Dispensa di

METODI NUMERICI PER IL CALCOLO

A.A. 2023/24

Giulio Casciola
(e-mail: giulio.casciola@unibo.it)

Ufficio F2

Dipartimento di Matematica,
P.zza di Porta S.Donato 5, Bologna

(rivista e corretta settembre 2023)

Indice

Parte I

Numeri Finiti e Aritmetica Floating Point

Capitolo 1

L'aritmetica computazionale

La moderna metodologia per la creazione di strumenti informatici per l'esame dei fenomeni reali, di supporto alle attività produttive, che possono spaziare dalla progettazione meccanica ai videogiochi di simulazione, comprende una fase di modellazione mediante astrazioni di tipo matematico e quindi la loro implementazione su computer mediante linguaggi di programmazione. In questo processo si introducono diversi livelli di approssimazione:

- *dal modello reale a quello matematico*: in tutte le discipline scientifiche, ogni volta che si introduce un modello matematico per descrivere un oggetto o fenomeno reale, si introduce un'approssimazione necessaria a limitare il numero di fattori da considerare altrimenti ingestibili; per esempio, nell'analisi agli elementi finiti per lo studip aerodinamico della carrozzeria di un'automobile, le superfici lisce e continue che la definiscono vengono approssimate da mesh poligonali, ossia sa una serie di piccole facce piane unite tra loro (vedi Fig. 1.1))

- *dal modello matematico alla sua rappresentazione computazionale*: i modelli matematici descritti teoricamente non sempre sono rappresentabili esattamente e i problemi relativi risolubili in modo esplicito ed esatto mediante un computer; scopo del *Calcolo Numerico* è proprio studiare gli effetti delle approssimazioni introdotte in questi passaggi. Nella presente trattazione porremo particolare attenzione a modelli e problemi che derivano dal settore della CAGD (Computer Aided Geometric Design).

La precisione con cui un modello matematico può essere implementato dipende dalle sue particolarità: più precisamente, essendo il computer una macchina discreta, esso potrà rappresentare esattamente modelli basati su insiemi discreti di simboli.
Quando un modello matematico è invece definito nel continuo, come accade

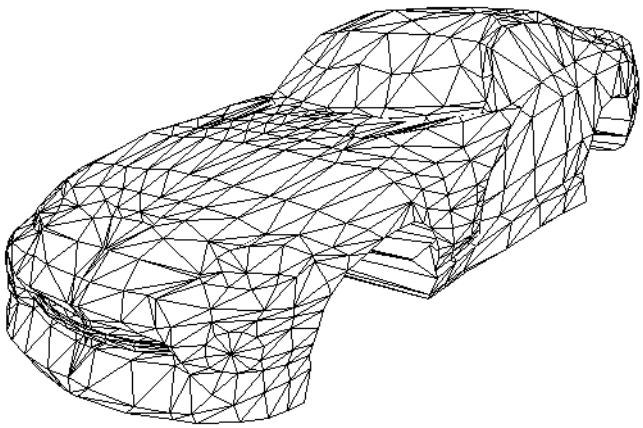

Figura 1.1: Modello poligonale

nella maggioranza dei casi, questo sottende una precisione infinita, mentre il computer utilizza necessariamente una rappresentazione discreta, con una precisione quindi limitata.
Si presenta poi la necessità di utilizzare tale modello per risolvere dei problemi e lavorando con un computer questo vuol dire risolvere algoritmicamente il problema matematico, cioè ottenere, mediante un numero finito di operazioni aritmetiche e/o logiche, un'informazione anche parziale della soluzione (spesso approssimata), ma adeguata alle richieste.
Ad esempio, per approssimare la curva di intersezione tra due superfici a forma libera viene calcolata una successione di punti che appartegono alla soluzione con una certa approssimazione.
Si parla così di *risoluzione numerica* del problema, intendendo che la soluzione ottenuta è calcolata a partire da un insieme finito di numeri (rappresentati con un numero finito di cifre), attraverso un numero finito di operazioni di macchina ed è espressa ancora mediante un insieme finito di numeri.

In questo primo capitolo si esaminano le differenze tra il modello teorico continuo dei numeri reali (insieme $\mathbb{R}$) e la sua approssimazione discreta all'interno dei computer, gli errori di approssimazione che ne derivano e le teorie per valutarne l'entità.

1.1 Rappresentazione dei numeri reali

Alla radice dell'impossibiltità di implementare in modo esatto i modelli matematici basati su un insieme continuo di simboli, sta la rappresentazione dei numeri reali utilizzata nei computer.
Questi, essendo macchine "finite", devono operare su numeri rappresentati mediante una sequenza finita di cifre; ad esempio, numeri come π o $\sqrt{2}$, la cui rappresentazione richiede infinite cifre, non possono che essere rappresentati se non in modo approssimato, commettendo cioè un errore. Inoltre, poiché l'insieme dei numeri rappresentabili è finito e quindi limitato, non potranno essere rappresentati numeri arbitrariamente piccoli o arbitrariamente grandi.
Un ulteriore problema nasce dalla base di rappresentazione; generalmente l'inserimento di valori numerici attraverso una interfaccia o shell avviene utilizzando la base decimale, usata naturalmente dall'uomo, mentre la rappresentazione interna al computer è binaria. Questo porta all'introduzione di approssimazioni in modo subdolo; ad esempio inserendo il numero decimale 0.1 questo viene memorizzato internamente in modo approssimato perché la sua rappresentazione binaria $0.000110011\ldots$ è periodica.

Teorema 1.1 (Rappresentazione in base) *Sia α un numero reale non nullo e $\beta \geq 2$ un numero intero; allora esistono e sono unici il numero intero p e la successione $\{\alpha_i\}_{i=1,2,\ldots}$ di numeri interi, $0 \leq \alpha_i < \beta, \alpha_1 \neq 0$, non definitivamente uguali a $\beta - 1$, tali che:*

$$\alpha = \pm(\sum_{i=1}^{\infty} \alpha_i \beta^{-i})\beta^p. \tag{1.1}$$

La (1.1) viene detta *rappresentazione in base β* del numero reale non nullo α.
Le quantità che compaiono vengono dette:

β	base
p	esponente
α_i	cifre della rappresentazione
$\sum_{i=1}^{\infty} \alpha_i \beta^{-i}$	mantissa

Se esiste un indice k tale che $\alpha_k \neq 0$ e $\alpha_i = 0$ per $i > k$, la rappresentazione in base β si dice *finita di lunghezza k*.

Definizione 1.1 (Numeri Finiti) *Siano $\beta, t, \lambda, \omega$, numeri interi tali che $\beta \geq 2$ e $t \geq 1$. Si definisce insieme dei numeri finiti, con rappresentazione normalizzata, in base β con t cifre significative, l'insieme:*

$$\mathbb{F}(\beta, t, \lambda, \omega) = \{0\} \cup \{\alpha \in \mathbb{R} : \alpha = \pm(\sum_{i=1}^{t} \alpha_i \beta^{-i})\beta^p,$$
$$con\ 0 \leq \alpha_i < \beta,\ per\ i = 1, 2, \ldots, t,\ \alpha_1 \neq 0,\ \lambda \leq p \leq \omega\}.$$

Gli n bit (o posizioni) consecutivi dedicati per la memorizzazione di un numero finito vengono suddivisi tra le t cifre della mantissa ed un certo numero di bit $(\omega - \lambda + 1)$ per l'esponente p, più un bit per il segno del numero. Alcune tipiche rappresentazioni sono:

$\mathbb{F}(2, 24, -127, 128)$ precisione singola: 32 bit
$\mathbb{F}(2, 53, -1023, 1024)$ precisione doppia: 64 bit

In precisione singola vengono destinati 24 bit alla mantissa (in realtà solo 23 [1]) e 8 all'esponente ($2^8 = 256 = \omega - \lambda + 1$, con $\lambda = -127$ e $\omega = 128$ [2]), mentre in precisione doppia le cifre della mantissa sono 53 (memorizzati 52 bit) e dell'esponente 11 ($2^{11} = 2048 = \omega - \lambda + 1$, con $\lambda = -1023$ e $\omega = 1024$ [3]).
Si noti che la normalizzazione della mantissa permette di sfruttare al meglio le

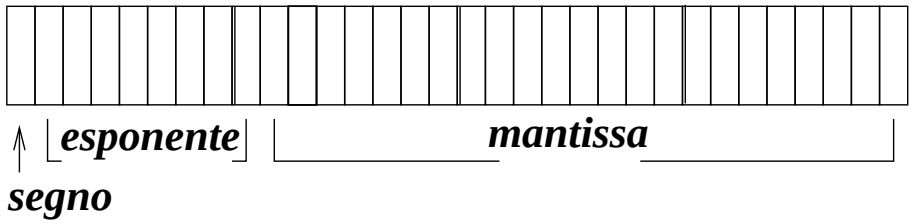

Figura 1.2: Rappresentazione binaria dei numeri finiti

cifre disponibili, evitando di rappresentare esplicitamente gli 0 non significativi. Ad esempio 0.000124 viene rappresentato come 0.124×10^{-3}.
La virgola quindi non ha una posizione fissa tra la parte intera e quella non, ma varia; per questo motivo tale rappresentazione è detta *floating-point* o a virgola mobile, in alternativa a *fixed-point* o a virgola fissa.
Per quel che riguarda l'esponente si osservi che questi viene memorizzato per traslazione (exponent biased) e che la costante di traslazione (bias) è $-\lambda$.

[1] Come vedremo parlando di standard ANSI/IEEE per la rappresentazione binaria, essendo sempre $\alpha_1 = 1$, la prima cifra di ogni numero può essere sottointesa senza mai essere fisicamente memorizzata.
[2] Come vedremo in ANSI/IEEE per la precisione singola sarà $\lambda = -126$ e $\omega = 127$.
[3] Come vedremo in ANSI/IEEE per la precisione doppia sarà $\lambda = -1022$ e $\omega = 1023$.

Le grandezze fondamentali che defiscono l'insieme dei numeri finiti, oltre alla base di rappresentazione β, sono:

- λ, ω che definiscono l'ordine di grandezza dei numeri rappresentabili e in genere è $\omega \simeq -\lambda$;

- il numero t di cifre della mantissa che fissa la precisione di rappresentazione di ogni numero; infatti la retta reale viene suddivisa in intervalli $[\beta^p,\ \beta^{p+1}]$ di ampiezza crescente esponenzialmente con il valore p, e all'interno di ciascuno di questi intervalli vengono rappresentati lo stesso numero $(\beta - 1)\beta^{t-1} - 1$ di valori. Si ha quindi una suddivisione molto densa per valori vicini allo 0 e più rada per valori grandi in valore assoluto.
  Tutti i numeri reali interni ad un sottointervallo di estremi due numeri finiti consecutivi vengono approssimati da uno dei due estremi, quindi maggiore è il numero t di cifre della mantissa, minore sarà l'ampiezza dei sotto-intervalli e quindi migliore l'approssimazione introdotta.

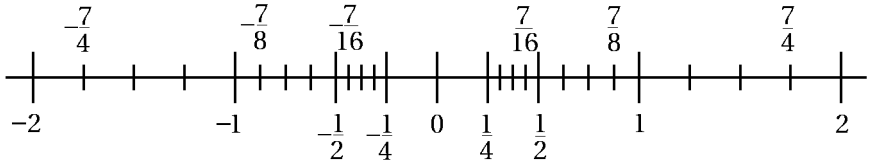

Figura 1.3: Discretizzazione della retta reale. in $\mathbb{F}(2, 3, -1, 2)$

Esempio 1.1 In figura 1.3 sono rappresentati con delle tacche alcuni numeri dell'insieme $\mathbb{F}(2, 3, -1, 2)$; poiché $\beta = 2$ e $t = 3$, questo insieme contiene i seguenti numeri:

$$\{0.100 \times 2^p,\ 0.101 \times 2^p,\ 0.110 \times 2^p,\ 0.111 \times 2^p\} \quad p = -1, 0, 1, 2$$

dove $0.100,\ 0.101,\ 0.110,\ 0.111$ sono tutte le possibili mantisse e p il valore dell'esponente.
In figura 1.3 sono rappresentati gli intervalli per $p = -1 :\ [\frac{1}{4},\ \frac{1}{2}],\ p = 0 :\ [\frac{1}{2},\ 1]$ e $p = 1 :\ [1,\ 2]$.
Risulta evidente l'aumento dell'ampiezza degli intervalli definiti dal valore p, in ognuno dei quali vengono localizzati i 4 valori della mantissa; ne risulta di conseguenza una diradazione delle suddivisioni verso gli estremi sinistro e destro della retta reale.

Formalmente: dato un numero reale non nullo $\alpha = \pm(.\alpha_1\alpha_2\ldots)\times\beta^p$ tale che $\lambda \le p \le \omega$, $\alpha_i = 0$ per $i > t$, cioè rappresentabile esattamente con un numero finito t di cifre, allora $\alpha \in \mathbb{F}(\beta, t, \lambda, \omega)$.
Altrimenti $\alpha \notin \mathbb{F}(\beta, t, \lambda, \omega)$ e quindi bisogna associargli un numero finito $\tilde{\alpha}$ che indicheremo anche con $fl(\alpha)$ (si pronuncia *float di* α).
Supposto per semplicità α positivo (analogamente per α negativo) e la base β pari (ipotesi sempre verificata in pratica) si hanno i seguenti casi:

- $p \notin [\lambda, \omega]$: viene segnalata una condizione d'errore

$$\begin{array}{ll} p < \lambda & underflow \\ p > \omega & overflow \end{array}$$

- $p \in [\lambda, \omega]$, ma $\alpha \notin \mathbb{F}(\beta, t, \lambda, \omega)$ perché le sue cifre α_i con $i > t$ non sono tutte nulle: viene assegnato un numero finito $fl(\alpha)$ seguendo due possibili criteri:

  Troncamento di α alla t-esima cifra

$$fl_T(\alpha) = \pm(\sum_{i=1}^{t} \alpha_i\beta^{-i})\beta^p \tag{1.2}$$

  Arrotondamento di α alla t-esima cifra

$$fl_A(\alpha) = \pm fl_T((\sum_{i=1}^{t+1} \alpha_i\beta^{-i} + \frac{\beta}{2}\beta^{-t-1})\beta^p) \tag{1.3}$$

Quindi

se $\alpha_{t+1} < \frac{\beta}{2}$, allora si ha $fl_A(\alpha) = fl_T(\alpha)$
se invece $\alpha_{t+1} \ge \frac{\beta}{2}$, allora si ha $fl_A(\alpha) = fl_T(\alpha) + \beta^{p-t}$

Osservazione 1.1 *Siano x ed y due numeri finiti consecutivi positivi tali che*

$$x \le \alpha < y$$

allora

$$x = (\sum_{i=1}^{t} \alpha_i\beta^{-i})\beta^p \qquad y = (\sum_{i=1}^{t} \alpha_i\beta^{-i} + \beta^{-t})\beta^p$$

e risulta

$$fl_T(\alpha) = x \qquad fl_A(\alpha) = \begin{cases} x & se\ \alpha < \frac{x+y}{2} \\ y & se\ \alpha \ge \frac{x+y}{2} \end{cases} \tag{1.4}$$

Ogni numero finito rappresenta se stesso e un intero intervallo di numeri reali.

Esempio 1.2 Definito l'insieme dei numeri finiti $\mathbb{F}(10, 5, -50, 49)$ resta definito il numero di posizioni (bit nel caso di base 2) necessarie per rappresentare in memoria un numero finito dell'insieme. Nel caso specifico saranno 1 per il segno (0 se positivo e $\beta - 1$ se negativo), 2 per l'esponente (si usa la tecnica di memorizzazione per traslazione, cioè nei due campi per l'esponente si memorizzano i valori da 00 a 99 intendendo gli esponenti da -50 a 49) e 5 posizioni per la mantissa (si memorizza a partire da sinistra). Vediamo qualche esempio numerico:

$$\alpha = 0.1039 \times 10^{-6} \qquad 04410390$$

dove 0 indica che il numero è positivo, 44 rappresenta l'esponente -06, quindi la mantissa 10390 arrotondata alla quinta cifra.

$$\begin{array}{ll} \alpha = 0.05302 & 04953020 \\ \alpha = -237.141 & 95323714 \\ \alpha = -0.00321665 & 94832167 \end{array}$$

1.2 Errori di rappresentazione

Risulta evidente che *l'insieme dei numeri finiti rappresenta solo un ristretto sottoinsieme di quello dei numeri reali*. La maggior parte dei valori $\alpha \in \mathbb{R}$ risulta $\notin \mathbb{F}(\beta, t, \lambda, \omega)$, quindi tali valori possono essere solamente approssimati mediante un $\tilde{\alpha} \in \mathbb{F}(\beta, t, \lambda, \omega)$, commettendo un certo errore di rappresentazione. Per valutarne l'entità si definiscono le seguenti quantità:

$$E_{abs} = |\tilde{\alpha} - \alpha| \qquad \text{errore assoluto}$$

$$E_{rel} = \left|\frac{\tilde{\alpha} - \alpha}{\alpha}\right| \text{ se } \alpha \neq 0 \qquad \text{errore relativo}$$

La rappresentazione discreta della retta reale descritta in precedenza (vedi Figura 1.3) è tale che fornisce un errore relativo di rappresentazione massimo costante per ogni α, mentre quello assoluto, di conseguenza, aumenta proporzionalmente al valore di α.

Nel calcolo scientifico, dove le risposte ai problemi possono variare grandemente in valore, solitamente si usa l'errore relativo in quanto è "scaling invariant", infatti per $\alpha \to s\alpha$ e $\tilde{\alpha} \to s\tilde{\alpha}$, E_{rel} resta uguale.

Teorema 1.2 *Per ogni $\alpha \in \mathbb{R}$ risulta:*

$$\begin{array}{rcl} |fl_T(\alpha) - \alpha| & < & \beta^{p-t} \\ |fl_A(\alpha) - \alpha| & \leq & \frac{1}{2}\beta^{p-t} \end{array}$$

Fass

β = base
t = cifre mantisse
λ = exp min
ω = exp max

$p = \omega - \lambda + 1$ $\qquad$ $p - t = \omega - \lambda - t + 1$

dove il segno di uguaglianza vale solo se $\alpha_{t+1} = \beta/2$ *e* $\alpha_{t+i} = 0$ $i \geq 2$.

Dim.
Siano x ed y i due numeri finiti consecutivi tali che $x \leq \alpha < y$; riprendendo l'osservazione 1.1 per la 1.4 sarà $\alpha - f\ell_T(\alpha) < y - x = \beta^{p-t}$; ancora sarà

$$|f\ell_A(\alpha) - \alpha| \leq \frac{y-x}{2} = \frac{1}{2}\beta^{p-t}$$

e l'uguaglianza vale solo se $\alpha = \dfrac{x+y}{2}$ cioè $\alpha_{t+1} = \beta/2$ e $\alpha_{t+i} = 0$ $i \geq 2$. $\odot$

Definizione 1.2 *Dato l'insieme dei numeri finiti* $\mathbb{F}(\beta, t, \lambda, \omega)$*, si dice* unità di arrotondamento *e la si indica con* u*, la quantità:*

$$u = \begin{cases} \beta^{1-t} & \textit{per Troncamento} \\ \frac{1}{2}\beta^{1-t} & \textit{per Arrotondamento} \end{cases}$$

Teorema 1.3 *Per ogni* $\alpha \in \mathbb{R}$ *e* $\alpha \neq 0$ *vale* [4]*:*

$$\left|\frac{f\ell(\alpha) - \alpha}{\alpha}\right| < u$$

Dim.
Caso di troncamento.

$$|\alpha| = (\alpha_1\beta^{-1} + \alpha_2\beta^{-2} + \ldots)\beta^p \geq \alpha_1\beta^{-1}\beta^p \geq \beta^{p-1}$$

e quindi

$$\left|\frac{\alpha - f\ell(\alpha)}{\alpha}\right| < \frac{\beta^{p-t}}{\beta^{p-1}} = \beta^{1-t}.$$

Caso di arrotondamento.

$$\left|\frac{\alpha - f\ell(\alpha)}{\alpha}\right| < \frac{1}{2}\frac{\beta^{p-t}}{\beta^{p-1}} = \frac{1}{2}\beta^{1-t}.$$

Il segno di uguaglianza in quest'ultima si potrebbe avere se e solo se $\alpha_{t+1} = \beta/2$ e $\alpha_{t+1} = 0; i \geq 2$, ma in tal caso risulterebbe

$$\alpha \geq (\alpha_1\beta^{-1} + \alpha_{t+1}\beta^{-t-1} + \ldots)\beta^p > \alpha_1\beta^{-1}\beta^p \geq \beta^{p-1}.$$

$\odot$

[4] Con la notazione $f\ell()$, d'ora in poi, s'indende una generica approssimazione del suo argomento secondo uno dei due criteri definiti nelle 1.2 e 1.3

L'errore relativo che si commette nel rappresentare il numero reale α con un numero finito $f\ell(\alpha)$ lo indicheremo con ϵ e lo chiameremo *errore relativo di rappresentazione*:

$$\epsilon = \frac{f\ell(\alpha) - \alpha}{\alpha} \qquad (1.5)$$

In definitiva il Teorema 1.2 dice che l'unità di arrotondamento u limita superiormente il modulo dell'errore relativo di rappresentazione.

Corollario 1.1 *Per ogni $\alpha \in \mathbb{R}$ e $\alpha \neq 0$ vale*

$$f\ell(\alpha) = \alpha(1+\epsilon), \qquad \text{con } |\epsilon| < u \qquad (1.6)$$

Dim.
Banalmente se si definisce $\epsilon := \dfrac{f\ell(\alpha) - \alpha}{\alpha}$, per il Teorema si ha che $|\epsilon| < u$ e $f\ell(\alpha) = \alpha(1+\epsilon)$. $\odot$

Osservazione 1.2 *Analogamente al Teorema 1.3, se $f\ell(\alpha) \neq 0$ vale*

$$\left| \frac{f\ell(\alpha) - \alpha}{f\ell(\alpha)} \right| < u;$$

da questo si ha poi che definendo

$$\epsilon = \frac{\alpha - f\ell(\alpha)}{f\ell(\alpha)} \qquad \text{vale} \qquad f\ell(\alpha) = \frac{\alpha}{1+\epsilon} \qquad \text{con } |\epsilon| < u. \qquad (1.7)$$

Esempio 1.3 *Si eseguano i passi necessari per rappresentare il numero reale $(-13.9)_{10}$ in un'area di memoria di 8 bit (1 per il segno, 3 per l'exponent biased e 4 per la mantissa), che permettono di memorizzare l'insieme $\mathbb{F}(2, 5, -3, 4)$ per troncamento; si converta poi quanto rappresentato nuovamente in base 10 e si valuti l'errore relativo di rappresentazione.*

Come osservato nella nota 1, lavorando in base 2 possiamo evitare di memorizzare il primo bit di mantissa che sarà sempre 1, così che memorizzeremo 5 cifre della mantissa in 4 bit. Per l'exponent biased avremo che -3 viene associato a $(0)_{10} = (000)_2$, mentre 4 viene associato a $(7)_{10} = (111)_2$.

Come primo passo convertiamo il numero in binario, prima la parte intera, quindi la parte decimale; sarà:

$$(13)_{10} = (1101)_2, \qquad (0.9)_{10} = (0.11100\ldots)_2$$

da cui

$$\begin{aligned} (-13.9)_{10} &= (-1101.11100\ldots)_2 \\ &= (-1.10111100\ldots)_2 \times 2^3 \\ &= (-1.10111100\ldots)_2 \times 2^{(110)_2}; \end{aligned}$$

da cui

$$f\ell((-13.9)_{10}) = (-1.1011)_2 \times 2^{(110)_2} = 11101011.$$

Per riconvertire il numero floating point appena determinato in base 10*, faremo:*

$$\begin{aligned}(-1.1011)_2 \times 2^{(110)_2} &= (-1.1011)_2 \times 2^{(3)_{10}} \\ &= (-1101.1)_2 \\ &= -(1 \times 2^3 + 1 \times 2^2 + 0^2 + 1 \times 2^0 + 1 \times 2^{-1}) \\ &= -(8 + 4 + 1 + 0.5)_{10} \\ &= -(13.5)_{10}.\end{aligned}$$

L'errore relativo di rappresentazione è:

$$E_{rel} = \left|\frac{-13.9 - (-13.5)}{-13.9}\right| = \frac{0.4}{13.9} = 0.0288$$

e ricordando che per l'insieme $\mathbb{F}$ *l'unità di arrotondamento è* $u = 2^{1-5} = 2^{-4} = 0.0625$ *possiamo verificare che* $0.0288 < 0.0625$.

Esercizio 1.1 *Si ripeta l'esempio per rappresentare il numero reale* $(-13.9)_{10}$ *nell'insieme* $\mathbb{F}(2, 5, -3, 4)$ *per arrotondamento.*

Esercizio 1.2 *Si ripeta l'esempio per rappresentare il numero reale* $(+13.9)_{10}$ *nell'insieme* $\mathbb{F}(2, 4, -7, 8)$ *per troncamento.*

1.3 Aritmetica floating-point

Oltre all'errore introdotto nella rappresentazione di un numero reale, anche le singole operazioni aritmetiche risultano approssimate. Ad esempio la somma di due numeri finiti $x, y \in \mathbb{F}(\beta, t, \lambda, \omega)$ è un numero che può non appartenere all'insieme $\mathbb{F}(\beta, t, \lambda, \omega)$.

Esempio 1.4 Siano 0.1×10^{-3} e $0.1 \times 10^{3} \in \mathbb{F}(10, 3, \lambda, \omega)$. Eseguendo la somma si ha:

$$\begin{array}{rl} 100.0000 & + \\ 0.0001 & = \\ \hline 100.0001 & \end{array}$$

ma, $100.0001 = 0.1000001 \times 10^3 \notin \mathbb{F}(10, 3, \lambda, \omega)$ perché la sua rappresentazione esatta richiede 7 cifre per la mantissa.

Si presenta quindi il problema di approssimare, nel modo più conveniente possibile, il risultato di un'operazione aritmetica fra due numeri finiti con un numero finito, occorre cioè definire *un'aritmetica di macchina*. Per convenzione si assume che:

> se $\tilde{op}$ è l'operazione di macchina che approssima l'operazione esatta op, per tutti i numeri finiti x, y per cui l'operazione non dia luogo a condizioni di overflow o di underflow, sia:
>
> $$x\ \tilde{op}\ y = f\ell(x\ op\ y)$$
>
> Questa convenzione e il Teorema 1.3, comportano che, se $x\ op\ y \neq 0$, vale:
>
> $$\left|\frac{x\ \tilde{op}\ y - (x\ op\ y)}{x op\ y}\right| < u.$$

Per il Corollario 1.1 possiamo anche scrivere:

$$x\ \tilde{op}\ y = (x\ op\ y)(1+\epsilon), \qquad |\epsilon| < u \tag{1.8}$$

con ϵ l'errore relativo commesso nell'operazione.

A parole possiamo dire che il risultato di un'operazione macchina deve essere uguale all'approssimazione con un numero finito del risultato dell'operazione esatta.

L'unità di arrotondamento u è detta *precisione di macchina* nel fissato sistema floating-point; con questa terminologia si indica sia il suo ruolo di precisione di rappresentazione (Teorema 1.3) che precisione di calcolo (risulto precedente).

La sua importanza numerica è data anche dalla seguente caratterizzazione:

> u è il più piccolo numero finito positivo tale che
>
> $$u \tilde{+} 1 = f\ell(u+1) > 1$$

$u \in \mathbb{N}^+$

Questo implica che per ogni numero finito $v < u$ sarà $v \tilde{+} 1 = f\ell(v+1) = 1$.

Esercizio 1.3 *Si verifichi la caratterizzazione di u sia per troncamento che per arrotondamento.*

Si noti che nell'aritmetica floating-point non valgono molte delle proprietà fondamentali dell'aritmetica classica, come la proprietà associativa, di cancellazione, ecc...

Esempio 1.5 Siano $\mathbb{F}(10, 2, \lambda, \omega)$, $a = 0.11 \times 10^0$, $b = 0.13 \times 10^{-1}$, $c = 0.14 \times 10^{-1}$
In aritmetica floating-point con arrotondamento:

$$a + (b + c) = a + (0.27 \times 10^{-1}) = 0.14 \times 10^0$$

$$(a + b) + c = (0.12 \times 10^0) + c = 0.13 \times 10^0$$

per cui:

$$a + (b + c) \neq (a + b) + c$$

1.4 ANSI/IEEE Std.754-1985

Uno degli scopi di questo standard è facilitare la portabilità del codice, così che sia possibile eseguire lo stesso programma su differenti architetture ottenendo gli stessi risultati.
Una proposta di standard fu presentata nel 1979, ma solo nel 1985, dopo varie modifiche, è stato adottato l'ANSI/IEEE Std.754-1985 e implementato su diversi microprocessori (fra i primi INTEL8087 e Motorola 68881). In questa sezione presenteremo tale standard, ma senza entrare troppo nel dettaglio rimandando a [Gol91] o [Ove01] per chi fosse interessato a saperne di più.
Lo standard definisce quattro formati floating-point in due gruppi, **basic** ed **extended**, ciascuno con due precisioni, **single** e **double**.
L'organizzazione del formato **basic** precisione **single** consiste in 1 bit per il segno (s), 8 bit per l'esponente (p) ($-127 \leq p \leq 128$ e $\bar{p} = p + 127 \in [0, 255]$, bias 127) e 23 bit per la mantissa (m), per un totale di 32 bit. Il valore v di un numero $f\ell(\alpha)$ è:

a. se $\bar{p} = 255$ ed $m \neq 0$ allora $v = NAN$ (Not A Number)

b. se $\bar{p} = 255$ ed $m = 0$ allora $v = (-1)^s \infty$ (Infinity)

c. se $0 < \bar{p} < 255$ allora $v = (-1)^s (1.m) \times 2^{\bar{p}-127} (Normalized)$

d. se $\bar{p} = 0$ e $m \neq 0$ allora $v = (-1)^s (0.m) \times 2^{\bar{p}-127}$ (Not Normalized)

e. se $\bar{p} = 0$ e $m = 0$ $v = (-1)^s 0$ (Zero)

Il caso più ricorrente è il caso **c**. Si osservi come dato $\mathbb{F}(\beta, t, \lambda, \omega)$ lo standard considera la mantissa nella forma

$$m_t = \alpha_0.\alpha_1\alpha_2 \cdots \alpha_t \qquad \alpha_0 \neq 0$$

e memorizza solo $\alpha_1, \ldots, \alpha_t$, comportando una differenza sull'esponente rispetto alle notazioni precedentemente introdotte. L'esponente è memorizzato in forma BIASED (per traslazione). Viene considerata una informazione non numerica NAN (Not A Number) per rendere il debugging più agevole; infatti quando viene effettuata un'operazione non valida come 0/0, o quando si utilizza una variabile non inizializzata, il risultato sarà un NAN. Questo comporta che in molti casi non viene più arrestata l'esecuzione di un programma quando viene effettuata un'operazione non valida.
Nel passato, quando un'operazione incorreva in underflow, era una pratica comune mettere il risultato a zero. Lo standard ANSI/IEEE definisce il "gradual underflow" (caso **d**).

Il formato **basic** precisone double è analogo alla precisione **single**. 1 bit per il segno, 11 bit per l'esponente (p) ($-1o23 \leq p \leq 1024$ e $\bar{p} = p+1023 \in [0, 2048]$, bias 1023) e 52 bit per la mantissa per un totale di 64 bit.

I dettagli del formato **extended** precisone **single** sono lasciati all'implementatore: nel caso di precisone **single** viene solo fatta la richiesta minima di 1 bit per il segno, almeno 32 bit per la mantissa e un esponenete che può essere BIASED e deve soddisfare $\lambda \leq -1023$ ed $\omega \geq 1024$; nel caso di precisione **double** si usano almeno 63 bit per la mantissa.

Un'implementazione dello standard fornisce le operazioni di addizione, sottrazione, moltiplicazione, divisione e radice quadrata, così come conversioni binario-decimale e viceversa. Tranne che per le conversioni, tutte le operazioni forniscono un risultato corrispondente all'approssimazione floating-point del risultato teorico o a precisione infinita.
La modalità standard di arrotondamento coincide con quella precedentemente descritta con la particolarità dell'**arrotondamento ai pari** quando un reale α è esattamente equidistante dai due numeri finiti x ed y consecutivi (questo si verifica quando $\alpha_{t+1} = \beta/2$ e $\alpha_i = 0$ per $i > t + 1$); in questa situazione l'arrotondamento comporta l'incremento di α_t di 1 se α_t è dispari, ma di non incrementarlo se è pari.
Nel caso di arrotondamento ai pari u viene caratterizzato come il più grande numero finito positivo tale che $f\ell(u + 1) = 1$.

Esercizio 1.4 *Si verifichi la caratterizzazione di u nello standard IEEE applicando l'arrotondamento ai pari.*

Nel 2008 lo standard ANSI/IEEE-754 è stato rivisitato per includere la rappresentazione a 128 bit e l'operazione "Fused Multiply-Add" (FMA). L'ope-

razione FMA calcola $f\ell(x * y + z)$ con un solo passo di arrotondamento. Senza l'operazione FMA il risultato sarebbe calcolato come $f\ell(f\ell(x * y) + z)$ con due arrotondamenti. Poiché FMA usa un solo arrotondamento il risultato risulta più accurato.

1.5 Analisi degli errori

In questa sezione affronteremo l'analisi degli errori che si possono avere nel risolvere un problema *ben posto* [5]. Al di là degli errori introdotti da singole operazioni, le situazioni più gravi nascono nel processo di soluzione del problema, che usualmente consiste nell'esecuzione di una sequenza di operazioni nella quale, ad ogni passo, vengono utilizzati come input valori calcolati precedentemente. In questo caso si verifica una *propagazione degli errori* la cui entità molto spesso è tutt'altro che trascurabile.
Il controllo e la gestione di tale fenomeno risulta fondamentale in tutte le applicazioni informatiche che implementano un metodo di calcolo, al fine di determinare l'attendibilità dei risultati ottenuti. La teoria della propagazione degli errori fornisce alcuni strumenti di base per questo scopo.

1.5.1 Precisione e Accuratezza

I termini *precisione* e *accuratezza* vengono spesso confusi o usati in modo intercambiabile, ma vale la pena distinguerli. Accuratezza si riferisce all'errore assoluto o relativo di una quantità approssimata. Precisione è l'accuratezza con la quale le operazioni aritmetiche $+, -, *, /$ vengono effettuate, e nell'aritmetica floating point questa viene misurata dall'unità di arrotondamento u. Accuratezza e precisione sono la stessa per il valore scalare $c = a * b$, ma per esempio l'accuratezza può essere molto peggiore della precisione nella soluzione di un sistema di equazioni lineari.

È importante infine sapere che l'accuratezza non è limitata dalla precisione, almeno in teoria. Questo può sembrare sorprendente e a volte contradittorio, tuttavia l'aritmetica di una data precisione può essere utilizzata per simulare l'aritmetica di una precisione arbitrariamente più grande (vedi librerie software per la precisione variabile, Error Free Transformation, Variable-Precision Arithmetic di Matlab)[Higham2002].

[5] Un problema è *ben posto* quando ammette, in un opportuno campo di definizione, una ed una sola soluzione ed inoltre tale soluzione dipende con continuità dai dati.

1.5.2 Analisi in avanti degli errori

Per valutare l'entità della propagazione degli errori in una sequenza di operazioni si considera l'errore totale dato dalla somma dei singoli errori.
Un problema può essere modellizzato come una funzione f che a partire da un dato x produce un risultato $f(x)$. Per ogni dato ed operazione si introduce un errore che rappresenta l'approssimazione introdotta nella rappresentazione e dall'aritmetica finita (espressioni 1.6, 1.8) e si ricava un corrispondete risultato approssimato $\tilde{f}(x)$.
La quantità

$$\left| \frac{f(x) - \tilde{f}(x)}{f(x)} \right|$$

fornisce un'indicazione sull'entità dell'errore relativo totale che affligge la soluzione del problema f con dato x (vedi Fig. 1.4).

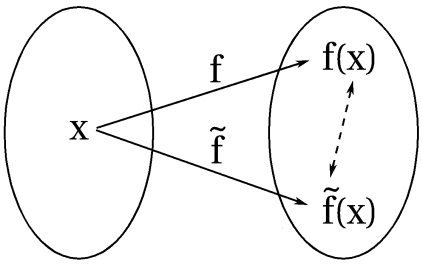

Figura 1.4: Analisi in avanti dell'errore

Esempio 1.6 Come esempio per chiarire questo tipo di analisi degli errori si consideri l'addizione di due numeri finiti x ed y; avremo

$$\left| \frac{f\ell(x+y) - (x+y)}{x+y} \right| < u$$

che indica che l'errore nell'effettuare la singola addizione è maggiorato da u.

1.5.3 Analisi all'indietro degli errori

Approccio opposto al precedente consiste nel considerare il risultato finale $\tilde{f}(x)$ come risultato esatto derivato da dati iniziali perturbati rispetto a quelli reali. La valutazione dell'entità dell'errore è data quindi da un fattore dx sul dato iniziale x tale che $f(x + dx) = \tilde{f}(x)$ (vedi Fig. 1.5).

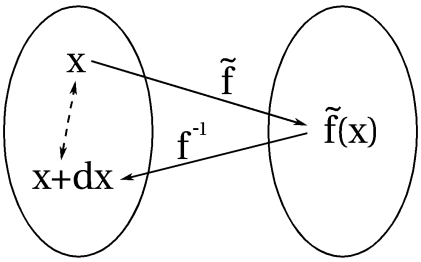

Figura 1.5: Analisi all'indietro dell'errore

Esempio 1.7 Come esempio per chiarire questo tipo di analisi degli errori si consideri l'addizione di due numeri finiti x ed y; avremo

$$\tilde{f}(x,y) = f\ell(x+y) = (x+y)(1+\epsilon) = x(1+\epsilon) + y(1+\epsilon)$$

quindi:

$$\tilde{f}(x,y) = f(x+dx, y+dy)$$

dove: $dx = x\epsilon$, $dy = y\epsilon$ che indica $\tilde{f}(x,y)$ come il risultato esatto a partire da due dati $x+dx$ e $y+dy$ che rappresentano delle perturbazioni dei dati iniziali.

Andare a pag 31

1.5.4 Errori nelle operazioni di macchina

Applicando l'analisi in avanti sulle quattro operazioni aritmetiche si ottengono alcuni importanti risultati; siano $x, y \in \mathbb{R}$, allora:

moltiplicazione

$$\left|\frac{f\ell(f\ell(x)f\ell(y)) - xy}{xy}\right| = \left|\frac{x(1+\epsilon_1)y(1+\epsilon_2)(1+\epsilon_3) - xy}{xy}\right|$$
$$= \left|\frac{xy + xy(\epsilon_1+\epsilon_2+\epsilon_3+\epsilon_1\epsilon_2+\epsilon_1\epsilon_3+\epsilon_2\epsilon_3+\epsilon_1\epsilon_2\epsilon_3) - xy}{xy}\right|$$
$$\simeq |\epsilon_1+\epsilon_2+\epsilon_3| < 3u$$

dove $|\epsilon_1|, |\epsilon_2|, |\epsilon_3| < u$ e le semplificazioni sono ottenute conducendo un'analisi del primo ordine[6] .
Risulta una propagazione lineare degli errori, per cui l'operazione di moltiplicazione è stabile.

[6]Nell'ipotesi realistica che gli errori ϵ siano dell'ordine della precisione di macchina u, che è un valore molto più piccolo di 1, è ragionevole condurre un'*analisi dell'errore del primo ordine*, cioè trascurare il contributo dei termini non lineari negli errori ϵ.

divisione

$$\left|\frac{f\ell(f\ell(x)/f\ell(y)) - x/y}{x/y}\right| = \left|\frac{x(1+\epsilon_1)\frac{(1+\epsilon_2)}{y}(1+\epsilon_3) - x/y}{x/y}\right|$$
$$= |(1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3) - 1|$$
$$\simeq |\epsilon_1 + \epsilon_2 + \epsilon_3| < 3u$$

Analogamente all'operazione di moltiplicazione anche la divisione è stabile.

addizione e sottrazione

$$\left|\frac{f\ell(f\ell(x) \pm f\ell(y)) - (x \pm y)}{x \pm y}\right|$$
$$= \left|\frac{[x(1+\epsilon_1) \pm y(1+\epsilon_2)](1+\epsilon_3) - (x \pm y)}{x \pm y}\right|$$
$$\simeq \left|\frac{x}{x \pm y}\epsilon_1 \pm \frac{y}{x \pm y}\epsilon_2 + \epsilon_3\right|$$

Nel caso dell'addizione (sottrazione) di due numeri x, y con segno concorde (discorde) l'errore finale è maggiorato da $3u$.
Altrimenti non è possibile dare una maggiorazione dell'errore relativo indipendente da x, y. In particolare se $(x \pm y) \to 0$ la limitazione dell'errore risulta molto elevata.
In questi casi, l'elevato errore che si può ritrovare nel risultato non è generato dall'operazione aritmetica floating-point (infatti l'errore locale dell'operazione ϵ_3 ha modulo minore di u), ma è dovuto alla presenza degli errori relativi non nulli ϵ_1, ϵ_2 su x, y che sono amplificati dalle operazioni aritmetiche di addizione e sottrazione.
Tale fenomeno viene definito di *cancellazione numerica* (vedi anche paragrafo 1.5.7) ed è senza dubbio la conseguenza più grave della rappresentazione con precisione finita dei numeri reali.

Esempio 1.8 Sia $\mathbb{F}(10, 6, \lambda, \omega)$ con rappresentazione per arrotondamento e siano dati i numeri reali $a = 0.147554326$ e $b = -0.147251742$. La loro approssimazione nell'insieme $\mathbb{F}$ sarà: $f\ell(a) = 0.147554$ e $f\ell(b) = 0.147252$. L'addizione esatta darà: $a + b = 0.302584 \times 10^{-3}$, mentre in aritmetica finita darà: $f\ell(f\ell(a) + f\ell(b)) = 0.302000 \times 10^{-3}$. L'errore relativo commesso sarà: $\epsilon \simeq 0.2 \times 10^{-2}$, mentre $u = 0.5 \times 10^{-5}$ comportando un grave errore.

1.5.5 Instabilità senza Cancellazione

Spesso si assume che una computazione che non effettua sottrazioni e quindi cancellazione deve essere accurata e stabile, specialmente se vengono coinvolte poche operazioni. I tre esempi che riportiamo mostrano l'erroneità di questa assunzione.

Per qualunque $x \geq 0$ la seguente computazione dovrebbe lasciare x immutato:

```
for i=1:60
  x=sqrt(x);
end
for i=1:60
  x=x^2;
end
```

Poiché il calcolo non coinvolge sottrazioni e tutte le quantità intermedie sono tra 1 ed x, ci possiamo aspettare che ritorni una approssimazione accurata di x in aritmetica floating point.

In Matlab, dando $x = 100$ viene prodotto $x = 1$; viene prodotto un risultato approssimato completamente inaccurato del risultato e compiendo appena 120 operazioni far quantità non negative. Come può essere possibile?

È ben noto che

$$\sum_{k=1}^{\infty} k^{-2} = \pi^2/6 = 1.644934066848\ldots$$

Supponiamo di non sapere di questa identità e di voler approssimare numericamente la somma. La strategia più ovvia è valutare la somma per k crescenti fino a che la somma calcolata non cambia. In precisione basic single questo porta al valore 1.6447253, che è ottenuto per $k = 4096$. Questo risultato è in accordo con la somma infinita per appena 4 cifre delle 7 possibili. Come può essere possibile?

```
somma1=single(0);
k=0;
somma=single(1);
while (somma ~= somma1)
    somma=somma1;
    k=k+1;
    somma1=somma+single(1/k^2);
end
disp('risultato atteso:')
disp(pi^2/6);
```

```
disp('invece ...')
disp(somma)
disp('k=')
disp(k)
disp('Errore relativo')
disp(abs(somma-pi^2/6)/(pi^2/6))
```

1.5.6 Alcuni risultati utili

Teorema 1.4 *Se $x, y, z \in \mathbb{F}$ e $x \leq y$ allora $f\ell(x+z) \leq f\ell(y+z)$.*

Dim.
$\odot$

Teorema 1.5 *Se $x, y \in \mathbb{F}$ allora $x \leq f\ell(\frac{1}{2}(x+y)) \leq y$ oppure $y \leq f\ell(\frac{1}{2}(x+y)) \leq x$.*

Dim.
$\odot$

Lemma 1.1 *Sia $t \in \mathbb{F}$ e $0 \leq t \leq 1$, se poniamo $s := f\ell(1-t)$ allora $f\ell(s+t) = 1$.*

Dim.
$\odot$

Teorema 1.6 *Se $x, y, t \in \mathbb{F}$ e $0 \leq t \leq 1$, $x \leq y$ allora la combinazione convessa*

$$(1-t)x + ty$$

è un numero di $\mathbb{F}$ tra x ed y, cioè

$$x \leq f\ell(f\ell(f\ell(1-t)x)) + f\ell(ty)) \leq y.$$

Dim.
$\odot$
Applichiamo l'analisi in avanti degli errori all'operazione di combinazione convessa.

$$\frac{f\ell(f\ell(f\ell(1-t)x) + f\ell(ty)) - ((1-t)x + ty)}{(1-t)x + ty}$$
$$= \frac{[(1-t)(1+\epsilon_1)x(1+\epsilon_2) + ty(1+\epsilon_3)](1+\epsilon_4) - ((1-t)x + ty)}{(1-t)x + ty}$$

$$= \frac{(1-t)x + ty + (1-t)x(\epsilon_1+\epsilon_2) + (1-t)x\epsilon_1\epsilon_2 + ty\epsilon_3}{(1-t)x+ty} +$$
$$+ \frac{[(1-t)x + ty + (1-t)x(\epsilon_1+\epsilon_2) + (1-t)x\epsilon_1\epsilon_2 + ty\epsilon_3]\epsilon_4 - ((1-t)x+ty)}{(1-t)x+ty}$$
$$\simeq \frac{(1-t)x}{(1-t)x+ty}(\epsilon_1+\epsilon_2) + \frac{ty}{(1-t)x+ty}\epsilon_3 + \epsilon_4$$

da questa analisi segue che se x ed y sono entrambi positivi o negativi l'errore è contenuto in $4u$, mentre ciò non accade se x ed y hanno segno opposto.

