ldots, m-1$$

Si dirà che l'interpolante a tratti è di classe C^k intendendo che è una funzione continua fino alla derivata di ordine k; quando $k = n-1$ si ha la massima continuità e l'interpolante polinomiale a tratti viene detta funzione **spline**.

Prima di procedere a presentare due metodi di interpolazione molto utilizzati in pratica si richiama l'interpolazione di Hermite con un polinomio cubico nella base di Bernstein; questa consiste nel determinare il polinomio cubico $p(t)$ che soddisfa le seguenti quattro condizioni di

interpolazione:

$$p(0) = y_0, \quad p'(0) = y'_0, \quad p(1) = y_1, \quad p'(1) = y'_1.$$

Si dovranno determinare i coefficienti b_0, b_1, b_2 e b_3. Due di loro sono banalmente determinati:

$$b_0 := y_0, \quad b_3 := y_1.$$

Per i rimanenti, richiamiamo il valore della derivata di un polinomio nella base di Berstein negli estremi (osservazione 1.5):

$$p'(0) = 3(b_1 - b_0), \quad p'(1) = 3(b_3 - b_2).$$

Da queste si possono ricavare b_1 e b_2:

$$b_1 := y_0 + \frac{1}{3}y'_0, \quad b_2 := y_1 - \frac{1}{3}y'_1.$$

2.6.1 Interpolazione locale

Per interpolazione locale si intende un metodo che determina il polinomio a tratti interpolante ricavando ogni singolo polinomio $p_i(x)$ indipendentemente dagli altri. Vediamo questo modo di procedere nel caso particolare di polinomi a tratti cubici con regolarità C^1; i singoli polinomi cubici $p_i(x)$ possono essere espressi nella base di Bernstein e determinati in modo da interpolare le due osservazioni y_i e y_{i+1} in corrispondenza di x_i e x_{i+1} sull'intervallo $[x_i, x_{i+1}]$ e i valori di derivata y'_i e y'_{i+1} sempre in corrispondenza di x_i e x_{i+1}. Si tratta di interpolanti cubici di Hermite che banalmente costituiranno un unico interpolante cubico a tratti dei punti (x_i, y_i) $i = 0, \ldots, m$ e di classe C^1; la Fig. 2.3 mostra il risultato di una tale interpolazione sulla funzione test di Runge su punti equidistanti.

N. punti	$\max_x \lvert R(x)\rvert$	N. punti	$\max_x \lvert R(x)\rvert$
11	0.0182	12	0.1114
21	0.0111	22	0.0181
31	0.0042	32	0.0048

Tabella 2.4: Errore Massimo per l'interpolazione polinomiale cubica a tratti C^1 della funzione di Runge su punti equidistanti e derivate stimate.

Operativamente ogni intervallo $[x_i, x_{i+1}]$ viene idealmente mappato nell'intervallo $[0, 1]$ ed in questo si applica una interpolazione cubica di Hermite vista, nella base di Bernstein. Le derivate assegnate y'_i dovranno essere

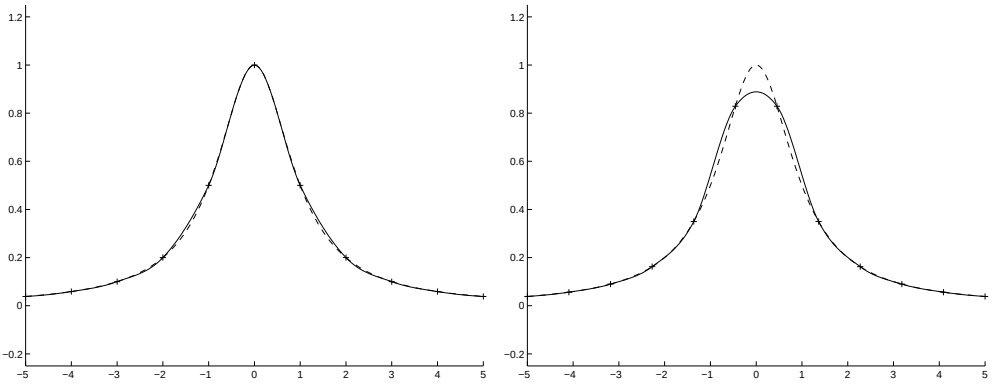


Figura 2.3: Interpolazione polinimiale cubica a tratti C^1 della funzione di Runge su punti equidistanti e derivate stimate; a sinistra 11 punti, a destra 12.

N. punti	$\max_x \lvert R(x)\rvert$	N. punti	$\max_x \lvert R(x)\rvert$
11	0.0129	12	0.0293
21	0.0013	22	0.0029
31	0.0005	32	0.0006

Tabella 2.5: Errore Massimo per l'interpolazione polinomiale cubica a tratti C^1 della funzione di Runge su punti equidistanti e derivate analitiche.

opportunamente scalate e per la precisione, con riferimento all'intervallo $[i, i+1]$, le osservazioni da interpolare saranno:

$$(0, y_i) \quad (0, y_i' h_{i+1}) \quad (1, y_{i+1}) \quad (1, y_{i+1}' h_{i+1})$$

con $h_i = x_{i+1} - x_i \quad i = 0, \ldots, m-1$.

Stima delle derivate

I metodi di interpolazione locale fanno uso delle derivate y_i' in corrispondenza dei punti assegnati x_i. Se queste non sono date, devono essere calcolate come parte del problema di interpolazione. In letteratura esistono un certo numero di metodi per ottenere stime delle derivate a partire dai dati assegnati. Qui ne riportiamo una semplice classe: siano

$$\begin{aligned} h_i &= x_{i+1} - x_i \quad i = 0, \ldots, m-1 \\ f_i &= y_{i+1} - y_i \quad i = 0, \ldots, m-1 \\ d_i &= \tfrac{f_i}{h_i} \quad i = 0, \ldots, m-1 \end{aligned}$$

allora si definisce una stima delle derivate come:

$$D_i = (1 - \alpha_i)d_{i-1} + \alpha_i d_i \quad i = 1, \ldots m-1.$$

A seconda di come vengono scelti i parametri α_i si hanno differenti schemi; la scelta

$$\alpha_i = \frac{h_{i-1}}{h_{i-1} + h_i} \quad i = 1, \ldots m-1$$

porta alla stima di Bessel. Si noti che nel primo ed ultimo punto le stime date non sono definite e che questi devono essere trattati come casi particolari. Per la stima di Bessel si definisce:

$$D_0 = 2d_0 - D_1 \qquad D_m = 2d_{m-1} - D_{m-1}.$$

2.6.2 Interpolazione globale

Per interpolazione globale si intende un metodo che determina il polinomio a tratti interpolante ricavando i polinomi $p_i(x)$ in modo dipendente dagli altri. Vediamo questo modo di procedere nel caso particolare di polinomi a tratti cubici con massima regolarità C^2, anche detta **spline cubica di interpolazione**.
Esprimiamo i singoli polinomi come polinomi di Bernstein in $[x_i, x_{i+1}]$ con coefficienti q_i, r_i, s_i e q_{i+1} per $i = 0, \ldots, m-1$. Affinché due polinomi consecutivi, nella base di Bernstein, si raccordino C^2 devono essere soddisfatte le seguenti condizioni o insieme di condizioni:

$$\begin{aligned} \frac{q_i - s_{i-1}}{h_{i-1}} &= \frac{r_i - q_i}{h_i} && i = 1, \ldots, m-1 \\ \frac{q_i - 2s_{i-1} + r_{i-1}}{h_{i-1}^2} &= \frac{s_i - 2r_i + q_i}{h_i^2} && i = 1, \ldots, m-1. \end{aligned} \tag{2.2}$$

Si tratta di $2m-2$ equazioni lineari nelle $2m$ incognite r_i ed s_i, $i = 0, \ldots, m-1$ (i q_i, per soddisfare le condizioni di interpolazione, devono valere y_i, $i = 0, \ldots, m$); solitamente si aggiungono due ulteriori condizioni o si assegnano liberamente i valori per due incognite (frequentemente per r_0 ed s_{m-1}), così da avere un sistema quadrato. Il dover risolvere tale sistema lineare equivale a determinare i polinomi a tratti in modo dipendente gli uni dagli altri.