Consideriamo il caso in cui $x \leq 0 \leq y$. Sia $\tilde{t}$ il parametro in $[0,1]$ con cui si vogliono combinare x ed y e si determini $s = (1-\tilde{t})x + \tilde{t}y$ nel seguente modo:

- sia $t_0 = \frac{-x}{y-x}$ risolvendo l'equazione $(1-t)x + ty = 0$;
- si considerino i seguenti tre casi:
  - se $\tilde{t} < t_0$, allora $s = \dfrac{t_0 - \tilde{t}}{t_0}x$;
  - se $\tilde{t} \equiv t_0$, allora $s = 0$;
  - se $\tilde{t} > t_0$, allora $s = \dfrac{\tilde{t} - t_0}{1-t_0}y$.

Questo procedura evita errori numerici lavorando sempre fra quantità positive.

1.5.7 Errori numerici più comuni

Nei paragrafi precedenti sono state descritte la rappresentazione dei numeri finiti e l'aritmetica floating-poing.
Da queste approssimazioni derivano una serie di errori che vanno tenuti in considerazione per valutare l'attendibilità del risultato floating-point di un problema. Nell'elenco che segue, si cerca di dare una descrizione di quelli più comuni (negli esempi si sottointende l'insieme dei numeri finiti $\mathbb{F}(10,5,\lambda,\omega)$) in cui $u = \frac{1}{2}10^{1-5} = 0.00005$:

Errori di conversione : poiché l'input dei dati avviene attraverso il formato decimale, mentre la base interna dei computer è binaria, non sempre è possibile rappresentare in modo esatto i valori numerici introdotti. Per esempio, il numero decimale 0.6 ha una rappresentazione binaria periodica $0.1001\,1001\ldots$, e quindi verrà arrotondato con un numero finito t di cifre introducendo un certo errore. Ancora, numeri periodici in base 10 o numeri irrazionali non potranno essere introdotti e rappresentati in modo esatto.

Errori di arrotondamento : poiché $a \cdot b$ in generale richiede una precisione maggiore dei rispettivi operandi a e b per essere rappresentato esattamente, il risultato di un prodotto può essere arrotondato risultando inesatto nell'ultima cifra. Per esempio, $0.24665 \cdot 0.63994 = 0.1578412010$ viene arrotondato a 0.15784 con un errore relativo circa di 1.2×10^{-6}

Errore di assorbimento : sommando due numeri a e b di due ordini di grandezza diversi, quello più piccolo può essere "assorbito" e non influenzare il risultato. Per esempio $0.1 \times 10^3 + 0.3 \times 10^{-3}$ è:

$$\begin{array}{rl} 100.0 & + \\ 0.0003 & \\ \hline 100.0003 & \end{array}$$

per cui il risultato finale approssimato è 0.1×10^3 e quindi $a + b = a$ anche se $b \neq 0$

Errore di cancellazione numerica : la differenza di due numeri a e b quasi uguali ha meno cifre significative rispetto sia ad a che b. Per esempio $0.90905 \times 10^2 - 0.90903 \times 10^2 = 0.2 \times 10^{-2}$. La mantissa risultante 0.20000, a meno che a e b siano rappresentabili in modo esatto, contiene quattro zeri non significativi. Infatti essi non derivano da un calcolo, ma sono dovuti alla perdita di precisione (vedi anche paragrafo 1.5.4).

1.5.8 Errore Analitico, Inerente e Algoritmico

In questo paragrafo si formalizza ulteriormente l'esame della propagazione dell'errore modelizzando il problema da risolvere con una funzione matematica e la soluzione del problema con la valutazione della funzione. Le funzioni effettivamente calcolabili su un computer sono solo le *funzioni razionali*, il cui valore è ottenuto mediante un numero finito di operazioni aritmetiche. Le funzioni non razionali, come ad esempio le trigonometriche, possono solo essere approssimate da opportune funzioni razionali.
In generale, data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ ed un vettore $x = (x_1, x_2, \ldots x_n)$ di dati in input, comprendente le variabili pure e i coefficenti numerici che definiscono f [7] , il valore effettivamente calcolato può essere affetto da errori indotti da diversi fenomeni:

[7] Si considera ogni funzione composta da una parte simbolica, rappresenta esattamente, ed una numerica. Ad esempio la rappresentazione di una retta è composta dalla sua equazione simbolica $y = mx + q$ e dai valori numerici dei coefficenti m, q. In questo caso il vettore di dati in input è (x, m, q) dove x è la variabile pura.

errore analitico : errore generato, se la funzione f non è razionale, dalla sua approssimazione con una funzione razionale

errore inerente : errore conseguente a quello iniziale sui dati di input $x_1, x_2, \ldots x_n$ e alla loro rappresentazione approssimata mediante numeri finiti

errore algoritmico : errore generato dal fatto che le operazioni sono effettuate in aritmetica finita (di macchina)

Se $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$ sono le rappresentazioni finite di $x_1, x_2, \ldots, x_n$ e $\tilde{f}$ è la funzione effettivamente calcolata, il valore che si ottiene al posto di $f(x)$ è allora $\tilde{f}(\tilde{x})$, dove $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$.
Si esamina il caso in cui f è razionale.

Teorema 1.7 *Siano gli $x_i \neq 0$, $i = 1, 2, \ldots, n$, e sia f una funzione razionale tale che $f(x) \neq 0$ e $f(\tilde{x}) \neq 0$. Indicati con*

$$E_{tot} = \left| \frac{\tilde{f}(\tilde{x}) - f(x)}{f(x)} \right|$$

*l'*errore totale relativo *di $\tilde{f}(\tilde{x})$ rispetto a $f(x)$, con*

$$E_{in} = \left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right|$$

*l'*errore inerente *e con*

$$E_{alg} = \left| \frac{\tilde{f}(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})} \right|$$

*l'*errore algoritmico, *risulta*

$$E_{tot} = E_{alg}(1 + E_{in}) + E_{in}. \tag{1.9}$$

Applicando l'analisi del primo ordine (vedi nota a pagina 18) la 1.9 risulta così semplificata:

$$E_{tot} \simeq E_{alg} + E_{in}. \tag{1.10}$$

Dim.
Si ha

$$\begin{aligned} E_{tot} &= \left| \frac{\tilde{f}(\tilde{x}) - f(x)}{f(x)} \right| = \left| \frac{\tilde{f}(\tilde{x})}{f(x)} - 1 \right| \\ &= \left| \frac{\tilde{f}(\tilde{x})}{f(\tilde{x})} \frac{f(\tilde{x})}{f(x)} - 1 \right| = (1 + E_{alg})(1 + E_{in}) - 1 \\ &= E_{alg}(1 + E_{in}) + E_{in}) \simeq E_{alg} + E_{in}. \end{aligned}$$

⊙

Se il problema $f : \mathbb{R}^n \to \mathbb{R}$ è differenziabile in un intorno di x, nelle ipotesi che i dati $x_i \neq 0$ per $i = 1, \ldots, n$, si può dare la seguente maggiorazione dell'errore inerente

$$\left|\frac{f(\tilde{x}) - f(x)}{f(x)}\right| \leq \sum_{i=1}^{n} |c_i \epsilon_i| \tag{1.11}$$

dove

$$c_i = \frac{x_i}{f(x)} \frac{\partial f(x)}{\partial x_i} \tag{1.12}$$

e $\epsilon_i = \frac{\tilde{x}_i - x_i}{x_i}$, con $|\epsilon_i| < u$, per $i = 1, 2, \ldots, n$ sono gli errori relativi sui dati del problema.

Infatti applicando lo sviluppo di Taylor di funzione in più variabili con arresto ai termini del primo ordine al problema f si ha:

$$f(\tilde{x}) = f(x) + \sum_{i=1}^{n} (\tilde{x}_i - x) \frac{\partial f(x)}{\partial x_i} + o(h^2)$$

dove $h^2 = \sum_{i=1}^{n} (\tilde{x}_i - x_i)^2 = \sum_{i=1}^{n} x_i^2 \epsilon_i^2$ con $\epsilon_i = \frac{\tilde{x}_i - x_i}{x_i}$, nelle ipotesi che $x_i \neq 0$ per $i = 1, \ldots, n$.

Allora

$$\begin{aligned} E_{in} &= \left|\frac{f(\tilde{x}) - f(x)}{f(x)}\right| \simeq \left|\frac{\sum_{i=1}^{n} (\tilde{x}_i - x) \frac{\partial f(x)}{\partial x_i}}{f(x)}\right| \\ &\leq \sum_{i=1}^{n} \left|\frac{(\tilde{x}_i - x)}{f(x)} \frac{\partial f(x)}{\partial x_i}\right| = \sum_{i=1}^{n} \left|\frac{x_i}{f(x)} \frac{\tilde{x}_i - x_i}{x_i} \frac{\partial f}{\partial x_i}\right| \\ &= \sum_{i=1}^{n} |c_i \epsilon_i| \end{aligned}$$

con c_i dato nella 1.12.

I coefficienti c_i sono detti *coefficienti di amplificazione* o *indici di condizionamento* e danno una misura di quanto influisce l'errore relativo ϵ_i, da cui è affetto il dato x_i, sul risultato: se i coefficienti c_i sono di modulo elevato, anche piccoli errori ϵ_i inducono grossi errori su $f(x)$; in questo caso il calcolo di $f(x)$ è detto *mal condizionato* [8].
Tale errore, come evidente dalla formula 1.12, dipende unicamente dalla funzione f ed in particolare dalla sua rappresentazione simbolica, mentre *i coefficenti*

[8] Con condizionamneto intendiamo un concetto più restrittivo di *ben posto* (vedi inizio sezione 1.5); in particolare si può dire che in un problema *ben condizionato* a piccole perturbazioni sui dati corrispondono piccole perturbazioni sui risultati

associati alle variabili pure sono costanti al variare dell'espressione usata per la f.
L'errore algoritmico E_{alg}, invece, è generato dal calcolo della funzione $\tilde{f}(\tilde{x})$ come composizione di un numero finito di operazioni di macchina. Poiché gli errori delle singole operazioni sono in modulo minori della precisione di calcolo, al primo ordine si può stimare

$$E_{alg} \simeq \theta(n, x) \cdot u$$

con n numero di operazioni effettuate. Se $\theta(n, x) = cn$ si dice che la crescita dell'errore è **lineare**, se $\theta(n, x) = c^n$ con $c > 1$ si dice che la crescita dell'errore è **esponenziale**. Una crescita lineare è normale (non evitabile) e non pericolosa, mentre una crescita esponenziale è insostenibile e invalida completamente i risultati di un calcolo. Un algoritmo che presenta una crescita esponenziale dell'errore si dice **instabile**. L'algoritmo risulterà tanto più **stabile**, in corrispondenza ad un insieme di dati, quanto più piccoli sono i valori assunti dalla funzione $\theta(n, x)$ in corrispondenza di quei dati.
Se la funzione f non è razionale, è necessario approssimarla con una funzione razionale g: tale approssimazione introduce l'errore analitico

$$E_{an} = \left| \frac{g(x) - f(x)}{f(x)} \right|$$

Con procedimenti equivalenti a quelli usati in precedenza risulta

$$E_{tot} \simeq E_{an} + E_{in} + E_{alg}.$$

1.5.9 Alcuni esempi

Esempio 1.9 *Si applica la stima 1.11 nei casi già visti di moltiplicazione, divisione e addizione/sottrazione fra numeri reali:*

moltiplicazione $f(x_1, x_2) = x_1 x_2$*; applicando la 1.12 si ha*

$$c_1 = \frac{x_1}{x_1 x_2} x_2 = 1 \qquad c_2 = \frac{x_2}{x_1 x_2} x_1 = 1$$

da cui si deduce che il problema in oggetto è ben condizionato e risulta

$$E_{in} = |\epsilon_1| + |\epsilon_2| \quad \textit{mentre} \quad E_{alg} = |\epsilon_3| \,.$$

divisione $f(x_1, x_2) = x_1/x_2$*; applicando la 1.12 si ha*

$$c_1 = \frac{x_1}{x_1/x_2} \frac{1}{x_2} = 1 \qquad c_2 = \frac{x_2}{x_1/x_2} \frac{-x_1}{x_2^2} = -1$$

da cui si deduce che il problema in oggetto è ben condizionato e risulta

$$E_{in} = |\epsilon_1| + |\epsilon_2| \quad \textit{mentre} \quad E_{alg} = |\epsilon_3| \,.$$

addizione/sottrazione *$f(x_1, x_2) = x_1 + x_2$; applicando la 1.12 si ha*

$$c_1 = \frac{x_1}{x_1 + x_2} \qquad c_2 = \frac{x_2}{x_1 + x_2}$$

da cui si deduce che il problema in oggetto è mal condizionato per $x_1+x_2 \to 0$ e risulta

$$E_{in} = \left|\frac{x_1}{x_1 + x_2}\epsilon_1\right| + \left|\frac{x_2}{x_1 + x_2}\epsilon_2\right| \quad \textit{mentre} \quad E_{alg} = |\epsilon_3|\,.$$

Esempio 1.10 *Si esamina il problema della valutazione della seguente espressione:*

$$y = \sqrt{x_1 + x_2} - \sqrt{x_1} \quad x_1 + x_2, x_1 \geq 0.$$

Si vede immediatamente che per $|x_2| << |x_1|$ si incorre in un errore di cancellazione numerica e quindi di instabilità del procedimento (infatti anche se x_1 e x_2 sono numeri finiti, non lo saranno $\sqrt{x_1 + x_2}$ e $\sqrt{x_1}$. In tal caso si cerca un algoritmo differente che eviti il problema. Nel caso in questione, razionalizzando si ha:

$$\begin{aligned} y &= (\sqrt{x_1 + x_2} - \sqrt{x_1})\frac{\sqrt{x_1 + x_2} + \sqrt{x_1}}{\sqrt{x_1 + x_2} + \sqrt{x_1}} \\ &= \frac{x_2}{\sqrt{x_1 + x_2} + \sqrt{x_1}}. \end{aligned}$$

L'espressione trovata è esente dal possibile errore segnalato; infatti a denominatore si effettua una addizione fra quantità non negative; l'operazione pericolosa è stata eseguita analiticamente.
Si osserva però che per $x_1 + x_2 \to 0$ anche la versione più stabile dà risultati scadenti; qual è la causa? analizziamo l'errore inerente utilizzando la stima 1.11: nel caso specifico sarà $E_{in} = c_1\epsilon_1 + c_2\epsilon_2$ con c_1 e c_2 calcolati tramite la 1.12; facendo i conti si ottiene:

$$c_1 = -\frac{1}{2}\sqrt{\frac{x_1}{x_1 + x_2}} \qquad c_2 = \frac{1}{2}\left(1 + \sqrt{\frac{x_1}{x_1 + x_2}}\right) = \frac{1}{2} - c_1.$$

e il problema risulta mal condizionato proprio quando $x_1 + x_2 \to 0$.
Ilustriamo la situazione con un esempio numerico, dove con y denotiamo il valore esatto, mentre con y_1 e y_2 rispettivamente i risultati dei due algoritmi ottenuti lavorando in $\mathbb{F}(2, 24, -128, 127)$.

- *siano dati $x_1 = 0.1 \times 10^1$ e $x_2 = 0.1 \times 10^{-3}$, allora $c_2 \simeq 1$ che indica un buon condizionamento in questo caso. I risultati esatto e calcolati con i due algoritmi sono:*

$$\begin{aligned} y &= 0.4999875 \times 10{-}6 \\ y_1 &= 0.4994869 \times 10{-}6 \\ y_2 &= 0.4999875 \times 10{-}6 \end{aligned}$$

che indica l'instabilità del primo algoritmo, infatti $|x_2| << |x_1|$.

- *siano dati* $x_1 = 1$ *e* $x_2 = -1 + 10^{-6}$, *allora* $c_2 \simeq 0.5 \times 10^3$ *che indica un cattivo condizionamento, infatti* $x_1 + x_2 \to 0$. *I risultati esatto e calcolati con i due algoritmi sono:*

$$y = -0.999$$
$$y_1 = y_2 = -0.9985858$$

che mostra come entrambi i risultati calcolati siano scadenti.

Esempio 1.11 *Si esamina il problema della risoluzione di un'equazione di* 2^o *grado. Data l'equazione*

$$ax^2 + bx + c = 0$$

sappiamo che ha esattamente due soluzioni date da:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \tag{1.13}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}; \tag{1.14}$$

se $b^2 - 4ac \geq 0$, *le soluzioni sono reali.*

Supponiamo che $a = 1$, $b = -10^5$, $c = 1$ *e che si vogliano calcolare (1.13) e (1.14) usando un'aritmetica con* 8 *cifre decimali per la mantissa* ($t = 8$).

Poiché in aritmetica finita $10^{10} - 4 = 10^{10}$ *si ha:*

$$x_1 = \frac{10^5 + \sqrt{10^{10} - 4}}{2} = \frac{10^5 + \sqrt{10^{10}}}{2} = 10^5,$$

$$x_2 = \frac{10^5 - \sqrt{10^{10} - 4}}{2} = \frac{10^5 - \sqrt{10^{10}}}{2} = 0.$$

Le soluzioni esatte sono invece:

$$x_1 = 9999.99999 \simeq 10^5 \qquad x_2 = 0.000010000000001 \simeq 1/10^5.$$

Se la prima soluzione ottenuta può essere considerata una approssimazione accettabile della soluzione vera, la seconda sicuramente no. L'errore nel risultato è dovuto al fatto che, nel calcolo del discriminante $b^2 - 4ac$, *i valori* b^2 *e* $4ac$ *hanno esponenti troppo diversi. Quindi nell'addizione/sottrazione* $4ac$ *non influenza minimamente* b^2, *ossia*

$$\sqrt{b^2 - 4ac} = \sqrt{b^2} = b.$$

L'errore di arrotondamento porta a calcolare la differenza tra due termini uguali. Questo errore è noto come cancellazione dovuto ad assorbimento. Per evitare questi inconvenienti è necessario calcolare le soluzioni di un'equazione di 2^o grado con altri metodi. Formule alternative a (1.13) e (1.14) sono:

$$x_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}, \tag{1.15}$$

$$x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}; \tag{1.16}$$

Se $a \neq 0$ e $c \neq 0$, si può notare che, se $b > 0$, la cancellazione numerica può verificarsi in (1.13) e (1.16), ma non in (1.14) e (1.15). Quindi si può usare (1.15) per calcolare x_1 e (1.14) per calcolare x_2. Se $b < 0$, non c'è cancellazione numerica in (1.13) e (1.16), così si può usare (1.13) per calcolare x_1 e (1.16) per x_2.

Nell'esempio precedente $(x^2 - 10^5 + 1 = 0)$ è $b < 0$, per cui:

$$x_1 = \frac{10^5 + \sqrt{10^{10}}}{2} = 10^5,$$

$$x_2 = \frac{2}{10^5 + \sqrt{10^{10}}} = \frac{1}{10^5}.$$

Dagli esempi precedenti risulta che per uno stesso problema esistono procedimenti di risoluzione che producono errori algoritmici diversi. Si è anche visto che, indipendentemente dall'algoritmo usato, esiste un *errore inerente*, intrinseco al problema, che non può essere controllato.

1.5.10 Considerazioni

La teoria sulla propagazione dell'errore, pur essendo fondamentale per capire l'essenza del problema, presenta molti limiti che la rendono inutilizzabile nella maggior parte dei casi concreti.
Primo fra tutti, la pesantezza del procedimento con cui si ricava l'espressione dell'errore.
Si è visto nei vari esempi come l'analisi anche di semplici problemi, comporti una notevole quantità di calcoli. Pensando ad un software composto da varie centinaia di funzioni che dipendono in modo complesso l'una dall'altra si capisce come una analisi del genere sia impraticabile (per una trattazione completa dei procedimenti per ricavare le espressioni dell'errore si consulti [BBCM92], paragrafi 2.8 - 2.10).
Un ulteriore limite è dovuto al fatto che le espressioni ricavate rappresentano

una maggiorazione dell'errore, e quindi una previsione pessimistica. Gli errori che si verificano nell'esecuzione reale delle varie operazioni possono risultare molto più limitati, e quindi essere mal rappresentati dai valori massimi calcolati. Per ottenere valutazioni dell'errore più aderenti ai calcoli reali esistono metodi che seguono i vari passi di esecuzione a run-time, considerando i valori numerici effettivamente in gioco.

1.6 L'indecidibilità dell'eguaglianza dei numeri floating-point

Una conseguenza delle approssimazioni che deve essere considerata nella stesura di codice, è il diverso significato che assume l'operatore di eguaglianza tra due numeri floating-point.

> Nella quasi totalità dei casi l'eguaglianza intesa come perfetta identità di tutti i bit che rappresentano due numeri floating point a, b non è affidabile.

Esempio 1.12 Il calcolo della soluzione di un'equazione

$$f(x) = 0$$

è uno dei più importanti problemi della matematica applicata.
In generale non sono disponibili formule esplicite per calcolare la soluzione, per cui si deve ricorre a metodi iterativi che convergono ad essa mediante una successione $\{x_i\}$ di approssimazioni successive.
L'iterazione viene terminata in base ad opportuni *criteri d'arresto*, uno di questi è dato dalla relazione

$$|f(x_i)| < \epsilon \tag{1.17}$$

dove ϵ *non può essere scelto arbitrariamente piccolo*, perchè, a causa degli errori di arrotondamento, la condizione 1.17 potrebbe non essere mai soddisfatta (per maggiori dettagli si rimanda a [BBCM92] capitolo 3).

Continua pag 43

Esempio 1.13 Si supponga un'aritmetica $\mathbb{F}(10, 3, \lambda, \omega)$ e si considerino due vettori bidimensionali $\vec{v_1} = (1, 0)$ e $\vec{v_2} = (0, 1.1)$.
Questi sono tra loro ortogonali, infatti il loro prodotto scalare

$$\vec{v_1} \cdot \vec{v_2} = 1 \cdot 0 + 0 \cdot 1.1 = 0$$

risulta perfettamente nullo.
Si esegua poi una rotazione di un'angolo $\alpha = 28$ gradi su entrambi; l'algoritmo è il seguente

$$\tilde{v} = (v_x \cos(\alpha) - v_y \sin(\alpha), v_x \sin(\alpha) + v_y \cos(\alpha))$$

I valori approssimati a 3 cifre decimali sono

$$\sin(28) = 0.469 \qquad \cos(28) = 0.883$$

Si ottiene quindi

$$\tilde{v_1} = (0.883,\ 0.469)$$

$$\tilde{v_2} = (0.469 \cdot 1.1, 0.883 \cdot 1.1) = (-0.516,\ 0.971)$$

Naturalmente ci si aspetta che i vettori $\tilde{v_1}$, $\tilde{v_2}$ siano ancora ortogonali tra loro, ma calcolando il prodotto scalare

$$\tilde{v_1} \cdot \tilde{v_2} = 0.883 \cdot (-0.516) + 0.469 \cdot 0.917 = -0.456 + 0.455 = -0.001$$

questo non è più perfettamente uguale a zero.

Come evidente dagli esempi è necessario considerare che i valori numerici sono approssimati e quindi definire un criterio di eguaglianza a meno di un'opportuna tolleranza. Questo approccio ormai entrato nella pratica, formalmente deriva dal fatto che l'uguaglianza fra numeri floating point è indecidibile. Questa affermazione è confermata anche nei seguenti termini piuttosto intuitivi:

> dati due numeri floating-point a e b, pur disponendo di una maggiorazione dell'errore assoluto ϵ su di essi, risulta che:
>
> - se $|a - b| > \epsilon$ si può affermare con sicurezza che $a \neq b$
> - se $|a - b| \leq \epsilon$ non si può affermare con sicurezza né che $a = b$ né che $a \neq b$.

Per far convivere quindi i test ispirati ad un modello continuo con le approssimazioni introdotte dalla rappresentazione discreta è necessario renderli meno rigidi introducendo un concetto di *tolleranza*.

> Dati due numeri floating-point a e b si definisce
>
> $a = b$ se $|a - b| < tol$
> dove tol è una tolleranza assoluta

oppure

$a = b$ se $|a - b| < tol \cdot max(|a|, |b|)$
dove *tol* è una tolleranza relativa (vedi [Mic02])

Estendendo l'approccio ai test descritti in precedenza, questi vengono implementati verificando che

$$|f(x)| < tol \qquad |\tilde{v_1} \cdot \tilde{v_2}| < tol.$$

In questo modo si ottengono dei miglioramenti immediati, infatti settando la tolleranza alcuni ordini di grandezza maggiore della precisione di macchina disponibile (vedi paragrafo 1.2), gli algoritmi risultano, in molti casi, insensibili agli errori di arrotondamento.
In molti software vengono definite più tolleranze ϵ, una per le misure di lunghezza, una per le misure angolari, una per le misure di aree, etc. Questa differenziazione evidenzia una delle difficoltà d'implementazione di questo approccio: *definire a priori un valore per la tolleranza ϵ che porti ad un comportamento ottimale degli algoritmi.*
Quindi, non esistendo il valore ottimale per ogni situazione, si ricorre ad un insieme di valori definiti arbitrariamente, di solito in base ad una serie di test che verificano il comportamento del programma al variare di questi (processo di tuning).

Parte II

Funzioni Polinomiali e Interpolazione

Capitolo 2

Funzioni Polinomiali

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 trattazione è un'introduzione alla modellazione. La Fig. 2.1 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 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

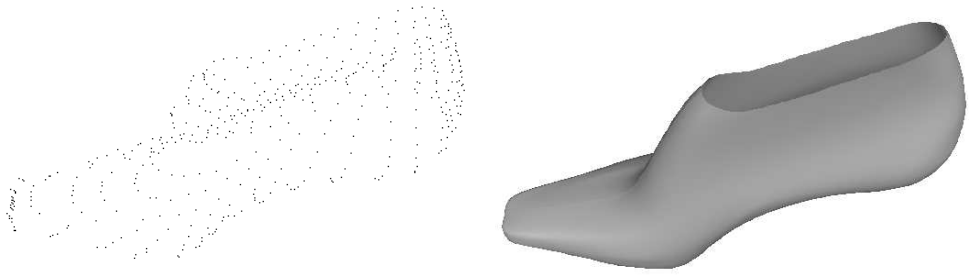

Figura 2.1: Interpolazione di punti 3D con una funzione vettoriale

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 2.1 (Polinomio) *Una funzione* $p : \mathbb{R} \to \mathbb{R}$ *definita da*

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i \tag{2.1}$$

dove n *è un intero non negativo e* $a_0, a_1, \ldots, a_n$ *sono numeri reali fissati è detta* funzione polinomiale. *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 detta funzione polinomiale nulla.*

Con $\mathbf{P}_n$ si denota l'insieme di tutte le funzioni polinomiali p con grado non superiore ad n, insieme alla funzione nulla. 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 2.1 (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 2.1 *Ogni equazione algebrica di grado n ha esattamente n radici reali o complesse, ciascuna con la sua molteplicità, cioè:*

$$p(x) = a_n(x - \alpha_1)^{m_1}(x - \alpha_2)^{m_2} \cdots (x - \alpha_k)^{m_k} \tag{2.2}$$

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 2.2 *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 $b(x)$.

2.1 Valutazione Numerica di un Polinomio

Assegnata una funzione polinomiale di grado n nella sua rappresentazione canonica (2.1), il metodo più immediato per la sua valutazione in corrispondenza di un assegnato valore $\bar{x}$ (`x_bar`) può essere come segue:

```
s=1
p=a(0)
for k=1,..,n
  s=s*x_bar
  p=p+s*a(k)
end
p(x_bar)=p
```

Il calcolo di $p(\bar{x})$ richiede $2n$ moltiplicazioni ed n addizioni/sottrazioni. Se si scrive il polinomio (2.1) nella seguente forma:

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots)) \tag{2.3}$$

si ricava differente metodo dovuto ad Horner:

```
p=a(n)
for k=n-1,..,0
  p=a(k)+x_bar*p
end
p(x_bar)=p
```

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

2.1.1 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 2.2 sulla divisione di due polinomi, nel caso particolare in cui il polinomio divisore sia il binomio $(x - \bar{x})$ per un assegnato valore reale $\bar{x}$, si ha che esistono unici i polinomi $q(x)$ ed $r(x)$ per cui

$$p(x) = q(x)(x - \bar{x}) + r(x)$$

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

Teorema 2.3 *Il resto della divisione del polinomio $p(x)$ per $(x - \bar{x})$ è $p(\bar{x})$.*

Dal teorema appena enunciato si deduce che un metodo per valutare una funzione polinomiale $p(x)$ in un punto $\bar{x}$ consiste nel determinare il resto della divisione fra $p(x)$ e il binomio $(x - \bar{x})$. 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
$\bar{x}$		$\bar{x}b_n$	$\bar{x}b_{n-1}$	...	$\bar{x}b_3$	$\bar{x}b_2$	$\bar{x}b_1$
	b_n	b_{n-1}	...	...	b_2	b_1	r

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

$$\begin{array}{lcl} b_n & \leftarrow & a_n \\ b_{n-1} & \leftarrow & a_{n-1} + \bar{x}b_n \\ b_{n-2} & \leftarrow & a_{n-2} + \bar{x}b_{n-1} \\ \vdots & \vdots & \vdots \\ b_1 & \leftarrow & a_1 + \bar{x}b_2 \\ r & \leftarrow & a_0 + \bar{x}b_1 \end{array}$$

dove

$$q(x) = b_1 + b_2 x + b_3 x^2 + \ldots b_n x^{n-1} = \sum_{i=0}^{n-1} b_{i+1} x^i.$$

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 2.1 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 anche manualmente in modo semplice;

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

2.1.2 Valutazione Numerica della Derivata

Si vuole ora procedere alla valutazione numerica della derivata di un polinomio $p(x)$ in un assegnato punto $\bar{x}$, cioè si vuole valutare $p'(\bar{x})$. La prima idea potrebbe essere quella di calcolare numericamente i coefficienti del polinomio derivato e valutarlo in $\bar{x}$ con Horner; questo equivale a determinare l'espressione $p'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \ldots n a_n x^{n-1}$ e a procedere nel valutare un polinomio di grado $n-1$. Nel caso in cui oltre al valore della derivata in $\bar{x}$ 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

$$p(x) = q(x)(x - \bar{x}) + p(\bar{x}),$$

derivando tale espressione si ha

$$p'(x) = q'(x)(x - \bar{x}) + q(x)$$

e valutandola in $\bar{x}$

$$p'(\bar{x}) = q(\bar{x})$$

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

Esempio 2.2 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 $p(\bar{x})$ e $p'(\bar{x})$ può essere organizzato nel seguente modo:

```
p1=0
p=a(n)
for k=n-1,...,0
  p1=p+x_bar*p1
  p=a(k)+x_bar*p
end
p(x_bar)=p
p'(x_bar)=p1
```

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

$$p''(x) = q''(x)(x - \bar{x}) + 2q'(x)$$

e valutando questa espressione in $\bar{x}$

$$p''(\bar{x}) = 2q'(\bar{x}).$$

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

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

allora

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

2.1.3 Errore algoritmico: un esempio famoso

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 canonica 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).

Esempio 2.3 *Si consideri il polinomio*

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

e lo si valuti 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, usando la precisione* **basic double**. *La Fig. 2.6 mostra il grafico della valutazione effettuata con il metodo di Horner ed il grafico corretto. La Fig.2.3 mostra il grafico della valutazione in* 257 *punti equispaziati nello stesso intervallo; ancora maggiormente si nota l'instabilità della valutazione in oggetto.*

2.1.4 Stima dell' errore algoritmico del metodo di Horner

Procedendo con un'analisi in avanti per determinare l'errore algoritmico del metodo di Horner per la valutazione di un polinomio $p(x)$ di grado n nella base canonica in un punto $\tilde{x}$, si ottiene:

$$\left| \frac{\tilde{p}(\tilde{x}) - p(\tilde{x})}{p(\tilde{x})} \right| \leq \frac{\gamma_{2n}}{|p(\tilde{x})|} \sum_{i=0}^{n} |a_i \tilde{x}^i|$$

con

$$\gamma_{2n} \leq \frac{2nu}{1 - 2nu} \leq 2.01nu.$$

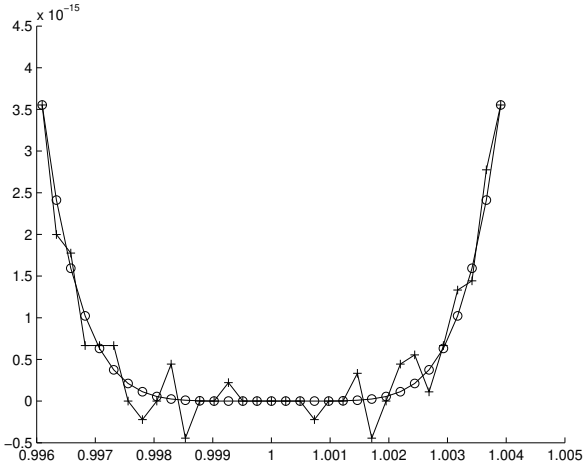

Figura 2.2: Grafico della valutazione effettuata in 33 punti con Horner (+) rispetto ad esatto (o).

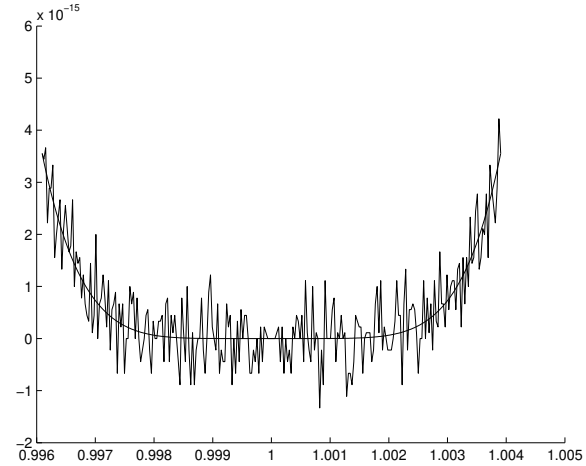

Figura 2.3: Grafico della valutazione effettuata in 257 punti con Horner rispetto ad esatto.

Chiaramente l'errore può essere grande, tuttavia se $a_i \geq 0$ per ogni i ed $\tilde{x} \geq 0$ o se $(-1)^i a_i \geq 0$ per ogni i e $\tilde{x} \leq 0$ sarà

$$\frac{\sum_{i=0}^{n} |a_i \tilde{x}^i|}{|p(\tilde{x})|} = 1.$$

2.2 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 2.4 Sia assegnato il polinomio

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

e lo si voglia valutare in punti $\bar{x} \in [100, 101]$. 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:

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

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. 2.4. 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 anche solo a_0, entrambi i coefficienti o valutando in altri punti dell'intervallo.

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 (1.11):

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \sum_i |c_i \epsilon_i|$$

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

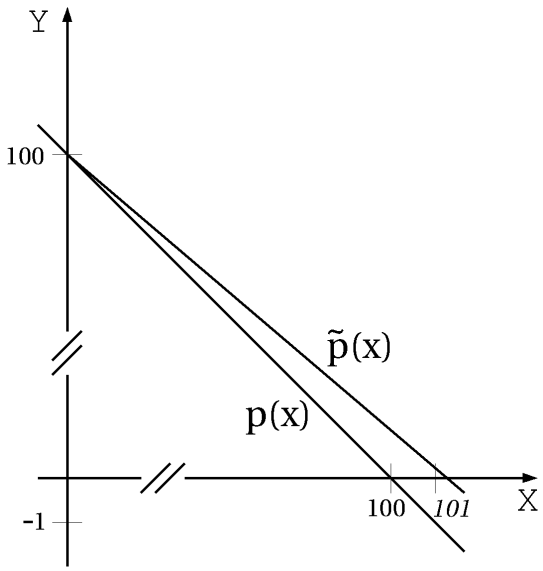

Figura 2.4: Amplificazione dell'errore

dove

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

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

$$\left| \frac{f(\tilde{a}_0, \tilde{a}_1, \tilde{x}) - f(a_0, a_1, x)}{f(a_0, a_1, x)} \right| \leq |c_0 \epsilon_0| + |c_1 \epsilon_1| + |c_x \epsilon_x|$$

$$= \left| \frac{a_0}{a_0 + a_1 x} \epsilon_0 \right| + \left| \frac{a_1 x}{a_0 + a_1 x} \epsilon_1 \right| + \left| \frac{x a_1}{a_0 + a_1 x} \epsilon_x \right|$$

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

$$= \left| \frac{100}{-1} \epsilon_0 \right| + \left| \frac{-101}{-1} \epsilon_1 \right| + \left| \frac{-101}{-1} \epsilon_x \right|$$

da qui si vede come una piccola perturbazione su uno dei coefficienti, per esempio a_1 dell'1% ($\epsilon_0 = 0$, $\epsilon_1 = 1/100$, $\epsilon_x = 0$) porti l'errore relativo ad assumere il valore $101 \frac{1}{100}$ e cioè 101 volte maggiore di quello sul dato 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_0 o $\epsilon_1 \neq 0$) viene grandemente amplificata (coefficienti c_0 e c_1). Si osservi inoltre che anche per $\epsilon_0 = \epsilon_1 = 0$ e $\epsilon_x \neq 0$ si avrà un errore inerente grande in questo intervallo di valutazione.

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

$$E_{in1} \simeq \left| \sum_{i=0}^{n} \frac{a_i}{p(x)} \frac{\partial p(x)}{\partial a_i} \epsilon_i \right| \leq \sum_{i=0}^{n} \left| \frac{a_i x^i}{p(x)} \epsilon_i \right| \quad \text{con} \quad \epsilon_i = \frac{\tilde{a}_i - a_i}{a_i}$$

e

$$E_{in2} \simeq \left| \frac{x}{p(x)} \frac{\partial p(x)}{\partial x} \epsilon_x \right| = \left| \frac{x p'(x)}{p(x)} \epsilon_x \right| \quad \text{con} \quad \epsilon_x = \frac{\tilde{x} - x}{x}.$$

Si può quindi desumere che:

- E_{in1} dipende da $p(x)$ e dai valori $a_i x^i$. Questi termini sono in parte controllabili riformulando il problema (la sua rappresentazione);
- E_{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 un grande 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.

2.2.1 Condizionamento di una base polinomiale

Sia $p(x) = \sum_{i=0}^{n} b_i \phi_i(x)$ $x \in [a, b]$, l'espressione di un generico polinomio nella base $\{\phi_i\}$. Se ripetiamo l'analisi fatta per determinare E_{in1}, cioè la parte dell'errore inerente che dipende dalla rappresentazione, nel caso che $p(x)$ sia rappresentato nella base $\{\phi_i\}$, avremo:

$$E_{in1} \simeq \left| \sum_{i=0}^{n} \frac{b_i \phi_i(x)}{p(x)} \epsilon_i \right| \quad \text{con} \quad \epsilon_i = \frac{\tilde{b}_i - b_i}{b_i}.$$

Se ripetiamo l'analisi fatta per determinare l'errore E_{in1} assoluto, anziché relativo, avremo:

$$E_{in1} \simeq \left| \sum_{i=0}^{n} b_i \phi_i(x) \epsilon_i \right| \quad \text{con} \quad \epsilon_i = \tilde{b}_i - b_i;$$

se, in quest'ultimo caso, assumiamo che gli errori assoluti sui coefficienti siano massimizzati dal valore ϵ, cioè:

$$|\epsilon_i| \leq |\epsilon| \quad i = 0, \ldots, n$$

avremo che

$$|\sum_{i=0}^{n} b_i \phi_i(x) \epsilon_i| \leq C_\Phi(p(x))|\epsilon| \quad \text{dove} \quad C_\Phi(p(x)) = \sum_{i=0}^{n} |b_i \phi_i(x)|.$$

$C_\Phi(p(x))$ è una stima del numero di condizione assoluto per la valutazione del polinomio $p(x)$ nella base $\Phi = \{\phi_0, \ldots, \phi_n\}$, in ciascun punto x.
Una base Φ è detta essere *non negativa* sull'intevallo $[a, b]$, se

$$\phi_i(x) \geq 0 \quad \text{per} \quad x \in [a, b] \quad i = 0, \ldots, n.$$

Una base non negativa permette una computazione in genere migliore dal punto di vista numerico.

2.3 Polinomi nella base di Bernstein

Con funzione polinomiale 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 i $B_{i,n}(x)$ sono i polinomi base di Bernstein definiti 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 $b_0, b_1, \ldots, b_n \in \mathbb{R}$ sono i coefficienti nella base di Bernstein.

2.3.1 Proprietà dei polinomi di Bernstein

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

- 1. $B_{i,n}(x) \geq 0 \quad i = 0, \ldots, n \quad \forall x \in [a, b]$;
- 2. Partizione dell'unità:

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

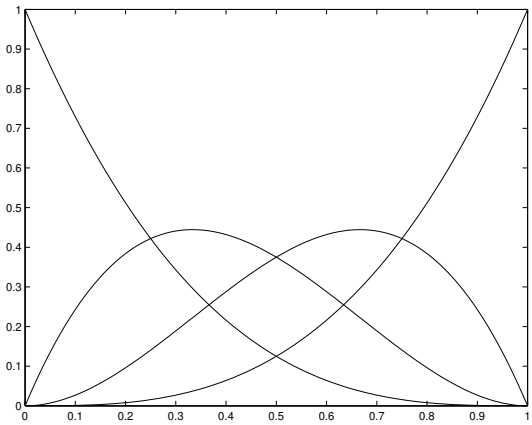

Figura 2.5: Polinomi base di Bernstein di grado 3.

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

$$\min_i\{b_i\} \leq p(x) \leq \max_i\{b_i\} \quad \forall x \in [a,b].$$

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

- 4. I polinomi base $B_{i,n}(x)$ per $i = 0, \ldots, n$ hanno uno zero in a con molteplicità[2] i e uno zero in b con molteplicità $n - i$.

- 5. 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 (i coefficienti nulli non vengono considerati).

2.3.2 Errore Inerente per la valutazione di polinomi di Bernstein

Riprendiamo l'esempio 2.4 dove ora si considera il polinomio nella base di Bernstein; sarà:

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

[2]Una funzione $f(x)$ regolare si dice che ha in $\bar{x}$ uno zero di molteplicità k se $f^{(j)}(\bar{x}) = 0$ per $j = 0, \ldots, k-1$ e $f^{(k)}(\bar{x}) \neq 0$.

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:

$$\left|\frac{f(\tilde{x}) - f(x)}{f(x)}\right| \leq |c_0\epsilon_0| + |c_1\epsilon_1| + |c_x\epsilon_x|$$

$$= \left|\frac{b_0}{p(x)}(101 - x)\epsilon_0\right| + \left|\frac{b_1}{p(x)}(x - 100)\epsilon_1\right| + \left|\frac{xp'(x)}{p(x)}\epsilon_x\right|$$

e perturbando solo b_1 (cioè $\epsilon_0 = \epsilon_x = 0$, ma $\epsilon_1 \neq 0$), valutando in un qualunque punto in $[100, 101]$ il coefficiente moltiplicatore di ϵ_1 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.