Osservazione 2.2 *Le condizioni sopra date derivano dalla definizione di $p_i(x)$ $i = 0, \ldots, m-1$ e dalle espressioni delle loro derivate prima e seconda;*

$$p_i(x) = q_i B_{0,3}(x) + r_i B_{1,3}(x) + s_i B_{2,3}(x) + q_{i+1} B_{3,3}(x)$$

$$p_i'(x) = \frac{3}{h_i}\left[(r_i - q_i)B_{0,2}(x) + (s_i - r_i)B_{1,2}(x) + (q_{i+1} - s_i)B_{2,2}(x)\right]$$

$$p_i''(x) = \frac{6}{h_i^2}\left[(s_i - 2r_i + q_i)B_{0,1}(x) + (q_{i+1} - 2s_i + r_i)B_{1,1}(x)\right]$$

Se poniamo

$$D_i = \frac{q_i - s_{i-1}}{h_{i-1}} = \frac{r_i - q_i}{h_i}$$

possiamo riscrivere le seconde condizioni 2.2 solo in termini di D_i e q_i: sostituendo $q_i - s_{i-1}$ con $h_{i-1}D_i$ e $-r_i + q_i$ con $-h_iD_i$ si ha:

$$\frac{h_{i-1}D_i - s_{i-1} + r_{i-1}}{h_{i-1}^2} = \frac{s_i - r_i - h_iD_i}{h_i^2};$$

sostituendo $-s_{i-1}$ con $h_{i-1}D_i - q_i$ e r_i con $h_iD_i + q_i$ si ha:

$$\frac{2h_{i-1}D_i - q_i + r_{i-1}}{h_{i-1}^2} = \frac{s_i - q_i - 2h_iD_i}{h_i^2};$$

infine sostituendo r_{i-1} con $h_{i-1}D_{i-1}+q_{i-1}$ e s_i con $-h_iD_{i+1}+q_{i+1}$ si ottiene:

$$\frac{2h_{i-1}D_i - q_i + h_{i-1}D_{i-1} + q_{i-1}}{h_{i-1}^2} = \frac{-h_iD_{i+1} - q_i - 2h_iD_i + q_{i+1}}{h_i^2}.$$

Semplificando

$$h_iD_{i-1} + 2D_i(h_i + h_{i-1}) + h_{i-1}D_{i+1} = \frac{h_i}{h_{i-1}}(q_i - q_{i-1}) + \frac{h_{i-1}}{h_i}(q_{i+1} - q_i)$$

per $i = 1, \ldots, m-1$.
Si tratta di un sistema di $m-1$ equazioni nelle $m+1$ incognite D_i; si possono allora assegnare D_0 e D_m (derivate agli estremi) e risolvere il seguente sistema quadrato $(m-1) \times (m-1)$ che, essendo la matrice a diagonale dominante, avrà una ed una sola soluzione

$$\begin{pmatrix} 2(h_1+h_0) & h_0 & 0 & \ldots & 0 \\ h_2 & 2(h_2+h_1) & h_1 & \ldots & 0 \\ \ldots & & & & \\ 0 & \ldots & 0 & h_{m-1} & 2(h_{m-1}+h_{m-2}) \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_{m-1} \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{h_1}{h_0}(q_1 - q_0) + \frac{h_0}{h_1}(q_2 - q_1) - h_1D_0 \\ \frac{h_2}{h_1}(q_2 - q_1) + \frac{h_1}{h_2}(q_3 - q_2) \\ \vdots \\ \frac{h_{m-1}}{h_{m-2}}(q_{m-1} - q_{m-2}) + \frac{h_{m-2}}{h_{m-1}}(q_m - q_{m-1}) - h_{m-2}D_m \end{pmatrix}.$$

Determinate le $D_1, D_2, \ldots D_{m-1}$ insieme alle assegnate D_0 e D_m si può procedere al calcolo degli m polinomi che costituiscono la polinomiale cubica a tratti di interpolazione. Tali polinomi possono essere valutati in $[0,1]$ facendo uso dei coefficienti:

$$q_i = y_i \quad i = 0, \ldots, m$$

$$q_i, \quad q_i + h_i D_i, \quad q_{i+1} - h_i D_{i+1}, \quad q_{i+1} \qquad i = 0, \ldots, m-1.$$

Osservazione 2.3 *Si noti che nel caso i punti siano tali per cui* $h_i \equiv 1 \quad \forall i$*, il sistema si semplifica in*

$$\begin{pmatrix} 4 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 \\ \ldots & & & & & \\ 0 & \ldots & 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{m-1} \end{pmatrix} = \begin{pmatrix} q_2 - q_0 - D_0 \\ q_3 - q_1 \\ q_4 - q_2 \\ \vdots \\ q_m - q_{m-2} - D_m \end{pmatrix}.$$

La Fig. 2.4 mostra il risultato dell'interpolazione della funzione test di Runge con una spline cubica su punti equidistanti.

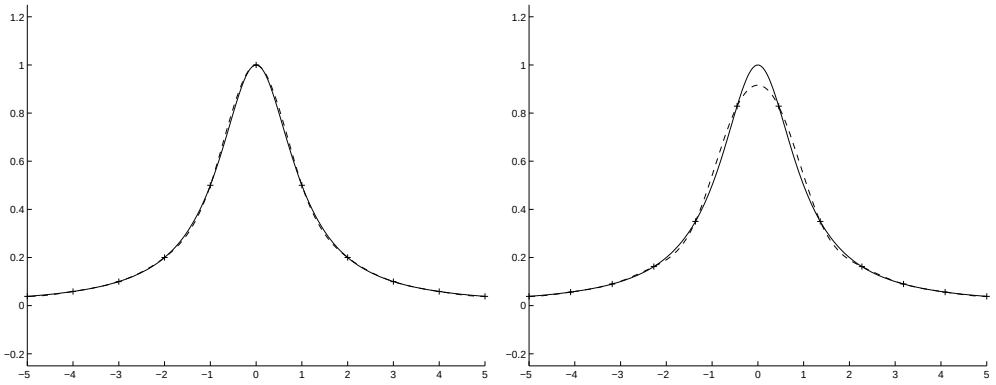


Figura 2.4: Interpolazione polinomiale cubica a tratti C^2 della funzione di Runge su punti equidistanti; a sinistra 11 punti, a destra 12.

Un esempio: controllo di un robot

Le curve spline sono utilizzate in numerose applicazioni industriali e sono alla base dei moderni sistemi CAD. In questo paragrafo vogliamo vedere un semplice esempio applicativo di una spline cubica di interpolazione. Nella robotica è frequente la situazione di avere dei motori che muovono

N. punti	$\max_x \lvert R(x)\rvert$	N. punti	$\max_x \lvert R(x)\rvert$
11	0.02193	12	0.08382
21	0.00318	22	0.00802
31	0.00084	32	0.00131
41	0.00063	42	0.00061

Tabella 2.6: Errore Massimo per l'interpolazione polinomiale cubica a tratti C^2 della funzione di Runge su punti equidistanti.

dei bracci meccanici i cui movimenti vengono gestiti da un calcolatore per compiere delle azioni ripetitive. Per focalizzare si può pensare ad un semplice braccio meccanico, come quello schematizzato in Fig. 2.5, composto da due parti (la spalla di lunghezza L_1 e il gomito di lunghezza L_2) rispettivamente mosse da due motori. Quando questi due motori ruotano, il braccio può posizionare la mano (la parte estrema del gomito) in un punto desiderato del piano su cui il braccio è posto. Il robot è gestito da un calcolatore che riceve l'input dell'utente (per esempio, la posizione desiderata della mano). Il computer calcola la rotazione che ogni motore deve effettuare per spostare la mano dalla posizione iniziale a quella finale. Successivamente, a intervalli regolari di tempo, il computer invia i comandi di rotazione ai due motori. In applicazioni come queste il computer usa una interpolazione spline per generare i comandi di rotazione. Il vantaggio di una spline è di fornire un movimento regolare ai motori del robot, evitando bruschi cambi di velocità e/o accelerazione, se regolare fino alla derivata

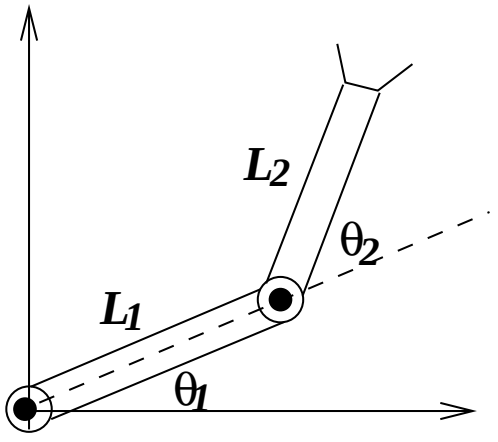


Figura 2.5: Schema di un braccio meccanico con due motori

seconda (nel caso cubico la spline è C^2).

Possiamo, grazie alla trigonometria, definire le espressione per le coordinate del gomito e della mano:

gomito:

$$\begin{aligned} x_{gomito} &= L_1 \cos(\theta_1) \\ y_{gomito} &= L_1 \sin(\theta_1) \end{aligned}$$

mano:

$$\begin{aligned} x_{mano} &= x_{gomito} + L_2 \cos(\theta_1 + \theta_2) = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \\ y_{mano} &= y_{gomito} + L_2 \sin(\theta_1 + \theta_2) = L_2 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \end{aligned}$$

Queste espresioni permettono di impostare gli angoli del gomito e della spalla che sono necessari per posizionare la mano nel punto specificato dalle coordinate (x_{mano}, y_{mano}). Per la precisione si ha:

$$R^2 = x_{mano}^2 + y_{mano}^2$$

$$\cos(\theta_2) = \frac{R^2 - L_1^2 - L_2^2}{2L_1 L_2}$$

$$\cos(\beta) = \frac{R^2 - L_1^2 - L_2^2}{2L_1 R}$$

$$\alpha = \arctan(\frac{y_{mano}}{x_{mano}})$$

$$\theta_1 = \begin{cases} \alpha + \beta & \text{se} \quad \theta_2 < 0 \\ \alpha - \beta & \text{se} \quad \theta_2 \geq 0 \end{cases}$$

Se si utilizza un polinomio di grado 3 per esprimere gli angoli dei giunti in funzione del tempo per $0 \leq t \leq T$, è posibile specificare quattro condizioni per ciascun angolo. Due di queste condizioni richiedono che il polinomio passi per il valore $\theta(0)$ e per il valore $\theta(T)$. Le altre due condizioni richiedono che la pendenza del polinomio debba essere nulla all'inizio e alla fine, in quanto il robot deve partire da una posizione di riposo e finire in una posizione di riposo.

Esempio 2.3 *Supponiamo che il braccio del robot abbia lunghezza* $L_1 = 4m$ *ed* $L_2 = 3m$. *Supponiamo inoltre che la mano debba spostarsi in* 2 *secondi dal punto* $x(0) = 6.5$, $y(0) = 0.0$ *al punto* $x(2) = 0.0$, $y(2) = 6.5$; *dalle espressioni otteniamo:*

$$\theta_1(0) = -18.717^o \quad \theta_1(2) = 71.283^o$$

$$\theta_2(0) = \theta_2(2) = 44.049^o$$

I polinomi di interpolazione questi valori e derivate nulle in 0 *e* 2 *sono:*

$$\theta_1(t) = -18.717 + 67.5t^2 - 22.5t^3$$

$$\theta_2(t) = 44.049$$

Se il computer utilizza questi polinomi per generare i comandi da inviare ai motori, si nota che la mano compie una traiettoria circolare in quanto l'angolo θ_2 fra i due bracci resta costante per l'intero percorso, mentre varia solo θ_1.

Osservazione 2.4 *Si noti che la traiettoria della mano non è una linea retta che congiunge la posizione iniziale e quella finale e che questo movimento potrebbe provocare delle collisioni con gli oggetti vicini.*

In molte aplicazioni è necessario controllare la traiettoria della mano con maggior precisione. Supponiamo di voler ottenere una traiettoria opportuna che vogliamo specificare con una serie di posizioni intermedie che devono essere raggiunte dalla mano del robot, in istanti unitari di tempo, partendo dalla posizione iniziale e terminando in quella finale.

Per ottenere un movimento regolare, non tanto della mano o traiettoria, quanto dei motori, per evitare bruschi cambi di velocità e accelerazione, si devono utilizzare delle funzioni spline per interpolare i valori di $\theta_1(t_i)$ e $\theta_2(t_i)$ in corrispondenza degli istanti t_i prefissati; $\theta_1(t_i)$ e $\theta_2(t_i)$ corispondono agli angoli che permettono di posizionare la mano nelle posizioni $(x(t_i), y(t_i))$ assegnate agli istanti t_i. La Fig.2.6 mostra l'ultimo fotogrammo generato da un programma di esempio che simula il comportamento di una tale robot che nel suo movimento con accelerazione continua deve partire da una posizione iniziale ed arrivare ad una finale passando in istanti unitari di tempo attraverso quattro posizioni predefinite. La corona circolare rappresentata in figura indica il raggio di azione possibile del robot avendo fissato per tale simulazione $L1 = 6$ ed $L_2 = 3$.