Generalizzando l'analisi fatta in questo esempio alla valutazione di un generico polinomio $p(x)$ nella base di Bernstein, la prima parte dell'errore inerente si può scrivere come:

$$E_{in1} \simeq \left|\sum_{i=0}^{n} \frac{b_i}{p(x)} \frac{\partial p(x)}{\partial a_i}\epsilon_i\right| \leq \sum_{i=0}^{n} \left|\frac{b_i B_{i,n}(x)}{p(x)}\epsilon_i\right| \quad \text{con} \quad \epsilon_i = \frac{\tilde{b}_i - b_i}{b_i}$$

che ci permette di affermare che i fattori di amplificazione degli errori ϵ_i sopra dati, possono essere controllati; infatti per le proprietà dei polinomi di Bernstein se i b_i sono tutti dello stesso segno sarà

$$\frac{b_i B_{i,n}(x)}{p(x)} < 1 \quad \forall i.$$

Per quel che riguarda la seconda componente dell'errore inerente, come già fatto notare anche nell'analisi precedente, se $\epsilon_x \neq 0$, $\frac{xp'(x)}{p(x)}$ 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 una applicazione (mapping) dal primo al secondo e viceversa

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

$$t = \frac{x - a}{b - a} \quad \text{o viceversa} \quad x = a + t(b - a) \tag{2.4}$$

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.

$$B_{i,n}(x) = \binom{n}{i}\left(\frac{b - x}{b - a}\right)^{n-i}\left(\frac{x - a}{b - a}\right)^{i}$$

$$= \binom{n}{i}(1 - t)^{n-i}t^{i} = B_{i,n}(t) \tag{2.5}$$

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 in quanto restano uguali.

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

$$E_{in} \leq \left| \frac{b_0}{p(t)}(1-t)\epsilon_0 \right| + \left| \frac{b_1}{p(t)}(t-0)\epsilon_1 \right| + \left| \frac{tp'(t)}{p(t)}\epsilon_x \right|$$

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 2.3.6).
Si noti che in $[0, 1]$ i polinomi di Bernstein hanno una definizione più semplice. Riassumendo, se si vuole valutare $p(\bar{x})$, si determina $\bar{t} := \frac{\bar{x}-a}{(b-a)}$, si calcola $p(\bar{t})$ con $p(t)$ polinomio di Bernstein in $[0, 1]$, e sarà $p(\bar{x}) \equiv p(\bar{t})$.
Da ora in poi useremo sempre i polinomi di Bernstein nell'intervallo $[0, 1]$.

2.3.3 Errore algoritmico: riprendiamo l'esempio famoso

Riprendiamo l'esempio precedentemente visto e notiamo che il polinomio considerato ammette una rappresentazione esatta nella base di Bernstein (i coefficienti nella base canonica che sono valori esattamente rappresentabili come numeri finiti portano ai coefficienti nella base di Bersntein che sono ancora valori esattamente rappresentabili). Si può allora procedere alla valutazione dello stesso polinomio, ma utilizzando la sua nuova espressione e quindi confrontare quanto ottenuto con la valutazione precedente.

Esempio 2.5 *Si riconsidera il polinomio*

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

che ammette la seguente rappresentazione nella base di Bernstein:

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

e lo si valuti 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, usando la precisione* **basic double**. *La Fig. 2.6 mostra i grafici delle valutazioni effettuate con la rappresentazione canonica e quella di Bernstein ed evidenzia che la nuova valutazione ricalca il grafico esatto.*

Osservazione 2.1 *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 (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.*

Teorema 2.4 *Siano* $\{\psi_i\}_{i=0,\dots,n}$ *e* $\{\phi_i\}_{i=0,\dots,n}$ *due basi polinomiali non negative di grado n su* $[a,b]$ *tali che la prima possa essere espressa come combinazione non negativa della seconda, cioè*

$$\psi_j(x) = \sum_{k=0}^{n} M_{jk}\phi_k(x)$$

dove $M_{jk} \geq 0$. *Allora i numeri di condizione delle due basi soddisfano:*

$$C_\Phi(p(x)) \leq C_\Psi(p(x)) \quad \forall x.$$

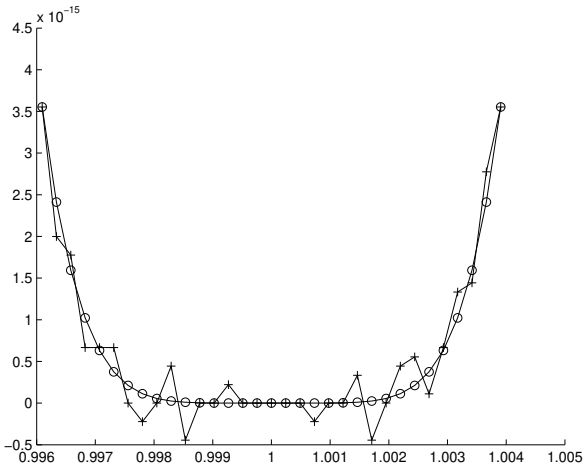

Figura 2.6: Grafici delle valutazioni effettuate in 33 punti con Bernstein (o) rispetto ad Horner (+).

Dim.
⊙

Osservazione 2.2 *Un esempio importante del Teorema 2.4 è il caso dove* Φ *è la base di Bernstein in* $[0,1]$ *e* Ψ *è la base canonica* $\{1, x, \ldots, x^n\}$.

Corollario 2.2 *Siano* $C_B(p(x))$ *e* $C_P(p(x))$ *i numeri di condizione per la base di Bernstein e la base canonica, allora*

$$C_B(p(x)) \leq C_P(p(x)).$$

Corollario 2.3 *I numeri di condizione* $\tilde{C}_B(p(x))$ *e* $C_B(p(x))$ *per le basi di Bernstein di grado* $n+r$ *ed* n *di un polinomio* $p(x)$ *di grado esattamente* n *soddisfano*

$$\tilde{C}_B(p(x)) \leq C_B(p(x)).$$

Corollario 2.4 *Sia* $[a,b] \subset [0,1]$. *Allora i numeri di condizione* $\bar{C}_B(p(x))$ *e* $C_B(p(x))$ *per polinomi* $p(x)$ *di grado* n *nella base di Bernstein rispettivamente in* $[a,b]$ *e* $[0,1]$ *soddisfano*

$$\bar{C}_B(p(x)) \leq C_B(p(x)).$$

2.3.4 Formula ricorrente base di Bernstein

Risultato 2.1 *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* $B_{i,n}(t) = 0 \quad \forall i \notin [0, n]$.

Dim.

Nella formula sostituiamo le $B_{i-1,n-1}$ e $B_{i,n-1}$ con la definizione 2.5

$$t\binom{n-1}{i-1}(1-t)^{n-i}t^{i-1} + (1-t)\binom{n-1}{i}(1-t)^{n-1-i}t^i$$

$$= \binom{n-1}{i-1}(1-t)^{n-i}t^i + \binom{n-1}{i}(1-t)^{n-i}t^i$$

$$= \left\{\binom{n-1}{i-1} + \binom{n-1}{i}\right\}(1-t)^{n-i}t^i$$

e poiché vale

$$\binom{n-1}{i-1} + \binom{n-1}{i} = \binom{n}{i}$$

resta verificata la formula data. ⊙

Viene riportato il codice Matlab che implementa la formula ricorrente dei 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 2.3 *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 2.6 *Polinomi base di Bernstein di grado 3 ottenuti mediante formula ricorrente:*

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

2.3.5 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

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

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:

$$b_i^{[k]} = t b_{i+1}^{[k-1]} + (1-t) b_i^{[k-1]} \tag{2.6}$$

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:

$$\begin{array}{lllll} b_0^{[0]} & b_1^{[0]} & b_2^{[0]} & \ldots & b_{n-1}^{[0]} \quad b_n^{[0]} \\ b_0^{[1]} & b_1^{[1]} & b_2^{[1]} & \ldots & b_{n-1}^{[1]} \\ \ldots & & & \ldots & \\ b_0^{[n-2]} & b_1^{[n-2]} & b_2^{[n-2]} & & \\ b_0^{[n-1]} & b_1^{[n-1]} & & & \\ b_0^{[n]} & & & & \end{array} \tag{2.7}$$

Osservazione 2.4 *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=1:np1-j
     w(i)=d1.*w(i+1)+d2.*w(i);
   end
 end
y(k)=w(1);
end
```

Osservazione 2.5 *Il valore di un polinomio di Bernstein agli estremi è banalmente dato dai suoi coefficienti b_0 e b_n, cioè:*

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

2.3.6 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:

$$b_0^{[0]}, b_0^{[1]}, \ldots, b_0^{[n]} \quad \text{e} \quad b_0^{[n]}, b_1^{[n-1]}, \ldots, b_n^{[0]} \tag{2.8}$$

che corrispondono ai lati verticale e diagonale dello schema triangolare (2.7). 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 (2.6) 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 (2.8) possono allora essere scritti come:

$$b_0^{[k]} = \sum_{j=0}^{k} b_j B_{j,k}(t_c) \quad \text{e} \quad b_k^{[n-k]} = \sum_{j=0}^{n-k} b_{k+j} B_{j,n-k}(t_c) \quad k = 0, \ldots, n. \tag{2.9}$$

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 2.5 *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

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

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 (2.5) 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 (2.9), 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 (2.9); analogamente per la seconda. $\odot$

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.

2.3.7 Derivata

Risultato 2.2 *Vale la seguente importante relazione:*

$$B'_{i,n}(t) = n\left[B_{i-1,n-1}(t) - B_{i,n-1}(t)\right]. \tag{2.10}$$

Dim.

Usando la definizione si ha:

$$\frac{d}{dt}\binom{n}{i}(1-t)^{n-i}t^i = -\binom{n}{i}(n-i)(1-t)^{n-i-1}t^i + \binom{n}{i}i(1-t)^{n-i}t^{i-1}$$

$$= -\frac{n!}{i!(n-i)!}(n-i)(1-t)^{n-i-1}t^i + \frac{n!}{i!(n-i)!}i(1-t)^{n-i}t^{i-1}$$

$$= -n\frac{(n-1)!}{i!(n-i-1)!}(1-t)^{n-i-1}t^i + n\frac{(n-1)!}{(i-1)!(n-i)!}(1-t)^{n-i}t^{i-1}$$

$$= -n\binom{n-1}{i}(1-t)^{n-1-i}t^i + n\binom{n-1}{i-1}(1-t)^{n-i}t^{i-1}$$

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

$\odot$

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

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

e sostituendo la (2.10) si ottiene

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

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

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

Suddivisione

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

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

$$p'(0) := d_0 = n(b_1 - b_0) \quad e \quad p'(1) := d_{n-1} = n(b_n - b_{n-1}).$$

Osservazione 2.7 *La derivata di un polinomio di Bernstein di grado n e definito su $[a, b]$ è data da:*

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

metodo 2 bisogna fare cambio di base

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

Riapplicando la formula ricorrente per le derivate si possono ottenere le espressioni delle funzioni derivate di ordini successivi, ossia le

derivata grado j

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

I loro coefficienti sono dati da:

$$b_i^{(j)} = (n - j + 1)(b_{i+1}^{(j-1)} - b_i^{(j-1)}), \quad j = 1, \ldots, n \quad i = 0, \ldots, n-j.$$

Nel caso il polinomio nella base di Bernstein sia definito in $[a, b]$, i coefficienti obbediscono alla formula:

$$b_i^{(j)} = (n - j + 1) \frac{b_{i+1}^{(j-1)} - b_i^{(j-1)}}{b-a}, \quad j = 1, \ldots, n.$$

Se poi si vogliono le derivate di ogni ordine negli estremi dell'intervallo di definizione, si avrà:

$$p^{(j)}(0) = b_0^{(j)}, \quad p^{(j)}(1) = b_{n-j}^{(j)}, \quad j = 1, \ldots, n.$$

Osservazione 2.8 *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 $p'(\bar{t}) = n(b_0^{[n]} - b_0^{[n-1]})/\bar{t} \equiv n(b_1^{[n-1]} - b_0^{[n]})/(1-\bar{t})$; 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
```

2.3.8 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 2.9 *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 2.10 *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:

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

2.3.9 La base di Legendre

Dato un polinomio $p(t) \in \mathbf{P}_n$ per $t \in [0,1]$ è spesso utile poterlo rappresentare in una *base ortogonale*,

$$p(t) = \sum_{i=0}^{n} a_i \phi_i(x),$$

caratterizzata dalla proprietà

$$\int_0^1 \phi_j(t)\phi_i(t)dt = \begin{cases} \beta_i & \text{se} \quad j = i \\ 0 & \text{se} \quad j \neq i, \end{cases}$$

La base di Bernstein vista non è chiaramente ortogonale, ma è strettamente legata ai polinomi di Legendre, una famiglia di polinomi ortogonali. L'interesse per tale legame è dato dalla semplice forma dei polinomi di Legendre nella base di Bernstein e dalla stabilità della trasformazione tra le due forme.
Per enfatizzare la simmetria, solitamente la base di Legendre è definita in $[-1, 1]$. Per esprimerla nella base di Bernstein è più conveniente usare l'intervallo $[0, 1]$. I polinomi di Legendre $L_i(t)$ su $[0, 1]$ possono essere generati mediante la seguente relazione ricorrente

$$(i+1)L_{i+1}(t) = (2i+1)(2t-1)L_i(t) - iL_{i-1}(t),$$

per $i = 1, 2, \ldots$, dove $L_0(t) = 1$ e $L_1(t) = 2t - 1$.

Lemma 2.1 *I polinomi di Legendre $L_k(t)$ possono essere espressi nella base di Bernstein $\{B_{0,k}, \ldots, B_{k,k}\}$ di grado k come*

$$L_k(t) = \sum_{i=0}^{k} (-1)^{k+i} \binom{k}{i} B_{i,k}(t).$$

Capitolo 3

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).

3.1 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 $f(x)$ equispaziati 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.

3.2 Approssimazione VD

Definizione 3.1 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 i coefficienti nulli.*

Esempio 3.1 *Sia* $\mathbf{c} = (-1, 3.2, 1.5, -3.7, 4)$ *allora* $V^-[\mathbf{c}] = 3$.

Definizione 3.2 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

$$V^-[\phi^*(x) - (a_0 + a_1 x)] \leq V^-[f(x) - (a_0 + a_1 x)] \quad x \in [a, b] \tag{3.1}$$

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

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

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

Teorema 3.2 *Sia $f(x) : [a,b] \rightarrow \mathbf{R}$ almeno continua, e si consideri l'approssimazione uniforme $B_n[f(x)]$ definita nella sezione 3.1; 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 3.1.
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 3.2 di $V^-[f(x)]$ per cui sarà

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

verificando così la seconda proprietà. $\odot$

La Fig.3.1 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 (3.1), la funzione data.

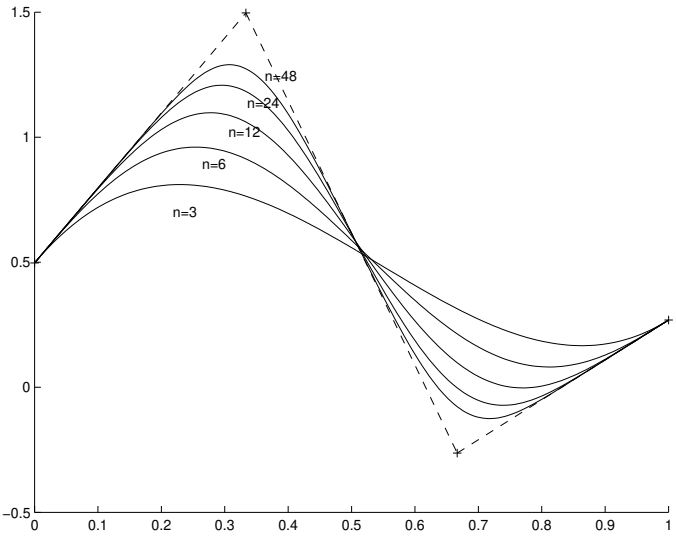

Figura 3.1: Approssimazione polinomiale VD di una funzione lineare a tratti.

3.3 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 (vedi sezione successiva su 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 3.1 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 3.1

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

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 3.1 *Dalla Fig.3.1 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 $[a, b]$.*

3.4 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 $P_i \in \mathbf{E}^2$ `(piano Euclideo)` $i = 0, \ldots, n$ 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. 3.2 a sinistra mostra una tale curva e i punti di controllo associati P_0, P_1, P_2 e P_3. Si può vedere

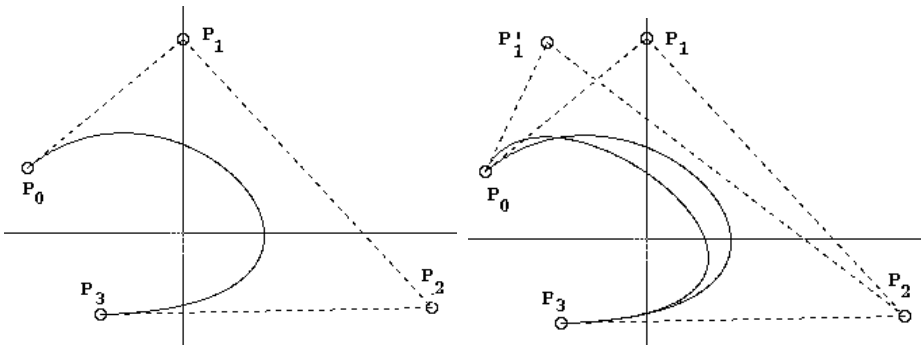

Figura 3.2: Esempio di curva di Bézier; caso $n = 3$ a sinistra; modifica di forma mediante spostamento di un punto di controllo a destra.

semplicemente che i punti estremi della curva coincidono con P_0 e P_3 e che i punti di controllo centrali non sono in genere punti della curva.

Spostando i punti di controllo si influenza la forma della curva; la Fig. 3.2 a destra mostra l'effetto di spostare il punto P_1 in P_1'.
E' facile pensare ad un software interattivo che permetta ad un utente di muovere i punti 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 oggi utilizzate in molti sistemi CAD 2D (Computer Aided Design), CAD 3D, sistemi di disegno, software e linguaggi grafici come Adobe Illustrator, Postscript, OpenGL, METAFONT, LaTex, Xfig, gimp, inkscape e tanti altri. Le curve di Bézier sono poi uno standard per la rappresentazione di fonti di caratteri, di modelli CAD in genere e sono alla base dello standard SVG (Scalable Vector Graphics) per la grafica vettoriale in alternativa alla grafica raster.

Una curva di Bézier è una curva in forma parametrica e le sue componenti sono polinomi definiti nell'intero intervallo parametrico $[a, b]$ di definizione della curva. Questo comporta un limite per la modellazione geometrica per vari motivi:

- per descrivere forme complesse occorrono molti punti di controllo e questo impone un alto grado n della curva;
- curve di grado elevato sono inefficienti nella valutazione e numericamente instabili;
- una modifica ad un punto di controllo influenza globalmente la forma dell'intera curva.

La soluzione consiste nell'utilizzare curve polinomiali a tratti, o meglio considerare più curve di Bézier.

Consideriamo $k+1$ curve di Bézier $C_i(u), i = 0, \ldots, k$, definite sulla seguente partizione dell'intervallo $[a, b]$

$$a \equiv u_0 < u_1 < u_2 < \ldots < u_{k+1} \equiv b$$

I valori u_i sono detti nodi e suddividono l'intervallo parametrico $[a, b]$ in $k + 1$ sottointervalli $I_i = [u_i, u_{i+1}], i = 0, \ldots, k$, ciascuno corrispondente ad un tratto di curva.

Chiameremo $C(u)$, la curva di Bézier complessiva definita a tratti dalle $C_i(u)$, cioè

$$C(u) := C_i(u) \qquad u \in I_i \quad i = 0, \ldots, k.$$

Ad ogni valore $\bar{u} \in [a, b]$ del parametro corrisponde un punto $C(\bar{u})$ sulla curva a tratti e per la precisione se $\bar{u} \in I_i$ per un certo indice i, allora $C(\bar{u}) := C_i(\bar{u})$.

Per semplicità di gestione e computazione, ogni tratto è conveniente che sia parametrizzato in $[0,1]$, cioè $C_i(t), \quad t \in [0,1], \quad i = 0, \ldots, k$.

Se $u \in I_i$ allora il corrispondente $t \in [0,1]$ parametro locale della curva i-esima è dato dalla trasformazione di coordinate:

$$t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{h_i}$$

dove $h_i = u_{i+1} - u_i$ è l'ampiezza dell'i-esimo intervallo.

L'introduzione delle coordinate parametriche locali comporta che la derivata prima della curva $C(t)$ sia data da:

$$\frac{d}{du}C(u) = \frac{d}{dt}C_i(t)\frac{d}{du}t(u). \tag{3.2}$$

Infatti, poiché t è funzione di u si applica la regola di composizione delle derivate.

Vediamo come si possono unire fra loro più tratti di curva di Bézier rispettando una certa regolarità (continuità) nei punti di raccordo.

I segmenti possono essere raccordati nei nodi con un certo grado di continuità, non necessariamente lo stesso in ogni nodo. $C(u)$ è detto essere C^k continua nel nodo u_i se

$$C_i^{(j)}(u_i) = C_{i+1}^{(j)}(u_i), \quad 0 \leq j \leq k, \quad k \leq n-1$$

indicando con $C_i^{(j)}$ la derivata di ordine j di $C_i(u)$.

Supponiamo di avere due curve di Bézier $C_1(u)$ e $C_2(u)$, di grado n, definite rispettivamente in $[u_0, u_1]$ e in $[u_1, u_2]$ con punti di controllo $P_0^1, \ldots, P_n^1$ e $P_0^2, \ldots, P_n^2$.

Si può pensare a queste due curve come a due entità separate o come due segmenti di una unica curva a tratti $C(u)$ definita nell'intervallo $u \in [u_0, u_2]$. Si possono presentare i seguenti casi.

- **Continuità C^0 nel nodo u_1.** Deve essere $P_n^1 = P_0^2$ ossia l'ultimo punto di controllo di C_1 e il primo di C_2 devono coincidere.

- **Continuità C^1 nel nodo u_1.** I due tratti devono essere C^0 ed inoltre si deve imporre che

  $$\frac{d}{du}C_1(u_1) = \frac{d}{du}C_2(u_1) \tag{3.3}$$

  da cui si ottiene

  $$\frac{P_n^1 - P_{n-1}^1}{h_1} = \frac{P_1^2 - P_0^2}{h_2}$$

  con $P_n^1 = P_0^2$, $h_1 = u_1 - u_0$ e $h_2 = u_2 - u_1$.

  Questa è condizione necessaria e sufficiente per avere continuità C^1 nel

nodo di raccordo u_1. *Significa che i tre punti* $P^1_{n-1}, P^1_n = P^2_0, P^2_1$ *devono essere collineari e nel rapporto* $\frac{h_1}{h_2}$.

- **Continuità** G^1 **nel nodo** u_1**.** *Se le due curve hanno nel punto di raccordo vettori tangente di stessa direzione e verso allora* $C(u)$ *è* G^1 *continua*.
  I due tratti devono essere C^0 ed inoltre si deve imporre

$$P^1_n - P^1_{n-1} = \beta_1(P^2_1 - P^2_0), \text{ con } P^1_n = P^2_0 \quad \text{e} \qquad \beta_1 > 0.$$

  La sola condizione di collinearità dei tre punti di controllo distinti garantisce una variazione continua della tangente alla curva. Visivamente la curva preserva la sua caratteristica di curva regolare nel punto di raccordo, tuttavia il generico parametro β_1 fa perdere la condizione C^1. Attenzione se i tre punti sono collineari, i vettori tangenti hanno la stessa direzione nel punto di raccordo, entrambe le curve presentano la stessa retta tangente, ma non é detto che siano G^1 continue.

- **Continuità** C^2 **nel nodo** u_1**.** Si assume che i due tratti siano C^1 e si deve imporre

$$\frac{d^2}{du^2}C_1(u_1) = \frac{d^2}{du^2}C_2(u_1) \tag{3.4}$$

  da cui si ottiene

$$\frac{P^1_n - 2P^1_{n-1} + P^1_{n-2}}{h_1^2} = \frac{P^2_2 - 2P^2_1 + P^2_0}{h_2^2}.$$

- **Continuità** G^2 **nel nodo** u_1**.** Si assume che i due tratti siano G^1 e si deve imporre

$$P^1_n - 2P^1_{n-1} + P^1_{n-2} = \beta_1^2(P^2_2 - 2P^2_1 + P^2_0) + \beta_2(P^2_1 - P^2_0)$$

  con $\beta_1 > 0$ e $\beta_2 \geq 0$. Le variabili β_1 e β_2 sono parametri liberi e posso essere usati come *parametri di forma*.

Un limite evidente delle curve di Bézier a tratti è dato dal fatto che per poter congiungere più segmenti di curve di Bézier polinomiali occorre gestire 'manualmente' i raccordi.

Una spline è una funzione costituita da un insieme di polinomi raccordati in modo 'automatico' con una certa continuità tra loro.

Nelle applicazioni spesso si preferiscono le curve spline alle curve di Bézier a tratti soprattuto nella fase di modellazione.

Capitolo 4

Interpolazione

4.1 Interpolazione di Lagrange

Siano assegnate $n+1$ osservazioni in corispondenza di $n+1$ punti distinti, cioè siano assegnati (x_i, y_i) $i = 0, \ldots, n$. Si consideri lo spazio $\mathbf{P}_n$ dei polinomi a coefficienti reali e una sua base $\phi_0, \phi_1, \ldots, \phi_n$ così che ogni elemento $p \in \mathbf{P}_n$ sia definito da una loro combinazione lineare, cioè

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

Allora il problema di interpolare i dati assegnati con una funzione polinomia $p \in \mathbf{P}_n$ consiste nel determinare la funzione polinomiale p^* (od anche i coefficienti $a_0^*, a_1^*, \ldots, a_n^*$ che la definiscono) tale che

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

Spesso, in pratica, questo problema viene applicato per determinare una approssimazione polinomiale su un intervallo di una data funzione non polinomiale; si considera una funzione continua $f(x)$, $x \in [a, b]$, la si campiona in $n+1$ punti dell'intervallo $[a, b]$ e tramite interpolazione polinomiale la si vuole approssimare in modo che

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

con ϵ una costante sufficientemente piccola legata all'applicazione. $R(x)$ viene detta errore di interpolazione. L'interpolazione polinomiale garantisce sempre l'esistenza e l'unicità della soluzione e, sotto opportune condizioni, una buona ricostruzione/approssimazione.

4.1.1 Forma Canonica

Teorema 4.1 *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 $p(x)$ in forma canonica

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

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:

$$V\mathbf{a} = \mathbf{y} \tag{4.1}$$

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. Questo prova 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 (4.1); 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.

4.1.2 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 (2.4) 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 con cui vogliamo interpolare 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 $O(n^3)$ operazioni floating point, ma la matrice in oggetto è totalmente positiva e per tali matrici esistono metodi di fattorizzazione $O(n^2)$.

Si noti inoltre 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. 4.1.2 in cui vengono mostrati gli indici di condizionamento delle matrici calcolate su punti equispaziati nell'intervallo di definizione mediante polinomi di Bernstein e base delle potenze.

4.1.3 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 sia dal punto di vista del condizionamento, ma anche che permetta di usare un

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 4.1: Indici di condizionamento delle matrici ottenute valutando i polinomi di Bernstein e la base delle potenze in punti equispaziati.

metodo più efficiente oltre che stabile. Nel caso dell'interpolazione polinomiale sicuramente una delle basi più interessanti è quella di Newton.

Assegnati i punti (x_i, y_i), $i = 0, \ldots, n$, con x_i distinti, si voglia determinare il polinomio interpolante nella base di Newton, cioè si voglia determinare il polinomio $p(x)$ nella forma

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

Le funzioni

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

costruite sui punti x_i, $i = 0, \ldots, n$ distinti, formano una base polinomiale per lo spazio $\mathbf{P}_n$ detta base di Newton.

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 $O(n^2)$ operazioni aritmetiche.

Forma di Newton e differenze divise

I coefficienti c_i della forma di Newton si possono anche determinare facendo uso del concetto di differenza divisa e di una semplice formula ricorrente per il loro calcolo. Infatti vale:

$$c_i = f[x_0, x_1, \ldots, x_i], \quad i = 0, \ldots, n,$$

dove con la notazione $f[x_0, x_1, \ldots, x_i]$ si intende la **differenza divisa** di ordine i della funzione continua f rispetto ai punti distinti $x_j, \quad j = 0, \ldots, i$[1]

Teorema 4.2 *Sia $p_k(x)$ il polinomio di interpolazione in $\mathbf{P}_k$ che soddisfa le condizioni $p_k(x_i) = f(x_i) \quad i = 0, \ldots, k$, allora la funzione*

$$p_{k+1}(x) = p_k(x) + (x - x_0)(x - x_1) \cdots (x - x_k) f[x_0, x_1, \ldots, x_{k+1}]$$

è il polinomio in $\mathbf{P}_{k+1}$ che soddisfa le codizioni $p_{k+1}(x_i) = f(x_i) \quad i = 0, \ldots, k+1$.

Dim.

Sia $q \in \mathbf{P}_{k+1}$ di interpolazione di $(x_i, f(x_i), \quad i = 0, \ldots, k+1$ e proviamo che $q \equiv p_{k+1}$. Per come è definito p_{k+1} si verifica banalmente che

$$q(x_i) - p_{k+1}(x_i) = 0 \quad i = 0, \ldots, k;$$

diremo che $q - p_{k+1} = 0$ ha $k+1$ radici nei punti $x_i, \quad i = 0, \ldots, k$. Ma $q - p_{k+1} \in \mathbf{P}_k$, infatti q e p_{k+1} hanno lo stesso coefficiente di x^{k+1} e nella differenza il grado si abbassa, segue che $q - p_{k+1}$ essendo un polinomio di grado k con $k+1$ radici deve essere il polinomio nullo. $\odot$

Formula ricorrente per le differenze divise

Teorema 4.3 *La differenza divisa $f[x_j, x_{j+1}, \ldots, x_{j+k+1}]$ di ordine $k+1$ è legata alle differenze divise $f[x_j, x_{j+1}, \ldots, x_{j+k}]$ e $f[x_{j+1}, x_{j+2}, \ldots, x_{j+k+1}]$ di ordine k dalla relazione:*

$$f[x_j, x_{j+1}, \ldots, x_{j+k+1}] = \frac{f[x_{j+1}, x_{j+2}, \ldots, x_{j+k+1}] - f[x_j, x_{j+1}, \ldots, x_{j+k}]}{x_{j+k+1} - x_j}.$$

[1] la differenza divisa $f[x_0, x_1, \ldots, x_i]$ è definita come il coefficiente della potenza massima x^i del polinomio $p \in \mathbf{P}_i$ che soddisfa le condizioni di interpolazione $p(x_j) = f(x_j), \quad j = 0, \ldots, i$.

Dim.

Sia $p_k \in \mathbf{P}_k$ tale che $p_k(x_i) = f(x_i), \quad i = j, \ldots, j+k$; questo polinomio avrà come coefficiente della potenza massima $f[x_j, x_{j+1}, \ldots, x_{j+k}]$. Sia $q_k \in \mathbf{P}_k$ tale che $q_k(x_i) = f(x_i), \quad i = j+1, \ldots, j+k+1$; questo polinomio avrà come coefficiente della potenza massima $f[x_{j+1}, x_{j+2}, \ldots, x_{j+k+1}]$. Si verifica facilmente che

$$p_{k+1}(x) = \frac{(x - x_j)q_k(x) + (x_{j+k+1} - x)p_k(x)}{x_{j+k+1} - x_j} \in \mathbf{P}_{k+1}$$

e $p_{k+1}(x_i) = f(x_i), \quad i = j, \ldots, j+k+1$; segue che il coefficiente di x^{k+1} di p_{k+1} è $f[x_j, x_{j+1}, \ldots, x_{j+k+1}]$ e quindi resta verificata la formula data. $\odot$

Utilizzando tale formula i coefficienti c_i possono essere calcolati con il seguente schema triangolare:

$$\begin{array}{llll} f[x_0] & & & \\ f[x_1] & f[x_0, x_1] & & \\ f[x_2] & f[x_1, x_2] & f[x_0, x_1, x_2] & \\ \vdots & \vdots & \vdots & \\ f[x_n] & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \cdots \quad f[x_0, x_1, \ldots, x_n] \end{array}$$

che può essere codificato semplicemente come:

```
function c=diffdiv(x,y)
% calcola le differenze divise relative agli argomenti
% x1, x2, ..., xn e ai valori y1, y2, ..., yn
% x,y --> coordinate dei punti
% c <-- valori delle differenze divise
c=y;
n=length(x);
for i=2:n
  for j=n:-1:i
   c(j)=(c(j)-c(j-1))./(x(j)-x(j-i+1));
  end
end
```

La soluzione al problema dell'interpolazione nella forma di Newton risulta ottimale, inoltre è possibile applicare una variante dell'algoritmo di Ruffini-Horner per la valutazione e la valutazione della derivata. Purtroppo se il polinomio così determinato deve essere sottoposto ad ulteriori elaborazioni come per esempio all'integrazione, allora tale forma non è più raccomandabile in quanto non sono noti algoritmi efficienti per procedere.

4.1.4 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 $L_{i,n}(x)$ di grado n che soddisfano le seguenti condizioni:

$$L_{i,n}(x_j) = \begin{cases} 1 & \text{se} \quad i = j \\ 0 & \text{se} \quad i \neq j \end{cases} \tag{4.2}$$

Questa base risulta la più semplice in cui risolvere un problema di interpolazione.

Assegnati i punti (x_i, y_i), $i = 0, \ldots, n$, con x_i distinti, il problema è risolto banalmente da

$$p(x) = \sum_{i=0}^{n} y_i L_{i,n}(x); \tag{4.3}$$

infatti per le condizioni (4.2) che definiscono i polinomi $L_{i,n}(x)$, banalmente in ogni punto x_j l'espressione polinomiale (4.3) varrà y_j per ogni $j = 0, \ldots, n$.
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. Al contrario la valutazione del polinomio interpolante comporta la determinazione e la valutazione dei polinomi $L_{i,n}(x)$.

Vediamo come possono essere espressi i polinomi base di Lagrange. Per il Teorema (2.2) i polinomi $L_{i,n}(x)$ avendo come radici i punti x_j con $j = 0, \ldots, n$ e $j \neq i$ saranno della forma

$$L_{i,n}(x) = w_i(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)$$

con w_i una costante da determinare in modo che $L_{i,n}(x_i) = 1$; allora

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

da cui in definitiva

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

La valutazione di questo polinomio interpolante può essere resa competitiva con il polinomio di interpolazione nella forma di Newton, mediante le così dette I e II forma di Lagrange:

I Forma di Lagrange. Inizialmente vengono determinati tutti i coefficienti pesi w_i come segue

$$w_i = \frac{1}{\prod_{j=0 j \neq i}^{n} (x_i - x_j)} \quad i = 0, \ldots, n$$

quindi al momento della valutazione in corrispondenza di un assegnato valore $x \neq x_i$ si determina

$$\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$$

e $p(x)$ come

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j.$$

II Forma di Lagrange. Come nel caso della I Forma si calcolano inizialmente i coefficienti pesi w_i e assegnato x si calcola $p(x)$ nel seguente modo:

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}};$$

infatti con riferimento al denominatore, se $y_i = 1, \quad i = 0, \ldots, n$, allora il polinomio nella I Forma di Lagrange corrispondente varrà 1, cioè:

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} 1 = 1,$$

e dividendo l'espressione con gli y_i dati della I Forma per questa espressione si ottiene la II Forma. Il vantaggio deriva dal fatto che avendo semplificato analiticamente $\ell(x)$, questa non deve essere calcolata; ancora, gli $n+1$ rapporti nelle sommatorie a numeratore e denominatore, essendo i medesimi, vengono calcolati una sola volta.

Come si vedrà nel capitolo dedicato all'integrazione numerica di funzioni, la rappresentazione nella forma di Lagrange risulterà molto importante.

4.1.5 Condizionamento dell'interpolazione

L'espressione del polinomio interpolante nella forma di Lagrange è ottimale per analizzare l'errore inerente di questo problema. Siano $\tilde{y}_i = y_i + \delta y_i$ le osservazioni perturbate in corrispondenza dei punti x_i che assumeremo non siano soggetti a perturbazione (questo deriva dall'interpolazione di una funzione $f(x)$ in cui le osservazioni altro non sono che la valutazione della $f(x)$ nei punti x_i, e se questi fossero perturbati si valuterebbe la funzione nei nuovi punti che quindi potrebbero essere pensati come dati esatti). Il polinomio di interpolazione sarà dato da:

$$\begin{aligned} \tilde{p}(x) &= \textstyle\sum_{i=0}^{n} \tilde{y}_i L_{i,n}(x) \\ &= \textstyle\sum_{i=0}^{n} (y_i + \delta y_i) L_{i,n}(x) \\ &= p(x) + \textstyle\sum_{i=0}^{n} \delta y_i L_{i,n}(x) \end{aligned}$$

con $p(x)$ il polinomio interpolante i dati (x_i, y_i) $i = 0, \ldots, n$. Allora

$$\begin{array}{ll} |\tilde{p}(x) - p(x)| & \leq \sum_{i=0}^{n} |\delta y_i||L_{i,n}(x)| \\ & \leq \max_{0\leq i\leq n} |\delta y_i| \sum_{i=0}^{n} |L_{i,n}(x)|. \end{array} \tag{4.4}$$

In particolare se $x \in [a, b]$

$$\frac{|\tilde{p}(x) - p(x)|}{\max_{0\leq i\leq n} |y_i|} \leq \frac{\max_{0\leq i\leq n} |\delta y_i| \sum_{i=0}^{n} |L_{i,n}(x)|}{\max_{0\leq i\leq n} |y_i|}$$

e poiché y_i è un valore assunto da $p(x)$, possiamo anche scrivere

$$\frac{|\tilde{p}(x) - p(x)|}{\max_{x\in[a,b]} p(x)} \leq \frac{\max_{0\leq i\leq n} |\delta y_i|}{\max_{0\leq i\leq n} |p(x_i)|} \sum_{i=0}^{n} |L_{i,n}(x)| \tag{4.5}$$

Le equazioni (4.4) e (4.5) mettono in relazione le perturbazioni assolute e relative sui risultati con quelle sui dati, segue che

$$C_{Int}(p(x)) = \sum_{i=0}^{n} |L_{i,n}(x)|$$

è il numero di condizione del problema di interpolazione nel punto x.

4.1.6 Errore di interpolazione di Lagrange

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 4.4 *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*

$$f(\bar{x}) = p(\bar{x}) + \frac{w(\bar{x})}{(n+1)!} f^{(n+1)}(\xi) \tag{4.6}$$

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

Dim.
Se $\bar{x}$ coincide con uno dei punti x_i, $i = 0, \ldots, n$ la tesi sarà banalmente vera.
Se $\bar{x}$ non coincide con nessuno fra tali punti si consideri la funzione $g(t)$ definita da

$$g(t) = f(t) - p(t) - \frac{w(t)}{w(\bar{x})}(f(\bar{x}) - p(\bar{x})).$$

Per $i = 0, \ldots, n$ sarà

$$\begin{aligned} g(x_i) &= f(x_i) - p(x_i) - \frac{w(x_i)}{w(\bar{x})}(f(\bar{x}) - p(\bar{x})) \\ &= y_i - y_i = 0, \end{aligned}$$

inoltre per $t = \bar{x}$

$$g(\bar{x}) = f(\bar{x}) - p(\bar{x}) - \frac{w(\bar{x})}{w(\bar{x})}(f(\bar{x}) - p(\bar{x})) = 0.$$

Allora la funzione $g(t)$ possiede $n + 2$ zeri distinti nell'intervallo $[a, b]$. Per il Teorema di Rolle la derivata $g'(t)$ deve possedere almeno $n + 1$ zeri in $[a, b]$, la derivata seconda $g''(t)$ deve possedere almeno n zeri e così via fino alla derivata $(n + 1)$-esima che possiede almeno uno zero in $[a, b]$. Chiamiamo ξ tale zero e sarà

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{(n+1)!}{w(\bar{x})}(f(\bar{x}) - p(\bar{x}))$$

infatti $p^{(n+1)}(t) \equiv 0$ essendo p di grado n e $w^{(n+1)}(t) = (n+1)!$ dalla definizione di $w(t)$.
Ricavando $f(\bar{x})$ dall'ultima espressione si ha

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

che è quanto volevamo provare. ⊙

Corollario 4.1 *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 4.1 *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 4.1 *Poiché in generale risulta difficile se non impossibile dare anche solo una stima del valore $f^{(n+1)}(\xi)$ è conveniente fornire un'altra espressione per l'errore di interpolazione $R(x)$.*

Teorema 4.5 (seconda formula dell'errore) *Nelle ipotesi del Teorema 4.4 che ci ha fornito l'espressione (4.6) dell'errore di interpolazione polinomiale si ha che vale*

$$f(\bar{x})=p(\bar{x})+w(\bar{x})f[x_0,x_1,\ldots,x_n,\bar{x}]. \tag{4.7}$$

Dim.
Per dimostrare questa seconda espressione dell'errore, basta osservare che

$$p(t)+w(t)f[x_0,x_1,\ldots,x_n,\bar{x}]$$

è il polinomio in t di interpolazione della f nei punti $x_0,x_1,\ldots,x_n,\bar{x}$ e per interpolazione tale polinomio in $\bar{x}$ coinciderà con $f(\bar{x})$. $\odot$

Corollario 4.2 *Confrontando le espressione (4.6) e (4.7) date per l'errore di interpolazione polinomiale si ottiene la relazione*

$$f[x_0,x_1,\ldots,x_n,\bar{x}]=\frac{1}{(n+1)!}f^{(n+1)}(\xi) \tag{4.8}$$

ove ξ dipende da $x_0,x_1,\ldots,x_n,\bar{x}$ e da $f(x)$.

Osservazione 4.2 *È importante 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 4.2 *Sia data la seguente funzione test:*

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

$n+1$ 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 4.2: Errore Massimo per l'interpolazione polinomiale della funzione di Runge su punti equispaziati in $[-5, 5]$.

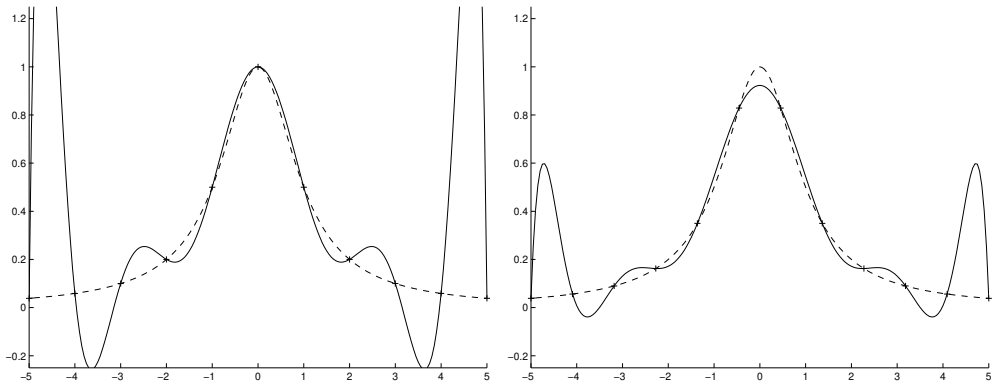

Figura 4.1: Interpolazione polinomiale della funzione di Runge su punti equispaziati; a sinistra 11 punti, a destra 12.

e si consideri il polinomio di interpolazione su $n+1$ punti equispaziati. Il matematico Runge dimostrò che al crescere di n la successione dei polinomi di interpolazione sui punti

$$x_i = \frac{10}{n} i - 5 \quad i = 0, \ldots, n$$

non converge puntualmente ad $f(x)$ e che i corrispondenti resti diventano in modulo arbitrariamente grandi in punti dell'intervallo $[-5, 5]$.

La Tab. 4.2 mostra l'errore massimo in valore assoluto fra la funzione di Runge e i polinomi interpolanti su punti equispaziati. In particolare l'errore è grande vicino agli estremi dell'intervallo ed aumenta con n (vedi anche Fig. 4.1).

Osservazione 4.3 *Nei problemi di interpolazione, la convergenza del polinomio interpolatore ad $f(x)$ dipende dalla regolarità della funzione e dalla legge con cui si infittiscono i punti. Come si è già visto su esempi, mantenendo i nodi equispaziati si ottiene convergenza solo per una classe ristretta di funzioni.*

Minimizzazione dell'errore di interpolzione

Supponiamo che $f(x)$ sia approssimata su $[a,b]$ dal polinomio interpolante $p(x)$ nei punti $x_0,\ldots,x_n$. Come visto l'errore di interpolazione è dato da (4.6) dove

$$w(x)=\prod_{i=0}^{n}(x-x_i).$$

Si noti che variando la posizione dei punti x_i, $i=0,\ldots,n$, cambia il polinomio $w(x)$ e quindi l'errore. Se $f^{(n+1)}(x)$ è maggiorabile da una costante M, il problema di minimizzare l'errore di interpolazione può essere posto come il problema della ricerca dei punti $x_i\in[a,b]$ tali da ottenere:

$$\min_{x_i\in[a,b]}\max_{x\in[a,b]} w(x).$$

La soluzione a questo problema sono i polinomi di Chebyshev di prima specie $T_{n+1}^{[a,b]}(x)$ di grado $n+1$ per cui vale

$$\max_{x\in[a,b]} T_{n+1}^{[a,b]}(x)=\frac{(b-a)^{n+1}}{2^{2n+1}}.$$

Inoltre tali polinomi sono monici e hanno tutti gli zeri reali in $[a,b]$. Se quindi come punti di interpolazione scegliamo i punti x_i zeri del polinomio $T_{n+1}^{[a,b]}(x)$ allora $w(x)=T_{n+1}^{[a,b]}(x)$ e

$$\max_{x\in[a,b]}|R(x)|\le\max_{x\in[a,b]}\left|\frac{f^{(n+1)}(\xi)}{(n+1)!}\right|\frac{(b-a)^{n+1}}{2^{2n+1}}$$

ed ora all'aumentare di n l'errore tende a zero.

Osservazione 4.4 *Se la funzione $f(x)$ è Lipschitziana e la si interpola su una opportuna distribuzione di punti (come per esempio gli zeri del polinomio di Chebishev di grado $n+1$[2]), si ha la convergenza dell'interpolante polinomiale. Più la f è regolare e più veloce è la convergenza.*

Teorema 4.6 *Sia $I=[-1,1]$ e i punti di interpolazione siano gli zeri del polinomio di Chebyshev di grado $n+1$.*
Se $f(x)\in C_I^0$ e soddisfa la condizione di Lipschitz

$$|f(\bar{x})-f(\tilde{x})|<K|\bar{x}-\tilde{x}|$$

[2] $x_i=cos(\frac{(2i+1)\pi}{2(n+1)})$, $i=0,\ldots,n$ sono gli $n+1$ zeri reali del polinomio di Chebishev di prima specie 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 $[a,b]$.

allora il polinomio interpolante converge uniformente a $f(x)$ su I per $n \to \infty$. Se inoltre $f(x) \in C_I^2$, si ha la seguente stima dell'errore:

$$max_{x \in I} |f(x) - p_n(x)| = O(\frac{1}{\sqrt{n}}).$$

Dim.
si veda Isaacson e Keller (1966), Ueberhuber (1995). $\odot$

$n+1$ 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 4.3: Errore Massimo per l'interpolazione polinomiale della funzione di Runge su punti zeri di Chebyshev in $[-5, 5]$.

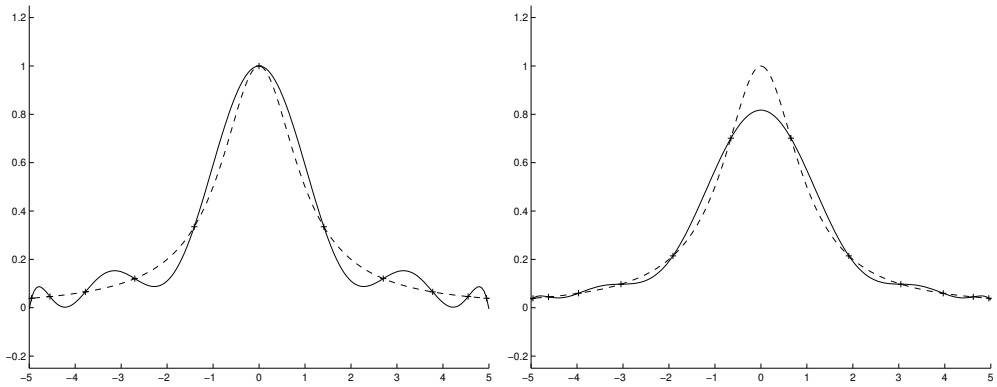

Figura 4.2: Interpolazione polinomiale della funzione di Runge su punti zeri di Chebishev; a sinistra 11 punti, a destra 12.

La Tab. 4.3 mostra l'errore massimo in valore assoluto fra la funzione di Runge e i polinomi interpolanti su punti zeri di Chebyshev (vedi anche Fig. 4.2).

4.2 Interpolazione di Hermite

L'interpolazione di Hermite si distingue da quella di Lagrange per il fatto che in corrispondenza dei punti distinti $x_i \quad i = 0, \ldots, n$ sono assegnate e si vogliono

interpolare valori e valori di derivata in genere di una funzione $f(x)$ che si vuole approssimare. Più precisamente si vuole determinare il polinomio $p(x) \in P_m$ tale che

$$p^{(k)}(x_i) = f^{(k)}(x_i) \quad i = 0, \ldots, n \quad k = 0, \ldots, m_i - 1.$$

La forma generale del polinomio di Hermite di grado m relativo a $n+1$ punti $x_0, \ldots, x_n \in [a, b]$ è data da

$$p(x) = \sum_{i=0}^{n} \sum_{k=0}^{m_i - 1} f^{(k)}(x_i) L_{i,k}(x)$$

dove i polinomi di grado m, $L_{i,k}(x)$ sono caratterizzati dalle condizioni

$$L_{i,k}^{(\ell)}(x_j) = \begin{cases} 1 & \text{se} \quad i = j \quad \text{ed} \quad \ell = k \\ 0 & \text{se} \quad i \neq j \end{cases} \tag{4.9}$$

e si suppone che $m + 1 = \sum_{i=0}^{n} m_i$.

Si hanno i seguenti risultati fondamentali.

4.2.1 Forma Cardinale

Teorema 4.7 *I polinomi $L_{i,k}$ che soddisfano la (4.9) possono essere calcolati, per $i = 0, \ldots, n$ nella seguente forma ricorrente*

$$L_{i,m_i-1}(x) = \ell_{i,m_i-1}(x)$$

$$L_{i,k}(x) = \ell_{i,k}(x) - \sum_{\ell=k+1}^{m_i-1} \ell_{i,k}^{(\ell)}(x) L_{i,\ell}(x) \quad k = m_i - 2, \ldots, 0$$

dove

$$\ell_{i,k}(x) = \frac{(x - x_i)^k}{k!} \prod_{j \neq i} \left(\frac{x - x_j}{x_i - x_j} \right)^{m_j} \quad k = 0, \ldots, m_i - 1.$$

Dim.

$\odot$

Osservazione 4.5 *Anche per la forma di Lagrange dell'interpolazione di Hermite si ha una forma baricentrica estremamente accurata e stabile.*

4.2.2 Errore di interpolazione di Hermite

Teorema 4.8 *Se* $[a,b]$ *contiene i punti* $x_i \quad i = 0,\dots,n$, $f \in C^{m+1}([a,b])$, *allora l'errore di approssimazione si rappresenta come*

$$f(x) - p(x) = \Omega_m(x)\frac{f^{(m+1)}(\xi)}{(m+1)!} \quad \xi \in (a,b) \tag{4.10}$$

dove $\Omega_m(x) = (x-x_0)^{m_0}(x-x_1)^{m_1}\cdots(x-x_n)^{m_n}$.

Dim.
$\odot$

Esempio di interpolazione di Hermite

Si esamina l'interpolazione di Hermite di valori e derivate prima su due punti con un polinomio cubico nella base di Bernstein, cioè della forma

$$p(x) = \sum_{i=0}^{3} b_i B_{i,3}(x) \qquad x \in [0,1];$$

questa consiste nel determinare il polinomio cubico $p(x)$ 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 Bernstein negli estremi (osservazione 2.6):

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

Da queste si possono ricavare b_1 e b_2:

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

4.3 Interpolazione polinomiale a tratti

Nonostante i risultati di convergenza, sia per l'interpolazione di Lagrange che di Hermite, 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 funzione. Si ha inoltre che i polinomi non sono sufficientemente flessibili per approssimare funzioni che hanno differenti andamenti in differenti parti dell'intervallo di definizione. Questo è il caso di molte funzioni 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. Per evitare problemi numerici si dovrebbero usare solo polinomi di grado basso.

Definizione 4.1 *Sia data la partizione* $\{x_i\}_{i=1,\ldots,m-1}$ *dell'intervallo* $[a,b]$*, cioè*

$$a = x_0 < x_1 < x_2 < \cdots < x_{m-1} < x_m = b,$$

allora si definisce polinomio a tratti *(piecewise polynomial) la funzione:*

$$pp(x) = \begin{cases} p_0(x) & x \in [x_0, x_1] \\ p_1(x) & x \in [x_1, x_2] \\ \vdots & \\ p_{m-1}(x) & x \in [x_{m-1}, x_m] \end{cases}$$

dove i $p_i(x)$, $i = 0, \ldots, m-1$ *sono tutti polinomi di grado* n *e soddisfano le seguenti condizioni:*

$$p_{i-1}^{(\ell)}(x_i) \equiv p_i^{(\ell)}(x_i), \quad \ell = 1, \ldots, k, \qquad i = 1, \ldots, m-1,$$

con $k \leq n-1$ *fissato.*

Si dirà che la funzione polinomiale a tratti $pp(x)$ 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 la funzione polinomiale a tratti viene chiamata funzione **spline**.

In conclusione, il passaggio da polinomi a polinomi a tratti permette una maggior flessibilità, ma a fronte della perdita di regolarità; il passaggio da semplici polinomi a tratti a spline recupera la regolarità utile alle applicazioni mantenendo la flessibilità.

Con *interpolazione polinomiale a tratti* di Lagrange si intende l'interpolazione di un set di dati in un intervallo $[a,b]$ con una funzione polinomiale a tratti. 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 interpolante poliniomiale a tratti consiste in una funzione $pp(x)$ definita come sopra, dove la partizione dell'intervallo di definizione $[a,b] = [x_0, x_m]$ viene definita dalle ascisse x_i dei punti di interpolazione e soddisfa le condizioni di interpolazione $pp(x_i) = y_i$, $i = 0, \ldots, m$, o anche:

1.
$$p_0(x_0) = y_0, \qquad pp_{m-1}(x_m) = y_m,$$

2.
$$p_{i-1}(x_i) \equiv p_i(x_i) = y_i \quad i = 1, \ldots, m-1$$

In modo del tutto analogo si può definire l'*interpolazione polinomiale a tratti* di Hermite.

Presentiamo due metodi di interpolazione polinomiale a tratti molto utilizzati nella pratica: l'interpolazione locale cubica di Hermite C^1 e l'interpolazione spline cubica C^2.

4.3.1 Interpolazione locale (cubica di Hermite C^1)

Per interpolazione locale si intende un metodo che determina il polinomio a tratti interpolante $pp(x)$ 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, chiamata **interpolazione cubica di Hermite** 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. 4.3.1 mostra il risultato di una tale interpolazione sulla funzione test di Runge su punti equispaziati.

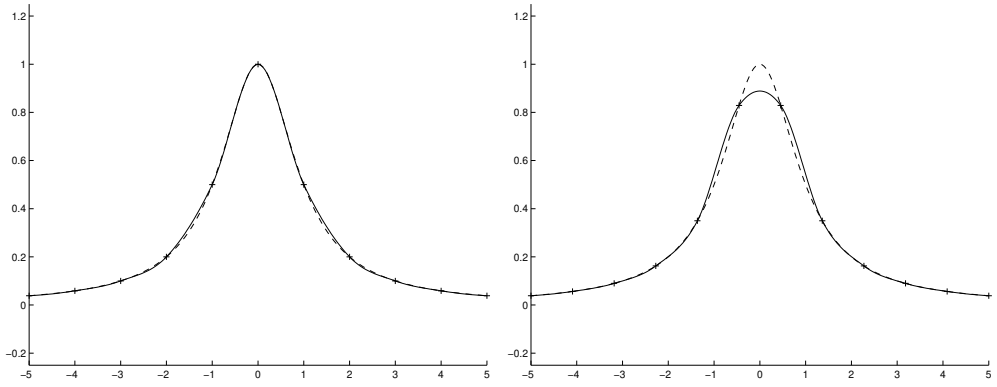

Figura 4.3: Interpolazione polinimiale cubica a tratti C^1 della funzione di Runge su punti equispaziati e derivate stimate; a sinistra 11 punti, a destra 12.

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

Tabella 4.4: Errore Massimo per l'interpolazione polinomiale cubica a tratti C^1 della funzione di Runge su punti equispaziati e derivate stimate.

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

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, nella base di Bernstein. Le derivate assegnate y'_i dovranno essere opportunamente scalate e per la precisione le osservazioni da interpolare saranno:

$$(0, y_i) \quad (0, y'_i h_i) \quad (1, y_{i+1}) \quad (1, y'_{i+1} h_i)$$

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

Teorema 4.9 (Convergenza) *Sia $pp_H(x)$ il polinomio a tratti cubico di Hermite C^1 di interpolazione di una funzione $f(x)$ di classe $C^4_{[a,b]}$. Allora posto $E(x) = f(x) - pp_H(x)$ per $x \in [a, b]$, si ha:*

$$|f(x) - pp_H(x)| \leq \frac{1}{384} h^4 max_{a \leq z \leq b} |f^{(4)}(z)| \quad \forall x \in [a, b].$$

con $h = \max_{0 \leq i \leq m-1}(x_{i+1} - x_i)$.

Dim.
Basta provare che vale la relazione indicata su ogni intervallo $[x_i, x_{i+1}]$ di ampiezza h_i. Per l'errore di interpolazione di Hermite (4.10) vale

$$|f(x) - pp_H(x)| = (x - x_i)^2 (x - x_{i+1})^2 \frac{f^{(4)}(\xi_i)}{4!} \quad \xi_i \in (x_i, x_{i+1}),$$

e poiché $(x - x_i)^2(x - x_{i+1})^2$ è un polinomio di grado 4 con radici doppie negli estremi, il suo max si avrà nel punto medio $(x_i + x_{i+1})/2$, così ché possiamo maggiorare l'errore con l'espressione

$$|f(x) - pp_H(x)| \leq \left(\frac{h_i}{2}\right)^4 \frac{f^{(4)}(\xi_i)}{4!} = \frac{1}{384} h_i^4 f^{(4)}(\xi_i) \quad x \in [x_i, x_{i+1}].$$

$\odot$

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:

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

Osservazione 4.6 *Si osservi che le stime delle derivate sopra introdotte, nel caso che $h_i = h \ \forall i$, altro non sono che le differenze centrate che come è ben noto si ottengono come la derivata della parabola nel punto centrale di tre punti di interpolazion e sono una approssimazione di ordine 2.*

4.3.2 Interpolazione globale (spline cubica C^2)

Per interpolazione globale si intende un metodo che determina il polinomio a tratti $pp(x)$ interpolante ricavando i polinomi $p_i(x)$ in modo non indipendente dagli altri. Vediamo questo modo di procedere nel caso particolare di polinomi a tratti cubici con massima regolarità C^2, anche detto **interpolazione spline cubica** C^2.
Se abbiamo i dati $(x_i, y_i), i = 0, \ldots, m$, allora una spline cubica $s(x)$ è data dalla funzione

$$s(x) = \begin{cases} p_0(x) & x \in [x_0, x_1] \\ p_1(x) & x \in [x_1, x_2] \\ \vdots & \\ p_{m-1}(x) & x \in [x_{m-1}, x_m] \end{cases}$$

dove ogni $p_i(x)$ è un polinomio cubico. Scriviamo i $p_i(x)$ nella base di Bernstein nel seguente modo:

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

Per determinare la spline dobbiamo determinare i coefficienti q_i, r_i, s_i e t_i per ogni i. Poiché ci sono m intervalli ci sono $4m$ coefficienti da determinare. Prima richiediamo che la spline interpoli i dati:

$$p_i(x_i) = y_i$$

$$p_i(x_{i+1}) = y_{i+1}$$

per ogni $i = 0, \ldots, m-1$. In altre parole dovrà essere:

$$\begin{aligned} q_i &= y_i \\ t_i &= y_{i+1} \qquad\qquad i = 0, \ldots, m-1. \end{aligned}$$

Per cui

$$p_i(x) = y_i B_{0,3}(x) + r_i B_{1,3}(x) + s_i B_{2,3}(x) + y_{i+1} B_{3,3}(x) \quad i = 0, \ldots, m-1. \tag{4.11}$$

Affinché poi due polinomi consecutivi si raccordini C^2 devono essere soddisfatte le seguenti ulteriori condizioni

$$p'_{i-1}(x_i) = p'_i(x_i)$$

$$p''_{i-1}(x_i) = p''_i(x_i)$$

su tutti i punti interni, cioè per $i = 1, \ldots, m-1$, che possiamo esplicitare nel seguente insieme di condizioni

$$\begin{array}{ll} \dfrac{y_i - s_{i-1}}{h_{i-1}} = \dfrac{r_i - y_i}{h_i} & i = 1, \ldots, m-1 \\ \dfrac{y_i - 2s_{i-1} + r_{i-1}}{h_{i-1}^2} = \dfrac{s_i - 2r_i + y_i}{h_i^2} & i = 1, \ldots, m-1. \end{array} \tag{4.12}$$

Si tratta di $2m-2$ equazioni lineari nelle $2m$ incognite r_i ed s_i, $i = 0, \ldots, m-1$. Solitamente si aggiungono due ulteriori condizioni, per esempio

$$p_0''(x_0) = p_{m-1}''(x_m) = 0;$$

queste sono chiamate condizioni **naturali**; un altro esempio consiste nelle condizioni **periodiche**

$$p_0'(x_0) = p_{m-1}'(x_m), \qquad p_0''(x_0) = p_{m-1}''(x_m).$$

Altre condizioni comuni sono quelle chiamate **derivate agli estremi**:

$$p_0'(x_0) = D_0$$

$$p_{m-1}'(x_m) = D_m$$

con D_0 e D_m assegnati, così da avere un sistema quadrato. Anziché aggiungere due condizioni si possono togliere due incognite; questo si può per esempio ottenere considerando che in $[x_0, x_2]$ e $[x_{m-2}, x_m]$ ci sia soltanto un polinomio, rispettivamente $p_1(x)$ e $p_{m-2}(x)$, portando ad un sistema quadrato con globalmente due incognite in meno; tali condizioni sono note in letteratura come **not a knot**.
Dover risolvere uno di questi sistemi lineari equivale a determinare i polinomi a tratti in modo dipendente gli uni dagli altri.

Osservazione 4.7 *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; se infatti è*

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

allora

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

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

Se poniamo

$$D_i = \frac{y_i - s_{i-1}}{h_{i-1}} = \frac{r_i - y_i}{h_i} \tag{4.13}$$

possiamo riscrivere le seconde condizioni 4.12 solo in termini di D_i e y_i sostituendo al posto di s_i ed r_i le espressioni trovate mediante la 4.13; sarà:

$$\frac{y_{i-1} - y_i + 2D_i h_{i-1} + D_{i-1} h_{i-1}}{h_{i-1}^2} = \frac{y_{i+1} - y_i - D_{i+1} h_i - 2D_i h_i}{h_i^2}; \tag{4.14}$$

moltiplicando entrambi i membri per $h_i h_{i-1}$ e raccogliendo opportunamente i termini si ottiene:

$$h_i D_{i-1} + 2D_i(h_i + h_{i-1}) + h_{i-1} D_{i+1} = \frac{h_i}{h_{i-1}}(y_i - y_{i-1}) + \frac{h_{i-1}}{h_i}(y_{i+1} - y_i)$$

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 $(m-1) \times (m-1)$ che, essendo la matrice a diagonale dominante, avrà una ed una sola soluzione

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

$$= \begin{pmatrix} \frac{h_1}{h_0}(y_1 - y_0) + \frac{h_0}{h_1}(y_2 - y_1) - h_1 D_0 \\ \frac{h_2}{h_1}(y_2 - y_1) + \frac{h_1}{h_2}(y_3 - y_2) \\ \vdots \\ \frac{h_{m-1}}{h_{m-2}}(y_{m-1} - y_{m-2}) + \frac{h_{m-2}}{h_{m-1}}(y_m - y_{m-1}) - h_{m-2} D_m \end{pmatrix}.$$

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 nella forma (4.11) saranno dati dai coefficienti

$$q_i = y_i, \quad r_i = y_i + h_i D_i, \quad s_i = y_{i+1} - h_i D_{i+1}, \quad t_i = y_{i+1} \qquad i = 0, \ldots, m-1.$$

Osservazione 4.8 *Si noti che nel caso i punti siano tali per cui* $h_i \equiv 1 \quad \forall i$, *il sistema si semplifica in*

$$\begin{pmatrix} 4 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 \\ \ldots & & & & & \\ 0 & \ldots & 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{m-1} \end{pmatrix} = \begin{pmatrix} y_2 - y_0 - D_0 \\ y_3 - y_1 \\ y_4 - y_2 \\ \vdots \\ y_m - y_{m-2} - D_m \end{pmatrix}.$$

La Fig. 4.4 mostra il risultato dell'interpolazione della funzione test di Runge con una spline cubica su punti equispaziati.

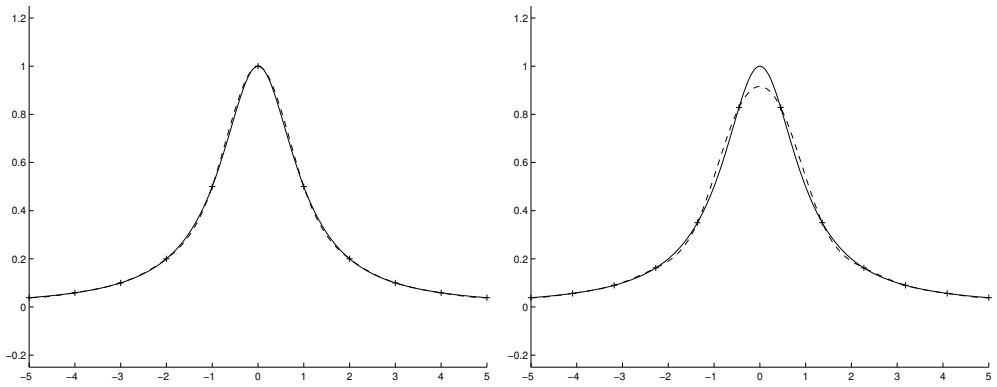

Figura 4.4: Interpolazione polinomiale cubica a tratti C^2 con derivate agli estremi della funzione di Runge su punti equispaziati; a sinistra 11 punti, a destra 12.

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

Tabella 4.6: Errore Massimo per l'interpolazione polinomiale cubica a tratti C^2 con derivate agli estremi della funzione di Runge su punti equispaziati.

Volendo imporre le condizioni agli estremi chiamate "naturali" e "periodiche", riprendiamo il sistema 4.14 di $m-1$ equazioni in $m+1$ incognite. Procediamo prima imponendo le condizioni "naturali"

$$p_0''(x_0) = p_{m-1}''(x_m) = 0;$$

Utilizzando l'espressione vista della derivata seconda di $p_0(x)$ si ha:

$$0 = p_0''(x_0) = \frac{6}{h_0^2}(s_0 - 2r_0 + y_0)$$

e usando la 4.13 per s_0 ed r_0 si ha

$$r_0 = D_0 h_0 + y_0 \qquad\qquad s_0 = y_1 - D_1 h_0$$

da cui sostituendo

$$= \frac{6}{h_0^2}(y_1 - y_0 - D_1 h_0 - 2D_0 h_0)$$

che porta all'equazione

$$2h_0 D_0 + h_0 D_1 = y_1 - y_0.$$

Analogamente da

$$0 = p''_{m-1}(x_m) = \frac{6}{h_{m-1}^2}(y_m - 2s_{m-1} + r_{m-1})$$

utilizzando la 4.13 per avere

$$s_{m-1} = y_m - h_{m-1} D_m \qquad\qquad r_{m-1} = y_{m-1} + h_{m-1} D_{m-1}$$

e sostituendo

$$= \frac{6}{h_{m-1}^2}(y_{m-1} - y_m + 2h_{m-1} D_m + h_{m-1} D_{m-1})$$

che porta all'equazione

$$h_{m-1} D_{m-1} + 2h_{m-1} D_m = y_m - y_{m-1}$$

da cui:

$$\begin{pmatrix} 2h_0 & h_0 & 0 & \dots & & 0 \\ h_1 & 2(h_1+h_0) & h_0 & 0 & \dots & 0 \\ 0 & h_2 & 2(h_2+h_1) & h_1 & \dots & 0 \\ \dots & & & & & \\ 0 & \dots & 0 & h_{m-1} & 2(h_{m-1}+h_{m-2}) & h_{m-2} \\ 0 & & \dots & 0 & h_{m-1} & 2h_{m-1} \end{pmatrix} \begin{pmatrix} D_0 \\ D_1 \\ \\ \vdots \\ D_{m-1} \\ D_m \end{pmatrix} =$$

$$= \begin{pmatrix} y_1 - y_0 \\ \frac{h_1}{h_0}(y_1 - y_0) + \frac{h_0}{h_1}(y_2 - y_1) \\ \frac{h_2}{h_1}(y_2 - y_1) + \frac{h_1}{h_2}(y_3 - y_2) \\ \vdots \\ \frac{h_{m-1}}{h_{m-2}}(y_{m-1} - y_{m-2}) + \frac{h_{m-2}}{h_{m-1}}(y_m - y_{m-1}) \\ y_m - y_{m-1} \end{pmatrix}.$$

Che consiste in un sistema $m+1 \times m+1$ a diagonale dominante.

Riprendendo il sistema 4.14 imponiamo le condizioni "periodiche"

$$p'_0(x_0) = p'_{m-1}(x_m), \qquad p''_0(x_0) = p''_{m-1}(x_m).$$

La prima condizione si traduce semplicemente in

$$D_0 = D_m$$

mentre per la seconda usiamo l'espressione della derivata seconda di $p_0(x)$ e di $p_{m-1}(x)$ ottenendo:

$$p_0''(x_0) = \frac{6}{h_0^2}(y_1 - y_0 - 2h_0 D_0 - h_0 D_1)$$

$$p_{m-1}''(x_m) = \frac{6}{h_{m-1}^2}(y_{m-1} - y_m + 2h_{m-1} D_m + h_{m-1} D_{m-1})$$

da cui uguagliandole si ha:

$$\frac{1}{h_0^2}(y_1 - y_0 - 2h_0 D_0 - h_0 D_1) = \frac{1}{h_{m-1}^2}(y_{m-1} - y_m + 2h_{m-1} D_m + h_{m-1} D_{m-1})$$

e moltiplicando entrambi i termini per $h_0 h_{m-1}$ si ottiene l'equazione:

$$2h_{m-1} D_0 + h_{m-1} D_1 + h_0 D_{m-1} + 2h_0 D_m = \frac{h_{m-1}}{h_0}(y_1 - y_0) + \frac{h_0}{h_{m-1}}(y_m - y_{m-1})$$

da cui:

$$\begin{pmatrix} 1 & 0 & \dots & & 0 & -1 \\ h_1 & 2(h_1+h_0) & h_0 & 0 & \dots & 0 \\ 0 & h_2 & 2(h_2+h_1) & h_1 & \dots & 0 \\ \dots & & & & & \\ 0 & \dots & 0 & h_{m-1} & 2(h_{m-1}+h_{m-2}) & h_{m-2} \\ 2h_{m-1} & h_{m-1} & \dots & 0 & h_0 & 2h_0 \end{pmatrix} \begin{pmatrix} D_0 \\ D_1 \\ \\ \vdots \\ D_{m-1} \\ D_m \end{pmatrix} =$$

$$= \begin{pmatrix} 0 \\ \frac{h_1}{h_0}(y_1 - y_0) + \frac{h_0}{h_1}(y_2 - y_1) \\ \frac{h_2}{h_1}(y_2 - y_1) + \frac{h_1}{h_2}(y_3 - y_2) \\ \vdots \\ \frac{h_{m-1}}{h_{m-2}}(y_{m-1} - y_{m-2}) + \frac{h_{m-2}}{h_{m-1}}(y_m - y_{m-1}) \\ \frac{h_{m-1}}{h_0}(y_1 - y_0) + \frac{h_0}{h_{m-1}}(y_m - y_{m-1}) \end{pmatrix}.$$

4.3.3 Spline cubiche C^2 e proprietà di minimo

Osservazione 4.9 *Si dice che le spline cubiche C^2 sono "smooth" fra i nodi. Questa proprietà deriva dal prossimo teorema che dice che le spline cubiche di*

interpolazioni non possono avere grandi oscillazioni. Infatti la derivata prima di una funzione oscillante assume valori positivi e negativi grandi e per il teorema del valor medio allora la derivata seconda avrà valori grandi; segue che l'integrale della derivata seconda al quadrato non potrebbe essere piccolo, come invece prova il prossimo teorema.

Proprietà 4.1 *Le spline cubiche C^2 di interpolazione con* **derivate agli estremi**, **naturali** *e* **periodiche** *(in quest'ultimo caso la $f(x)$ che si interpola deve essere periodica, ossia $f(a) = f(b)$, $f'(a) = f'(b)$ e $f''(a) = f''(b)$), soddisfano la seguente proprietà di minimo che le caratterizza.*

Teorema 4.10 (Proprietà di minimo) *Sia $s(x)$ la spline cubica C^2 di interpolazione con* **derivate agli estremi**, **naturale** *o* **periodica**. *Sia $f(x)$ una funzione di classe $C^2_{[a,b]}$ che soddisfi le stesse condizioni di interpolazione della $s(x)$; allora vale la seguente disuguaglianza:*

$$\int_a^b (s''(x))^2 dx \leq \int_a^b (f''(x))^2 dx$$

il segno di uguaglianza vale solo se $s(x) \equiv f(x)$.

Dim.
Per la dimostrazione si veda [PRE75]. $\odot$

Osservazione 4.10 *La relazione integrale vista, consente una dimostrazione di esistenza e unicità per la spline cubica di interpolazione di una funzione $f(x)$ con condizioni di "derivate agli estremi", "naturale" o "periodica" ([PRE75]).*

Teorema 4.11 *$s(x) \equiv 0$ è l'unica spline cubica di interpolazione con "derivate agli estremi", "naturale" o "periodica" della funzione $f(x) \equiv 0$.*

Dim.
Sia $s(x)$ una spline cubica di interpolazione con "derivate agli estremi", "naturale" o "periodica" della funzione nulla. Allora $s(x)$ soddisfa la relazione integrale. Segue che $s''(x) \equiv 0$ in $[a, b]$. Ma allora $s(x)$ è un polinomio di grado 1 su $[a, b]$. Poiché per interpolazione deve essere $s(a) = 0$ ed $s(b) = 0$ si trova che $s(x) \equiv 0$ su $[a, b]$. $\odot$

È ora facile provare il seguente teorema.

Teorema 4.12 (Esistenza e unicità) *La spline cubica C^2 interpolante con "derivate agli estremi", "naturale" o "periodica" esiste ed è unica.*

Dim.
Supponiamo che il sistema di equazioni da risolvere imponendo le condizioni di interpolazioni sia

$$A\mathbf{c} = \mathbf{b}$$

dove A è la matrice dei coefficienti $(k+4) \times (k+4)$. È chiaro che la A è indipendente dalla funzione da interpolare e resterà uguale qualunque sia la funzione da interpolare.
Poiché $s(x) \equiv 0$ è l'unica spline cubica C^2 interpolante di $f(x) \equiv 0$, sarà $\mathbf{b} = \mathbf{0}$ e da $A\mathbf{c} = \mathbf{0}$ (avendo una ed una sola soluzione) si deduce che A deve essere non singolare. Ma allora $A\mathbf{c} = \mathbf{b}$ ha una ed una sola soluzione $\forall \mathbf{b}$. $\odot$

Teorema 4.13 (Convergenza) *Sia $s(x)$ la spline cubica C^2 di interpolazione con "derivate agli estremi", "naturale" o "periodica" di una funzione $f(x)$ di classe $C^2_{[a,b]}$. Allora posto $E(x) = f(x) - s(x)$ per $x \in [a,b]$, si ha:*

$$|E(x)| = |f(x) - s(x)| \leq \sqrt{8}\, h^{\frac{3}{2}} \max_{a \leq z \leq b} |f''(z)| \quad \forall\, x \in [a,b]$$

$$|E'(x)| = |f'(x) - s'(x)| \leq \sqrt{8}\, h^{\frac{1}{2}} \max_{a \leq z \leq b} |f''(z)| \quad \forall\, x \in [a,b]$$

con $h = \max_{0 \leq i \leq m-1}(x_{i+1} - x_i)$.

Dim.
si veda [PRE75]. $\odot$

Teorema 4.14 (Convergenza) *Sia $s(x)$ la spline cubica C^2 di interpolazione con "derivate agli estremi", "naturale" o "periodica" di una funzione $f(x)$ di classe $C^4_{[a,b]}$. Allora posto $E(x) = f(x) - s(x)$ per $x \in [a,b]$, si ha:*

$$|E(x)| = |f(x) - s(x)| \leq \frac{5}{384}\, h^4 \max_{a \leq z \leq b} |f^{(4)}(z)| \quad \forall\, x \in [a,b]$$

$$|E'(x)| = |f'(x) - s'(x)| \leq \left(\frac{\sqrt{3}}{216} + \frac{1}{24}\right)\, h^3 \max_{a \leq z \leq b} |f^{(4)}(z)| \quad \forall\, x \in [a,b]$$

con $h = \max_{0 \leq i \leq m-1}(x_{i+1} - x_i)$.

Dim.
si veda [PRE75]. $\odot$

Osservazione 4.11 *Questo teorema asserisce che all'aumentare dei punti di interpolazione, la spline cubica C^2 di interpolazione converge alla funzione da interpolare. Ancora la derivata della spline cubica C^2 interpolante converge alla derivata della funzione da interpolare.*

Esempio di analisi dell'errore

Se usiamo l'interpolazione spline cubica C^2 per approssimare $f(x) = \cos(x)$ sull'intervallo $[0, 3.1]$ con $h = 0.1$, qual è l'errore atteso?

Poiché sappiamo che $f^{(4)}(x) = \cos(x)$, $\max_{a \leq z \leq b} |f^{(4)}(z)| \leq 1$. Sostituendo nell'espressione dell'errore si ha:

$$\frac{5}{384} \cdot 1 \cdot (0.1)^4 = \frac{5}{3840000} = \frac{1}{768000} = 1.302 \times 10^{-6}$$

Si noti che questo piccolo errore richiede solo $31 + 1 = 32$ valutazioni della funzione $f(x)$.

Possiamo usare la stessa espressione dell'errore per decidere quante valutazioni della funzioni $f(x)$ sono necessarie per ottenere un errore di approssimazione di livello accettabile. Questo è un problema di importanza pratica; per esempio si vuole determinare in anticipo quante valutazioni della funzione sono necessarie per ottenere un certo tipo di errore, prima di eseguire un costoso esperimento in termini di utilizzo della funzione.

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 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. 4.5, 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 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

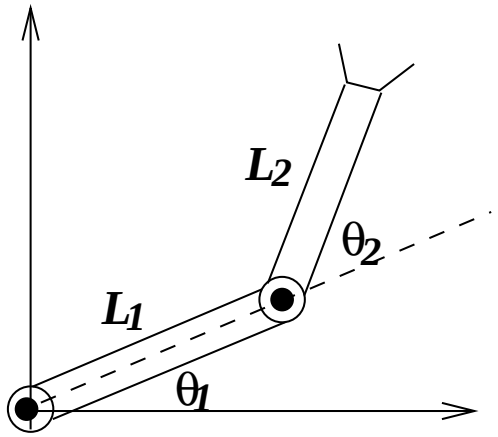

Figura 4.5: Schema di un braccio meccanico con due motori

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 4.3 *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 4.12 *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.4.6 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 $L_2 = 3$.

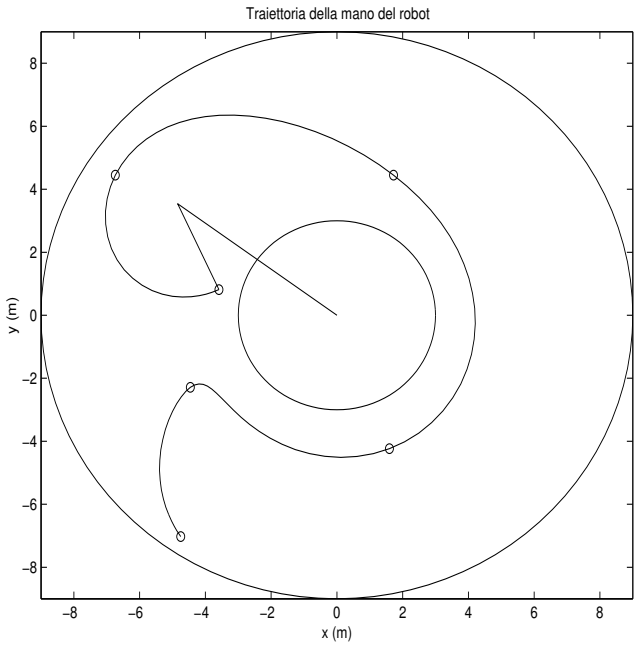

Figura 4.6: Schema di un braccio meccanico con due motori: output di un programma esempio

4.4 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 curva cubica a tratti di Hermite C^1 e con una curva spline cubica C^2.
Siano dati $m+1$ punti $Q_i \equiv (x_i, y_i)_{i=0,\ldots,m}$; si vogliono determinare la curva cubica a tratti di Hermite C^1 e la spline cubica C^2 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 equispaziati:

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

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 di Hermite C^1 e spline cubica 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.4.7 mostra tre interpolazioni di 8 punti disposti a forma di "S" con cubiche a tratti di Hermite C^1 dove i vettori derivati, inizialmente stimati come indicato nella sezione 4.3.1, sono stati globalmente scalati di 1/2 e poi 1/4 evidenziando il tensionamento relativo. La Fig.4.8 mostra invece una curva di interpolazione spline cubica C^2 con parametrizzazione della corda di 32 punti disposti nel piano a definire un profilo 2D di Paperino.

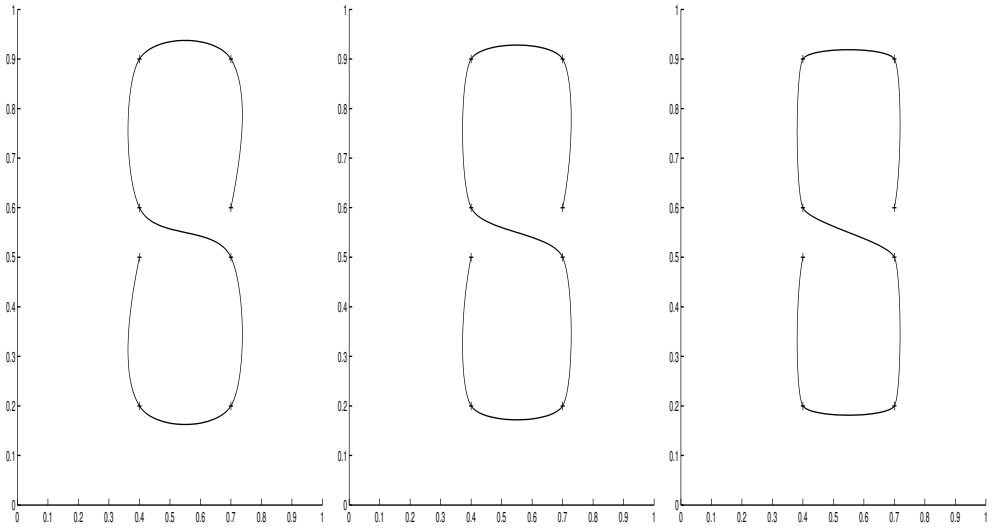

Figura 4.7: Interpolazione con curva polinomiale cubica a tratti di Hermite C^1 con stima dei vettori derivati nei punti (sinistra); vettori derivati scalati di 1/2 (centro); vettori derivati scalati di 1/4 (destra).

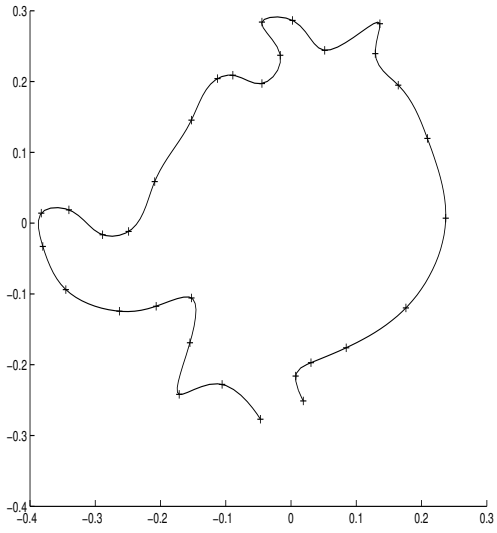

Figura 4.8: Interpolazione con curva spline cubica di 33 punti.

Capitolo 5

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

$$\frac{\partial g}{\partial a_i}=-2\sum_{k=1}^{m}y_k\phi_i(x_k)+2\sum_{j=0}^{n}a_j\left(\sum_{k=1}^{m}\phi_i(x_k)\phi_j(x_k)\right)=0\quad i=0,\ldots,n$$

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 5.1 *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.

5.1 Forma Monomiale

Teorema 5.1 *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. $\odot$

5.2 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 2.4 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.5.2).

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

5.3 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 Householder e Stiefel.

5.3.1 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:

$$\begin{cases} p_0(x) = & 1 \\ p_1(x) = & xp_0(x) - \alpha_1 p_0(x) \end{cases}$$

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

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

5.4 Esempi numerici

Le Fig.5.1 e 5.2 mostrano il risultato di un'approssimazione ai minimi quadrati di 51 punti equispaziati della funzione di Runge rispettivamente con polinomi di grado 10, 15, 30 e 35 nella base monomiale. I differenti indici di condizionamento della Tab.5.2 indicano una elaborazione numerica potenzialmente meno accurata nel caso monomiale che non nella base di Bernstein; in realtà questa differenza non risulta graficamente apprezzabile né usando Bernstein né usando

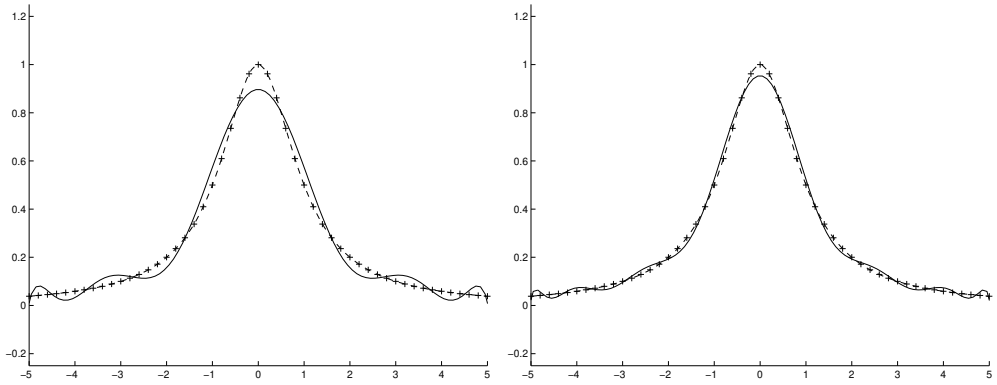

Figura 5.1: Approssimazione polinomiale nella base delle potenze della funzione di Runge di 51 punti equispaziati; a sinistra grado 10, a destra 15.

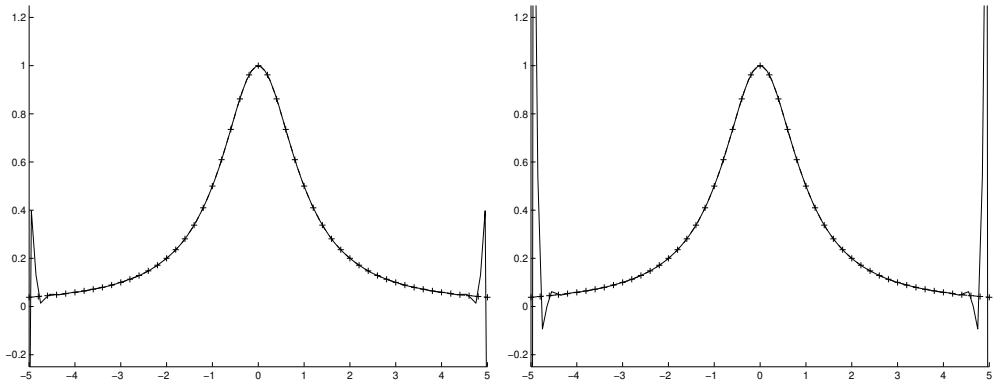

Figura 5.2: Approssimazione polinomiale nella base delle potenze della funzione di Runge di 51 punti equispaziati; a sinistra grado 30, a destra 35.

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.5.2, 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 poca flessibilità tipici dei polinomi di grado alto.