2.7 Un'applicazione: curve cubiche a tratti di interpolazione

Nei pacchetti grafici di disegno 2D, nei sistemi CAD e in numerose altre applicazioni la determinazione di curve mediante interpolazione di punti risulta una tecnica molto usuale. In questa sezione si vogliono presentare i due tipi sicuramente più frequentemente usati; l'interpolazione con una

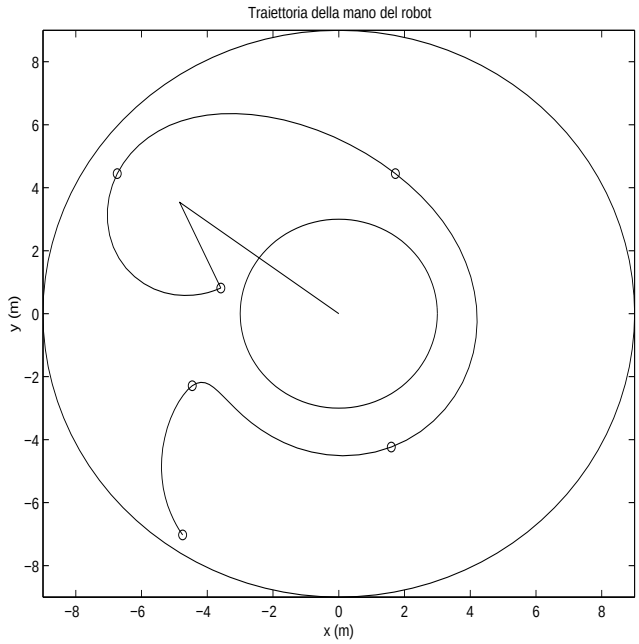


Figura 2.6: Schema di un braccio meccanico con due motori: output di un programma esempio

curva cubica a tratti C^1 e con una spline cubica.
Siano dati $m+1$ punti $Q_i \equiv (x_i, y_i)_{i=0,\ldots,m}$; si vogliono determinare la curva cubica a tratti C^1 e la spline cubica che interpolano i punti Q_i; si cerca una curva in forma parametrica

$$\mathbf{C}(t) = \begin{pmatrix} C_1(t) \\ C_2(t) \end{pmatrix}$$

tale che siano verificate le condizioni di interpolazione:

$$\begin{cases} C_1(t_i) = x_i \\ C_2(t_i) = y_i \end{cases} \qquad i = 0, \ldots, m$$

Il problema di interpolazione con una curva (funzione vettoriale a due componenti) si riduce a due problemi di interpolazione con funzioni scalari, cioè si cercano: la funzione $C_1(t)$ che interpola i punti (t_i, x_i) e la funzione $C_2(t)$ che interpola i punti (t_i, y_i).
Si noti che i valori parametrici t_i in corrispondenza dei quali interpolare non sono assegnati e devono essere determinati come parte del problema stesso. Vediamo due possibili modi.

Metodo della parametrizzazione uniforme. Consiste nel suddividere l'intervallo $[0,1]$ in m sottointervalli di uguali ampiezze, ovvero nel prendere $m+1$ punti equidistanti:

$$t_i = \frac{i-1}{m} \qquad i = 1, \ldots, m.$$

Questo tipo di parametrizzazione, però, non tiene conto della distanza relativa dei punti sulla curva.
Per ovviare a questo inconveniente si usa il metodo seguente:

Metodo della parametrizzazione della corda. In questo caso si tiene conto della geometria dei punti, poiché si mantengono invariati i rapporti:

$$\frac{t_{i+1} - t_i}{t_{i+2} - t_{i+1}} = \frac{\|Q_{i+1} - Q_i\|_2}{\|Q_{i+2} - Q_{i+1}\|_2}.$$

Per determinare i valori t_i si può procedere come segue:

$$\begin{aligned} &\tau_1 = 0 \\ &\tau_i = \tau_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad i = 2, \ldots, m+1 \end{aligned}$$

Infine si ottengono i valori:

$$t_i = \frac{\tau_i}{\tau_n} \quad i = 1, \ldots, m+1.$$

Determinati i parametri t_i, i metodi più utilizzati sono quelli visti nelle sezioni precedenti e per la precisione l'interpolazione polinomiale cubica a tratti C^1 e spline cubica che fornisce una curva C^2. La prima è molto interessante per il bassissimo costo computazionale e per la possibilità di avere $m+1$ parametri di forma consistenti nei moduli dei vettori tangenti nei punti di interpolazione; tali scalari giocano come parametri di tensione locale per la curva o globali se modificati tutti contemporaneamente. La Fig.2.7 mostra tre interpolazioni di 8 punti disposti ad esse con cubiche a tratti C^1 dove i vettori derivati, inizialmente stimati come indicato nella sezione 2.6.1, sono stati globalmente scalati di $1/2$ e poi $1/4$ evidenziando il tensionamento relativo. La Fig.2.8 mostra invece una curva di interpolazione spline cubica con parametrizzazione della corda di 32 punti disposti nel piano a definire il profilo di Paperino.

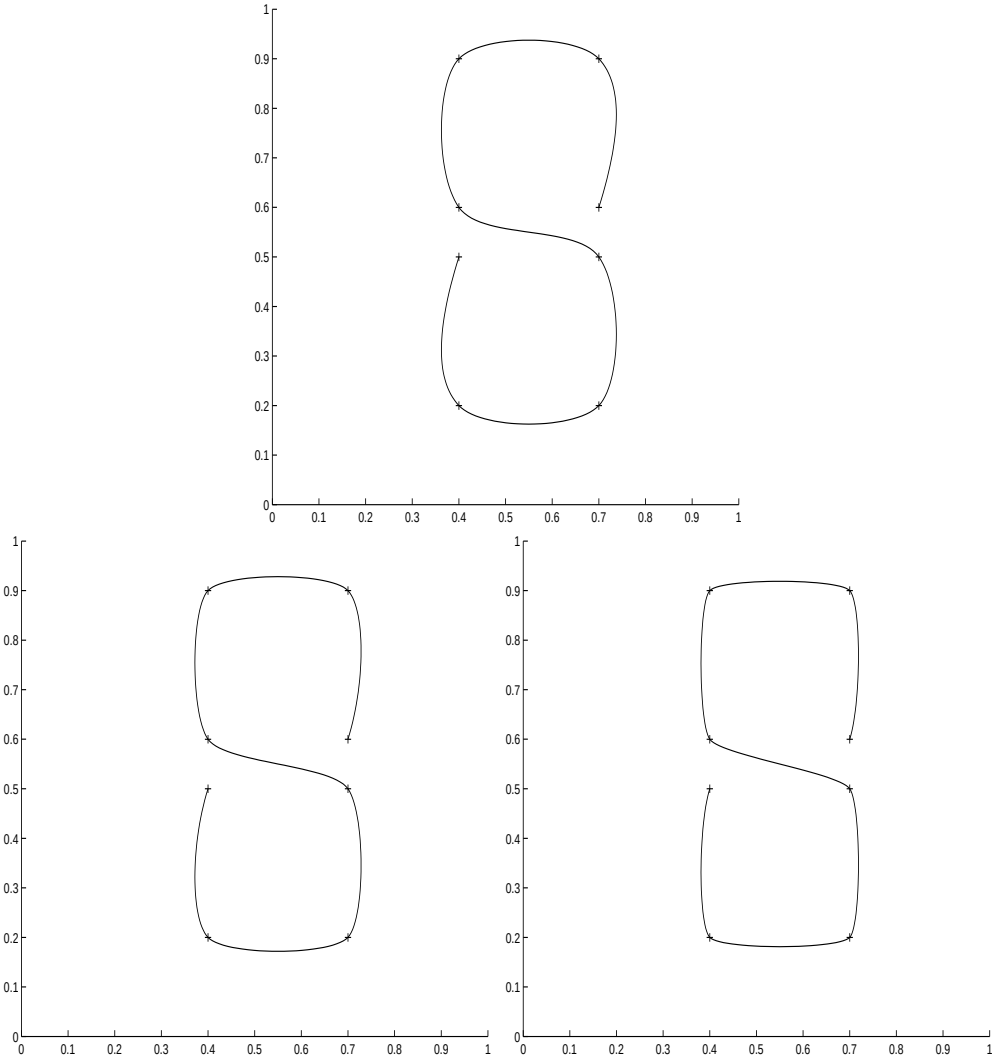


Figura 2.7: Interpolazione con curva polinomiale cubica a tratti C^1 con stima dei vettori derivati nei punti (alto); vettori derivati scalati di 1/2 (basso a sinsitra); vettori derivati scalati di 1/4 (basso a destra).