5.5 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

$$a_0 = \bar{y} - a_1\bar{x} \qquad a_1 = \frac{\sum_{k=1}^{m} x_k y_k - m\bar{x}\bar{y}}{\sum_{k=1}^{m} x_k^2 - m\bar{x}^2} \tag{5.1}$$

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 5.1 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

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

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

5.6 Un'applicazione: curve di approssimazione

Possiamo risolvere problemi di approssimazione ai minimi quadrati con funzioni vettoriali o curve polinomiali o polinomiali a tratti, per esempio nel piano.
Siano assegnati m punti nel piano euclideo $Q_i = (x_i, y_i)_{i=1,\dots,m}$. 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 $i = 1, \dots, m$. 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.

Parte III

Integrazione Numerica

Capitolo 6

Integrazione Numerica

L'integrazione numerica o quadratura numerica consiste nel calcolare un valore approssimato di

$$\int_a^b f(x)dx.$$

Ovviamente il problema sorge quando l'integrazione non può essere eseguita esattamente (non si può trovare una funzione primitiva, come per esempio nel caso di $\int_a^b e^{-x^2}dx$ o $\int_0^{\pi/2}\sqrt{1+\cos^2 x}\,dx$) o quando la funzione integranda è nota soltanto in un numero finito di punti. Ancora, anche se è nota una funzione primitiva, il calcolo può essere così costoso che è preferibile utilizzare un metodo numerico per determinarne un'approssimazione.
I metodi che vedremo consistono nell'approssimare la funzione integranda mediante polinomi o polinomi a tratti che risultano facilmente integrabili.

6.1 Formule di quadratura di Newton-Cotes

Nella sua formulazione più generale una **formula di quadratura** esprime un'approssimazione di un integrale come una combinazione lineare di valori della funzione integranda, cioè:

$$\sum_{i=0}^{n} W_i f(x_i).$$

Le formule di quadratura di Newton-Cotes si ottengono considerando $n+1$ punti equispaziati nell'intervallo chiuso $[a,b]$ dati da:

$$x_i = a + ih \quad i = 0,\ldots,n \quad \text{con} \quad h = \frac{b-a}{n} \quad \text{per} \quad n > 0,$$

e approssimando la funzione integranda $f(x)$ mediante il polinomio interpolante $p \in \mathbf{P}_n$, cioè di al più grado n, dei punti $(x_i, f(x_i))$ $i = 0, \ldots, n$. Sarà:

$$\int_a^b f(x)dx = \int_a^b p(x)dx + R_T \simeq \int_a^b p(x)dx.$$

Dal teorema che dà l'errore dell'interpolazione polinomiale si ha che l'errore di troncamento R_T è dato da:

$$R_T = \int_a^b (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{(n+1)}(\xi_x)}{(n+1)!} dx. \tag{6.1}$$

Se l'integrale richiesto viene approssimato dall'integrale di un polinomio di interpolazione di grado n, si ottiene una formula che è esatta per tutti i polinomi di grado minore od uguale ad n (grado di precisione n).
È facile mostrare che tale approsimazione è una formula di quadratura, cioè è esprimibile come una combinazione lineare dei valori $f(x_i)$ $i = 0, \ldots, n$ della funzione. Per vedere questo, pensiamo $p(x)$ nella forma di Lagrange

$$p(x) = \sum_{i=0}^{n} f(x_i) L_{i,n}(x)$$

dove gli $L_{i,n}(x)$ sono i polinomi di Lagrange di grado n sui punti x_i. Si ottiene

$$\int_a^b p(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_{i,n}(x)dx = \sum_{i=0}^{n} W_i f(x_i). \tag{6.2}$$

Dove i pesi W_i sono calcolati per integrazione dei polinomi $L_{i,n}(x)$ $i = 0, \ldots, n$. Procediamo alla loro integrazione. Effettuiamo un cambio di variabile di integrazione da $x \in [a, b]$ a $t \in [0, n]$ mediante la $x = a + ht$ con $h = \frac{b-a}{n}$.

$$\int_a^b L_{i,n}(x)dx = \int_a^b \prod_{j=0, i \neq j}^{n} \frac{x - x_j}{x_i - x_j} dx$$

$$= h \int_0^n \prod_{j=0, i \neq j}^{n} \frac{a + ht - a - hj}{a + hi - a - hj} dt$$

$$= h \int_0^n \prod_{j=0, i \neq j}^{n} \frac{t - j}{i - j} dt$$

Allora:

$$\int_a^b p(x)dx = h \sum_{i=0}^{n} f(x_i) w_i$$

avendo posto

$$w_i = \int_0^n \prod_{j=0, i\neq j}^{n} \frac{t-j}{i-j} dt; \tag{6.3}$$

e i W_i della (6.2) sono dati dagli $h \cdot w_i$ per $i = 0, \ldots, n$.
Si osservi che i coefficienti o pesi w_i dipendono solamente da n ed in particolare non dipendono dalla funzione $f(x)$ che deve essere integrata, e nemmeno dagli estremi a e b di integrazione.

Un modo alternativo per determinare i pesi W_i è quello dei "coefficienti incogniti". Qui i coefficienti o pesi sono determinati richiedendo che R_T sia uguale a zero per le funzioni

$$f(x) = x^k \quad k = 0, \ldots, n.$$

Queste richieste portano ad un sistema lineare di $n+1$ equazioni nelle $n+1$ incognite W_i.

Le formule di Newton-Cotes più comunemente usate sono quelle per $n = 1$ (formula dei Trapezi) ed $n = 2$ (formula di Simpson); nelle prossime sezioni deriveremo queste formule.

6.1.1 Formula dei Trapezi

Si procede al calcolo dei w_i per $i = 0, 1$ (caso $n = 1$) dalla (6.3).

$$w_0 = \int_0^1 \frac{t-1}{0-1} dt = \int_0^1 (1-t) dt = \left[-\frac{t^2}{2} + t \right]_0^1 = \frac{1}{2};$$

$$w_1 = \int_0^1 \frac{t-0}{1-0} dt = \int_0^1 t dt = \left[\frac{t^2}{2} \right]_0^1 = \frac{1}{2}.$$

Allora

$$\int_a^b p_1(x) dx = \frac{h}{2} \left[f(x_0) + f(x_1) \right]$$

e quindi

$$\int_a^b f(x) dx = \frac{h}{2} \left[f(a) + f(b) \right] + R_T.$$

La $f(x)$ è approssimata da una retta passante per i punti $(a, f(a))$ e $(b, f(b))$; cioè l'integrale è approssimato dall'area del trapezio rappresentato in Fig.6.1

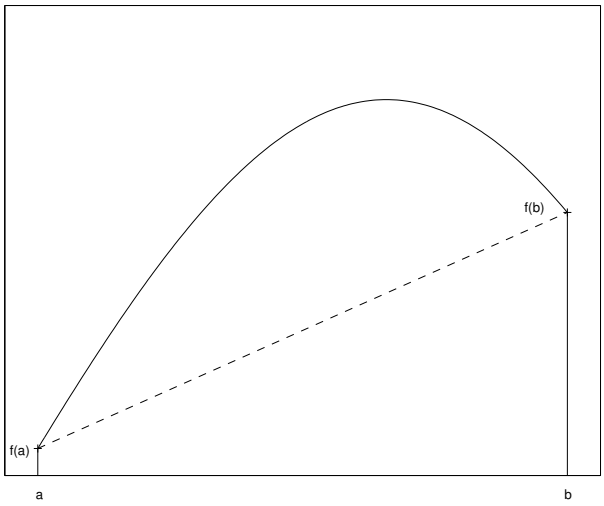

Figura 6.1: Interpolazione lineare per formula dei trapezi.

6.1.2 Formula di Simpson

Si procede al calcolo dei w_i per $i = 0, 1, 2$ (caso $n = 2$) dalla (6.3).

$$w_0 = \int_0^2 \frac{t-1}{0-1}\frac{t-2}{0-2}dt = \frac{1}{2}\int_0^2 (t^2 - 3t + 2)dt = \frac{1}{2}\left[\frac{t^3}{3} - \frac{t^2}{2} + 2t\right]_0^2 =$$

$$\frac{1}{2}\left[\frac{8}{3} - \frac{12}{2} + 4\right] = \frac{1}{3};$$

$$w_1 = \int_0^2 \frac{t-0}{1-0}\frac{t-2}{1-2}dt = -\int_0^2 (t^2 - 2t)dt = -\left[\frac{t^3}{3} - 2\frac{t^2}{2}\right]_0^2 = -\left[\frac{8}{3} - 4\right] = \frac{4}{3};$$

$$w_2 = \int_0^2 \frac{t-0}{2-0}\frac{t-1}{2-1}dt = \frac{1}{2}\int_0^2 (t^2 - t)dt = \frac{1}{2}\left[\frac{t^3}{3} - 2\frac{t^2}{2}\right]_0^2 = \frac{1}{2}\left[\frac{8}{3} - \frac{4}{2}\right] = \frac{1}{3};$$

e quindi

$$\int_a^b p_2(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

cioè

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] + R_T.$$

La $f(x)$ è approssimata da una parabola passante per $(a, f(a))$, $(\frac{a+b}{2}, f(\frac{a+b}{2}))$ e $(b, f(b))$ (vedi Fig 6.2).

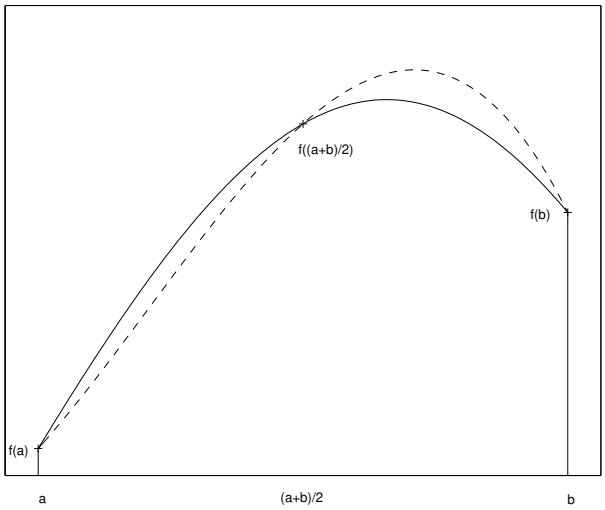

Figura 6.2: Interpolazione quadratica per formula di Simpson.

Esempio 6.1 *Applichiamo le formule dei Trapezi e di Simpson per il calcolo di* $\int_0^1 e^{-x^2}dx$*, che è un esempio di integrale non risolubile tramite il calcolo della funzione primitiva. Il valore esatto alle prime 6 cifre significative è* 0.746824.

Applichiamo la formula dei Trapezi:

$$\int_0^1 e^{-x^2}dx \simeq \frac{h}{2}(e^0 + e^{-1}) = \frac{1}{2}(1 + 0.367879) = 0.683939.$$

Applichiamo la formula di Simpson:

$$\int_0^1 e^{-x^2}dx \simeq \frac{h}{3}(e^0 + 4e^{-1/4} + e^{-1}) = \frac{1}{6}(1 + 4 \cdot 0.778801 + 0.367879) = 0.747180.$$

6.1.3 Grado di Precisione ed Errore di Integrazione

Definizione 6.1 *Una formula di quadratura si dice che ha grado di precisione* n *se fornisce il valore esatto dell'integrale per polinomi di grado* $\leq n$*.*

Teorema 6.1 *Il grado di precisione delle formule di quadratura di Newton-Cotes è* $\mu = n$ *se* n *è dispari,* $\mu = n + 1$ *se* n *è pari.*

Dim.
Ci limiteremo a dimostrare che $\mu \geq n$ nel primo caso e $\mu \geq n + 1$ nel secondo. Poiché n è il grado del polinomio interpolante con cui viene costruita la formula di quadratura, il grado di precisione della formula è almeno n. Se $f(x)$ è un

polinomio di grado $n+1$, allora $f^{(n+1)}(x) = costante$, e se n è pari, dalla formula dell'errore di integrazione (6.1) si ha

$$R_T = \int_a^b \omega(x) \frac{f^{(n+1)}(\xi_x)}{(n+1)!} dx = \frac{costante}{(n+1)!} \int_a^b \omega(x) dx = 0$$

e che valga zero è dato dal fatto che $\omega(x)$ è definita su $n+1$ punti equispaziati ed è quindi una funzione dispari rispetto al punto centrale, da cui il suo integrale è nullo; questo implica che il grado di precisione è almeno $n+1$. ⊙

Teorema 6.2 *Si considerino le formule di Newton-Cotes; allora se n è dispari ed* $f(x) \in C^{n+1}_{[a,b]}$*, si ha:*

$$R_T = \frac{h^{n+2} f^{(n+1)}(\eta)}{(n+1)!} \int_0^n t(t-1)\cdots(t-n)dt; \tag{6.4}$$

se n è pari ed $f(x) \in C^{n+2}_{[a,b]}$*, si ha:*

$$R_T = \frac{h^{n+3} f^{(n+2)}(\eta)}{(n+2)!} \int_0^n t^2(t-1)\cdots(t-n)dt. \tag{6.5}$$

Dim.
⊙

Errore della formula dei Trapezi

Nel caso $n = 1$ sarà:

$$R_T = \frac{h^3 f^{(2)}(\eta)}{2} \int_0^1 t(t-1)dt$$

dove $h = b - a$; si ha:

$$\int_0^1 t(t-1)dt = \left[\frac{t^3}{3} - \frac{t^2}{2}\right]_0^1 = -\frac{1}{6}$$

e quindi

$$R_T = -\frac{1}{12} h^3 f^{(2)}(\eta). \quad M = 1$$

Errore della formula di Simpson

Nel caso $n = 2$ sarà:

$$R_T = \frac{h^5 f^{(4)}(\eta)}{4!} \int_0^2 t^2(t-1)(t-2)dt$$

dove $h = \frac{b-a}{2}$; si ha:

$$\int_0^2 t^2(t-1)(t-2)dt = \int_0^2 (t^4 - 3t^3 + 2t^2)dt = \left[\frac{t^5}{5} - \frac{3}{4}t^4 + \frac{2}{3}t^3\right]_0^2$$
$$= \left[\frac{32}{5} - 12 + \frac{16}{3}\right] = -\frac{4}{15}h^5$$

e quindi

$$R_T = -\frac{1}{90}h^5 f^4(\eta). \quad n = 2$$

Questa espressione dell'errore mostra come la formula di Simpson (così come tutte quelle con n pari) sia esatta non solo per polinomi di secondo grado, come ci si aspetterebbe, ma anche per polinomi cubici.

Esempio 6.2 *Si calcoli l'integrale della seguente funzioni polinomiale in* $[0,1]$*:*

$$p(x) = 8x^3 - 12x^2 + 3x + 1,$$

applicando la formula di Simpson.

$$\int_0^1 (8x^3 - 12x^2 + 3x + 1)dx \simeq \frac{h}{3}(p(x_0) + 4p(x_1) + p(x_2)) = \frac{1}{3}\frac{1}{2}(1 + 4\frac{1}{2} + 0) = \frac{1}{2}.$$

Osservazione 6.1 *Se con s si indica il denominatore comune per i pesi frazionari w_i, e indichiamo con $\sigma_i := sw_i$ $i = 0, \ldots, n$ i conseguenti numeri interi, allora la formula di Newton-Cotes si può riscrivere come:*

$$\int_a^b f(x)dx \simeq \frac{b-a}{ns}\sum_{i=0}^{n}\sigma_i f(x_i).$$

La seguente Tab.6.1 riporta le formule di Newton-Cotes per $n = 1, \ldots, 8$; si osservi che $\sigma_i \equiv \sigma_{n-i}$ per $i = 0, \ldots, n/2$. Per $n \geq 8$ alcuni coefficienti σ_i diventano negativi e aumentano in modulo. Questo fatto, che rende instabile la formula dal punto di vista della propagazione degli errori (cancellazione numerica), insieme alla considerazione che l'aumento del grado di precisione non significa necessariamente la convergenza della formula di quadratura all'integrale quando la funzione non è polinomiale, rende interessanti le formule di Newton-Cotes soltanto per $n \leq 7$.

6.1.4 Formule Composte

Le formule composte consistono nel considerare dei polinomi a tratti come funzioni approsimanti della funzione integranda nell'intervallo $[a,b]$. In pratica tale intervallo viene suddiviso in sottointervalli, in generale di uguale ampiezza, $[x_i, x_{i+1}]$ $i = 0, \ldots, m-1$, e su ciasuno di essi si applica una formula di quadratura di grado basso.

n	σ_i	ns	Errore	Nome
1	1 ...	2	$-\frac{1}{12}h^3 f^{(2)}(\eta)$	Trapezi
2	1 4 ...	6	$-\frac{1}{90}h^5 f^{(4)}(\eta)$	Simpson
3	1 3 ...	8	$-\frac{3}{80}h^5 f^{(4)}(\eta)$	3/8
4	7 32 12 ...	90	$-\frac{8}{945}h^7 f^{(6)}(\eta)$	Milne
5	19 75 50 ...	288	$\cdots h^7 f^{(6)}(\eta)$	
6	41 216 27 272 ...	840	$\cdots h^9 f^{(8)}(\eta)$	Weddle
7	751 3577 1323 2989 ...	17280	$\cdots h^9 f^{(8)}(\eta)$	
8	989 5888 -928 10496 -4540 ...	28350	$\cdots h^{11} f^{(10)}(\eta)$	

Tabella 6.1: Formule Newton-Cotes per $n = 1, \ldots, 8$.

Formula dei Trapezi

Se si usa la formula dei trapezi sul sottointervallo $[x_i, x_{i+1}]$ si ottiene:

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}(f(x_i) + f(x_{i+1})) - \frac{1}{12}h^3 f^{(2)}(\eta_i)$$

con $h = x_{i+1} - x_i$ e $x_i \leq \eta_i \leq x_{i+1}$. L'errore è detto **errore locale di troncamento.**
Per l'intero intervallo $[a, b]$ si ottiene:

$$\int_a^b f(x)dx = \sum_{i=0}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx$$

$$= h\left(\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{m-1}) + \frac{1}{2}f(x_m)\right) + R_T$$

(vedi Fig.6.3). L'errore globale di troncamento R_T è la somma degli errori locali. Si noti che $b - a = mh$, e perciò

$$R_T = -\frac{b-a}{12}h^2 \frac{\sum_{i=0}^{m-1} f^{(2)}(\eta_i)}{m}.$$

Se $f^{(2)}$ è continua in $]a, b[$, allora esiste un η in questo intervallo, tale che

$$\frac{1}{m}\sum_{i=0}^{m-1} f^{(2)}(\eta_i) = f^{(2)}(\eta).$$

Da cui l'errore di troncamento globale per la formula dei trapezi composta è

$$R_T = -\frac{b-a}{12}h^2 f^{(2)}(\eta); \tag{6.6}$$

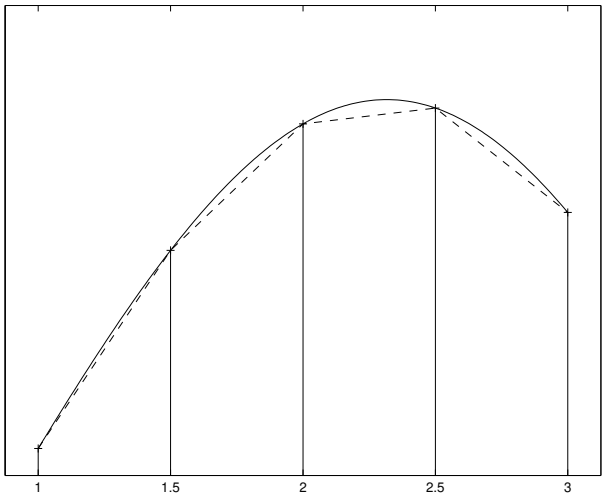

Figura 6.3: Interpolazione lineare a tratti per formula composta dei trapezi.

in particolare si osservi che vale

$$\lim_{h\to 0} R_T = 0.$$

Perciò più fine è la suddivisione dell'intervallo e migliore risulta l'approssimazione dell'integrale.

Esempio 6.3 *Si vuole determinare il passo h da utilizzare nella formula dei trapezi composta, affinché l'$\int_0^1 \frac{1}{1+x}dx$ sia approssimato alla tolleranza 0.5×10^{-3}. Si considera la formula (6.6) per l'errore di integrazione dei trapezi composta e si cerca un limite superiore per la $f^{(2)}$ in $[a,b]$; nel caso specifico sarà:*

$$f^{(1)}(x) = -\frac{1}{(1+x)^2} \qquad f^{(2)}(x) = \frac{2}{(1+x)^3}.$$

Allora

$$\max_{0\le x\le 1} f^{(2)}(x) = f^{(2)}(0) = 2$$

e ricordando che $h = \frac{b-a}{m} = \frac{1}{m}$ si ha:

$$\left|\frac{b-a}{12}h^2 f^{(2)}(\eta)\right| \le \frac{1}{12}h^2 2 = \frac{1}{6m^2} \le 0.5\times 10^{-3}$$

da cui

$$m^2 \ge \frac{1}{3}\times 10^3 \quad \textit{e quindi} \quad m \ge 18.25.$$

Segue che per approssimare l'integrale dato alla tolleranza fissata è sufficiente usare $m = 19$, cioè il più piccolo intero che soddisfa $m \ge 18.25$.

Formula di Simpson

Se $m = 2k$, cioè l'intervallo $[a, b]$ è suddiviso in un numero pari di sottointervalli, allora l'integrale in ogni sottointervallo $[x_{2i}, x_{2i+2}]$ può essere calcolato con la formula di Simpson:

$$\int_{x_{2i}}^{x_{2i+2}} f(x)dx = \frac{h}{3}[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})] - \frac{1}{90}h^5 f^{(4)}(\eta_i)$$

con $h = x_{2i+2} - x_{2i}$ e $x_{2i} \leq \eta_i \leq x_{2i+2}$.
Per l'intero intervallo $[a, b]$ si ottiene:

$$\int_a^b f(x)dx = \sum_{i=0}^{k-1} \int_{x_{2i}}^{x_{2i+2}} f(x)dx$$

$$= \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{m-1}) + f(x_m)] + R_T$$

(vedi Fig.6.4), dove

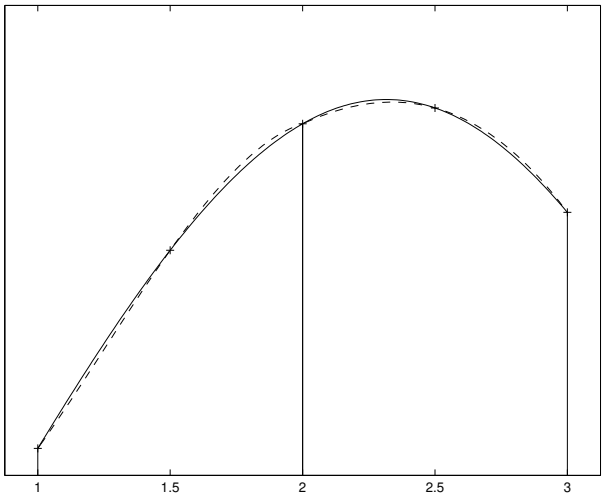

Figura 6.4: Interpolazione quadratica a tratti per formula composta di Simpson.

$$R_T = -\frac{h^5}{90}\sum_{i=0}^{k-1} f^{(4)}(\eta_i).$$

Per le stesse argomentazioni usate per la formula dei trapezi, questo si può scrivere

$$R_T = -\frac{b-a}{180}h^4 f^{(4)}(\eta); \tag{6.7}$$

dove $\eta \in]a, b[$ con $f^{(4)}$ continua in $[a, b]$.

Esempio 6.4 *Si vuole determinare il passo h da utilizzare nella formula di Simpson composta, affinché l'$\int_0^1 \frac{1}{1+x}dx$ sia approssimato alla tolleranza 0.5×10^{-3}. Si considera la formula (6.7) per l'errore di integrazione di Simpson composta e si cerca un limite superiore per la $f^{(4)}$ in $[a,b]$; nel caso specifico sarà:*

$$f^{(1)}(x) = -\frac{1}{(1+x)^2}, \qquad f^{(2)}(x) = \frac{2}{(1+x)^3},$$
$$f^{(3)}(x) = -\frac{6}{(1+x)^4}, \qquad f^{(4)}(x) = \frac{24}{(1+x)^5}.$$

Allora

$$\max_{0\le x\le 1} f^{(4)}(x) = f^{(4)}(0) = 24$$

e ricordando che $h = \frac{b-a}{2k} = \frac{1}{2k}$ si ha:

$$\left|\frac{b-a}{180}h^4 f^{(4)}(\eta)\right| \le \frac{1}{180}h^4 24 = \frac{2}{15}\frac{1}{16k^4} = \frac{1}{120k^4} \le 0.5 \times 10^{-3}$$

da cui

$$k^4 \ge \frac{2}{120} \times 10^3 \quad \textit{e quindi} \quad k \ge 2.0205.$$

Segue che per approssimare l'integrale dato alla tolleranza fissata è sufficiente usare $k = 3$, cioè il più piccolo intero che soddisfa $k \ge 2.0205$.

6.1.5 Metodi Adattivi

Nelle applicazioni pratiche, chi è interessato a calcolare numericamente l'integrale di una funzione, solitamente ha la necessità che l'approssimazione sia contenuta in una tolleranza fissata; cioè se indichiamo con I_A l'integrale approssimato con una formula di quadratura e con *tol* la tolleranza fissata, l'utente vuole che

$$\left|\int_a^b f(x)dx - I_A\right| \le tol.$$

A meno di casi particolari, come quello dell'esempio, quanto visto fino ad ora non permette di soddisfare tale richiesta. Per fare ciò è necessario un criterio per determinare una stima dell'errore di integrazione; ci sono due tipi di approcci: non adattivo e adattivo. In un approccio non adattivo i punti in cui si valuta la funzione $f(x)$ sono scelti senza tener conto del comportamento della $f(x)$; in questo caso può accadere che se la funzione ha un comportamento oscillante in una parte dell'intervallo d'integrazione, questo comporti un numero elevato di punti su tutto l'intervallo, anche dove la funzione ha un comportamento lineare. Invece in un approccio adattivo il numero di punti viene scelto in base

al comportamento della funzione. Si suddivide l'intervallo di integrazione in sottointervalli e si applica ricorsivamente a questi una formula di quadratura, sfruttando un stima dell'errore di integrazione per il test di arresto. La funzione integranda viene così valutata in pochi punti nei sottointervalli in cui ha un andamento regolare e in molti punti negli intervalli dove sono presenti irregolarità.
Nella prossima sezione vedremo una tecnica nota come estrapolazione di Richardson che deriva da un risultato più generale da cui si può estrarre una stima per l'errore di integrazione.

Estrapolazione di Richardson

Con **estrapolazione di Richardson** ci si riferisce ad una tecnica generale che permette di ottenere dall'applicazione di due formule di integrazione composte con passi rispettivamente h ed $\frac{h}{2}$ un valore di approssimazione per l'integrale, più preciso dei due precedenti. Vediamo questa tecnica sia nel caso dei trapezi composti che nel caso di Simpson composto al fine di progettare poi una formula di integrazione adattiva.

Caso Trapezi

Teorema 6.3 *Si indichi con $T(h)$ la formula dei trapezi composta con $h = \frac{b-a}{m}$; allora vale*

$$\int_a^b f(x)dx - T(h) = \frac{h^2}{12}(f^{(1)}(b) - f^{(1)}(a)) - \frac{h^4}{720}(f^{(3)}(b) - f^{(3)}(a)) +$$

$$\frac{h^6}{30240}(f^{(5)}(b) - f^{(5)}(a)) + \cdots + c_{2k}h^{2k}(f^{(2k-1)}(b) - f^{(2k-1)}(a)) + O(h^{2k+2})$$

nelle ipotesi che la f sia derivabile in $[a, b]$ almeno $2k + 2$ volte.

Corollario 6.1 *Assumendo che la $f(x)$ sia derivabile almeno 4 volte su $[a, b]$, si ha:*

$$\int_a^b f(x)dx - T(h/2) \simeq \frac{T(h/2) - T(h)}{3} + O(h^4).$$

Dim.

Dal teorema si ha:

$$\int_a^b f(x)dx - T(h) = c_2h^2 + O(h^4)$$

e se il passo usato è $h/2$

$$\int_a^b f(x)dx - T(h/2) = c_2(h/2)^2 + O(h^4)$$

e questo in quanto il coefficiente c_2 è indipendente da h. Eliminando il termine in h^2 fra le due equazioni, si ottiene:

$$4\left(\int_a^b f(x)dx - T(h/2)\right) = \int_a^b f(x)dx - T(h) + O(h^4)$$

da cui

$$3\left(\int_a^b f(x)dx - T(h/2)\right) = T(h/2) - T(h) + O(h^4)$$

e quindi ciò che si voleva provare. $\odot$

Osservazione 6.2 *Dal Corollario si può ricavare una formula che fornisce un'approssimazione dell'integrale del quarto ordine, cioè*

$$\int_a^b f(x)dx = \frac{4T(h/2) - T(h)}{3} + O(h^4).$$

Questo metodo di eliminare la potenza più bassa di h nello sviluppo dell'errore, usando due differenti passi h ed $h/2$, è noto come estrapolazione di Richardson. Il metodo è attraente, perché, una volta calcolato $T(h)$ e $T(h/2)$, si può ottenere l'approssimazione di ordine superiore ad un extra costo inesistente.

Esempio 6.5 *Si stima l'$\int_0^1 e^{\sin x}dx$ usando la formula composta dei trapezi con $h = 1, 0.5, 0.25$. La stima viene poi migliorata mediante estrapolazione di Richardson. Il valore esatto dell'integrale è* 1.63186961.

h	$T(h)$	$Estrap.$	$\lvert \int_a^b f dx - T(h)\rvert$	$\lvert \int_a^b f dx - Estrap.\rvert$
1	1.65988841		0.0280188	
0.5	1.63751735	1.63006033	0.0056477	0.0018093
0.25	1.63321154	1.63177627	0.0013419	0.0000934

Tabella 6.2: Esempio di estrapolazione di Richardson

Osservazione 6.3 *Il Corollario fornisce una stima del termine principale dell'errore e quindi fornisce un metodo pratico per raggiungere l'obiettivo che ci si*

è posti. Si può progettare un metodo iterativo in cui si dimezza il passo fino a che

$$\left|\frac{T(h/2) - T(h)}{3}\right| \leq tol;$$

quando questo si verifica, per il Corollario, sarà anche

$$\left|\int_a^b f(x)dx - T(h/2)\right| \leq tol.$$

Questo metodo iterativo può essere applicato in modo adattivo. Si rimanda la descrizione di un tale metodo al paragrafo successivo.

Caso Simpson

A differenza del caso trapezi, qui non esiste un teorema che possa essere utilizzato per sapere l'ordine dello schema che si ottiene applicando alle formule di Simpson composte l'estrapolazione di Richardson.
Sia $h = (b-a)/2$, allora si calcoli un Simpson con passo h ($S(h)$) e un Simpson composto con passo $h/2 = (b-a)/4$ ($S(h/2)$ o analogamente due Simpson semplici con passo $h/2$); avremo

$$\int_a^b f(x)dx = S(h) - \frac{h^5}{90}f^{(4)}(\xi) \quad \xi \in]a,b[$$

$$\begin{aligned}\int_a^b f(x)dx &= S(h/2) - \frac{b-a}{180}(h/2)^4 f^{(4)}(\eta) \quad \eta \in]a,b[\\ &= S(h/2) - \frac{1}{16}\frac{h^5}{90}f^{(4)}(\eta). \end{aligned} \tag{6.8}$$

Nelle ipotesi che $f^{(4)}(\xi) \simeq f^{(4)}(\eta)$ si può procedere ad eliminare il termine h^4 fra le due equazioni, ottenendo

$$S(h) - S(h/2) \simeq \frac{h^5}{90}f^{(4)}(\eta)(1 - \frac{1}{16})$$

da cui

$$\frac{1}{15}[S(h) - S(h/2)] \simeq \frac{1}{16}\frac{h^5}{90}f^{(4)}(\eta)$$

e sostituendo questa nella (6.8) si ottiene

$$\begin{aligned}\int_a^b f(x)dx &\simeq S(h/2) - \frac{1}{15}[S(h/2) - S(h)] \\ &= \frac{1}{15}[16 \cdot S(h/2) - S(h)] \end{aligned} \tag{6.9}$$

che è una nuova formula che fornisce un'approssimazione dell'integrale di ordine superiore. Dalla prima relazione (6.9) si ottiene

$$\left| \int_a^b f(x)dx - S(h/2) \right| \simeq \left| \frac{1}{15}[S(h/2) - S(h)] \right| \tag{6.10}$$

e quindi la possibilità di innescare un procedimento iterativo in cui si dimezza il passo fino a che

$$\left| \frac{1}{15}[S(h/2) - S(h)] \right| \leq tol;$$

quando ciò si verifica sarà anche

$$\left| \int_a^b f(x)dx - S(h/2) \right| \leq tol.$$

Si può ora procedere alla progettazione di un metodo adattivo.

Metodo di Simpson adattivo

L'utente di una routine che implementa un metodo adattivo specifica l'intervallo $[a, b]$, fornisce una routine per la valutazione di $f(x)$ per $x \in [a, b]$ e sceglie una tolleranza tol. La routine tenta di calcolare un valore approssimato I_A così che:

$$\left| \int_a^b f(x)dx - I_A \right| \leq tol.$$

La routine può determinare che la richiesta tolleranza non sia ottenibile nei limiti definiti di massimo numero di livelli di ricorsione e ritornare la miglior approssimazione da lei fornibile. Per quel che riguarda l'efficienza di una routine di questo tipo, è noto che il costo maggiore di calcolo consiste nella valutazione della funzione integranda $f(x)$.
Durante l'esecuzione, ogni intervallo viene determinato per bisezione di un intervallo ottenuto precedentemente durante il calcolo. Il numero effettivo di sottointervalli, così come la loro posizione e ampiezza, dipende dalla funzione $f(x)$ e dalla tolleranza tol. Lo schema più classico applica due differenti formule di quadratura ad ogni sottointervallo $[x_i, x_{i+1}]$, per esempio la formula di Simpson semplice ($S(h_i)$ con $h_i = x_{i+1} - x_i$) e la formula di Simpson composta con $m = 4$ ($S(h_i/2)$)(che equivale a due Simpson semplici applicati rispettivamente a $[x_i, x_i + h_i/2]$ e $[x_i + h_i/2, x_{i+1}]$). Entrambe $S(h_i)$ ed $S(h_i/2)$ sono approssimazioni per

$$I_{Ai} = \int_{x_i}^{x_{i+1}} f(x)dx.$$

L'idea base del metodo adattivo è utilizzare le due approssimazioni per ottenere una stima dell'errore di integrazione (stima (6.10)). Se la tolleranza è raggiunta, $S(h_i/2)$ o una combinazione delle due approssimazioni (estrapolazione di Richardson) viene presa come valore dell'integrale su quell'intervallo. Se la tolleranza non è raggiunta, il sottointervallo viene suddiviso a metà e il processo viene ripetuto su ognuno dei sottointervalli più piccoli.
Per ridurre il numero totale di valutazioni della funzione integranda, solitamente si organizza che le due formule utilizzate richiedano valori della funzione in punti comuni; per esempio, con la formula di Simpson, $S(h_i/2)$ richiede cinque valori della funzione, tre dei quali sono anche usati in $S(h_i)$. Come conseguenza, processare un nuovo sottointervallo richiede solo due nuove valutazioni di funzione.

La Fig.6.5 mostra gli intervalli considerati, e i valori della funzione nei loro estremi, nell'applicare il metodo di Simpson adattivo alla funzione

$$f(x) = e^{\sqrt{x}} \sin x + 2x - 4 \quad x \in [0, 20]$$

richiedendo una tolleranza 1.0×10^{-6}; il valore determinato è 294.872 ed è stato ottenuto effettuando 361 valutazioni della $f(x)$.

La Fig.6.6 mostra gli intervalli considerati e i valori della funzione nei loro

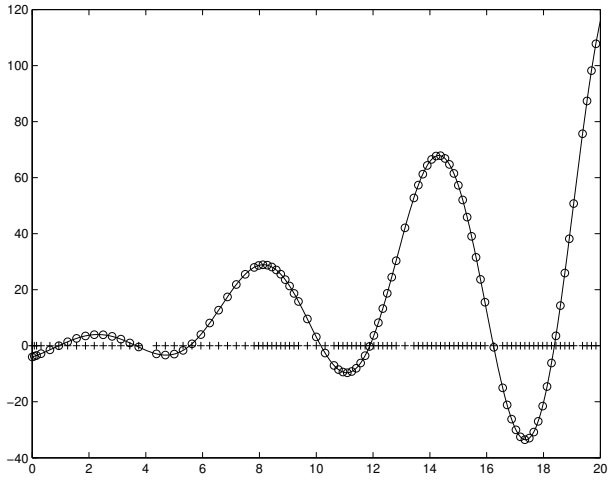

Figura 6.5: Simpson adattivo: 361 valutazioni di funzione.

estremi, nell'applicare il metodo di Simpson adattivo alla funzione

$$f(x) = \frac{32}{1 + 1024x^2} \quad x \in [0, 4]$$

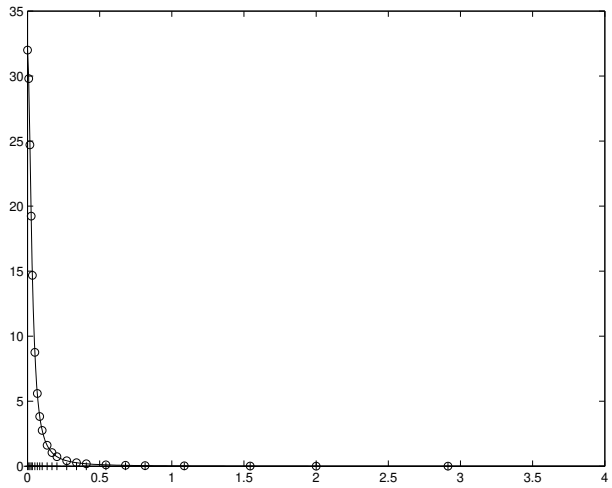

Figura 6.6: Simpson adattivo: 93 valutazioni di funzione.

richiedendo una tolleranza 1.0×10^{-6}; il valore determinato è 1.563 ed è stato ottenuto effettuando 93 valutazioni della $f(x)$. Si noti che in ogni intervallo in cui si è raggiunta la tolleranza la funzione è stata valutata 3 volte oltre ai due estremi.
Per tali elaborazioni si è utilizzata la function **quad** presente nel sistema MATLAB 6.5.1.

6.2 Formule di quadratura di Gauss

Se ci si svincola dalla condizione di avere i punti di integrazione equispaziati, come nella formule di Newton-Cotes, si può cercare di sceglierli, insieme ai pesi, in una maniera ottimale. In una formula del tipo

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} W_i f(x_i) + E_n \qquad n \geq 1$$

si sceglieranno i punti $x_1, x_2, \ldots, x_n$ e i pesi $W_1, W_2, \ldots, W_n$ per rendere la formula esatta per tutti i polinomi di grado $\leq 2n-1$; ogni formula in cui i punti e i pesi vengono scelti con questa finalità è detta **formula di quadratura di Gauss**.

Senza perdere in generalità continuiamo ad assumere che l'intervallo di integrazione sia $[-1, 1]$ e vediamo operativamente come ricavare formule di Gauss ad 1 e 2 punti.

Nel caso di $n = 1$ deve essere:

$$\int_{-1}^{1} f(x)dx = W_1 f(x_1) + E_1$$

che deve essere esatta quando $f(x)$ è un polinomio di grado 1 per opportuni valori di W_1 e x_1 che possiamo quindi determinare ad hoc nel seguente modo:

$$\int_{-1}^{1} (a_0 + a_1 x)dx = a_0 \int_{-1}^{1} dx + a_1 \int_{-1}^{1} x dx = a_0 W_1 + a_1 W_1 x_1$$

e uguagliando i coefficienti

$$\begin{cases} W_1 & = \int_{-1}^{1} dx \\ W_1 x_1 & = \int_{-1}^{1} x dx \end{cases} \qquad \begin{cases} W_1 & = 2 \\ W_1 x_1 & = 0 \end{cases} \qquad \begin{cases} W_1 & = 2 \\ x_1 & = 0 \end{cases}$$

così ché la formula di Gauss ad 1 punto è data da:

$$\int_{-1}^{1} f(x)dx = 2f(0) + E_1.$$

Si determinano ora i punti e i pesi per la formula di Gauss a 2 punti:

$$\int_{-1}^{1} f(x)dx = W_1 f(x_1) + W_2 f(x_2) + E_2.$$

I pesi W_1 e W_2 ed i punti x_1 e x_2 devono essere scelti così ché la formula sia esatta quando $f(x)$ è un polinomio di grado 3. Eguagliando i coefficienti, si ottengono le quattro equazioni non lineari:

$$\begin{cases} 1) & W_1 + W_2 & = \int_{-1}^{1} dx = 2 \\ 2) & W_1 x_1 + W_2 x_2 & = \int_{-1}^{1} x dx = 0 \\ 3) & W_1 x_1^2 + W_2 x_2^2 & = \int_{-1}^{1} x^2 dx = 2/3 \\ 4) & W_1 x_1^3 + W_2 x_2^3 & = \int_{-1}^{1} x^3 dx = 0 \end{cases}$$

Le equazioni 2) e 4) sono soddisfatte per $W_1 = W_2$ e $x_1 = -x_2$; dalla 1) segue che $W_1 = W_2 = 1$ e dalla 3) $x_1^2 = x_2^2 = 1/3$. Allora la formula di Gauss a 2 punti è data da

$$\int_{-1}^{1} f(x)dx = f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right) + E_2.$$

In generale si potrebbero determinare i pesi e i punti per qualsiasi intero positivo n risolvendo $2n$ equazioni che derivano dall'uguagliare i coefficienti in un polinomio di grado $2n - 1$. Tuttavia le equazioni sono non lineari e la soluzione diviene già complicata per piccoli valori di n. Fortunatamente, c'è un elegante metodo matematico che fornisce i pesi ed i punti in modo abbastanza facile. Questo metodo è basato sulla teoria dei polinomi ortogonali.

Consideriamo, per il nostro problema, l'insieme dei polinomi di Legendre $\{p_0, p_1, \ldots, p_n, \ldots\}$ che ha le seguenti proprietà:

1. per ogni n, p_n è un polinomio di grado n;
2. $\int_{-1}^{1} p(x)p_n(x)dx = 0$ se $p(x)$ è un polinomio di grado minore di n.

I primi polinomi di Legendre sono:

$$p_0(x) = 1, \qquad p_1(x) = x, \qquad p_2(x) = x^2 - \frac{1}{3},$$
$$p_3(x) = x^3 - \frac{3}{5}x, \qquad p_4(x) = x^4 - \frac{6}{7}x^2 + \frac{3}{35}, \qquad \cdots.$$

Gli zeri di questi polinomi sono distinti nell'intervallo $[-1, 1]$, sono simmetrici rispetto all'origine e, la cosa più importante, sono la scelta corretta per determinare i parametri che risolvono i nostri problemi. Infatti, che i punti $x_1, x_2, \ldots, x_n$, necessari per produrre una formula di integrazione approssimata per ottenere risultati esatti per ogni polinomio di grado $\leq 2n-1$, siano gli zeri del polinomio di Legendre di grado n, si può dimostrare nel seguente modo.

Teorema 6.4 *Siano $x_1, x_2, \ldots, x_n$ gli zeri del polinomio di Legendre p_n di grado n e per ogni $i = 1, 2, \ldots, n$ siano definiti i coefficienti W_i*

$$W_i = \int_{-1}^{1} L_{i,n-1}(x)dx$$

con $L_{i,n-1}(x)$ i polinomi elementari di Lagrange di grado $n-1$ definiti sui punti non equispaziati x_j $j = 1, 2, \ldots, n$ [1]. Se $p(x)$ è un qualunque polinomio di grado $< 2n$, allora

$$\int_{-1}^{1} p(x)dx = \sum_{i=1}^{n} W_i p(x_i).$$

Dim. Si consideri prima la situazione di un polinomio $r(x)$ di grado $< n$. Riscriviamo $r(x)$ come polinomio interpolante nella forma di Lagrange di grado $n-1$ con punti di interpolazione gli zeri del polinomio di Legendre p_n di grado n. Questa rappresentazione di $r(x)$ è esatta poiché l'errore di interpolazione dipende dalla derivata n-esima di $r(x)$ che è zero. Quindi

$$\begin{aligned} \int_{-1}^{1} r(x)dx &= \int_{-1}^{1} \left[\sum_{i=1}^{n} r(x_i)L_{i,n-1}(x)\right] dx \\ &= \sum_{i=1}^{n} \left[\int_{-1}^{1} L_{i,n-1}(x)dx\right] r(x_i) \\ &= \sum_{i=1}^{n} W_i r(x_i). \end{aligned}$$

Questo verifica il risultato per polinomi di grado $< n$. Se il polinomio $p(x)$ è di grado $\leq 2n-1$ e viene diviso per il polinomio di Legendre $p_n(x)$, si ottengono due polinomi $q(x)$ ed $r(x)$ di grado $< n$

$$p(x) = q(x)p_n(x) + r(x).$$

Si osserva che:

[1] Sarà $L_{i,n-1}(x) = \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$, per $i = 1, \ldots, n$.