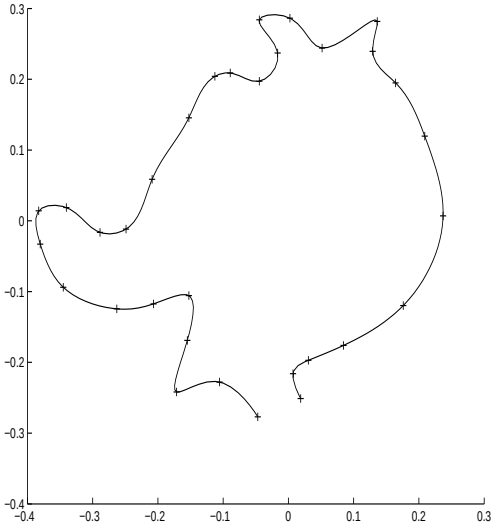


Figura 2.8: Interpolazione con curva spline cubica di 33 punti.

Capitolo 3

Approssimazione ai minimi quadrati

Siano assegnate m osservazioni in corispondenza di m punti distinti, cioè siano assegnati (x_i, y_i) $i = 1, \ldots, m$. Nel caso generale viene individuato uno spazio Φ a dimensione finita $n + 1 \leq m$, di funzioni reali e una loro base di rappresentazione $\phi_0, \phi_1, \ldots, \phi_n$ così che ogni elemento $\phi \in \Phi$ sia definita da una loro combinazione lineare, cioè

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

Il problema della migliore approssimazione nel senso dei minimi quadrati consiste nel determinare una funzione funzione $\phi^* \in \Phi$ (od anche i coefficienti $a_0^*, a_1^*, \ldots, a_n^*$ che la definiscono) tale che

$$\sum_{k=1}^{m} [\phi^*(x_k) - y_k]^2 \leq \sum_{k=1}^{m} [\phi(x_k) - y_k]^2 \quad \forall \phi \in \Phi.$$

Questo equivale a determinare il minimo della seguente funzione in $n + 1$ variabili:

$$g(a_0, a_1, \ldots, a_n) = \sum_{k=1}^{m} [\sum_{i=0}^{n} a_i \phi_i(x_k) - y_k]^2$$

cioè

$$\min_{a_i \in \mathbf{R}} g(a_0, a_1, \ldots, a_n).$$

Un metodo per risolvere questo problema di minimo consite nell'impostare e risolvere il sistema delle **equazioni normali**, che deriva dall'imporre

$$\frac{\partial g(a_0, a_1, \ldots, a_n)}{\partial a_i} = 0 \quad i = 0, \ldots, n$$

o in forma esplicita

$$\frac{\partial}{\partial a_i}\sum_{k=1}^{m}\left[\sum_{i=0}^{n}a_i\phi_i(x_k)-y_k\right]^2=0 \quad i=0,\ldots,n.$$

Infatti, essendo la funzione g quadratica con coefficienti positivi per i termini a_i^2, avrà un minimo. Pima di procedere alla derivazione riscriviamo la funzione g sviluppando il quadrato:

$$g(a_0,a_1,\ldots,a_n)=\sum_{k=1}^{m}y_k^2-2\sum_{i=0}^{n}a_i\sum_{k=1}^{m}y_k\phi_i(x_k)+\sum_{i=0}^{n}\sum_{j=0}^{n}a_ia_j\sum_{k=1}^{m}\phi_i(x_k)\phi_j(x_k)$$

allora

$$\frac{\partial g}{\partial a_i}=-\sum_{k=1}^{m}y_k\phi_i(x_k)+\sum_{j=0}^{n}a_j\left(\sum_{k=1}^{m}\phi_i(x_k)\phi_j(x_k)\right) \quad i=0,\ldots,n$$

da cui il sistema lineare

$$G\mathbf{a}=\mathbf{r}$$

dove

$$G=\{g_{i,j}\}_{i,j=0,\ldots,n} \quad \text{con} \quad g_{i,j}=\sum_{k=1}^{m}\phi_i(x_k)\phi_j(x_k)$$

ed

$$\mathbf{r}=\{r_i\}_{i=0,\ldots,n} \quad \text{con} \quad r_i=\sum_{k=1}^{m}y_k\phi_i(x_k).$$

Osservazione 3.1 *Si noti che algoritmicamente, il sistema* $G\mathbf{a}=\mathbf{r}$ *può essere costruito calcolando la matrice* $H=\{h_{i,j}\}_{i=1,\ldots,m,j=0,\ldots,n}$ *delle funzioni base nei punti, cioè* $h_{i,j}=\phi_j(x_i)$ *e quindi ottenere la matrice* G *ed il vettore* $\mathbf{r}$ *come:*

$$G:=H^TH \qquad \mathbf{r}:=H^T\mathbf{y}.$$

Nel caso generale qui descritto, la scelta dello spazio Φ è solitamente dettato dal fatto che esista sempre una soluzione unica al problema, dalla regolarità desiderata per l'approssimante e dalla particolarità dei dati.

In queste dispense si tratterà solamente il caso dell'approssimazione polinomiale, in cui la matrice G risulta simmetrica e definita positiva, il ché garantisce sempre l'esistenza e l'unicità della soluzione ed una buona regolarità della stessa.

3.1 Forma Monomiale

Teorema 3.1 *Siano dati m punti (x_i, y_i), $i = 1, \ldots, m$ con x_i distinti, allora esiste ed è unico il polinomio $p \in \mathbf{P}_n$, con $n \leq m$, di approssimazione nel senso dei minimi quadrati.*

dim. Si consideri $p(x)$ nella forma monomiale e si noti la forma della matrice H:

$$H = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ 1 & x_3 & x_3^2 & \ldots & x_3^n \\ \ldots & & & & \\ \ldots & & & & \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{pmatrix}$$

si tratta di una matrice di Vandermonde rettangolare, ed essendo i punti x_i distinti, sarà di rango massimo $n+1$; segue che la matrice $G = H^T H$ sarà non singolare e che il sistema lineare delle equazioni normali avrà un'unica soluzione; segue che $p(x)$ di migliore approssimazione esiste ed è unico.

3.2 Forma di Bernstein

Nella sezione precedente si è dimostrata l'esistenza ed unicità del polinomio approssimante nella base monomiale, ma tale polinomio può essere rappresentato in una qualunque altra base. Procediamo quindi alla sua determinazione via soluzione di un sistema lineare, ma a partire dalla base di Bernstein. Siano dati m punti (x_i, y_i), $i = 1, \ldots, m$ con x_i distinti in $[a, b]$ e si vuole determinare $p(x)$ di grado $n \leq m$, nella base di Bernstein tale che risulti il polinomio di approssimazione nel senso dei minimi quadrati. Siano (t_i, y_i), $i = 1, \ldots, m$ i punti traslati e scalati in $[0, 1]$ mediante la 1.6 e sia

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \quad \text{con} \quad t \in [0, 1]$$

la forma polinomiale che vogliamo approssimi i valori assegnati. Costruendo la marice H delle funzioni di Bernstein nei punti si può impostare e risolvere il sistema delle equazioni normali. Il vettore soluzione $\mathbf{b}$ del sistema lineare fornirà sia il polinomio approssimante in $[0, 1]$ che in $[a, b]$.