1. avendo $q(x)$ grado $< n$, per la propietà 2. prima vista sarà
$$\int_{-1}^{1} q(x)p_n(x)dx = 0;$$
2. poiché $r(x)$ è un polinomio di grado $< n$, per quanto visto all'inizio
$$\int_{-1}^{1} r(x)dx = \sum_{i=1}^{n} W_i r(x_i).$$
3. poiché x_i è uno zero di $p_n(x)$ per ogni $i = 1, 2, \ldots, n$, si ha
$$p(x_i) = q(x_i)p_n(x_i) + r(x_i) = r(x_i);$$

Mettendo insieme queste osservazioni si verifica che la formula è esatta per polinomi $p(x)$ di grado $\leq 2n-1$, infatti

$$\begin{array}{ll} \int_{-1}^{1} p(x)dx & = \int_{-1}^{1} [q(x)p_n(x) + r(x)]\, dx \\ & = \int_{-1}^{1} r(x)dx \\ & = \sum_{i=1}^{n} W_i r(x_i) \\ & = \sum_{i=1}^{n} W_i p(x_i). \end{array}$$

$\odot$

Osservazione 6.4 *I polinomi di Legendre soddisfano la relazione ricorrente:*
$$p_{n+1}(x) = \frac{2n+1}{n+1} x p_n(x) - \frac{n}{n+1} p_{n-1}(x) \quad n \geq 0$$
con $p_0(x) = 1, p_{-1}(x) = 0$.

Osservazione 6.5 *Il calcolo dei punti e dei pesi va fatto per via numerica non essendo disponibili, in generale, in forma esplicita. Solitamente sono forniti in tabelle pronte all'uso. I pesi* W_i *necessari per la formula di quadratura possono essere calcolati dall'espressione data nel seguente Teorema; questo nel caso li si volesse avere alla precisione di macchina.*

Teorema 6.5 *I punti di interpolazione* x_i *per le formule di Gauss a n punti sono gli zeri del polinomio* $p_n(x)$ *di Legendre di grado n, ed i pesi sono dati da:*
$$W_i = \frac{1}{[p_n'(x_i)]^2} \frac{2}{1 - x_i^2} \quad i = 1, 2, \ldots, n.$$
Inoltre l'errore di troncamento delle formule di Gauss a n punti è dato da
$$E_n = \frac{2^{2n+1}[n!]^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi) \qquad \xi \in (-1, 1).$$

Esercizio 6.1 *Applicare il Teorema precedente per riottenere punti e pesi per le formule di Gauss ad 1 e 2 punti.*

6.2.1 Formule di Gauss Composte

Come già visto nel caso delle formule composte di Newton-Cotes, l'idea è quella di partizionare l'intervallo di integrazione $[a, b]$ in m sottointervalli di uguale ampiezza e nell'applicare le formule di quadratura di Gauss a n punti, in ogni sottointervallo. Se determiniamo $x_i = a+ih$ per $i = 0, \ldots, m$, con $h = (b-a)/m$, l'integrale si scriverà come

$$\int_a^b f(x)dx = \sum_{i=0}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx.$$

Prima di poter applicare ad uno di questi integrali una formula di Gauss si dovrà trasformare l'intervallo di integrazione da $[x_i, x_{i+1}]$ a $[-1, 1]$. Si deve cioè applicare la seguente trasformazione lineare che mappa $x \in [x_i, x_{i+1}]$ in $t \in [-1, 1]$:

$$x = \frac{x_{i+1} + x_i}{2} + \frac{x_{i+1} - x_i}{2} t$$

ottenendo

$$\int_{x_i}^{x_{i+1}} f(x)dx = \int_{-1}^{1} f\left(\frac{x_{i+1} + x_i}{2} + \frac{x_{i+1} - x_i}{2} t\right) \frac{x_{i+1} - x_i}{2} dt.$$

6.3 Confronto Newton-Cotes e Gauss

Si vogliono fare alcune semplici considerazioni sulle formule viste al fine di una maggior comprensione dei pro e dei contro nell'usarle in pratica.

- Le formule di Gauss sono formule di integrazione *aperte*, cioè tutti i punti di integrazione sono strettamente interni all'intervallo di integrazione, al contrario delle formule di Newton-Cotes che sono *chiuse*, cioè gli estremi sono inclusi. Questo implica che le formule di Gauss possano essere usate per integrare funzioni singolari negli estremi, come per esempio $\int_0^1 \log(x)\cos(x)dx$, mentre le formule di Newton-Cotes no!

- Le formule di Gauss sono per costruzione quasi il doppio più precise delle formule di Newton-Cotes.

- Se gli estremi di integrazione sono numeri razionali, allora le formule di Newton-Cotes avranno come nodi e pesi dei numeri razionali, mentre in quelle di Gauss saranno numeri irrazionali; questo ptrebbe comportare una maggior propagazione degli errori di arrotondamento nelle formule gaussiane.

- I pesi delle formule gaussiane sono sempre positivi, mentre quelli delle formule di Newton-Cotes lo sono solo per $n \leq 7$, e quindi per un noto risultato si ha che le formule di Gauss convergono per $n \to \infty$, mentre le formule di Newton-Cotes possono non convergere.

- Gli n nodi di una formula di Gauss con $n \geq 1$ non sono un sottoinsieme degli m nodi di alcuna formula di Gauss con $m > n$. Questo è un vero inconveniente nelle formule di Gauss in quanto risulta impossibile in un metodo adattivo e iterativo poter usufruire dei valori della funzione integranda precedentemente calcolati, cosa invece sempre possibile con le formule di Newton-Cotes.

6.4 Applicazione: lunghezza di una curva

Sia data una curva di Bézier $\mathbf{C}(t)$ con $t \in [0, 1]$. Siamo interessati a determinare la lunghezza di tale curva e a progettare curve di lunghezza assegnata mantenendo la stessa forma.
È noto che la lunghezza di una curva in questa forma è espressa da

$$L = \int_0^1 \|\mathbf{C}'(t)\| dt$$

$$= \int_0^1 \sqrt{x'(t)^2 + y'(t)^2} dt$$

con $x(t)$ ed $y(t)$ le componenti della $\mathbf{C}(t)$ e $x'(t)$ ed $y'(t)$ le componenti della $\mathbf{C}'(t)$. Infatti l'infinitesimo di curva sarà dato da $\delta\mathbf{C}(t)$ che approssimeremo con $\mathbf{C}'(t)\delta t$ e la cui lunghezza sarà $\|\mathbf{C}'(t)\|\delta t$.
Per calcolare tale lunghezza sarà necessario utilizzare una formula di integrazione numerica e se si desidera l'approssimazione con una certa tolleranza si può utilizzare il metodo di Simpson adattivo.
Supponiamo di aver progettato la curva $\mathbf{C}(t)$ e di aver stimato che la sua lunghezza è L_C. Siamo interessati ad una curva $\mathbf{G}(t)$ con la stessa forma della $\mathbf{C}(t)$, ma di lunghezza fissata L_G. A tal fine si noti che se la curva $\mathbf{C}(t)$ viene scalata di un fattore s la sua lunghezza viene scalata dello stesso fattore; infatti sia

$$\mathbf{G}(t) = \begin{pmatrix} s \cdot x(t) \\ s \cdot y(t) \end{pmatrix}$$

la curva $\mathbf{C}(t)$ scalata del fattore s, allora sarà

$$L_G = \int_0^1 \sqrt{[s \cdot x'(t)]^2 + [s \cdot y'(t)]^2} dt = s \int_0^1 \sqrt{x'(t)^2 + y'(t)^2} dt = sL_C.$$

Segue che assegnata la lunghezza L_G sarà sufficiente calcolare $s := L_G/L_C$ e scalare la curva $\mathbf{C}(t)$ per tale fattore ottenendo una curva $\mathbf{G}(t)$ della forma della $\mathbf{C}(t)$ e di lunghezza desiderata. La Fig6.7 mostra una curva polinomiale cubica a tratti C^1 ottenuta per interpolazione; la curva tratteggiata è stata ottenuta per scala dalla precedente, rispetto all'origine degli assi, affinché avesse lunghezza 2.

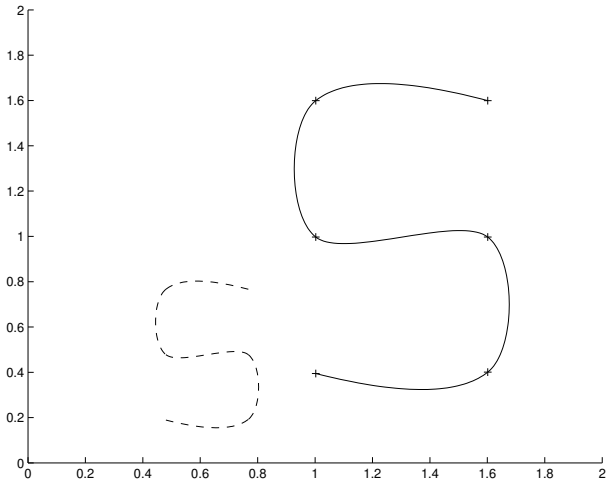

Figura 6.7: Lunghezza di una curva: la curva tratteggiata è stata scalata rispetto all'origine degli assi per avere lunghezza 2.

6.5 Applicazione: area di una curva

Sia data una curva di Bézier $\mathbf{C}(t)$ con $t \in [0, 1]$. Siamo interessati a determinare l'area che tale curva sottende con l'origine degli assi (se la curva è chiusa questo equivale all'area della regione che resta definita dalla curva) e a progettare curve di area assegnata mantenendo la stessa forma.
È noto che l'area di una curva in questa forma è espressa da

$$A = \pm\frac{1}{2}\int_0^1 \mathbf{C}(t) \times \mathbf{C}'(t)dt$$

$$= \pm\frac{1}{2}\int_0^1 [x(t)y'(t) - x'(t)y(t)]dt$$

con $x(t)$ ed $y(t)$ le componenti della $\mathbf{C}(t)$ e $x'(t)$ ed $y'(t)$ le componenti della $\mathbf{C}'(t)$. Il segno di quest'area è positivo se la curva è parametrizzata in senso antiorario nel piano xy.

L'infinitesimo di area sotteso dalla curva sarà dato da $\frac{1}{2}[\mathbf{C}(t) \times \mathbf{C}'(t)]\delta t$. Per calcolare tale area sarà necessario utilizzare una formula di integrazione numerica e se si desidera l'approssimazione con una certa tolleranza si può utilizzare il metodo di Simpson adattivo, ma si può anche osservare che la funzione integranda, a differenza del caso della lunghezza della curva, è una funzione polinomiale. Se per esempio la curva fosse una cubica, allora la funzione integranda sarebbe un polinomio di grado 5 e potrebbe essere integrata in modo esatto, a meno di errori di approsimazione numerica, usando una formula di Newton-Cotes con $n = 4$.
Supponiamo di aver progettato la curva $\mathbf{C}(t)$ e di aver stimato che la sua area è A_C. Siamo interessati ad una curva $\mathbf{G}(t)$ con la stessa forma della $\mathbf{C}(t)$, ma di area fissata A_G. A tal fine si noti che se la curva $\mathbf{C}(t)$ viene scalata di un fattore s la sua area viene scalata dello stesso fattore al quadrato; infatti sia

$$\mathbf{G}(t) = \begin{pmatrix} s \cdot x(t) \\ s \cdot y(t) \end{pmatrix}$$

la curva $\mathbf{C}(t)$ scalata del fattore s, allora sarà

$$A_G = \int_0^1 [s^2 x(t)y'(t) - s^2 x'(t)y(t)]dt = s^2 \int_0^1 [x(t)y'(t) - x'(t)y(t)]dt = s^2 A_C.$$

Segue che assegnata l'area A_G sarà sufficiente calcolare $s := \sqrt{A_G/A_C}$ e scalare la curva $\mathbf{C}(t)$ per tale fattore ottenendo una curva $\mathbf{G}(t)$ della forma della $\mathbf{C}(t)$ e di area desiderata. La Fig.6.8 mostra una curva chiusa polinomiale cubica a tratti C^1 ottenuta per interpolazione; la curva tratteggiata è stata ottenuta per scala dalla precedente, rispetto al baricentro dei punti di interpolazione, affinché avesse area unitaria.

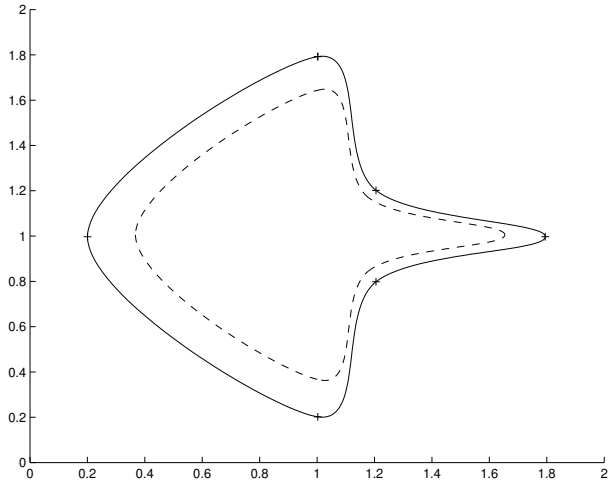

Figura 6.8: Area di una curva: la curva tratteggiata è stata scalata per avere area 1.

Parte IV

Equazioni Non Lineari

Capitolo 7

Equazioni non Lineari

Il problema di determinare le radici di un'equazione non lineare

$$f(x) = 0$$

nella variabile $x \in \mathbb{R}$, si presenta frequentemente nelle applicazioni. In generale non sono disponibili formule esplicite per calcolare le radici di equazioni, per cui si deve ricorrere a tecniche che consentono di approssimare le soluzioni con una fissata precisione. Tali metodi numerici sono di tipo iterativo, ovvero consistono nel definire una successione che a partire da una approssimazione iniziale converga alla radice in un processo al limite. In questa trattazione verrano dati gli elementi di base necessari per affrontare tale problema nel caso generale, quindi si particolarizzerà al caso di funzioni polinomiali per cui si discuteranno due fra i tantissimi approcci presenti in letteratura e si presenteranno alcune applicazioni significative.

7.1 Errore Inerente

L'input per il problema di trovare le radici di una equazione non lineare $f(x) = 0$ consiste nell'espressione assegnata per definire la f; nel caso polinomiale si tratterebbe della combinazione di certi coefficienti numerici per una certa base di rappresentazione. Lavorando in aritmetica finita, la funzione che verrà considerata sarà differente dalla f, chiamiamola $\hat{f}$ (nel caso polinomiale nella base canonica la $\hat{f}$ sarà il polinomio che si ottiene considerando i coefficienti di f approssimati con numeri finiti). In definitiva il problema che andremo a risolvere sarà determinare le radici di $\hat{f}(x) = 0$ e non di $f(x) = 0$. Sia x^* una radice di $f(x) = 0$ e $\hat{x}^*$ una radice di $\hat{f}(x) = 0$. Si indica l'errore sui dati con $E(x) = |\hat{f}(x) - f(x)|$ e si cerca una relazione con l'errore assoluto sul risultato

$|\hat{x}^* - x^*|$. Si può dimostrare che

$$|\hat{x}^* - x^*| = \frac{E(\hat{x}^*)}{|f'(\xi)|} \quad \text{con} \quad \xi \in (x^*, \hat{x}^*). \tag{7.1}$$

Per il teorema del valor medio vale

$$\frac{f(\hat{x}^*) - f(x^*)}{\hat{x}^* - x^*} = f'(\xi) \quad \xi \in (x^*, \hat{x}^*);$$

ricordando che $f(x^*) = 0$ e che per definizione $E(\hat{x}^*) = |\hat{f}(\hat{x}^*) - f(\hat{x}^*)| = |f(\hat{x}^*)|$ perché $\hat{f}(\hat{x}^*) = 0$, risulterà

$$\frac{E(\hat{x}^*)}{|\hat{x}^* - x^*|} = |f'(\xi)|$$

da cui la (7.1).

Osservazione 7.1 *L'errore $|\hat{x}^* - x^*|$ dipende criticamente sia dalla derivata della f in prossimità della radice, quanto dall'errore di valutazione della f.*

7.2 Metodo di Bisezione

Sia $f(x) \in C_{[a,b]}$ tale che $f(a)f(b) < 0$. Allora esiste almeno una soluzione x^* di $f(x) = 0$ con $a < x < b$. Il metodo più semplice e intuitivo per approssimare x^* è il metodo di bisezione; questo procede suddividendo ripetutamente l'intervallo $[a, b]$ a metà e determinando in quale dei due sottointervalli si trova la soluzione. Così facendo l'ampiezza dell'intervallo che contiene x^* risulta ad ogni passo dimezzata fino ad essere piccola come la tolleranza desiderata. Più in dettaglio si considera il punto medio di $[a, b]$, $x_m = (a + b)/2$; si valuta $f(x_m)$ e si considera quale dei due intervalli $[a, x_m]$ o $[x_m, b]$ è nelle condizioni per contenere x^* ($f(a)f(x_m) < 0$, allora $x^* \in [a, x_m]$, altrimenti $x^* \in [x_m, b]$). Il procedimento viene ripetuto (metodo iterativo) sull'intervallo che è risultato contenere x^* e verrà arrestato quando l'ampiezza dell'intervallo in esame risulterà minore di una prefissata tolleranza *tol*.

Osservazione 7.2 *La costante tol non può essere scelta arbitrariamente piccola, perché lavorando con numeri finiti la condizione di arresto*

$$\frac{|b_k - a_k|}{\min(|a_k|, |b_k|)} < tol, \quad con \quad 0 \notin [a_k, b_k] \quad \forall k,$$

dove con $[a_k, b_k]$ *si intende l'intervallo ottenuto al* $k-$*esimo passo del metodo, potrebbe non essere mai soddisfatta. Senza perdere in generalità sia* $0 < X < Y$ *con* X *ed* Y *numeri finiti consecutivi, e quindi*

$$X = (\sum_{i=1}^{t} \alpha_i \beta^{-i})\beta^p \quad e \quad Y = (\sum_{i=1}^{t} \alpha_i \beta^{-i} + \beta^{-t})\beta^p$$

allora

$$Y - X = \beta^{p-t} \quad e \quad X \geq \beta^{p-1}$$

per cui

$$\frac{Y - X}{X} \leq \frac{\beta^{p-t}}{\beta^{p-1}} = \beta^{1-t} = 2u.$$

Se poi $\alpha_1 = 1$ *e* $\alpha_i = 0$ *per* $i > 1$ *allora*

$$\frac{Y - X}{X} = 2u.$$

Ne consegue che tol non può essere scelto arbitrariamente, ma deve essere maggiore di $2u$*; a tal fine il test di arresto deve essere così formulato:*

$$|b_k - a_k| \leq TOL + 2u \min(|a_k|, |b_k|)$$

dove ora TOL *può essere scelto arbitrariamente.*

Osservazione 7.3 *Nella pratica questo test può essere usato anche nel caso in cui* $0 \in [a_k, b_k]$*; se però* $x^* \equiv 0$ *sarà sempre* $a_k < 0$ *e* $b_k > 0$*, per cui*

$$\frac{|b_k - a_k|}{\min(|a_k|, |b_k|)} > 1;$$

per evitare questo caso basta controllare se $f(0) \equiv 0$ *quando* $a < 0 < b$*.*

Di seguito viene presentato un semplice codice MATLAB che implementa quanto descritto.

```
function [xm,n]=sbisez(fun,a,b,tol)
% Questa routine determina una radice di una funzione con
% il metodo di bisezione
% fun --> funzione di cui det. uno zero
% a,b --> intervallo di definizione della funzione
% tol --> tolleranza richiesta
% xm  <-- approssimazione desiderata della radice
% Questa routine fa uso della variabile eps di MATLAB che
```

```
% corrisponde a 2*u con u unita' di arrotondamento
      n=0;
      if ((a<0) & (b>0) & (feval(fun,0)==0)) then
        xm=0;
      else
       fa=feval(fun,a);
       fb=feval(fun,b);
       while (abs(b-a)>(tol+eps*min([abs(a),abs(b)]));
         n=n+1;
         xm=(a+b)/2;
         fxm=feval(fun,xm);
         if (sign(fa)==sign(fxm))
           a=xm;
           fa=fxm;
         else
           b=xm;
           fb=fxm;
         end
       end
       xm=(a+b)/2;
      end
```

Osservazione 7.4 *Si noti che se nell'intervallo* $[a, b]$ *è contenuto un numero dispari di radici di* $f(x) = 0$*, allora il metodo di bisezione è ancora applicabile, ma determinerà l'approssimazione di una sola di esse.*

7.3 Metodi di Iterazione Funzionale

Il metodo di bisezione può essere applicato ad un'ampia classe di funzioni. Tuttavia ha lo svantaggio di essere lento, infatti ad ogni passo l'approssimazione della radice migliora di una sola cifra binaria. Per ridurre l'errore di una cifra decimale sono mediamente necessari 3.3 iterazioni. Ancora, la velocità di convergenza non dipende dalla funzione $f(x)$ in quanto il metodo usa il segno che la funzione assume negli estremi per determinare il successivo intervallo che viene solo e comunque dimezzato. Il metodo delle bisezioni può essere comunque utilizzato per determinare delle buone approssimazioni iniziali della radice che possono essere utilizzate dai metodi iterativi che andiamo a descrivere. Richiedendo alla f condizioni di essere più regolare che solo continua è possibile individuare una vasta classe di metodi che forniscono le stesse approssimazioni del metodo di bisezione utilizzando un numero di iterazioni molto minore. In

generale questi metodi sono del tipo:

$$x_{k+1} = g(x_k) \quad k = 0, 1, \ldots \tag{7.2}$$

con cui, a partire da un valore iniziale x_0 è possibile approssimare le soluzioni dell'equazione

$$x = g(x). \tag{7.3}$$

Le soluzioni di (7.3) sono dette **punti fissi** di $g(x)$. Per poter applicare lo schema (7.2) all'equazione $f(x) = 0$, bisogna trasformarla in una equazione equivalente alla (7.3). Uno dei modi con cui questa trasformazione può essere fatta è quella di usare una funzione $h(x) \neq 0$ e definire

$$g(x) = x - \frac{f(x)}{h(x)}$$

cosí ché $x = g(x)$ abbia le stesse soluzioni di $f(x) = 0$ in un opportuno intervallo contenente la soluzione che si vuole approssimare. Ovviamente è banale vedere che ogni punto fisso di g è uno zero di f e viceversa.

Teorema 7.1 *Se $g(x)$ possiede un punto fisso x^* e se $g(x) \in C^1_{[x^*-\rho, x^*+\rho]}$ con $\rho > 0$ e soddisfa la condizione*

$$|g'(x)| \leq \lambda < 1 \quad \textit{per} \quad x \in [x^* - \rho, x^* + \rho],$$

allora per ogni $x_0 \in [x^* - \rho, x^* + \rho]$, tutti gli iterati x_k generati dalla (7.2) appartengono a questo intervallo, la successione converge ad x^* e x^* è l'unico punto fisso della g nell'intervallo $[x^* - \rho, x^* + \rho]$.

Dim.

Diremo che $x_k \to x^*$ per $k \to \infty$ se $\forall k$ vale

$$|x_{k+1} - x^*| < |x_k - x^*|.$$

Si ha

$$x_{k+1} - x^* = g(x_k) - g(x^*)$$

e per il teorema del valor medio

$$= g'(\xi_k)(x_k - x^*) \quad \text{con} \quad \xi_k \in (x^*, x_k);$$

passando ai moduli si ha

$$|x_{k+1} - x^*| \leq \lambda |x_k - x^*| \quad \forall k$$

da cui

$$|x_{k+1} - x^*| \leq \lambda |x_k - x^*| \leq \cdots \leq \lambda^k |x_0 - x^*|$$

e quindi

$$\lim_{k\to\infty} |x_{k+1} - x^*| \leq \lim_{k\to\infty} \lambda^k |x_0 - x^*| = 0.$$

Questo prova che tutti gli iterati appartengono all'intervallo $[x^* - \rho, x^* + \rho]$ e che la successione converge ad x^*. Per dimostrare l'unicità del punto fisso ragioniamo per assurdo e supponiamo che i punti fissi siano due, $x_1^*, x_2^* \in [x^* - \rho, x^* + \rho]$. Allora

$$|x_1^* - x_2^*| = |g(x_1^*) - g(x_2^*)| = |g'(\xi)||x_1^* - x_2^*| \quad \texttt{con} \quad \xi \in [x^* - \rho, x^* + \rho]$$

ma poiché $|g'(\xi)| < 1|$ si ha

$$|x_1^* - x_2^*| < |x_1^* - x_2^*|$$

e ciò è assurdo. $\odot$

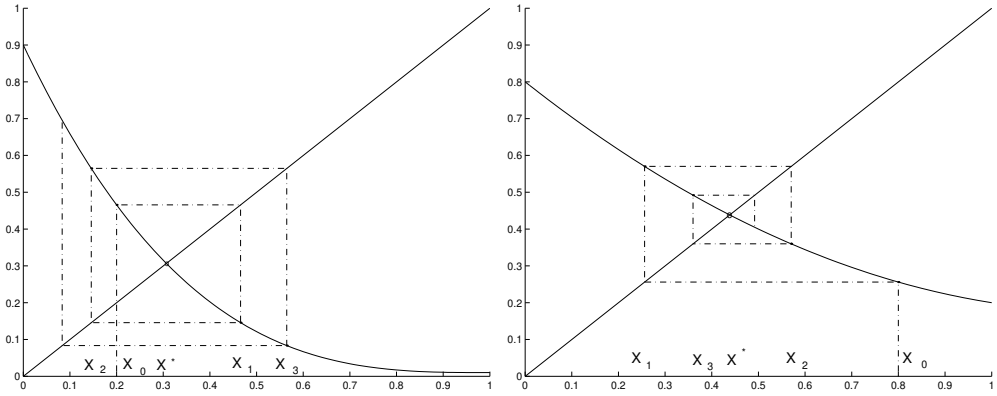

Figura 7.1: $g'(x) < -1$ (sinistra) e $-1 < g'(x) < 0$ (destra) per $x \in [x^* - \rho, x^* + \rho]$.

Le Fig.7.1 e 7.2 mostrano graficamente i possibili andamenti della successione $\{x_k\}$ a seconda della pendenza della funzione $g(x)$ in un intorno di x^*; si osservi come le successioni $\{x_k\}$ convergono o meno a x^* a seconda che la $g(x)$ soddisfi o meno le condizioni di convergenza.

7.3.1 Ordine di Convergenza

Per confrontare differenti metodi iterativi che approssimano la stessa soluzione x^*, si può considerare la velocità con cui le successioni ottenute convergono alla soluzione. Lo studio della velocità di convergenza viene generalmente ricondotto a quello dell'ordine di convergenza del metodo.

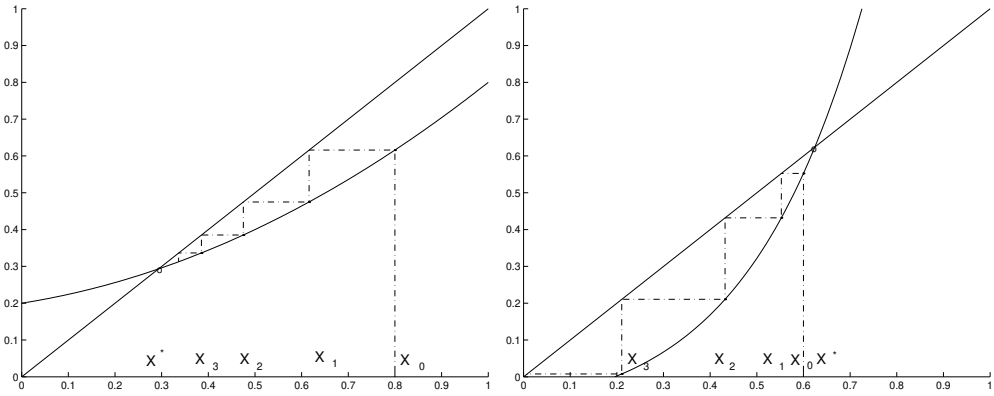

Figura 7.2: $0 < g'(x) < 1$ (sinistra) e $1 < g'(x)$ (destra) per $x \in [x^* - \rho, x^* + \rho]$.

Definizione 7.1 *Sia $\{x_k\}$ una successione convergente ad x^* e sia $x_k \neq x^* \quad \forall k$. Se esiste un numero reale $p \geq 1$, tale che*

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = \gamma \quad dove \quad e_k = x_k - x^*$$

con

$$\begin{cases} 0 < \gamma \leq 1 & se \quad p = 1 \\ \gamma > 0 & se \quad p > 1 \end{cases}$$

si dice che la successione ha **ordine di convergenza** *p. La costante γ è detta* **fattore di convergenza**.

Osservazione 7.5 *Terminologia:*
se $p = 1$ e $0 < \gamma < 1$ si dice che la convergenza è **lineare**;
se $p = 1$ e $\gamma = 1$ si dice che la convergenza è **sublineare**;
se $p > 1$ e $\gamma > 0$ si dice che la convergenza è **superlineare**;

Osservazione 7.6 *Dalla definizione si ha che esiste una costante c tale che da un certo k in poi valgono*

$$|e_{k+1}| \leq c|e_k|^p \quad e \quad \left|\frac{e_{k+1}}{x^*}\right| \leq c|x^*|^{p-1}\left|\frac{e_k}{x^*}\right|^p \quad se \quad x^* \neq 0$$

che misurano la riduzione dell'errore assoluto e relativo ad ogni iterazione.

Teorema 7.2 *Sia $\{x_k\}$ una successione generata da (7.2) convergente ad x^* e sia $g(x)$ sufficientemente regolare in un intorno di x^*. Allora la successione ha ordine di convergenza $p \geq 1$ se e solo se*

$$g'(x^*) = g''(x^*) = \cdots = g^{(p-1)}(x^*) = 0, \quad g(x^*)^{(p)} \neq 0.$$

Dim.

$$\begin{aligned} x_{k+1} - x^* &= g(x_k) - g(x^*) \\ &= g(x^*) + g'(x^*)(x_k - x^*) + \tfrac{1}{2}g''(x^*)(x_k - x^*)^2 + \dots \\ &\dots + \tfrac{1}{(p-1)!}g^{(p-1)}(x^*)(x_k - x^*)^{p-1} + \tfrac{1}{p!}g^{(p)}(\xi_k)(x_k - x^*)^p - g(x^*) \end{aligned}$$

dove ξ_k è compreso tra x_k e x^*. Per le ipotesi risulta

$$|e_{k+1}| = \frac{1}{p!}|g^{(p)}(\xi_k)||e_k|^p.$$

$\odot$

7.3.2 Metodo di Newton

Supponiamo che $f(x) \in C^2_{[a,b]}$; sia $\bar{x} \in [a,b]$ un'approssimazione di x^* tale che $f'(\bar{x}) \neq 0$ e $|\bar{x} - x^*|$ sia piccolo. Si consideri il polinomio di Taylor per $f(x)$ in un intorno di $\bar{x}$ e lo si valuti in x^*

$$f(x^*) = f(\bar{x}) + (x^* - \bar{x})f'(\bar{x}) + \frac{(x^* - \bar{x})^2}{2!}f''(\bar{x}) + \cdots$$

Poiché $|\bar{x} - x^*|$ si è assunto piccolo, $(x^* - \bar{x})^2$ sarà ancora più piccolo e ancor di più i successivi termini; trascurando allora i termini non lineari si ha:

$$0 \simeq f(\bar{x}) + (x^* - \bar{x})f'(\bar{x});$$

risolvendo per x^* si ottiene:

$$x^* \simeq \bar{x} - \frac{f(\bar{x})}{f'(\bar{x})}$$

che viene ad essere un'approssimazione di x^* migliore di $\bar{x}$.
Questa relazione fornisce l'idea per il **metodo di Newton**, che consiste nel generare una successione $\{x_k\}$ definita da:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad k \geq 0 \quad \text{e} \quad f'(x_k) \neq 0 \; \forall k. \tag{7.4}$$

L'interpretazione geometrica indica che ogni nuovo iterato x_{k+1} è dato dall'intersezione fra la retta tangente alla $y = f(x)$ nel punto x_k e l'asse x (ciò deriva dalla linearizzazione della $f(x)$ in x_k, da cui anche il nome **metodo delle tangenti**).
Infatti, con riferimento alla Fig.7.3, se chiamiamo $\Delta y = f(x_k)$ e $\Delta x = x_k - x_{k+1}$ si ha che vale

$$\Delta x = \frac{\Delta y}{\tan\alpha} \quad \text{con} \quad \tan\alpha = f'(x_k).$$

Sostituendo si ottiene proprio la relazione (7.4) che dà il metodo di Newton.

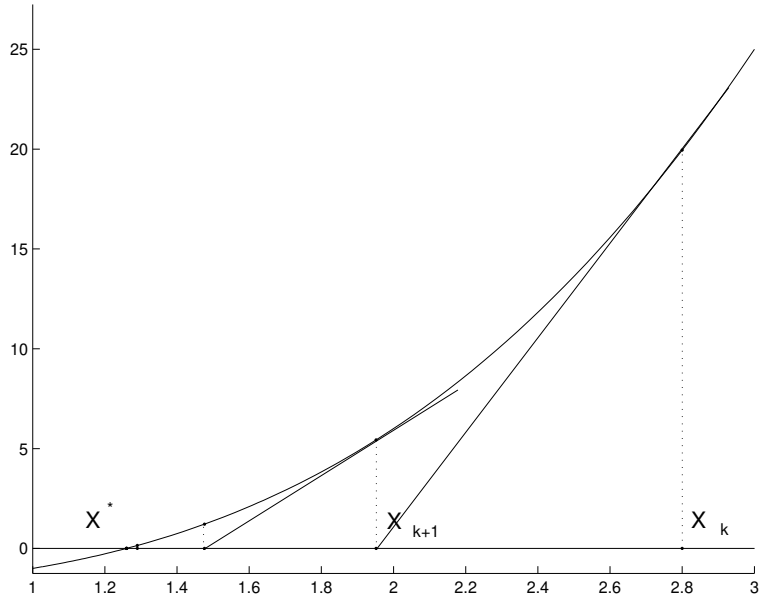

Figura 7.3: Iterazione del metodo di Newton.

7.3.3 Convergenza del metodo di Newton

Si noti che il metodo di Newton equivale al metodo di iterazione funzionale in cui la funzione $g(x)$ è

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{7.5}$$

Si vuole sfruttare il Teorema 7.1 per dimostrare il seguente risultato:

Teorema 7.3 *Se $f(x) \in C^2_{[a,b]}$, $f(x^*) = 0$ e $f'(x^*) \neq 0$, allora esiste un intervallo $[x^* - \rho, x^* + \rho]$ tale che se $x_0 \in [x^* - \rho, x^* + \rho]$ il metodo di Newton converge ad x^*.*

Dim.

Il Teorema 7.1 regola la convergenza ad x^* della successione $x_{k+1} = g(x_k)$ e dice che questo avviene se $|g'(x)| < 1$ per $x \in [x^* - \rho, x^* + \rho]$; nel nostro caso per la (7.5) è:

$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$$

ed è facile vedere che nelle ipotesi date sulla f, la $g'(x)$ è continua e si annulla in x^*; allora esiste un intorno di x^* per cui $|g'(x)| < 1$ c.v.d. $\odot$

Teorema 7.4 *Se* $f(x) \in C^3_{[a,b]}$, $f(x^*) = 0$ *e* $f'(x^*) \neq 0$, *allora esiste un intervallo* $[x^* - \rho, x^* + \rho]$ *tale che se* $x_0 \in [x^* - \rho, x^* + \rho]$ *il metodo di Newton ha ordine di convergenza* 2.

Dim.

Per il Teorema 7.2, essendo $g'(x^*) = 0$ si ha che l'ordine di convergenza è 2.

Se poi calcoliamo la derivata seconda di $g(x)$ troviamo:

$$g''(x) = \frac{[f'(x)f''(x) + f(x)f'''']^2 - 2f(x)f'(x)[f''(x)]^2}{[f'(x)]^4}.$$

Valutandola in x^* si ha

$$g''(x^*) = \frac{f''(x^*)}{f'(x^*)} \tag{7.6}$$

ne segue che se $f''(x^*) \neq 0$ allora anche $g''(x^*) \neq 0$ e quindi, per il teorema 7.2, l'ordine p è esattamente 2. Se invece $f''(x^*) = 0$ allora l'ordine è almeno pari a 3. Dalla relazione (7.6) si ha che il fattore di convergenza vale

$$c = \frac{1}{2}\left|\frac{f''(x^*)}{f'(x^*)}\right|.$$

$\odot$

Il Teorema 7.3 vale nell'ipotesi che $f'(x^*) \neq 0$, cioè se x^* è una radice semplice di $f(x)$. Se invece la radice x^* ha molteplicità $m > 1$, l'ordine di convergenza del metodo non sarà più 2. Infatti se scriviamo

$$f(x) = (x - x^*)^m q(x), \qquad q(x^*) \neq 0,$$

la funzione $g(x)$ nel caso di Newton avrà la seguente espressione

$$g(x) = x - \frac{(x - x^*)q(x)}{mq(x) + (x - x^*)q'(x)},$$

da cui si può facilmente determinare che

$$g'(x^*) = 1 - \frac{1}{m}. \tag{7.7}$$

Allora, poiché $m > 1$, risulta $|g'(x)| < 1$ in un intorno di x^* e per il Teorema 7.1 il metodo risulta ancora convergente ma, per il Teorema 7.2 l'ordine di convegenza è solo 1.

Se si conosce la molteplicità della radice si può modificare la formula iterativa di Newton per ottenere un metodo con ordine 2. Infatti definendo

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, \ldots$$

si ha un metodo con la seguente funzione iteratrice

$$g(x) = x - m\frac{f(x)}{f'(x)}$$

da cui, per la (7.7) si ha

$$g'(x^*) = 0.$$

7.3.4 Propagazione degli errori

A causa degli errori di arrotondamento che si commettono nel calcolo di $f(x)$ e conseguentemente di $g(x)$, la successione effettivamente calcolata $\{\tilde{x}_k\}$ può non essere convergente anche quando sono soddisfatte le ipotesi del Teorema 7.1. Sia δ_k l'errore assoluto introdotto nel calcolo effettivo, in aritmetica finita, alla k-esima iterazione

$$\tilde{x}_k = g(\tilde{x}_{k-1}) + \delta_k \quad \text{con} \quad |\delta_k| < \delta \quad k = 0, 1, \ldots,$$

allora vale il seguente

Teorema 7.5 *Sia x^* punto fisso di $g(x) \in C^1_{[x^*-\rho, x^*+\rho]}$ con $\rho > 0$ e sia $|g'(x)| \leq \lambda < 1$ per $x \in [x^* - \rho, x^* + \rho]$. Allora per ogni $\tilde{x}_0 \in [x^* - \rho, x^* + \rho]$, se*

$$\frac{\delta}{1-\lambda} < \rho,$$

gli iterati $\tilde{x}_k$ con gli errori δ_k (tutti limitati da δ) apparterranno all'intervallo $[x^* - \rho, x^* + \rho]$ e soddisfano la

$$|\tilde{x}_k - x^*| \leq \frac{\delta}{1-\lambda} + \lambda^k \left(\rho - \frac{\delta}{1-\lambda}\right). \tag{7.8}$$

Osservazione 7.7 *Il secondo membro della (7.8) si compone di due termini; il primo può essere molto grande quando $\lambda \simeq 1$ e questo indipendentemente dall'indice di iterazione k; il secondo termine tende a zero per $k \to \infty$, essendo $\lambda < 1$. Pertanto se $\lambda \simeq 1$ non si può essere certi che la successione $\{\tilde{x}_k\}$ converga ad x^*.*

Osservazione 7.8 *Si consideri la differenza di due iterati consecutivi; sfruttando il risultato del Teorema 7.5 si ha:*

$$\begin{aligned}|\tilde{x}_{k+1} - \tilde{x}_k| &= |(\tilde{x}_{k+1} - x^*) + (x^* - \tilde{x}_k)| \\ &\leq |\tilde{x}_{k+1} - x^*| + |x^* - \tilde{x}_k| \\ &\leq \frac{2\delta}{1-\lambda} + (\lambda^{k+1} + \lambda^k)\left(\rho - \frac{\delta}{1-\lambda}\right) \\ &= \frac{2\delta}{1-\lambda} + \lambda^k(1+\lambda)\left(\rho - \frac{\delta}{1-\lambda}\right).\end{aligned}$$

Ciò mostra che, comunque grande sia l'indice di iterazione k, non è possibile ottenere una maggiorazione della differenza di due iterati consecutivi più piccola di $\frac{2\delta}{1-\lambda}$ a causa degli errori di arrotondamento.
Questa stima è una misura dell'incertezza con cui è possibile determinare la soluzione per effetto degli errori di arrotondamento. Di questo è opportuno tenere conto nella scelta della tolleranza per le condizioni di arresto del metodo.

7.3.5 Test di arresto

Come test di arresto per un metodo di iterazione funzionale, ed in particolare per il metodo di Newton, si vorrebbe richiedere

$$\left|\frac{x_k - x^*}{x^*}\right| \leq tol,$$

ma non conoscendo x^*, ci si limiterà a chiedere

$$\frac{|x_k - x_{k-1}|}{\min(|x_k|, |x_{k-1}|)} \leq tol \quad \text{con} \quad tol > \frac{2\delta}{1-\lambda}.$$

Ancora però, pur pensando di approssimare δ con u, non è noto λ. Si osservi che nel caso del metodo di Newton, $\lambda \simeq 1$ può essere interpretato come un indice di mal condizionamento del problema cosa che si verifica, come abbiamo visto, sia a causa di errori nella valutazione della funzione ($E(\hat{x}^*)$) che del valore che assume la $f'(x)$ nell'intorno di x^*. In definitiva, nell'implementare il metodo di Newton, è bene prevedere almeno tre test al verificarsi di uno dei quali ci si deve arrestare:

1. $|x_k - x_{k-1}| \leq tol_1 + 2u\min(|x_k|, |x_{k-1}|)$
2. $|f(x_k)| \leq tol_2$
3. $k > MaxIter$

Con tol_1 tolleranza richiesta di approssimazione, tol_2 per stimare il malcondizionamento e $MaxIter$ come limite massimo del numero di iterazioni. Di seguito viene presentato un semplice codice MATLAB che implementa quanto descritto.

```
function [xstar,n]=stangmet(fun,funp,x0,tol)
% Questa routine determina una radice di una funzione con
% il metodo di Newton o delle tangenti
% fun --> funzione
% funp--> funzione derivata
% x0  --> iterato iniziale
% tol --> tolleranza richiesta
% xstar <-- approssimazione desiderata della radice
% n <-- iterazioni effettuate
% Questa routine fa uso della variabile eps di MATLAB che
% corrisponde a 2*u con u unita' di arrotondamento e
% della variabile realmin di MATLAB che corrisponde al piu'
% piccolo numero finito positivo
  n=0;
  maxiter=50;
  smallreal=100*realmin;
  xstar=x0;
  x0=x0+1;
  while ((abs(xstar-x0)>(tol+eps*min([abs(x0),abs(xstar)])) & (n<maxiter))
    x0=xstar;
    fx0=feval(fun,x0);
    if (abs(fx0)>=smallreal)
      xstar=x0-fx0/feval(funp,x0);
      n=n+1;
    end
  end
```

7.3.6 Applicazioni: $\sqrt{a}$ e $1/a$

La radice quadrata positiva di un numero reale positivo o il reciproco di un reale (operazioni a tutt'oggi previste in hardware dallo standard IEEE) possono essere calcolate risolvendo rispettivamente le equazioni:

$$x^2 - a = 0 \qquad \text{e} \qquad \frac{1}{x} - a = 0.$$

Si osservi che agli inizi del calcolo automatico, la radice quadrata e la divisione erano realizzate via software trovando le radici delle equazioni suddette; le sole operazioni implementate in hardware erano l'addizione, la sottrazione e la moltiplicazione.
Procediamo nel trovare le radici di queste due equazioni applicando in entrambi i casi il metodo di Newton.

Equazione: $x^2 - a = 0$.
Sarà $f(x) = x^2 - a$ e $f'(x) = 2x$, allora

$$x_{k+1} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{x_k^2 + a}{2x_k} = \frac{1}{2}(x_k + \frac{a}{x_k})$$

che è la così detta formula di Erone per il calcolo della radice quadrata, nota già agli antichi greci.

Nel'innescare il metodo iterativo si può assumere come valore iniziale $x_0 = 1$; infatti essendo interessati alla radice positiva vale $|g'(x)| < 1$ in tutto il semiasse positivo (vedi Fig.7.4 sinistra). Se il valore iniziale 1 è molto distante dal valore cercato $\sqrt{a}$, il metodo impiegherà solo più iterazioni.

Equazione: $\dfrac{1}{x} - a = 0$.
Sarà $f(x) = \dfrac{1}{x} - a$ e $f'(x) = -\dfrac{1}{x^2}$, allora

$$x_{k+1} = x_k - \frac{\frac{1}{x_k} - a}{-\frac{1}{x_k^2}} = x_k + \frac{1 - ax_k}{x_k} x_k^2 = x_k(2 - ax_k).$$

Nell'innescare il metodo iterativo è importante assumere un opportuno valo-

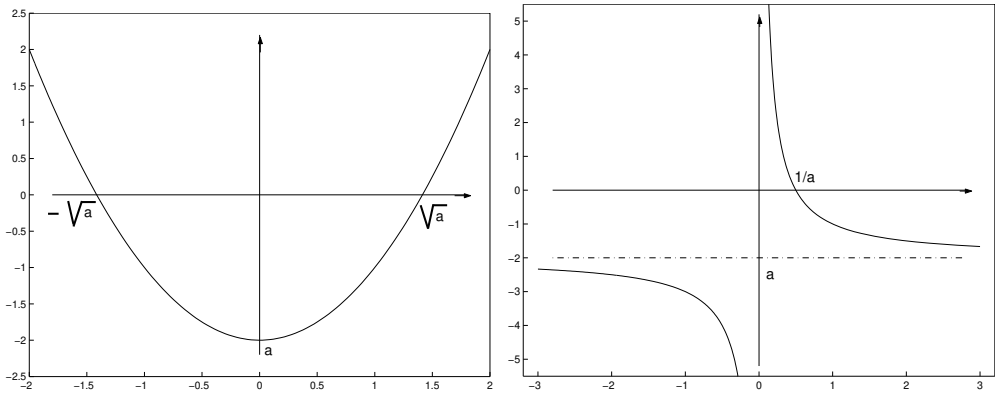

Figura 7.4: Equazione $x^2 - a = 0$ (sinistra) $\dfrac{1}{x} - a = 0$ (destra).

re iniziale x_0 (vedi Fig.7.4 destra). Il metodo infatti converge sicuramente se $0 < x_0 < 1/a$, ma negli altri casi potrebbe divergere. Assegnato a in forma normalizzata si consideri la sua parte esponente; se per esempio $a = 15000 = 0.15 \times 10^5$, l'esponente sarà 10^5. Banalmente risulta $a \leq 10^5$ e $0 < 10^{-5} \leq \frac{1}{a}$. Allora si può assumere come valore iniziale il reciproco della parte esponente di a.

Esercizio 7.1 *Approssimare il numero* $\sqrt[m]{a}$ *con* $m \in \mathbb{R}$, $m \geq 2$, $a > 0$.

Il numero cercato è lo zero della funzione $f(x) = x^m - a$. *Il metodo di Newton fornisce lo schema*

$$x_{k+1} = x_k - \frac{x_k^m - a}{m x_k^{m-1}} = \frac{1}{m}[(m-1)x_k + c x_k^{1-m}], \quad k = 0, 1, \ldots$$

Osservazione 7.9 *Se* $f'(x^*) \neq 0$, *cioè* x^* *è una radice semplice di* $f(x) = 0$, *ogni iterazione dà un errore che è circa il quadrato di quello dell'iterazione precedente. Per* $e_k \to 0$ *il numero di cifre decimali esatte (anche cifre binarie) vengono approssimativamente raddoppiate ad ogni iterazione.*

Esempio 7.1 *Sia* $f(x) = x^3 - 3x + 2$*; si vuole applicare il metodo di Newton per determinare le due radici* $x_1^* = -2$ *e* $x_2^* = 1$.
Si calcola la derivata prima $f'(x) = 3x^2 - 3$ *e si costruisce il metodo iterativo:*

$$x_{k+1} = x_k - \frac{x_k^3 - 3x_k + 2}{3x_k^2 - 3} = \frac{2}{3}\frac{x_k^3 - 1}{x_k^2 - 1}.$$

Partendo da $x_0 = -2.4$ *si calcola la successione per determinare* $x_1^* = -2$*; la Tab.7.1 mostra quanto si ottiene.*

k	x_k	$e_k = 1 - x_k$	$\lvert e_{k+1}\rvert / \lvert e_k\rvert^2$
0	-2.4	0.4	0.476190475
1	-2.076190476	0.076190476	0.619469086
2	-2.003596011	0.003596011	0.664202613
3	-2.000008589	0.000008589	
4	-2.000000000	0.000000000	

Tabella 7.1: Esempio di esecuzione

Si osservi che numericamente si ha

$$|e_{k+1}| \simeq \frac{2}{3}|e_k|^2.$$

Partendo da $x_0 = 1.2$ *si calcola la successione per determinare la radice doppia* $x_2^* = 1$*; La Tab.7.2 mostra quanto si ottiene.*
Si osservi che numericamente si ha

$$|e_{k+1}| \simeq \frac{1}{2}|e_k|.$$

k	x_k	$e_k = 2 - x_k$	$\lvert e_{k+1}\rvert/\lvert e_k\rvert$
0	1.2	0.2	0.515151515
1	1.103030303	0.103030303	0.508165253
2	1.052356420	0.052356420	0.496751115
3	1.026400811	0.026400811	0.509753688
4	1.013257730	0.013257730	0.501097775
5	1.006643419	0.006643419	

Tabella 7.2: Esempio di esecuzione

Osservazione 7.10 *La difficoltà nel metodo di Newton è trovare un valore iniziale x_0 sufficientemente prossimo ad x^* così da soddisfare le ipotesi del Teorema 7.3. Per questo motivo, il metodo di Newton deve essere preceduto dall'applicazione di metodi globalmente convergenti come quello di bisezione. In definitiva ed in pratica, il metodo di Newton deve essere visto come un metodo terminale veloce (che agisce localmente) a seguito di un metodo iniziale più lento (che agisce globalmente).*

7.3.7 Metodo delle Secanti

Ci sono funzioni per cui è molto economica la valutazione della $f'(x)$ una volta valutata la $f(x)$. Per altre funzioni, la valutazione della $f'(x)$ è tanto costosa quanto una seconda valutazione della $f(x)$. Per altre funzioni ancora, valutare $f'(x)$ è quasi impossibile.
Quando si cercano gli zeri di una funzione per la quale la valutazione della $f'(x)$ è difficoltosa, il metodo delle secanti è spesso una scelta migliore che quella di Newton. Tale metodo parte da due valori x_0 e x_1. Ad ogni iterazione si determina x_{k+1} come l'intersezione fra la retta passante per i punti $(x_k, f(x_k))$ e $(x_{k-1}, f(x_{k-1}))$ e l'asse x (vedi Fig.7.5).
Da cui si ha la relazione:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}).$$

Questa si può anche vedere considerando il metodo di Newton in cui alla $f'(x_k)$ si sostituisce il rapporto incrementale

$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

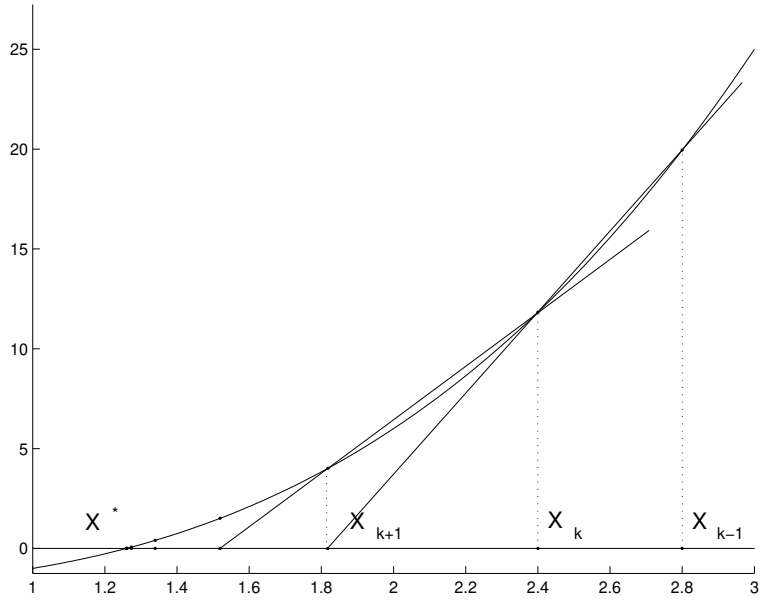

Figura 7.5: Iterazione del metodo delle secanti.

Teorema 7.6 *Se* $f(x) \in C^2_{[a,b]}$, $f(x^*) = 0$, $f'(x^*) \neq 0$ *e* $f''(x^*) \neq 0$, *allora esiste un intervallo* $[x^* - \rho, x^* + \rho]$ *tale che se* $x_0, x_1 \in [x^* - \rho, x^* + \rho]$ *il metodo delle secanti converge ad* x^*. *Inoltre il*

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = \gamma \neq 0$$

dove $p = \frac{1}{2}(\sqrt{5}+1) \simeq 1.618\ldots$

Questo Teorema afferma che, per buoni valori iniziali, il metodo delle secanti converge ad una radice semplice con ordine di convergenza $1.618\ldots$. Come già nel metodo di Newton, il più grosso problema del metodo delle secanti è determinare x_0 e x_1 abbastanza prossimi ad x^* così da garantire la convergenza.
Di seguito viene presentato un semplice codice MATLAB che implementa quanto descritto.

```
function [xstar,n]=ssecmet(fun,x0,x1,tol)
% Questa routine determina una radice di una funzione
% con il metodo delle secanti
% fun --> funzione analitica
% x0,x1  --> iterati iniziali
% tol --> tolleranza richiesta
```

```
% xstar <-- approssimazione desiderata della radice
% n <-- iterazioni effettuate
% Questa routine fa uso della variabile eps di MATLAB che
% corrisponde a 2*u con u unita' di arrotondamento e
% della variabile realmin di MATLAB che corrisponde al piu'
% piccolo numero finito positivo
  n=0;
  maxiter=50;
  smallreal=100*realmin;
  xstar=x1;
  x1=x0;
  fx1=feval(fun,x1);
  while ((abs(xstar-x1)>(tol+eps*min(abs(x1),abs(xstar)))) & (n<maxiter));
    x0=x1;
    fx0=fx1;
    x1=xstar;
    fx1=feval(fun,x1);
    if (abs(fx1)>=smallreal)
      xstar=x1-fx1*(x1-x0)/(fx1-fx0);
      n=n+1;
    end
  end
```

7.4 Zeri di polinomi

In questa sezione si affronta il problema di determinare le radici di una equazione polinomiale od anche zeri di una funzione polinomiale. Come è noto, il teorema fondamentale dell'algebra, assicura che ogni polinomio di grado n ha n radici reali o complesse. Se il polinomio è a coefficienti reali e una radice è complessa anche la complessa coniugata sarà una radice.
In letteratuta sono stati proposti numerosi metodi numerici per la determinazione di tutte le radici di un polinomio e ogni libreria scientifica è dotata di una tale routine. Per esempio il sistema MATLAB prevede la routine **roots**.
Nelle applicazioni pratiche, soprattutto della Computer Graphics e Modellazione Geometrica, è molto utile determinare solo le radici reali in un assegnato intervallo. Questa è la motivazione che ci spinge a presentare alcuni risultati per questo particolare problema a partire da un polinomio espresso sia nella base canonica che nella base di Bernstein. Nel primo caso viene presentato un metodo per isolare le radici dovuto a Sturm a cui può far seguito un metodo locale di determinazione delle singole radici. Nel secondo caso di polinomi nella base

di Bernstein ci sono molti metodi in letteratura e tutti sfruttano il significato geometrico che hanno i coefficienti di un polinomio in tale base (vedi capitolo su "Interpolazione con Funzioni Polinomiali"), dando luogo ad una classe di metodi geometrico/numerici.

Viene prima presentato il teorema di Sturm che permette di determinare il numero di zeri reali di un polinomio in un intervallo assegnato.

Sia dato un polinomio a coefficienti reali $p(x)$, e la sua derivata prima $p'(x)$. Applichiamo l'algoritmo di Euclide, opportunamente modificato; otteniamo la cosiddetta *sequenza di Sturm*, nel seguente modo: dividiamo $p(x)$ per $p'(x)$, ottenendo come quoziente $q_1(x)$ e come resto $r_1(x)$:

$$p(x) = q_1(x)p'(x) + r_1(x).$$

Si ponga $s_1(x) = -r_1(x)$, e si divida ora per $s_1(x)$:

$$p'(x) = q_2(x)s_1(x) + r_2(x).$$

Sia $s_2(x) = -r_2(x)$, e si prosegua dividendo $s_1(x)$ per $s_2(x)$:

$$s_1(x) = q_3(x)s_2(x) + r_3(x),$$

e così via, fino ad ottenere l'ultimo polinomio $s_k(x)$ non nullo. La sequenza di polinomi (di gradi decrescenti)

$$p(x), p'(x), s_1(x), s_2(x), s_3(x), \ldots, s_k(x)$$

è detta *sequenza di Sturm*.

Per esempio, se $p(x) = x^4 - 3x - 1$, la sequenza di Sturm è

$$p(x) = x^4 - 3x - 1$$
$$p'(x) = 4x^3 - 3$$
$$s_1(x) = \frac{9}{4}x + 1$$
$$s_2(x) = \frac{2443}{729}.$$

Per ogni numero reale c, la sequenza di Sturm individua la corrispondente sequenza di numeri reali

$$p(c), p'(c), s_1(c), s_2(c), s_3(c), \ldots, s_k(c).$$

In questa sequenza è possibile determinare il *numero di variazioni di segno*: una variazione di segno si verifica quando un valore della sequenza è positivo e il successivo (non nullo) è negativo, o viceversa. Definiamo così la funzione $S(x)$ da $\mathbb{R}$ in $\mathbb{N}$, che associa ad ogni reale x_0 il numero delle variazioni di segno nella sequenza di Sturm individuata dalla funzione $p(x)$ e valutata in x_0.

Teorema 7.7 *Dato un polinomio $p(x)$ a coefficienti in $\mathbb{R}$, il numero di zeri reali distinti di $p(x)$ compresi tra a e b ($a < b, f(a) \neq 0, f(b) \neq 0$) è*

$$S(a) - S(b).$$

Mostriamo l'applicazione di questo teorema al polinomio di cui abbiamo prima calcolato la sequenza di Sturm. Valutiamo le quattro funzioni della sequenza in $-2, -1, 0, 1, 2, 3$. Otteniamo la seguente tabella:

x	-2	-1	0	1	2	3
$p(x)$	21	3	-1	-3	9	71
$p'(x)$	-35	-7	-3	1	29	105
$s_1(x)$	-7/2	-5/4	1	13/4	11/2	31/4
$s_2(x)$	2443/729	2443/729	2443/729	2443/729	2443/729	2443/729
$S(x)$	2	2	1	1	0	0

Come si può vedere, la funzione $S(x)$ decresce e segnala gli intervalli nei quali è compreso uno zero di $p(x)$; non ci sono zeri tra -2 e -1, né tra 0 e 1, né tra 2 e 3, mentre uno zero è compreso tra -1 e 0 e un altro tra 1 e 2. Inoltre deduciamo che non ci sono altri zeri per $x > 2$ (infatti $S(x)$, per definizione, non può essere negativa).

Il metodo di Sturm appena descritto individua quindi il numero di radici reali per $p(x)$. Per ottenere le soluzioni, qualora ce ne fossero più di una, viene applicato un metodo per il root-isolation che isola la radice e successivamente viene innescata uno routine che ne approssima il valore.

Viene ora presentato un metodo, dovuto a Lane e Riesenfeld [LaRi81] per isolare e approssimare le radici reali di un polinomio nella base di Bernstein. Tale metodo è basato sulla proprietà di *Variation Diminishing* dei coefficienti del polinomio nella base di Bernstein e nell'uso di una tecnica di bisezione ricorsiva (vedi Suddivisione nel sopracitato capitolo).
La proposta consiste in una procedura *root_isolate* per determinare degli intervalli contenenti una sola radice controllando che il polinomio, ristretto a quell'intervallo nella base di Bernstein, abbia una sola variazione di segno dei coefficienti. Praticamente si esamina il numero di variazioni di segno dei coefficienti del polinomio: se questo è 0 non ci sono radici; se è 1 c'è una radice; se è > 1 si suddivide il polinomio nel punto medio e si procede sui due polinomi fino a che questi non abbiano 1 radice o nessuna. Alla fine si avranno in uno stack tanti polinomi che hanno una sola radice nell'intervallo di definizione. Viene quindi utilizzata una procedura *find_root* che per ogni polinomio memorizzato nello stack, cerca la sua unica radice. Si tratta di una routine che suddivide il polinomio fino a che uno dei due coefficienti estremi non sia così piccolo (tolleranza richiesta), da indicare nell'estremo corrispondente la radice. La tecnica

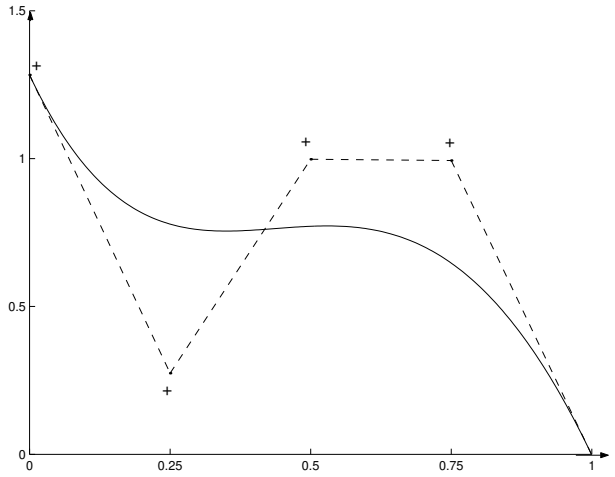

Figura 7.6: Zero in un estremo dell'intervallo.

di suddivisione non avviene nel punto medio, ma nel punto in cui la poligonale interseca l'asse x. Questa scelta è dettata dal fatto che per suddivisione la poligonale converge alla funzione e quindi lo zero della poligonale deve convergere allo zero della funzione.

Osservazione 7.11 *Si osservi che una situazione come quella presentata in Fig.7.6, dove la radice del polinomio coincide con un estremo dell'intervallo di definizione, non presenta variazioni di segno dei coefficienti, e quindi la radice non viene localizzata. Una implementazione corretta deve accorgersi di una tale situazione.*

7.4.1 Metodo di Newton per polinomi

Nel caso polinomiale, il metodo di Newton, come procedura locale per convergere velocemente ad una radice, è molto attraente in quanto, una volta valutato il polinomio, è poco costoso valutare anche la sua derivata prima. Il vero problema, come precedentemente sottolineato, è sapere quando si sia nelle ipotesi per poter applicare il metodo, certi che converga od anche sapere quando ci si è localizzati ad un intervallo da un punto del quale, innescando Newton, si ha sicuramente convergenza della successione. A tal fine è molto interessante il seguente risultato:

Teorema 7.8 *Sia $f(x) \in C^2_{[a,b]}$ e siano soddisfatte le seguenti condizioni: 1.$f(a)f(b) < 0$;*
2.$f'(x) \neq 0 \quad \forall x \in [a,b]$;

3.$f''(x) \geq 0 \quad o \quad \leq 0 \quad \forall x \in [a,b]$;
4.sia c l'estremo di $[a,b]$ *nel quale* $|f'(x)|$ *risulta minore e sia*

$$\left|\frac{f(c)}{f'(c)}\right| \leq b-a.$$

Sotto queste ipotesi il metodo di Newton converge all'unica radice di $f(x)=0$ *per un qualsiasi iterato iniziale* x_0 *in* $[a,b]$.

Osservazione 7.12 *Se la* $f(x)$ *è un polinomio nella base di Bernstein e cioè*

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

è banale verificare le ipotesi del teorema e quindi stabilire se nell'intervallo di definizione si può applicare il metodo di Newton con la certezza di convergere all'unica radice presente; infatti avremo:
1.(segno di b_0*)* $\neq$ *(segno di* b_n*);*
2.$b'_i := b_{i+1} - b_i \quad i = 0,\ldots,n-1$ *tutti dello stesso segno;*
3.$b''_i := b'_{i+1} - b'_i = b_{i+2} - 2b_{i+1} + b_i \quad i = 0,\ldots,n-2$ *tutti dello stesso segno;*
4.($|b'_0| < |b'_{n-1}|$ *e* $|b_0/b'_0| \leq b-a$*) oppure (*$|b'_{n-1}| < |b'_0|$ *e* $|b_n/b'_{n-1}| \leq b-a$*).*
se queste sono verificate si consideri $x_0 = (a+b)/2$ *e si applichi il metodo di Newton, il valore del polinomio e della sua derivata possono essere calcolati applicando l'algoritmo di de Casteljau una sola volta.*

Spesso, in pratica, un test per controllare se siamo nelle ipotesi di applicazione del metodo di Newton può risultare più costoso che innescare Newton, fare un paio di iterazioni e controllare se queste stanno convergendo o meno. Più precisamente localizzate le radici con la procedura **root_isolate**, in ogni intervallo individuato si può innescare il metodo di Newton a partire dal punto medio dell'intervallo e controllare semplicemente che ogni iterato resti interno all'intervallo di partenza; se questo accade la successione sta convergendo e soddisfatto il test di arresto si restituisce la soluzione trovata, altrimenti si suddivide ricorsivamente l'intervallo in due parti e si innesca Newton su ciascuno.

Viene ora presentato l'algoritmo *Bézier-Clipping* [NisSedKak90] che si basa sulla proprietà del gusio convesso delle curve di Bézier. Gli algoritmi tradizionali di suddivisione che utilizzano tale proprietà individuano gli zeri per mezzo di una suddivisione binaria del dominio, fornendo così una velocità di convergenza lineare. L'algoritmo *Bézier-Clipping* attraverso un più efficiente utilizzo del guscio convesso consente di raggiungere una velocità di convergenza superiore, tipica degli algoritmi numerici come quelli basati sul metodo degli intervalli di

Newton. Per tale motivo, esso può essere considerato in parte come algoritmo geometrico e in parte come lgoritmo numerico.

Sia data la funzione polinomiale nella base di Bernstein

$$d(u) = \sum_{i=0}^{n} d_i B_{i,n}(u) \qquad u \in [u_0, u_1]$$

di cui si vogliono determinare gli zeri. La funzione $d(u)$ può essere rappresentata come curva di Bézier:

$$D(u) = \sum_{i=0}^{n} D_i B_{i,n}(u) = (u, d(u), \qquad D_i = (\xi_i, d_i) = \left(\frac{i}{n}(u_1 - u_0), d_i\right).$$

È possibile, tramite la proprietà del guscio convesso, individuare l'intervallo parametrico di u in cui sicuramente la curva $D(u)$ non interseca l'asse u. Con riferimento alla figura 7.7, che mostra un esempio di curva $D(u)$, si vede che il guscio convesso determinato a partire dai punti di controllo D_i interseca l'asse u nei punti u_{min} e u_{max}. Tenendo presente che la curva è interna al guscio convesso, si può facilmente concludere che essa non ha intersezioni con l'asse u negli intervalli $[u_0, u_{min})$ e $(u_{max}, u_1]$. A questo punto la curva D può essere divisa

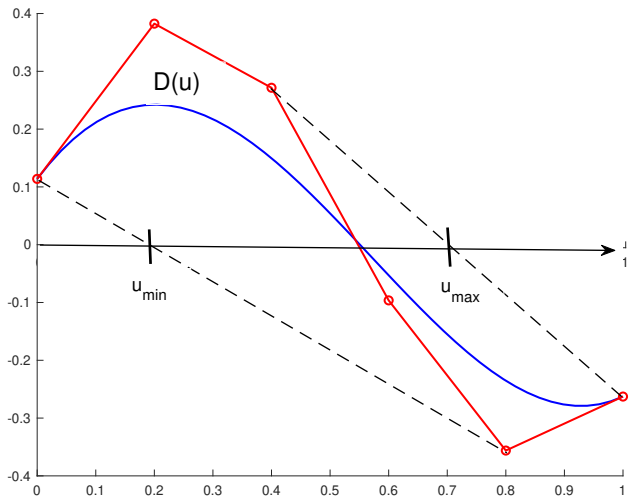

Figura 7.7: Itersezione guscio convesso con asse u.

in tre segmenti, definiti negli intervalli del dominio parametrico individuati, applicando l'algoritmo di suddivisione di de Casteljau nei punti u_{min} e u_{max}.

Inoltre, per quanto osservato, la ricerca dell'intersezione può essere ristretta al solo segmento definito nell'intervallo centrale $[u_{min}, u_{max}]$.

Il procedimento può essere reiterato, con la certezza di ridurre ad ogni passo l'intervallo parametrico entro cui è racchiusa l'intersezione, fino all'individuazione di un intervallo che soddisfi la precisione prefissata. In presenza di più radici l'intervallo non potrà essere ridotto ulteriormente, allora viene proposto di effettuare una suddivisione a metà dell'intervallo nel caso in cui la riduzione sia inferiore al 20%.

Per il calcolo del gusio convesso si è individuato un algoritmo ottimale avente complessità $O(n \log n)$, che diventa $O(n)$, dove n è il numero dei punti, se questi sono ordinati in senso circolare. Gli algoritmi disponibili in letteratura sono molti e hanno comportamenti diversi a seconda della distribuzione dei punti nel piano xy. L'algoritmo da noi utilizzato è dovuto a *Graham* e si può trovare in [PreSha85]. Nell'implementazione non si determina l'intero guscio convesso, contenente la funzione, ma si individuano due lati estremi del guscio da intersecare con l'asse u. Prendendo la retta congiungente il primo ed ultimo punto di controllo, la chiameremo ℓ, si divide il piano in due parti: quella superiore in cui i vertici sono ordinati rispetto a $+\infty$, mentre quella inferiore rispetto a $-\infty$. L'algoritmo esamina triple di punti consecutivi, nell'ordine circolare descritto, per determinare quali di essi formano un angolo interno riflesso $\geq \pi$ (rispetto al guscio convesso). Diremo che i tre punti $P_1P_2P_3$, formano un *left turn* se l'angolo $P_1\hat{P}_2P_3$ è riflesso, viceversa diremo che costituiscono un *right turn*. È evidente che nell'attraversamento di un guscio convesso, in base alle definizioni date, si incontrano solo sequenze di coefficienti *right turn* e in effetti, se la sequenza $P_1P_2P_3$ è *left turn* si può concludere che il punto P_2 non è un vertice del guscio convesso. L'attraversamento dei punti può allora essere così riassunto:

1. $P_1P_2P_3$ formano un *left turn*, allora si elimina P_2 e si controlla la terna $P_1P_3P_4$.

2. $P_1P_2P_3$ formano un *right turn*, allora procede all'esame di $P_2P_3P_4$.

Nella Figura 7.8 i punti $P_1P_2P_3$ formano un *right turn*, si procede quindi con $P_2P_3P_4$, ma questi formano un *left turn* per cui viene eliminato il punto P_3 e resta come ultimo lato del poligomo convesso P_2P_4 con cui si intersecherà l'asse u.

L'algoritmo, a partire dal primo coefficiente, se > 0 cercando fra i punti sotto la retta ℓ, e se < 0 cercando fra i punti sopra la retta ℓ, determina l'intersezione u_a tra il lato estremo del guscio convesso e l'asse u. Successivamante considerato l'ultimo coefficiente, se > 0 cercando fra i punti sotto la retta ℓ, individuerà un altro punto del guscio, o se < 0 cercando fra i punti sopra la retta ℓ, individuerà

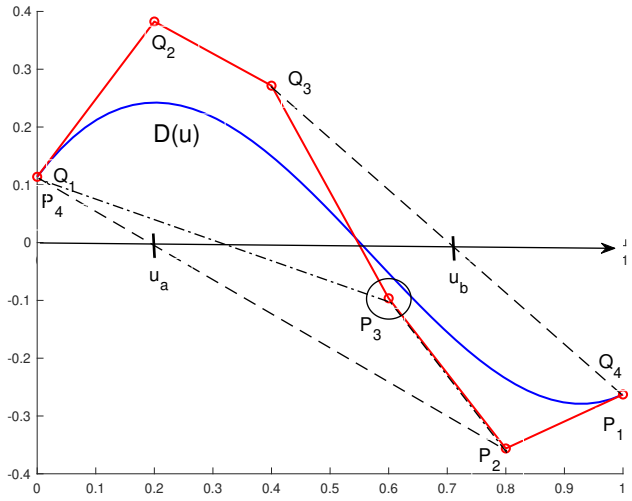

Figura 7.8: Eliminazione da una sequenza *left turn*.

un altro punto di intersezione tra l'altro lato estremo del poligono e l'asse u (u_b). Individuati i due punti u_a e u_b si potrà procedere al clipping o suddivisione.

7.5 Applicazioni

Vengono di seguito presentate alcune semplici, ma significative applicazioni della determinazione delle radici di una equazione polinomiale in un intervallo, per problemi di modellazione e interrogazione per curve piane di Bézier.

7.5.1 Punto interno/esterno ad una curva

Data una curva chiusa piana $\mathbf{C}(t)$ $t \in [0,1]$ nella forma di Bézier (o Bézier a tratti) e un punto Q del piano, si vuole determinare se il punto è interno o esterno alla regione di piano delimitata dalla curva. La soluzione di questo problema potrebbe consistere nel determinare le intersezioni fra una semiretta, che per semplicità possiamo considerare orizzontale, uscente da Q e la curva $\mathbf{C}(t)$; se il numero di intersezioni risulta dispari, il punto è interno, se pari o zero, il punto è esterno (vedi Fig.7.9 sinistra).

Analizziamo, in generale, come poter determinare le intersezioni fra una curva

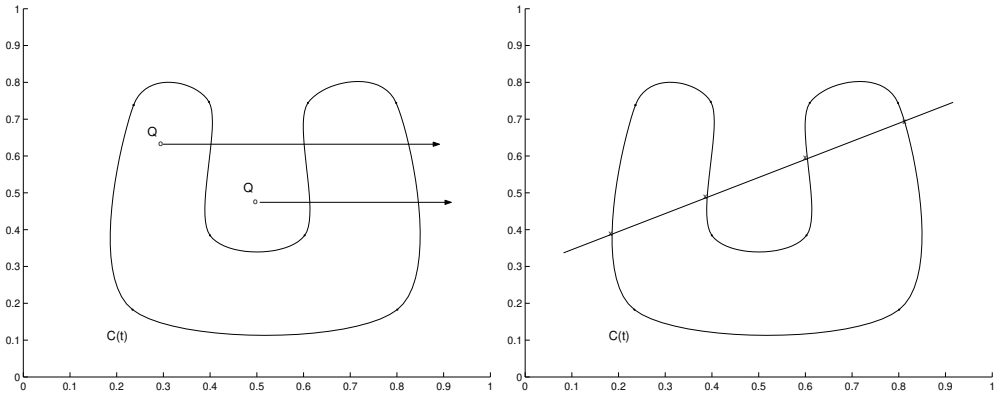

Figura 7.9: Punto interno/esterno ad una curva (sinistra); intersezione retta/curva (destra).

ed una retta del piano (vedi Fig.7.9 destra). Siano

$$\mathbf{C}(t) = \begin{pmatrix} C_1(t) \\ C_2(t) \end{pmatrix} \quad \text{e} \quad ax + by + c = 0$$

rispettivamente l'espressione della curva in forma parametrica con $t \in [0, 1]$ e l'equazione della retta in forma cartesiana. Sostituendo le componenti x ed y della curva nell'equazione della retta si ha:

$$aC_1(t) + bC_2(t) + c = 0.$$

Se $\mathbf{C}(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$ con $P_i = (x_i, y_i)^T$, sostituendo

$$a\sum_{i=0}^{n} x_i B_{i,n}(t) + b\sum_{i=0}^{n} y_i B_{i,n}(t) + c\sum_{i=0}^{n} B_{i,n}(t) = 0$$

e quindi

$$\sum_{i=0}^{n}(ax_i + by_i + c)B_{i,n}(t) = 0.$$

Si tratta di determinare le radici di un polinomio di grado n nell'intervallo $[0, 1]$. Le soluzioni saranno i parametri della curva in corrispondenza dei quali la curva e la retta si intersecano. La valutazione della curva in corrispondenza di questi parametri fornisce le coordinate cartesiane dei punti di intersezione.
Tornando al problema di determinare se un punto $Q = (qx, qy)^T$ è interno o esterno, è sufficiente considerare la retta orizzontale $y = qy$ e determinare le radici di

$$\sum_{i=0}^{n}(y_i - qy)B_{i,n}(t) = 0.$$

Si valuti la componente x della curva in corrispondenza delle soluzioni trovate, e si contino i valori trovati maggiori di qx (numero di intersezioni fra la curva e la semiretta orizzontale di origine Q e diretta come l'asse x positivo).

7.5.2 Punti estremi di una curva

Data una curva piana $\mathbf{C}(t)$ $t \in [0,1]$ nella forma di Bézier (o Bézier a tratti), si vogliono determinare i punti estremi (vedi Fig.7.10 sinistra). Siano

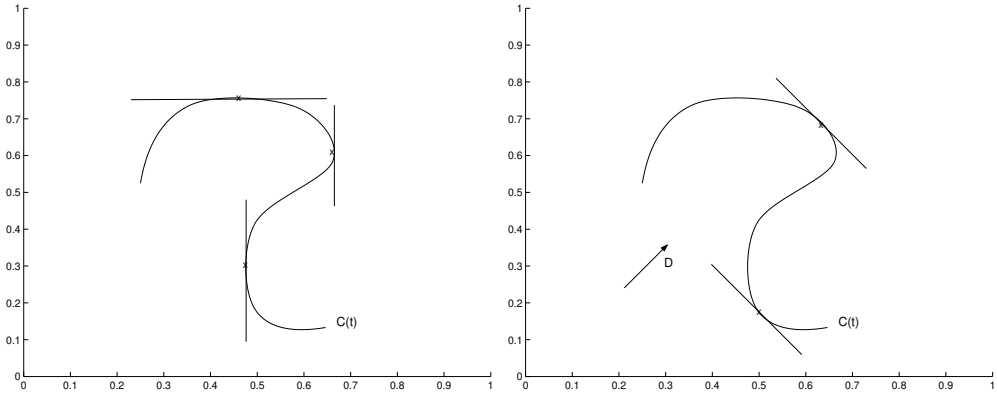

Figura 7.10: Punti estremi di una curva (sinistra); punti estremi secondo una fissata direzione D (destra).

$$\mathbf{C}(t) = \begin{pmatrix} C_1(t) \\ C_2(t) \end{pmatrix} \quad \text{e} \quad \mathbf{C}'(t) = \begin{pmatrix} C_1'(t) \\ C_2'(t) \end{pmatrix}$$

rispettivamente l'espressione della curva in forma parametrica con $t \in [0,1]$ e l'espressione della curva derivata prima che rappresenta per ogni punto t il vettore tangente alla curva $\mathbf{C}(t)$; i punti estremi saranno i punti della curva che hanno vettore tangente orizzontale o verticale e che quindi possono essere individuati come i punti della $\mathbf{C}'(t)$ con una delle componenti nulla e l'altra no. In pratica si cercano i punti t tali che:

$$C_1'(t^*) = 0 \quad \text{e} \quad C_2'(t^*) \neq 0$$

e

$$C_2'(t^*) = 0 \quad \text{e} \quad C_1'(t^*) \neq 0$$

Si tratta di determinare le radici di equazioni poinomiali e verificare che in corrispondenza di tali soluzioni l'altra componente non si annulli.

Se si cercano i punti estremi in una data direzione $D = (dx, dy)$ (vedi Fig.7.10 destra), basta determinare le radici dell'equazione polinomiale

$$\mathbf{C}'(t) \cdot D = 0$$

cioè

$$C_1'(t)dx + C_2'(t)dy = 0.$$

Se $\mathbf{C}(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$ con $P_i = (x_i, y_i)^T$, sarà $\mathbf{C}'(t) = n \sum_{i=0}^{n-1} (P_{i+1} - P_i) B_{i,n-1}(t)$ e si cercheranno le radici di

$$\sum_{i=0}^{n-1} [(x_{i+1} - x_i)dx + (y_{i+1} - y_i)dy] B_{i,n-1}(t) = 0.$$

7.5.3 Distanza di un punto da una curva

Data una curva piana $\mathbf{C}(t)$ $t \in [0, 1]$ nella forma di Bézier (o Bézier a tratti) e un punto Q del piano, si vuole determinare il punto della curva più prossimo a Q (vedi Fig.7.11). Si tratta di trovare quel punto della curva per cui

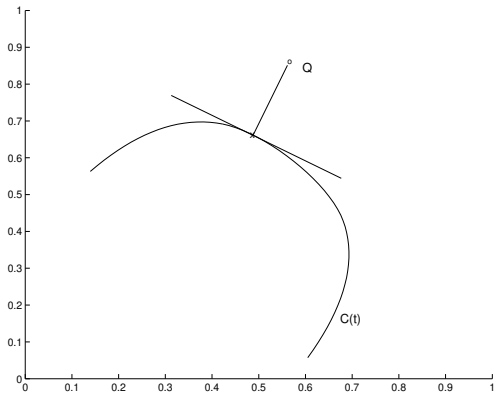

Figura 7.11: Distanza di un punto da una curva.

$$\mathbf{C}(t) - Q$$

risulta ortogonale alla tangente alla curva, cioè

$$(\mathbf{C}(t) - Q) \cdot \mathbf{C}'(t) = 0.$$

Si devono determinare gli zeri di un polinomio di grado $n(n-1)$.

7.5.4 Intersezioni fra due curve

Date due curve $\mathbf{C}(t)$ $t \in [0,1]$ e $\mathbf{D}(s)$ $s \in [0,1]$ si vogliono determinare i punti di intersezione (vedi Fig.7.12), che consistono a seconda delle applicazioni nelle coppie (t^*, s^*), nei punti $Q^* = (qx^*, qy^*)$ o in entrambi. I metodi noti in

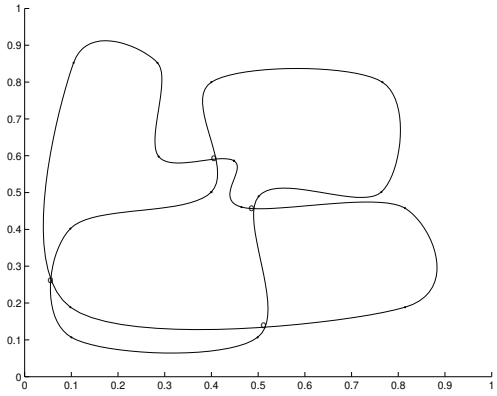

Figura 7.12: Intersezione fra due curve di Bézier a tratti.

letteratura sono numerosi e tutti interessantissimi; qui ne vedremo uno molto semplice che necessita in fase di preprocessing di trovare i punti estremi delle curve assegnate e nella fase finale di determinare l'intersezione fra due segmenti retti.
Il metodo consiste nel suddividere le due curve date in corrispondenza dei loro punti estremi ottenendo così due insiemi di tratti di curve che rappresentano rispettivamente la $\mathbf{C}(t)$ e la $\mathbf{D}(s)$; l'algoritmo continua nel confrontare ogni tratto della $\mathbf{C}(t)$ con quelli della $\mathbf{D}(s)$. Si osservi che ognuno di tali segmenti di curva è sicuramente contenuto nel rettangolo di vertici opposti gli estremi del segmento di curva (vedi Fig 7.13 sinistra). Questa osservazione è tanto semplice quanto potente. Infatti, in questo modo si può procedere con un test molto semplice e veloce per controllare se due tratti di curva si intersecano o meno. Il controllo consiste nel determinare se il rettangolo definito dagli estremi di un tratto si interseca con il rettangolo di un altro (vedi Fig.7.13 destra). Se non si intersecano i rettangoli, non si intersecheranno nemmeno i tratti di curva; se si intersecano i rettangoli si procede suddividendo i due tratti a metà e ricontrollando i quattro accoppiamenti. Si osservi ulteriormente che anche eventuali suddivisioni di un tratto saranno sicuramente contenute nel rettangolo di vertici opposti gli estremi. In questo caso i rettangoli così determinati vengono utilizzati come convessi contenenti i segmenti di curva e la suddivisione, che consiste nel valutare la curva e memorizzare i coefficienti intermedi, serve solo per controllare quando arrestare il procceso. Infatti se due tratti che si intersecano risultano

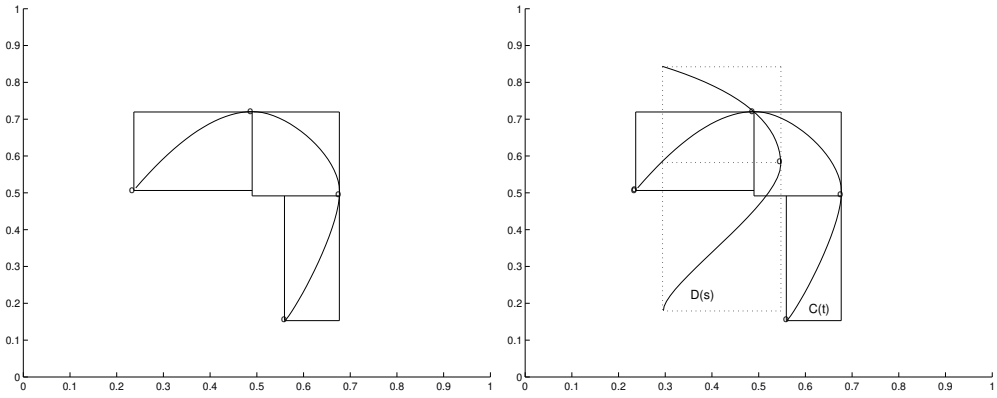

Figura 7.13: Punti estremi e rettangoli definiti dagli estremi dei tratti di curva (sinistra); intersezione di due curve (destra).

essere dei segmenti retti a meno di una prefissata tolleranza (test di linearità dei punti di controllo associati), allora si arresta il processo di test e suddivisione e si intersecano i due segmenti (vedi algoritmo di intersezione fra due segmenti).

Parte V

Algebra Lineare Numerica

Capitolo 8

Sistemi Lineari e Condizionamento

Uno dei problemi più frequenti nel calcolo scientifico è la soluzione di un sistema lineare; se questo è composto da tante equazioni quante incognite si dice sistema quadrato o normale. In forma matriciale può essere scritto come

$$A\mathbf{x} = \mathbf{b}$$

dove A è una data matrice di ordine $n \times n$, $\mathbf{b}$ è un dato vettore colonna con n elementi ed $\mathbf{x}$ è il vettore delle incognite.
In algebra lineare si studiano metodi per risolvere sistemi lineari non singolari. Un metodo noto è quello di Cramer (od anche regola di Cramer) nel quale ogni componente della soluzione è espressa come:

$$x_i = \frac{detA_i}{detA},$$

con A_i la matrice ottenuta da A sostituendo alla i-esima colonna il vettore $\mathbf{b}$, così che la soluzione del sistema lineare si riduce al calcolo di $n+1$ determinanti di ordine n e ad n divisioni.

Se si cerca di risolvere un sistema di 20 equazioni con la regola di Cramer, sarebbe necessario calcolare 21 determinanti di ordine 20. Come è noto per calcolare il determinante di una matrice $n \times n$, per esempio con la formula di Laplace o quella di Leibniz, servono $(n-1)n!$ moltiplicazioni; nel nostro esempio numerico sarà $19 \cdot 20!$ e quindi l'intero sistema lineare comporterà $19 \cdot 20! \cdot 21$ moltiplicazioni, più un ugual numero di addizioni. Su un normale PC, oggi si possono fare circa 10^9 moltiplicazioni al secondo (Giga FLOPS FLoating point OPeration per Second), così che, solo le moltiplicazioni richiederanno un tempo di calcolo di circa 30782 anni.

Nei corsi di algebra lineare si insegna che la soluzione di $A\mathbf{x} = \mathbf{b}$ può essere scritta come $\mathbf{x} = A^{-1}\mathbf{b}$, dove A^{-1} è l'inversa di A. Nella maggioranza dei problemi pratici non è necessario, se non inopportuno, calcolare A^{-1} per risolvere il sistema lineare. Con un esempio estremo, ma illustrativo, consideriamo un sistema di appena una equazione, come

$$7x = 21.$$

Il miglior modo per risolvere tale sistema è con una divisione

$$x = \frac{21}{7} = 3.$$

L'uso della matrice inversa porterebbe invece a fare due operazioni

$$x = 7^{-1} \cdot 21 = 0.142857\ldots \cdot 21 = 2.99997\ldots.$$

L'inversa richiede una divisione ed una moltiplicazione invece di appena una divisione. È il maggior numero di operazioni che ci induce ad evitare il calcolo dell'inversa. Nel seguito ci concentreremo quindi sulla soluzione diretta o iterativa di sistemi piuttosto che sul calcolo dell'inversa. Prima di passare ai metodi di soluzione in questo capitolo ci occuperemo brevemente dell'errore inerente o condizionamneto del problema di risolvere un sistema lineare.