Si noti che la matrice G delle equazioni normali, nel caso si usi la base di Bernstein nei punti, risulta meglio condizionata che non la matrice G nel caso si usi la forma monomiale (si veda Tab.3.1).

grado n	Bernst.	Potenze
10	2.77E+05	1.01E+14
15	2.73E+08	6.12E+21
20	3.95E+11	1.96E+28
25	9.68E+15	1.95E+36
30	2.18E+16	3.64E+39
35	1.75E+17	1.15E+51

Tabella 3.1: Indici di condizionamento delle matrici G ottenute valutando i polinomi di Bernstein e la base delle potenze in 51 punti equidistanti.

3.3 Forma ortogonale

Come già detto sia nell'introduzione che nel capitolo dell'interpolazione, a seconda del problema, si può determinare una base più idonea delle altre sia dal punto di vista del condizionamento, ma anche che permetta di usare un metodo più efficiente oltre che stabile. Nel caso dell'approssimazione polinomiale sicuramente la base più idonea è quella dei polinomi $\{p_i(x)\}$ $i = 0, \ldots, n$ ortogonali sui punti dati $\{x_k\}$ $k = 1, \ldots, m$, ossia caratterizzati dalla proprietà:

$$\sum_{k=1}^{m} p_i(x_k)p_j(x_k) = 0 \quad i \neq j.$$

Con tale base la matrice G delle equazioni normali risulta diagonale e la soluzione è semplicemente data da:

$$a_i = \frac{\sum_{k=1}^{m} y_k p_i(x_k)}{\sum_{k=1}^{m} [p_i(x_k)]^2} \quad i = 0, \ldots, n$$

Per costruire tale base ortogonale di polinomi di grado n su m punti $\{x_k\}$ assegnati, si può usare lo schema ricorrente proposto da Hoseholder e Stiefel.

3.3.1 Generazione polinomi ortogonali

Vediamo i polinomi ortogonali su un insieme discreto di punti, suggeriti da Householder e Stiefel.

$$\begin{cases} p_0(x) = & 1 \\ p_i(x) = & xp_{i-1}(x) - \alpha_i p_{i-1}(x) - \beta_i p_{i-2}(x) \quad i = 1, 2, \ldots \end{cases}$$

dove

$$\begin{cases} \alpha_i = & \dfrac{\sum_{k=1}^{m} x_k [p_{i-1}(x_k)]^2}{w_{i-1}} \quad i = 1, 2, \ldots \\ \beta_1 = & 0 \\ \beta_i = & \dfrac{w_{i-1}}{w_{i-2}} \quad i = 2, 3, \ldots \end{cases}$$

con

$$w_i = \sum_{k=1}^{m} [p_i(x_k)]^2 \quad i = 0, 1, \ldots .$$

Esempio
Siano assegnati in $[0, 1]$ i cinque punti $x_i = (i-1)h$ $i = 1, \ldots, 5$ con $h = 1/4$. Allora i polinomi ortogonali su tali punti che generano $\mathbf{P}_1$ sono:

$$\begin{cases} p_0(x) = & 1 \\ p_1(x) = & x p_0(x) - \alpha_1 p_0(x) \end{cases}$$

dove

$$\alpha_1 = \frac{\sum_{k=1}^{5} x_k [p_0(x_k)]^2}{w_0} = \frac{0 + 0.25 + 0.5 + 0.75 + 1}{5} = 0.5$$

essendo

$$w_0 = \sum_{k=1}^{5} [p_0(x_k)]^2 = 5$$

da cui

$$p_1(x) = x - 0.5$$

Verifichiamo che

$$\sum_{k=1}^{5} p_0(x_k) p_1(x_k) = 0;$$

infatti

$$1 \cdot (-0.5) + 1 \cdot (-0.25) + 1 \cdot 0 + 1 \cdot 0.25 + 1 \cdot 0.5 = 0.$$

3.4 Esempi numerici

Le Fig.3.1 e 3.2 mostrano il risultato di un'approssimazione ai minimi quadrati di 51 punti equidistanti della funzione di Runge rispettivamente con polinomi di grado 10, 15, 30 e 35 nella base monomiale. I differenti indici di condizionamento della Tab.3.1 indicano una elaborazione numerica potenzialmente meno accurata nel caso monomiale che non nella base di

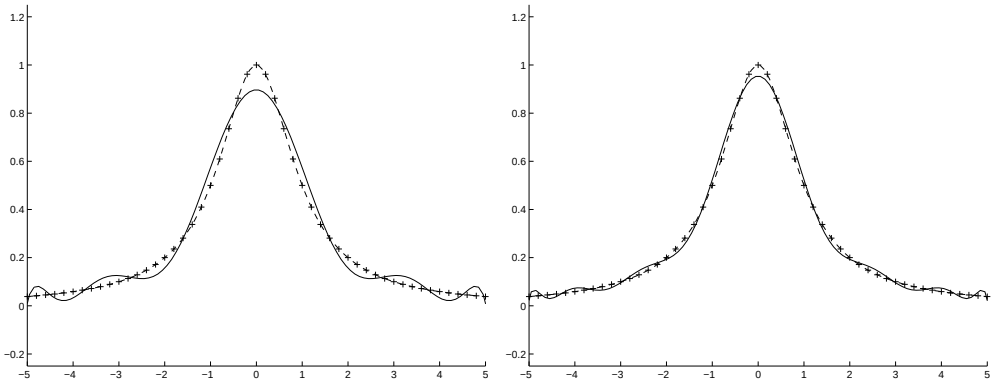


Figura 3.1: Approssimazione polinomiale nella base delle potenze della funzione di Runge di 51 punti equidistanti; a sinistra grado 10, a destra 15.

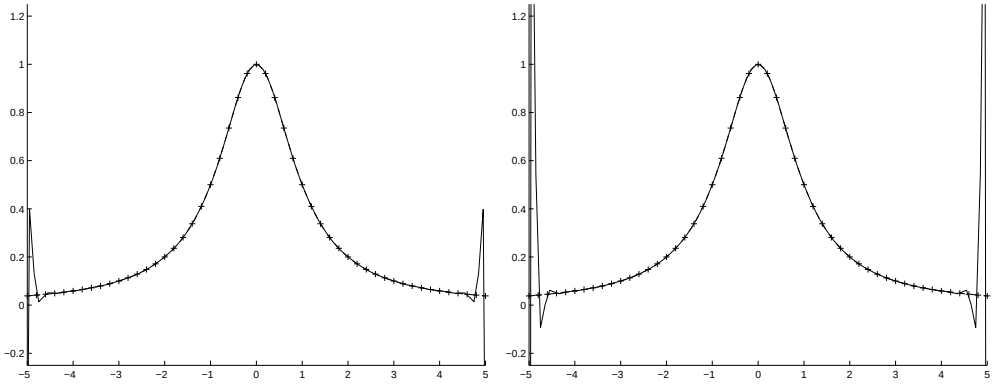


Figura 3.2: Approssimazione polinomiale nella base delle potenze della funzione di Runge di 51 punti equidistanti; a sinistra grado 30, a destra 35.

bernstein; in realtà questa differenza non risulta graficamente apprezzabile né usando Bernstein né usando i polinomi ortogonali. Determinato il polinomio $p^*(x)$ di grado n di migliore approssimazione, chiameremo residuo la quantità

$$d_n = \sum_{i=0}^{n} [p^*(x_k) - y_k]^2.$$

Si nota che all'aumentare di n ($n < m$), sarà $d_{n+1} < d_n$, e per $n = m - 1$, $d_{m-1} \equiv 0$: questo, come è ben visibile dalle Fig.3.2, non assicura che $p^*(x)$ converga alla funzione test né analiticamente né numericamente. Anche in

questo caso i polinomi risultano poco flessibili e numericamente instabili per gradi alti.

Per l'approssimazione nel senso dei minimi quadrati è possibile utilizzare polinomi a tratti o spline tipicamente di grado basso, con il vantaggio di avere delle ottime funzioni di approssimazione senza i problemi numerici e di flessibilità tipici dei polinomi di grado alto.

3.5 Retta di regressione lineare

Nelle applicazioni pratiche, noto un set di dati sperimentali acquisiti da un fenomeno di andamento lineare, è molto utilizzata la tecnica nota come **retta di regressione lineare** che consiste nel determinare il polinomio lineare di approsimazione nel senso dei minimi quadrati dei dati assegnati. La determinazione di tale polinomio viene effettuata mediante la soluzione del sistema lineare delle equazioni normali, che in questo caso risulterà di dimensioni 2×2 e la cui soluzione può essere data in forma esplicita risolvendo analiticamente il sistema suddetto. Sarà:

$$G = \begin{pmatrix} \sum_{k=1}^{m} 1 & \sum_{k=1}^{m} x_k \\ \sum_{k=1}^{m} x_k & \sum_{k=1}^{m} x_k^2 \end{pmatrix}$$

e definendo

$$\bar{x} = \frac{1}{m} \sum_{k=1}^{m} x_k \quad \bar{y} = \frac{1}{m} \sum_{k=1}^{m} y_k$$

il baricentro dei punti dati, il sistema può essere scritto in modo semplificato come:

$$\begin{pmatrix} m & m\bar{x} \\ m\bar{x} & \sum_{k=1}^{m} x_k^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} m\bar{y} \\ \sum_{k=1}^{m} x_k y_k \end{pmatrix}$$

da cui con semplici passaggi si ricava

$$a_0 = \bar{y} - a_1 \bar{x} \qquad a_1 = \frac{\sum_{k=1}^{m} x_k y_k - m\bar{x}\bar{y}}{\sum_{k=1}^{m} x_k^2 - m\bar{x}^2} \tag{3.1}$$

In modo ancor più operativo, l'espressione del polinomio di approssimazione può essere scritta come

$$p(x) = a_0 + a_1 x = \bar{y} + a_1 (x - \bar{x})$$

che comporta solo la valutazione di a_1 dalla seconda delle 3.1 e geometricamente dice che la retta di regressione lineare passa per il baricentro

del set di dati ed ha pendenza a_1. In questo caso diremo che la retta di regressione lineare è rappresentata nella base con centro $\bar{x}$ data dalle funzioni $\{1, (x - \bar{x})\}$.

Se richiesto, la retta di regressione lineare si può esprimere semplicemente nella base lineare di Bernstein in $[0, 1]$ come

$$p(t) = a_0(B_{0,1}(t) + B_{1,1}(t)) + a_1 B_{1,1}(t) =$$

$$= a_0 B_{0,1}(t) + (a_0 + a_1) B_{1,1}(t).$$

3.6 Un'applicazione: curve di approssimazione

Possiamo risolvere problemi di approssimazione ai minimi quadrati con funzioni vettoriali o curve polinomiali o polonomiali a tratti, per esempio nel piano.
Siano assegnati m punti nel piano euclideo $Q_i = (x_i, y_i)_{i=1,\ldots,m}$. Il nostro obiettivo è determinare una curva polinomiale nella base di Bersntein:

$$\underline{C}(t) = \begin{pmatrix} C_1(t) \\ C_2(t) \end{pmatrix} = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

che rende minima la seguente espressione:

$$\sum_{i=1}^{m} \|Q_i - \underline{C}(t_i)\|_2^2 = \sum_{i=1}^{m} ((x_i - C_1(t_i))^2 + (y_i - C_2(t_i))^2)$$

Come nel caso dell'interpolazione, il problema della determinazione di una curva polinomiale che approssimi i punti Q_i, può essere scomposto in due sottoproblemi, cioè nella ricerca di due funzioni polinomiali $C_1(t)$ e $C_2(t)$ che approssimino nel senso dei minimi quadrati rispettivamente i punti (t_i, x_i) e (t_i, y_i) per $i = 1, \ldots, m$. Risulta sufficiente trovare le due funzioni componenti della curva applicando due volte il metodo dei minimi quadrati.
Il primo passo da fare è, ancora una volta, la determinazione dei parametri t_i a cui far corrispondere i punti Q_i. Metodi utilizzabili sono, come già visto nel caso dell'interpolazione con curve, la parametrizzazione uniforme e la parametrizzazione della corda.

Capitolo 4

Approssimazione di forma

In questa sezione si vuole mostrare come i polinomi di Bernstein, inizialmente introdotti da Bernstein stesso per l'approssimazione uniforme su un intervallo, godono anche della proprietà di essere approssimanti di forma o in modo più tecnico di essere approssimanti variation diminishing (VD).

4.1 Approssimazione uniforme

I polinomi di Bernstein furono originariamente introdotti nell'approssimazione di funzioni continue $f(x)$ su un intervallo $[a, b]$. Dati $n + 1$ punti di $f(x)$ equidistanti in $[a, b]$, l'approssimazione di Bernstein di grado n, $B_n[f(x)]$ di $f(x)$ è definita come

$$B_n[f(x)] = \sum_{i=0}^{n} f(\xi_i) B_{i,n}(x)$$

con $\xi_i = f(a + i(b - a)/n)$ e $B_{i,n}(x)$ i polinomi di Bernstein di grado n in $[a, b]$. Questa approssimazione può essere costruita soddisfando un qualunque errore δ usando polinomi di grado sufficientemente elevato, cioè esiste un valore n_δ tale che:

$$|B_n[f(x)] - f(x)| < \delta$$

$\forall x \in [a, b]$ quando $n > n_\delta$. Così si dice che l'approssimazione di Bernstein converge uniformemente ad $f(x)$:

$$\lim_{n\to\infty} B_n[f(x)] = f(x) \quad x \in [a, b].$$

Questa proprietà della base di Bernstein permette di dimostrare il teorema di Weierstrass che una funzione continua su un intervallo può essere approssimata con una data tolleranza da un polinomio di opportuno grado n.

4.2 Approssimazione VD

Definizione 4.1 Variazione di segno di un vettore
Sia $\mathbf{c} = (c_0, c_1, \ldots, c_n)$ *un vettore ad elementi reali; si definisce numero di variazioni di segno di* $\mathbf{c}$ *in* **senso forte**, *e lo si indica con*

$$V^-[\mathbf{c}]$$

il numero di variazioni di segno nella sequenza $c_0, c_1, \ldots, c_n$ *dove si ignorano gli zeri.*

Esempio 4.1 *Sia* $\mathbf{c} = (-1, 3.2, 1.5, -3.7, 4)$ *allora* $V^-[\mathbf{c}] = 3$.

Definizione 4.2 Variazione di segno di una funzione
Sia f *una funzione reale in* $[a, b]$*; si dirà che:*

$$V^-[f(z)] = sup_{n+1}\{V^-[f(z_0), \ldots, f(z_n)] \text{ con } z_0 < z_1 < \ldots < z_n\}.$$

Sia data una funzione $f(x)$ almeno continua nell'intervallo $[a, b]$. Nel caso generale viene individuato uno spazio Φ a dimensione finita $n+1$, di funzioni reali e una loro base di rappresentazione $\phi_0, \phi_1, \ldots, \phi_n$ così che ogni elemento $\phi \in \Phi$ è definito da una loro combinazione lineare, cioè

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

Il problema dell'approssimazione di forma o **variation diminishing** consiste nel determinare una funzione $\phi^* \in \Phi$ tale che

1. se $f(x)$ è lineare, allora la ϕ^* è lineare e coincide con la $f(x)$;
2. $$V^-[\phi^*(x)] \leq V^-[f(x)] \quad x \in [a, b]$$
   che significa che la variazione in segno in senso forte della ϕ^* è minore o uguale alla variazione in segno in senso forte della $f(x)$.

Se queste due proprietà vengono soddisfatte si ha come conseguenza che

$$V^-[\phi^*(x) - (a_0 + a_1 x)] \leq V^-[f(x) - (a_0 + a_1 x)] \quad x \in [a, b] \qquad (4.1)$$

che dice che la $\phi^*(x)$ non può essere più oscillante della $f(x)$ che approssima, in quanto il numero di intersezioni della $\phi^*(x)$ con una retta è sempre minore o uguale al numero di intersezioni della $f(x)$ con quella retta.

Teorema 4.1 *Sia $n \geq 1$, allora $\forall x \in [a,b]$ si ha:*

$$\sum_{i=0}^{n} \xi_i B_{i,n}(x) = x$$

con $\xi_i = \frac{a+i(b-a)}{n}$.

Teorema 4.2 *Sia $p : [a,b] \to \mathbf{R}$ un polinomio nella base di Bernstein e si consideri come approssimazione di una $f(x) : [a,b] \to \mathbf{R}$ almeno continua, la $B_n[f(x)]$ definita nella sezione 4.1; allora $B_n[f(x)]$ è approssimante VD della $f(x)$.*

dim.
Verifichiamo le due proprietà caratterizzanti:
1)$B_n[f(x)]$ deve preservare le funzioni lineari:

$$B_n[a_0 + a_1 x] = \sum_{i=0}^{n} (a_0 + a_1 \xi_i) B_{i,n}(x) =$$

$$= a_0 \sum_{i=0}^{n} B_{i,n}(x) + a_1 \sum_{i=0}^{n} \xi_i B_{i,n}(x) = a_0 + a_1 x$$

dove nella prima sommatoria si è applicata la proprietà di partizione dell'unità mentre nella seconda il Teorema 4.1.
2) La $B_n[f(x)]$ gode della proprietà di variazione in segno dei coefficienti (vedi proprietà sui polinomi di Bernstein), cioè:

$$V^-[B_n[f(x)]] \leq V^-[f(\xi_0), \ldots, f(\xi_n)]$$

poichè gli ξ_i sono in ordine crescente, individueranno sicuramente una $n+1$-pla che soddisfa la definizione 4.2 di $V^-[f(x)]$ per cui sarà

$$V^-[f(\xi_0), \ldots, f(\xi_n)] \leq V^-[f(x)]$$

verificando così la seconda proprietà.

La Fig.4.1 mostra una funzione f lineare a tratti e alcuni polinomi di approssimazione VD di grado crescente. Dai grafici si vede come al crescere del grado i polinomi approssimano sempre meglio la funzione f; vale infatti il teorema di Weierstrass che la $B_n[f(x)]$ verifica. La stessa figura mostra anche come ogni polinomio approssima in forma, secondo la proprietà conseguente 4.1, la funzione data.

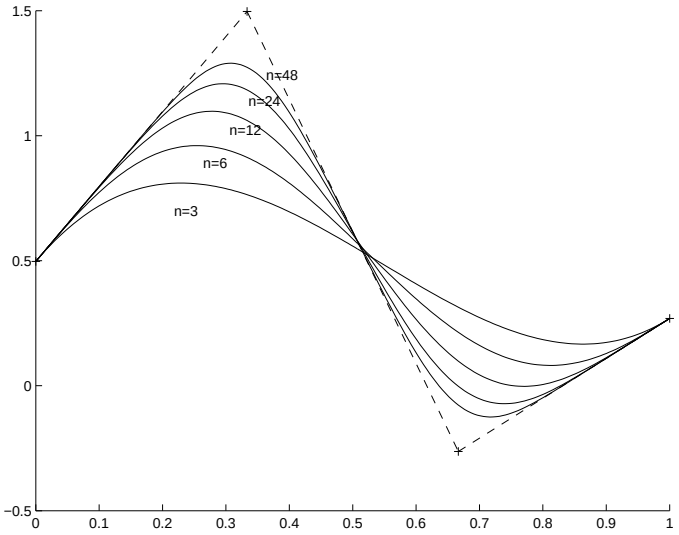


Figura 4.1: Approssimazione polinomiale VD di una funzione lineare a tratti.

4.3 Significato geometrico dei coefficienti

Dato un polinomio di grado n nella base di Bernstein, i coeffcienti b_i, tipicamente scalari, assumono un importante significato geometrico derivante dall'approssimazione VD vista. Infatti, tali coefficienti possono essere interpretati come i valori di una funzione continua in corrispondenza delle ascisse ξ_i $i = 0, \ldots, n$ che il polinomio approssima nel senso VD. Per analogia con il caso di funzioni vettoriali (curve di Bézier) la funzione continua viene scelta come la lineare a tratti di vertici i punti (ξ_i, b_i).
È noto che una funzione scalare può essere sempre rappresentata come una funzione vettoriale in modo banale. Questo si può applicare ad un polinomio nella base di Bernstein e rappresentarlo come una funzione vettoriale, ed in particolare per il Teorema 4.1 come una curva di Bézier.

Sia $p(x) = \sum_{i=0}^{n} b_i B_{i,n}(x)$ con $x \in [a, b]$, un polinomio scalare nella base di Bernstein. Allora questo può essere rappresentato in forma vettoriale come:

$$\mathbf{C}(t) = \begin{pmatrix} t \\ \sum_{i=0}^{n} b_i B_{i,n}(t) \end{pmatrix}$$

e per il teorema 4.1

$$\mathbf{C}(t) = \begin{pmatrix} \sum_{i=0}^{n} \xi_i B_{i,n}(t) \\ \sum_{i=0}^{n} b_i B_{i,n}(t) \end{pmatrix}.$$

Questa rappresentazione giustifica, che nel caso scalare, la funzione che $p(x)$ approssima nel senso VD sia assunta come la lineare a tratti di vertici i punti (ξ_i, b_i), $i = 0, \ldots, n$.

Osservazione 4.1 *Dalla Fig.4.1 si può notare che la convergenza del polinomio approssimante VD è molto lenta, ossia si ottiene per polinomi di grado molto elevato. Anche in questo caso si può ricorrere all'uso di polinomi a tratti o spline tipicamente di grado basso (cubiche) e ottenere un'approsimazione VD che converge alla* $f(x) \in C_{[a,b]}$ *all'aumentare dei tratti o intervalli in cui si partiziona* $[a, b]$.

Elenco delle figure

Bibliografia

[BBCM92] R. Bevilacqua, D. Bini, M. Capovani, O. Menchi. *Metodi Numerici*. Zanichelli, 1992.

[Bez70] P. Bézier. Numerical Control; Mathematics and Applications. *Johm Wiley & Sons*, 1970.

[FaPr87] I. D. Faux, M. J. Pratt. *Computational Geometry for Design and Manufacture*. John Wiley & Sons, 1987.

[FaRa87] R. T.Farouki, V. T.Ragian. On the numerical condition of polynomials in Bernstein form; *Computer Aided Geometric Design*, 4 (1987), 191-216.

[FaRa88] R. T.Farouki, V. T.Ragian. Algorithms for polynomials in Bernstein form; *Computer Aided Geometric Design*, 5 (1988), 1-26.