8.1 Condizionamento del Problema $A\mathbf{x} = \mathbf{b}$

In questa sezione si vuole esaminare come, perturbazioni sugli elementi della matrice A e sugli elementi del termine noto $\mathbf{b}$ influenzano la soluzione $\mathbf{x}$ del sistema lineare. Queste perturbazioni sono tipicamente dovute agli errori di approssimazione quando la matrice A ed il termine noto $\mathbf{b}$ vengono rappresentati con numeri finiti. Per essere in grado di stimare gli errori, dobbiamo introdurre una misura delle *dimensioni* di un vettore o *distanza fra vettori*

8.1.1 Richiami sul concetto di norma

Siano $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Una **norma vettoriale** $\|\cdot\|$ è una funzione $\mathbb{R}^n \to \mathbb{R}$ che associa ad un vettore di $\mathbb{R}^n$ un valore reale (la lunghezza di quel vettore). Tale funzione, per essere una norma vettoriale deve soddisfare le tre seguenti proprietà:

1. $\|\mathbf{x}\| \geq 0 \quad \forall \mathbf{x} \quad \text{e} \quad \|\mathbf{x}\| = 0 \quad \text{se e solo se} \quad \mathbf{x} = \mathbf{0}$
2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (disuguaglianza triangolare)

3. $\|\alpha\mathbf{x}\| = |\alpha| \ \|\mathbf{x}\| \quad \text{con} \quad \alpha \in \mathbb{R}$.

Le norme vettoriali più importanti sono:

- $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$
- $\|\mathbf{x}\|_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}} = \sqrt{\mathbf{x}^T\mathbf{x}} = (\mathbf{x}^T\mathbf{x})^{\frac{1}{2}}$
- $\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n} |x_i|$

Questi sono tutti casi particolari della norma p

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}.$$

Osservazione 8.1 *La $\|\cdot\|_2$ è una generalizzazione ad $\mathbb{R}^n$ dell'usuale distanza in $\mathbb{R}^2$ o $\mathbb{R}^3$ ed è detta norma Euclidea. La $\|\cdot\|_\infty$ è detta norma infinito o norma del massimo.*

Con le norme, possiamo introdurre i concetti di **distanza** e **continuità** in $\mathbb{R}^n$. Sia $\tilde{\mathbf{x}}$ un vettore approssimazione di un vettore $\mathbf{x}$ non nullo. Per una data norma vettoriale $\|\cdot\|$, si definisce
errore assoluto

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|$$

ed **errore relativo**

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|}.$$

Teorema 8.1 (di equivalenza delle norme) *Siano $\|\cdot\|'$ e $\|\cdot\|''$ due norme vettoriali. Allora le due norme sono equivalenti, cioè esistono α e $\beta \in \mathbb{R}$ con $0 < \alpha \leq \beta$, tali che per ogni $\mathbf{x} \in \mathbb{R}^n$ è*

$$\alpha \ \|\cdot\|'' \leq \|\cdot\|' \leq \beta \ \|\cdot\|''.$$

Teorema 8.2 *Per ogni $\mathbf{x} \in \mathbb{R}^n$ si ha:*

- $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \ \|\mathbf{x}\|_\infty$
- $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n} \ \|\mathbf{x}\|_2$
- $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n \ \|\mathbf{x}\|_\infty$

Sia A una matrice $m \times n$ cioè $A \in \mathbb{R}^{m\times n}$. Possiamo pensare di definire una norma di matrici per misurare la dimensione di una matrice e la distanza fra matrici.
Una **norma matriciale** $\|\cdot\|$ è una funzione $\mathbb{R}^{m\times n} \to \mathbb{R}$ che associa ad una matrice di $\mathbb{R}^{m\times n}$ un valore reale. Tale funzione, per essere una norma matriciale deve soddisfare le tre seguenti proprietà:

1. $\|A\| \geq 0 \quad \forall A \quad$ e $\quad \|A\| = 0 \quad$ se e solo se $\quad A = 0$
2. $\|A + B\| \leq \|A\| + \|B\|$ (disuguaglianza triangolare)
3. $\|\alpha A\| = |\alpha| \ \|A\| \quad$ con $\quad \alpha \in \mathbb{R}$.

Definizione 8.1 (Norma indotta di matrice) *Per ogni norma vettoriale, possiamo definire una corrispondente norma matriciale come*

$$\|A\| = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\ \mathbf{x}\|}{\|\mathbf{x}\|}.$$

Osservazione 8.2 *Si noti che se $A \in \mathbb{R}^{m\times n}$, deve essere $\mathbf{x} \in \mathbb{R}^n$ da cui si tratta del max di una norma in $\mathbb{R}^m$ su una norma in $\mathbb{R}^n$.*

Così definita, $\|A\|$ è una norma di matrice, cioè soddisfa le tre proprietà date.

Risultato 8.1 *Se $\|\cdot\|$ denota una norma vettoriale e la corrispondente norma matriciale indotta, allora*

$$\|A\ \mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$$
$$\|A\ B\| \leq \|A\| \cdot \|B\|$$

Le norme matriciali più importanti sono:

- $\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{i,j}|$
- $\|A\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{i,j}|$
- $\|A\|_2 = \left(\lambda_{max}(A^T\ A)\right)^{1/2}$ (norma spettrale)

Risultato 8.2 (Relazioni fra le norme matriciali) *Per le norme che sono state introdotte valgono le seguenti relazioni, che possono essere dimostrate dalle relazioni fre le norme vettoriali e la definizione di norma indotta:*

- $\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{n}\ \|A\|_\infty$
- $\frac{1}{\sqrt{n}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\ \|A\|_1$
- $\max_{i,j} |a_{i,j}| \leq \|A\|_2 \leq n\ \max_{i,j} |a_{i,j}|$
- $\|A\|_2 \leq \sqrt{\|A\|_1\ \|A\|_\infty}$

8.1.2 Errore Inerente

Affrontiamo lo studio dell'errore inerente del problema $A\ \mathbf{x} = \mathbf{b}$ considerando separatamente eventuali perturbazioni sulla matrice A e sul vettore dei termini noti $\mathbf{b}$.

Introduciamo un vettore di perturbazione $\delta\mathbf{b} \in \mathbb{R}^n$ sul termine noto; cerchiamo $\mathbf{x} + \delta\mathbf{x} \in \mathbb{R}^n$ soluzione del sistema perturbato

$$A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}.$$

poiché è $A\ \mathbf{x} = \mathbf{b}$ risulterà

$$A\delta\mathbf{x} = \delta\mathbf{b}$$

da cui

$$\delta\mathbf{x} = A^{-1}\ \delta\mathbf{b}.$$

Passando alle norme

$$\|\delta\mathbf{x}\| = \|A^{-1}\ \delta\mathbf{b}\| \leq \|A^{-1}\| \cdot \|\delta\mathbf{b}\|$$

inoltre vale

$$\|\mathbf{b}\| = \|A\ \mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|.$$

Allora

$$\begin{aligned} \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} &\leq \frac{\|A^{-1}\| \cdot \|\delta\mathbf{b}\|}{\|\mathbf{x}\|} \\ &\leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}. \end{aligned}$$

Dunque il condizionamento di $A\ \mathbf{x} = \mathbf{b}$ dipende dalla costante

$$K = \|A\| \cdot \|A^{-1}\|$$

detto **numero di condizione** di A (rispetto a perturbazioni sul termine noto).

Introduciamo ora una matrice di perturbazione $\delta A \in \mathbb{R}^{n\times n}$ sulla matrice dei coefficienti, in modo che $A+\delta A$ sia ancora invertibile (si può dimostrare che per essere sicuri che $A + \delta A$ sia invertibile è sufficiente che $r = \|A^{-1}\| \cdot \|\delta A\| < 1$). Cerchiamo $\mathbf{x} + \delta\mathbf{x} \in \mathbb{R}^n$ soluzione del sistema lineare perturbato

$$(A + \delta A)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b}.$$

Sviluppando si ha

$$A\mathbf{x} + \delta A\ \mathbf{x} + A\ \delta\mathbf{x} + \delta A\ \delta\mathbf{x} = \mathbf{b}$$

$$\delta A\ \mathbf{x} + A\ \delta\mathbf{x} + \delta A\ \delta\mathbf{x} = \mathbf{0}$$

$$A\ \delta\mathbf{x} = -\delta A\ \mathbf{x} - \delta A\ \delta\mathbf{x}$$
$$\delta\mathbf{x} = -A^{-1} \cdot \delta A(\mathbf{x} + \delta\mathbf{x}).$$

Passando alle norme

$$\begin{aligned} \|\delta\mathbf{x}\| &= \|A^{-1} \cdot \delta A(\mathbf{x} + \delta\mathbf{x})\| \\ &\leq \|A^{-1}\| \cdot \|\delta A\| \cdot (\|\mathbf{x}\| + \|\delta\mathbf{x}\|) \end{aligned}$$

da cui

$$\|\delta\mathbf{x}\| \leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|\mathbf{x}\| + \|A^{-1}\| \cdot \|\delta A\| \cdot \|\delta\mathbf{x}\|$$
$$(1 - \|A^{-1}\| \cdot \|\delta A\|)\|\delta\mathbf{x}\| \leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|\mathbf{x}\|$$

e ricordando che $\|A^{-1}\| \cdot \|\delta A\| < 1$ e $\|A^{-1}\| \cdot \|\delta A\| = K\|\delta A\|/\|A\|$, si ottiene

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{K}{1-r}\frac{\|\delta A\|}{\|A\|};$$

dunque ancora il condizionamento di $A\ \mathbf{x} = \mathbf{b}$ dipende dalla costante $K = \|A\| \cdot \|A^{-1}\|$ (ora rispetto alla perturbazione sulla matrice dei coefficienti).

Teorema 8.3 *Sia A non sigolare e sia $r = \|A^{-1}\| \cdot \|\delta A\| < 1$; allora la matrice $A + \delta A$ è non singolare, e*

$$\|(A + \delta A)^{-1}\| \leq \frac{\|A^{-1}\|}{1-r}.$$

La soluzione del sistema perturbato

$$(A + \delta A)\mathbf{y} = \mathbf{b} + \delta\mathbf{b}$$

soddisfa

$$\frac{\|\mathbf{y} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{K(A)}{1-r}\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right).$$

Capitolo 9

Sistemi Lineari: metodi diretti

In questo capitolo vedremo due metodi diretti per risolvere un sistema lineare $A\ \mathbf{x} = \mathbf{b}$. Entrambi i metodi si basano su una *fattorizzazione* della matrice A, ossia costruiscono due matrici il cui **prodotto** sia uguale ad A:

1. Fattorizzazione LU (L (Low) matrice triangolare inferiore e diagonale unitaria, U (Up) matrice triangolare superiore)

$$A = L\ U$$

2. Fattorizzazione QR (Q matrice ortogonale, R (Right) matrice triangolare superiore)

$$A = Q\ R$$

Lo scopo è quello di ricondurre la soluzione del sistema (pieno e/o senza struttura particolare) $A\mathbf{x} = \mathbf{b}$ alla soluzione di due sistemi (con matrici strutturate); la soluzione del sistema si ottiene in due fasi:

1. fattorizzazione di A;
2. soluzione di due sistemi lineari le cui caratteristiche di soluzione sono più semplici e meno costose (rispetto a quelle del sistema originario).

9.1 Fattorizzazione LU

In questa sezione si studia la possibilità di fattorizzare la matrice A nel prodotto di una matrice L triangolare inferiore per una matrice U triangolare superiore:

$$A_{n\times n} = L_{n\times n} \cdot U_{n\times n}$$

con

$$L = \begin{pmatrix} 1 & & & \\ \ell_{2,1} & 1 & & \\ \ell_{3,1} & & 1 & \\ \vdots & & & \ddots \\ \ell_{n,1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix} \quad U = \begin{pmatrix} u_{1,1} & \dots & & u_{1,n} \\ & u_{2,2} & & \\ & & u_{3,3} & \\ & & & \ddots \quad \vdots \\ & & & u_{n,n} \end{pmatrix}.$$

L'esistenza di una tale fattorizzazione renderebbe facile la determinazione della soluzione del sistema normale, infatti si avrebbe:

$$LU\mathbf{x} = \mathbf{b}$$

e ponendo $U\mathbf{x} = \mathbf{y}$ si potrebbe risolvere il sistema

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

per sostituzione in avanti (forward-substitution); determinato $\mathbf{y}$ si risolve il sistema

$$U\mathbf{x} = \mathbf{y}$$

per sostituzione all'indietro (backward-substitution).

9.1.1 Sostituzione in Avanti

Il primo sistema da risolvere è

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

in cui L è una matrice triangolare inferiore.

- La soluzione si ottiene con la sostituzione in avanti, mediante l'algoritmo che segue

```
y(1) = b(1) / L(1,1)
 per i = 2,..,n
   per k=1,..,i-1
     b(i) = b(i) - L(i,k) * y(k)
   y(i) = b(i) / L(i,i)
```

  La struttura dell'algoritmo è triangolare. Per generare $y(i)$ si compiono $2i-1$ operazioni e per la precisione $i-1$ moltiplicazioni, 1 divisione e $i-1$ sottrazioni.

- Il costo computazionale della sostituzione in avanti è dato da

$$1 + \sum_{i=2}^{n}(2i-1) = \sum_{i=1}^{n}(2i-1) = 2\sum_{i=1}^{n} i - n = n(n+1) - n = n^2$$

- Se L è una matrice triangolare inferiore con elementi diagonali unitari, allora nell'algoritmo non ci sono divisioni:

```
y(1) = b(1)
 per i = 2,..,n
   per k=1,..,i-1
     b(i) = b(i) - L(i,k) * y(k)
   y(i) = b(i)
```

ed il costo computazionale diventa

$$\sum_{i=2}^{n} 2(i-1) = \sum_{i=1}^{n} 2(i-1) = n(n+1) - 2n = n^2 - n$$

Esempio 9.1 *Sostituzione in avanti.*

$$\begin{pmatrix} \ell_{1,1} & 0 & 0 \\ \ell_{2,1} & \ell_{2,2} & 0 \\ \ell_{3,1} & \ell_{3,2} & \ell_{3,3} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$y_1 = b_1/\ell_{1,1}$ — 1 *operazione*
$\downarrow$
$y_2 = (b_2 - \ell_{2,1} * y_1)/\ell_{2,2}$ — 3 *operazioni*
$\downarrow$
$y_3 = (b_3 - \ell_{3,1} * y_1 - \ell_{3,2} * y_2)/\ell_{3,3}$ — 5 *operazioni*

Costo computazionale: $n^2 = 3^2 = 9$ *operazioni.*

9.1.2 Sostituzione all'indietro

Il secondo sistema da risolvere è

$$U\,\mathbf{x} = \mathbf{y}$$

in cui U è una matrice triangolare superiore.

- La soluzione si ottiene con la sostituzione all'indietro, mediante l'algoritmo che segue

```
x(n) = y(n)/U(n,n)
 per i=n-1,..,1
   per k=i+1,..,n
     y(i) = y(i) - U(i,k) * x(k)
   x(i) = y(i) / U(i,i)
```

La struttura dell'algoritmo è triangolare. Per generare $x(i)$ si compiono $2i-1$ operazioni e per la precisione $i-1$ moltiplicazioni, 1 divisione e $i-1$ sottrazioni.

- Il costo computazionale della sostituzione all'indietro è dato da

$$1 + \sum_{i=2}^{n}(2i-1) = ... = n^2$$

Esempio 9.2 *Sostituzione all'indietro*

$$\begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$x_3 = y_3/u_{3,3}$ — 1 *operazione*

$\downarrow$

$x_2 = (y_2 - u_{2,3} * x_3)/u_{2,2}$ — 3 *operazioni*

$\downarrow$

$x_1 = (y_1 - u_{1,2} * x_2 - u_{1,3} * x_3)/u_{1,1}$ — 5 *operazioni*

Costo computazionale: $n^2 = 3^2 = 9$ *operazioni.*

Riassumendo: determinata una fattorizzazione LU di A, il vettore soluzione $\mathbf{x}$ si può ottenere facilmente risolvendo due sistemi triangolari, uno inferiore e l'altro superiore.

Ci si concentrerà ora sulla fattorizzazione LU di A; si osserva che l'ipotesi che A sia non singolare non garantisce l'esistenza di una fattorizzazione LU, infatti vale il seguente teorema.

Teorema 9.1 *Se i minori principali di ordine k di A per $k = 1, \ldots, n-1$ sono diversi da zero, allora esiste una ed una sola fattorizzazione LU di A.*

9.1.3 Metodo di Gauss

Procediamo in un caso semplice (matrice $A_{3\times 3}$) ad illustrare come funziona il metodo di Gauss per fattorizzare LU una matrice A (il procedimento in oggetto può essere applicato anche a matrici rettangolari).
Sia

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

con

$$a_{1,1} \neq 0 \quad \text{e} \quad det\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = a_{1,1}a_{2,2} - a_{1,2}a_{2,1} \neq 0.$$

Consideriamo la matrice L_1 siffatta:

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{a_{2,1}}{a_{1,1}} & 1 & 0 \\ -\frac{a_{3,1}}{a_{1,1}} & 0 & 1 \end{pmatrix}$$

(si noti la necessità che $a_{1,1} \neq 0$; $a_{1,1}$ è detto perno o pivot) allora sarà:

$$L_1 A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a^{(1)}_{2,2} & a^{(1)}_{2,3} \\ 0 & a^{(1)}_{3,2} & a^{(1)}_{3,3} \end{pmatrix}$$

con $a^{(1)}_{i,j} = a_{i,j} - \frac{a_{i,1}}{a_{1,1}} a_{1,j},\ i,j = 2,3.$
Consideriamo ora la matrice L_2 siffatta:

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a^{(1)}_{3,2}}{a^{(1)}_{2,2}} & 1 \end{pmatrix}$$

(si noti la necessità che $a^{(1)}_{2,2} = \frac{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}}{a_{1,1}} \neq 0$, ossia che $a_{1,1}a_{2,2} - a_{1,2}a_{2,1} \neq 0$; $a^{(1)}_{2,2}$ è detto perno o pivot) allora sarà:

$$L_2(L_1 A) = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a^{(1)}_{2,2} & a^{(1)}_{2,3} \\ 0 & 0 & a^{(2)}_{3,3} \end{pmatrix}$$

con $a^{(2)}_{3,3} = a^{(1)}_{3,3} - \frac{a^{(1)}_{3,2}}{a^{(1)}_{2,2}} a^{(1)}_{2,3}$.
Chiamiamo U la $L_2 L_1 A$ e notiamo che la U è del tipo cercato. Si noti che la L_1 e la L_2 sono non singolari, allora

$$A = L_1^{-1} L_2^{-1} U.$$

Chi sono L_1^{-1} e L_2^{-1} e chi è $L_1^{-1} L_2^{-1}$? Saranno molto semplicemente e senza richiedere calcoli espliciti:

$$L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{a_{2,1}}{a_{1,1}} & 1 & 0 \\ \frac{a_{3,1}}{a_{1,1}} & 0 & 1 \end{pmatrix} \quad L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{a^{(1)}_{3,2}}{a^{(1)}_{2,2}} & 1 \end{pmatrix}$$

$$L_1^{-1} L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{a_{2,1}}{a_{1,1}} & 1 & 0 \\ \frac{a_{3,1}}{a_{1,1}} & \frac{a^{(1)}_{3,2}}{a^{(1)}_{2,2}} & 1 \end{pmatrix}$$

e se chiamiamo quest'ultima L avremo che $A = L\,U$ come volevamo.

```
U=A
L=I
per k=1,..,n-1
  per j=k+1,..,n
    L(j,k) = U(j,k) / U(k,k)
    per i=1,..,k
      U(j,i) = 0
    per i=k+1,..,n
      U(j,i) = U(j,i) - L(j,k) * U(k,i)
```

Per determinare la fattorizzazione suddetta sono necessarie $n(n-1)/2$ moltiplicazioni/divisioni floating point per il calcolo degli elementi di L e $(n-1)^2 + (n-2)^2 + \ldots + 2^2 + 1 = (n-1)n(2n-1)/6$ moltiplicazioni/divisioni floating point per gli elementi di U. La complessità computazionale, in termini di operazioni di moltiplicazione e divisione, è quindi:

$$\frac{(n-1)n(2n-1)}{6} + \frac{(n-1)n}{2}$$
$$= \frac{(n-1)n(2n+3)}{6}$$

$$= \frac{(n-1)n(n+1)}{3} = \frac{n^3 - n^2}{3} \approx \frac{1}{3}n^3.$$

Esempio 9.3 *Fattorizzazione $A = L\ U$ di Gauss.*

$$\textit{Sia} \quad A\mathbf{x} = \mathbf{b} \qquad \textit{con} \qquad A = \begin{pmatrix} 2 & 1 & 0 \\ 4 & 5 & 2 \\ 6 & 15 & 12 \end{pmatrix} \qquad e \qquad \mathbf{b} = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}.$$

L_1 deve essere triangolare inferiore e tale da rendere nulli gli elementi della prima colonna di A sotto l'elemento $a_{11} = 2$.

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}$$

$$L_1\ A = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 1 & 0 \\ 4 & 5 & 2 \\ 6 & 15 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 0 & 12 & 12 \end{pmatrix}$$

L_2 deve essere triangolare inferiore e tale da rendere nulli gli elementi della seconda colonna di $L_1\ A$ sotto ad $a^{(1)}_{2,2} = 3$.

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{pmatrix}$$

$$L_2\ (L_1\ A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 0 & 12 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 0 & 0 & 4 \end{pmatrix} = U$$

Allora

$$L_2\ L_1\ A = U,$$

$$A = L_1^{-1}\ L_2^{-1}\ U = L\ U$$

e risulta

$$L = L_1^{-1}\ L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}.$$

Fattorizzata la matrice, si procede alla soluzione dei sistemi triangolari

$$L\mathbf{y} = \mathbf{b} \quad e \quad U\mathbf{x} = \mathbf{y}.$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} \quad y_1 = 2;$$

$$\begin{pmatrix} 1 & 0 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} -3 \\ -4 \end{pmatrix} \quad y_2 = -3;$$

$$(1)(y_3) = (8) \quad y_3 = 8;$$

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 8 \end{pmatrix} \quad x_3 = 2;$$

$$\begin{pmatrix} 2 & 1 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ -7 \end{pmatrix} \quad x_2 = -7/3;$$

$$(2)(x_1) = (13/3) \quad x_1 = 13/6.$$

9.1.4 Calcolo di A^{-1}

Sia $A \in \mathbb{R}^{n \times n}$, invertibile; vogliamo calcolare A^{-1}, risolvendo il sistema

$$A\ X = I_n, \qquad X = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \ldots\ldots \mid \mathbf{x}_n], \qquad I_n = [\mathbf{e}_1 \mid \mathbf{e}_2 \mid \ldots\ldots \mid \mathbf{e}_n]$$

I_n è la matrice identità di dimensione n; i vettori $\mathbf{e}_1$, ... $\mathbf{e}_n$ sono la base canonica di $\mathbb{R}^n$. $X \in \mathbb{R}^{n \times n}$ è la matrice incognita.

Risolvere $A\ X = I_n$, per ottenere $X = A^{-1}$, è equivalente a risolvere n sistemi lineari, tutti con la stessa matrice A dei coefficienti

$$\begin{cases} A\ \mathbf{x}_1 = \mathbf{e}_1 \\ A\ \mathbf{x}_2 = \mathbf{e}_2 \\ \quad \vdots \\ A\ \mathbf{x}_n = \mathbf{e}_n \end{cases}$$

Si fattorizzi $A = L\ U$ una sola volta, e poi si risolvano 2 n sistemi lineari di

forma triangolare

$$\begin{cases} \textbf{(i)} & A = L\,U & \frac{1}{3}n^3 - \frac{1}{3}n^2 \\ \textbf{(ii)} & \begin{cases} L\mathbf{y}_1 = \mathbf{e}_1 & n^2 - n \\ U\mathbf{x}_1 = \mathbf{y}_1 & n^2 \end{cases} \\ & \vdots \\ & \vdots \\ & \begin{cases} L\mathbf{y}_n = \mathbf{e}_n & n^2 - n \\ U\mathbf{x}_n = \mathbf{y}_n & n^2 \end{cases} \end{cases}$$

9.1.5 Calcolo del $det(A)$

Fattorizzare $A = L\,U$ permette di calcolare determinante della matrice A con un basso costo computazionale; vale

$$det(A) = det(L)\ det(U) = det(U) = \prod_{i=1}^{n} u_{i,i}$$

Abbiamo utilizzato il Teorema di Binet, che $det(L) = 1$ e che il determinante di una matrice triangolare è dato dal prodotto degli elementi della diagonale.

9.1.6 Metodo di Gauss con scambio delle righe

Sia dato il sistema

$$\begin{pmatrix} 0 & 3 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix};$$

essendo $a_{1,1} = 0$, non si può applicare l'algoritmo di fattorizzazione di Gauss alla matrice dei coefficienti, ma si osserva che questo sistema è equivalente a

$$\begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix};$$

ed ora $a_{1,1} \neq 0$.
Allora è possibile trovare una matrice di permutazione P per cui $P\ A$ sia fattorizzabile $L\,U$, cioè

$$P\ A = L\,U.$$

Matrice di Permutazione: una tale matrice si ottiene dalla matrice identità I_n permutando le righe. Applichiamo P ad una matrice $A \in \mathbb{R}^{n\times n}$, *allora* :

$$P\ A \quad \longrightarrow \quad \text{permuta righe di } A$$

$$A\ P \quad \longrightarrow \quad \text{permuta colonne di } A$$

Ad esempio:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$P\ A = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \end{pmatrix} \qquad A\ P = \begin{pmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 9 & 8 \end{pmatrix}.$$

Se esiste una opportuna matrice P che permette la fattorizzazione $PA{=}LU$, questa non è detto sia determinabile a priori, mentre si può procedere alla sua determinazione costruttivamente. Questo è ciò che fa l'algoritmo di eliminazione di Gauss con scambio delle righe (pivoting parziale). Si illustra il metodo con un esempio. Sia data la matrice

$$A = A_0 = \begin{pmatrix} 4 & -8 & 2 \\ 2 & -4 & 6 \\ 1 & -1 & 3 \end{pmatrix};$$

il pivot $a_{i,1}^{(0)}$ è 4 e quindi $P_1 = I$ e

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ -1/4 & 0 & 1 \end{pmatrix};$$

si costruisce

$$A_1 = L_1\ P_1\ A_0 = \begin{pmatrix} 4 & -8 & 2 \\ 0 & 0 & 5 \\ 0 & 1 & 5/2 \end{pmatrix}.$$

Il pivot $a_{i,2}^{(1)}$ è 1; le matrici P_2 ed L_2 saranno:

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Si costruisce

$$A_2 = L_2\ P_2\ A_1 = \begin{pmatrix} 4 & -8 & 2 \\ 0 & 1 & 5/2 \\ 0 & 0 & 5 \end{pmatrix} = U.$$

In definitiva è

$$L_2\ P_2\ L_1\ P_1\ A = U.$$

Si noti che

$$L_2\ P_2\ L_1\ P_1 = L_2\ P_2\ L_1\ P_2^{-1}\ P_2 P_1$$

e se poniamo $T = L_2\ P_2\ L_1\ P_2^{-1}$ si ha che T è triangolare inferiore e quindi

$$T\ P_2\ P_1\ A = U$$

e ponendo

$$L^{-1} = T \quad \text{e} \quad P = P_2\ P_1$$

si ha

$$P\ A = L\ U.$$

Completando l'esempio si ha

$$L = P_2\ L_1^{-1}\ P_2^{-1}\ L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix}$$

ed infatti

$$L\ U = \begin{pmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & -8 & 2 \\ 0 & 1 & 5/2 \\ 0 & 0 & 5 \end{pmatrix} = \begin{pmatrix} 4 & -8 & 2 \\ 1 & -1 & 3 \\ 2 & -4 & 6 \end{pmatrix} = P\ A.$$

Nell'implementare tale metodo ci si accorge immediatamente che non è possibile perdere l'informazione P, infatti se fattorizziamo $P\ A$ il sistema da risolvere sarà

$$P\ A\ \mathbf{x} = P\ \mathbf{b}.$$

9.1.7 Fattorizzazione $P\ A = L\ U$

Riassumendo, abbiamo visto cosa possiamo fare se abbiamo una matrice A per la quale il metodo di fattorizzazione semplice di Gauss fallisce. Arrichiamo allora il metodo visto in precedenza con dei passi di pivotaggio. Formalizzando quanto già detto vale il seguente teorema.

Teorema 9.2 (Esistenza di $P\ A = L\ U$) *Per ogni matrice $A \in \mathbb{R}^{n\times n}$, esiste una matrice di permutazione P tale che*

$$P\ A = L\ U$$

con L triangolare inferiore con elementi unitari e U triangolare superiore.

metodo di gauss con scambio delle righe

Il teorema enunciato assicura che, se la fattorizzazione di Gauss di una matrice A sembra fallire (perché ad un certo passo k si ottiene un perno $a_{k,k}^{(k-1)} \approx 0$), è possibile superare questo passo andando a cercare (nella colonna $k-$esima della matrice A_{k-1} corrente) un nuovo perno $a_{i,k}^{(k-1)} \neq 0$ ed introducendo una permutazione di righe in modo che $a_{i,k}^{(k-1)}$ occupi la posizione di $a_{kk}^{(k-1)}$

$$L_{k-1} \ldots L_1 \, A = \begin{pmatrix} \ddots & \vdots & & \\ \ldots & a_{k,k}^{(k-1)} = 0 & \ldots & \leftarrow \text{ riga } k \\ & a_{k+1,k}^{(k-1)} & & \\ & \vdots & & \\ & a_{i,k}^{(k-1)} \neq 0 & \ldots & \leftarrow \text{ riga } i \\ & \vdots & & \\ & a_{n,k}^{(k-1)} & \ldots & \\ & \uparrow & & \\ & \text{colonna } k & & \end{pmatrix} = A_{k-1}.$$

Se ad un certo passo k si ottiene un perno $a_{k,k}^{(k-1)} = 0$ e nella colonna k-esima della matrice A_{k-1} sotto l'elemento diagonale tutti gli elemnti sono nulli, allora il procedimento si arresta e quanto trovato fino a questo punto porta alla fattorizzazione cercata; si veda il seguente esempio:

$$A = A_0 = \begin{pmatrix} 4 & -8 & 2 \\ 1 & -2 & 1/2 \\ 2 & -4 & 1 \end{pmatrix};$$

il pivot $a_{i,1}^{(0)}$ è 4 e quindi $P_1 = I$ e

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1/4 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix};$$

si costruisce

$$A_1 = L_1 \, P_1 \, A_0 = \begin{pmatrix} 4 & -8 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Poiché $a_{2,2}^{(1)} = 0$ e nella colonna, sotto di lui, gli elementi sono tutti nulli, il procedimento si arresta; infatti A_1 è già triangolare superiore. In definitiva è

$$L_1 \, P_1 \, A \; = \; A_1 \; = \; U,$$

con $L = L_1$ e $P = P_1$.

Riassumendo, se la matrice $A \in \mathbb{R}^{n\times n}$ è non singolare, come richiesto nel caso si voglia risolvere un sistema lineare, la fattorizzazione di Gauss con scambio delle righe effettuerà esattamente $n-1$ passi; nel caso in cui la matrice sia singolare verrà effettuato un numero di passi uguale al rango della matrice A.

Prima di formalizzare l'algoritmo di fattorizzazione $L\ U$ con scambio delle righe, applichiamo l'analisi all'indietro per stimare la stabilità dell'algoritmo di fattorizzazione descritto.

9.1.8 Stabilità della Fattorizzazione $L\ U$

Poichè le operazioni aritmetiche che intervengono nell'algoritmo di fattorizzazione sono effettuate in aritmetica finita, segue che questi algoritmi, anziché generare i fattori L ed U per A, generano dei fattori non esatti

$$\tilde{L} = L + \delta L \quad \text{e} \quad \tilde{U} = U + \delta U.$$

Posto $A + \delta A = \tilde{L}\ \tilde{U}$, cioè la matrice di cui $\tilde{L}\ \tilde{U}$ è effettivamente una fattorizzazione, si ha

$$A + \delta A = \tilde{L}\ \tilde{U} = (L + \delta L)(U + \delta U) =$$

$$= L\ U + \delta L\ U + L\ \delta U + \delta L\ \delta U$$

da cui

$$\delta A = \delta L\ U + L\ \delta U + \delta L\ \delta U$$

da cui segue che se gli elementi di L ed U sono grandi, gli elementi di δA sono grandi e quindi, gli errori di arrotondamento si amplificano (analisi dell'errore all'indietro). Perciò diremo che la fattorizzazione $A = L\ U$ è **stabile** numericamente se gli elementi di L ed U non sono troppo grandi rispetto agli elementi di A. In particolare se esistono delle costanti a e b indipendenti dagli elementi e dall'ordine di A tali che $|\ell_{i,j}| \leq a$ e $|u_{i,j}| < b$, allora si dice che la fattorizzazione $L\ U$ è **stabile in senso forte**; se le costanti a e b dipendono dall'ordine di A, allora si dice che la fattorizzazione $L\ U$ è **stabile in senso debole**.

Sebbene si sia visto che per ogni matrice, a meno di una permutazione di righe, esiste sempre la fattorizzazione $L\ U$, in generale questa potrebbe non essere stabile. Infatti gli elementi

$$\ell_{i,k} = \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}$$

possono assumere valori grandi e quindi gli elementi di L possono crescere oltre ogni limite.
Questo inconveniente può essere eliminato facendo un opportuno scambio di righe ad ogni passo dell'algoritmo, cioè scegliendo i pivot $a_{k,k}^{(k-1)}$ in modo che

$$|a_{k,k}^{(k-1)}| \geq \{|a_{i,k}^{(k-1)}|\}_{i=k,\ldots,n}.$$

Questa strategia è indicata con **scambio delle righe e perno massino** e limita ad 1 il valore degli elementi della matrice L, mentre non riesce a limitare gli elementi di U che possono crescere esponenzialmente con l'ordine n di A. Infatti si ha che:

$$\max |u_{i,j}| \leq 2^{n-1} \max |a_{i,j}|.$$

Perciò l'algoritmo di Gauss con scambio delle righe e perno massimo genera una fattorizzazione **stabile in senso debole** infatti si ha $a = 1$ e $b = 2^{n-1} \max |a_{i,j}|$.

9.1.9 Metodo di Gauss con scambio delle righe e perno massimo

Le considerazioni sopra esposte portano a modificare l'algoritmo di Gauss introducendo lo scambio delle righe non solo per individuare un pivot non nullo, ma quello massimo. Questo si realizza al passo $k-$esimo scegliendo come pivot fra gli $a_{i,k}^{(k-1)}$ con $i = k, \ldots, n$ il più grande in valore assoluto, così che a scambio delle righe effettuato risulti

$$|a_{k,k}^{(k-1)}| \geq \max_{i=k+1,\ldots,n} \{|a_{i,k}^{(k-1)}|\}.$$

Quando si usa questo criterio per la scelta dei pivot, il metodo di Gauss si chiama metodo di eliminazione di Gauss con scambio delle righe e perno (o pivot) massimo.

Esempio 9.4 *Fattorizzazione di Gauss con scambio delle righe e perno massimo.*

$$\textit{Sia} \quad A\mathbf{x} = \mathbf{b} \qquad \textit{con} \qquad A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix} \qquad \textit{e} \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

La soluzione analitica è data da:

$$x_1 = 1 + \frac{1}{10^{20} - 1} \simeq 1 \qquad x_2 = 1 - \frac{1}{10^{20} - 1} \simeq 1.$$

Procediamo a risolvere il sistema lineare in aritmetica finita a doppia precisione, prima con Gauss senza perno massimo e poi con perno massimo; procedendo senza perno massimo avremo:

$$L_1 = \begin{pmatrix} 1 & 0 \\ -10^{20} & 1 \end{pmatrix}, \quad L_1\, A = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix} = U, \quad L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}.$$

Fattorizzata la matrice, si procede alla soluzione dei sistemi triangolari

$$L\mathbf{y} = \mathbf{b} \quad e \quad U\mathbf{x} = \mathbf{y}.$$

$$\begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad y_1 = 1;$$

$$(1)(y_2) = 2 - 10^{20}, \qquad y_2 = -10^{20}.$$

$$\begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -10^{20} \end{pmatrix}, \quad x_2 = 1;$$

$$(10^{-20})(x_1) = -1 + 1, \qquad x_1 = 0.$$

Come si vede la seconda componente del vettore soluzione è completamente sbagliata. Durante il procedimento (algoritmo) sono stati commessi gravi errori numerici che hanno invalidato il risultato (algoritmo non stabile). Se invece usiamo il perno massimo, avremo:

$$P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad P_1\, A = \begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix}, \quad P = P_1, \quad e \quad P_1\, \mathbf{b} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

procedendo si ha:

$$L_1 = \begin{pmatrix} 1 & 0 \\ -10^{-20} & 1 \end{pmatrix}, \quad L_1\, P_1\, A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = U, \quad L = \begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix}.$$

Fattorizzata la matrice, si procede alla soluzione dei sistemi triangolari

$$L\mathbf{y} = P\mathbf{b} \quad e \quad U\mathbf{x} = \mathbf{y}.$$

$$\begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad y_1 = 2;$$

$$(1)(y_2) = 1 - 2 \cdot 10^{-20}, \qquad y_2 = 1.$$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad x_2 = 1;$$

$$(1)(x_1) = 2 - 1, \qquad x_1 = 1.$$

Ora la soluzione numerica coincide nella precisione di calcolo con la soluzione analitica; l'algoritmo con scambio delle righe e perno massimo è numericamente stabile (anche se solo in senso debole).

9.1.10 Caso Tridiagonale

Sia dato il sistema lineare $A\mathbf{x} = \mathbf{b}$ con A matrice tridiagonale della forma

$$A = \begin{pmatrix} c_1 & e_1 & 0 & \dots & 0 \\ d_2 & c_2 & e_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & d_{n-1} & c_{n-1} & e_{n-1} \\ 0 & \dots & 0 & d_n & c_n \end{pmatrix}.$$

L'algorimo di fattorizzazione LU di A risulta semplificato e prende il nome di algoritmo di Thomas. Segue che le matrici L ed U sono bidiagonali, ed hanno la forma

$$L = \begin{pmatrix} 1 & & & \\ \beta_2 & 1 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \ddots \\ & & & \beta_n \quad 1 \end{pmatrix}, \quad U = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ & \alpha_2 & \ddots & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & \gamma_{n-1} \\ & & & & \alpha_n \end{pmatrix}.$$

Moltiplicando L ed U e uguagliando i valori con quelli di A, si ottiene

$$\gamma_i = e_i \quad i = \dots, n-1,$$

$$\alpha_1 = c_1, \quad \beta_i = \frac{d_i}{\alpha_{i-1}}, \quad \alpha_i = c_i - \beta_i \gamma_{i-1}, \quad i = 2, \dots, n.$$

Questo algoritmo richiede $3n - 3$ operazioni floating point, invece delle $\frac{2}{3}n^3 + O(n^2)$ richieste per l'eliminazione di Gauss.

Una volta determinati i coefficienti di L ed U, la soluzione $\mathbf{x}$ richiede la soluzione a cascata dei due sistemi

$$L\mathbf{y} = \mathbf{b}, \qquad U\mathbf{x} = \mathbf{y}.$$

La struttura bidiagonale di L e U può essere sfruttata anche nella risoluzione di questi sistemi; infatti per $L\mathbf{y} = \mathbf{b}$ si ha

$$y_1 = b_1, \quad y_i = b_i - \beta_i y_{i-1}, \quad i = 2, \dots, n,$$

al costo di $2(n-1)$ operazioni floating point. Applicando la sostituzione all'indietro al secondo sistema, $U\mathbf{x} = \mathbf{y}$, si ha

$$x_n = \frac{y_n}{\alpha_n}, \quad x_i = \frac{y_i - \gamma_i x_{i+1}}{\alpha_i}, \quad i = n-1, \ldots, 1,$$

al costo di $3(n-1)+1$ operazioni floating point.

9.1.11 Metodo di Cholesky

La fattorizzazione $A = L\ U$ vista può essere scritta nella forma:

$$A = L\ D\ R$$

dove sia L che R sono triangolari, rispettivamente inferiore e superiore, con diagonali unitarie e D è diagonale con $d_{i,i} = u_{i,i}\ i = 1, \ldots, n$. Quando A è simmetrica abbiamo $R = L^T$; inoltre gli elementi $d_{i,i}$ sono tutti positivi se e solo se A è definita positiva[1]. Queste considerazioni portano al seguente risultato:

Teorema 9.3 *Se A è simmetrica e definita positiva, esiste un'unica matrice triangolare superiore S con elementi positivi sulla diagonale, tale che*

$$A = S^T\ S. \tag{9.1}$$

Infatti è sufficiente porre $S = L\ \tilde{D}$, dove con $\tilde{D}$ denominiamo la matrice diagonale che ha come elementi sulla diagonale $\tilde{d}_{i,i} = \frac{\sqrt{d_{i,i}}}{d_{i,i}}\ i = 1, \ldots, n$.

Gli elementi della matrice S possono essere determinati anche con le seguenti formule esplicite che si deducono dalla relazione (9.1):

$$s_{i,i} = \sqrt{a_{i,i}^{[i-1]}} \qquad s_{i,j} = \frac{a_{i,j}^{[i-1]}}{\sqrt{a_{i,i}^{[i-1]}}}, \quad j > i$$

dove

$$a_{i,j}^{[k]} = a_{i,j}^{[k-1]} - s_{k,i}s_{k,j} \quad i,j = k+1, \ldots, n \quad k = 1, \ldots, n-1$$

e con inizialmente $A^{[0]} = A$.

Questa fattorizzazione è nota come fattorizzazione di Cholesky e può essere determinata con una complessità di $n^3/6$ operazioni, cioè la metà di quelle necessarie per una generica fattorizzazione $L\ U$.

[1] Una matrice A di ordine n si dice simmetrica e definita positiva, se $A = A^T$ e $x^T Ax > 0\ \forall x \neq 0\ ,\ x \in \mathbb{R}^n$

9.2 Fattorizzazione QR

Data una matrice A di dimensione $n \times n$, esiste sempre una fattorizzazione Q R con Q matrice ortogonale ed R matrice triangolare superiore:

$$A_{n\times n} = Q_{n\times n} \cdot R_{n\times n}$$

con

$$\begin{array}{l} Q^T\ Q\ =\ I \\ \text{od anche} \\ Q^T\ =\ Q^{-1} \end{array} \qquad R = \begin{pmatrix} r_{1,1} & \dots & & & r_{1,n} \\ & r_{2,2} & & & \\ & & r_{3,3} & & \\ & & & \ddots & \vdots \\ 0 & & & & r_{n,n} \end{pmatrix}.$$

Osservazione 9.1 *Sia $Q_{n\times n}$ ortogonale ed $\mathbf{a}$ un vettore $n \times 1$, allora $\|Q\mathbf{a}\|_2 = \|\mathbf{a}\|_2$, infatti*

$$\|Q\mathbf{a}\|_2^2 = (Q\mathbf{a})^T(Q\mathbf{a}) = (\mathbf{a}^T Q^T)(Q\mathbf{a}) =$$
$$= \mathbf{a}^T(Q^T Q)\mathbf{a} = \mathbf{a}^T I\mathbf{a} = \mathbf{a}^T\mathbf{a} = \|\mathbf{a}\|_2^2.$$

L'esistenza di una tale fattorizzazione rende facile la determinazione della soluzione del sistema normale non singolare $A\mathbf{x} = \mathbf{b}$, infatti si avrebbe:

$$QR\mathbf{x} = \mathbf{b}$$

e ponendo $R\mathbf{x} = \mathbf{y}$ si potrebbe risolvere il sistema

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

sfruttando la definizione di Q ortogonale cioè $\mathbf{y} = Q^T\mathbf{b}$; determinato $\mathbf{y}$ si risolve il sistema

$$R\mathbf{x} = \mathbf{y}$$

per sostituzione all'indietro (backward-substitution).

9.2.1 Matrici elementari di Householder

Per matrice elementare di Householder si intende una matrice H tale che

$$H\ \mathbf{a} = \pm\|\mathbf{a}\|_2\ \mathbf{e}_1 = \begin{pmatrix} \pm\|\mathbf{a}\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Limitandoci al caso reale, cioè a vettori ad elementi reali si può dire che le matrici elementari di Householder sono simmetriche ($H = H^T$) ed ortogonali ($H^T H = I$ cioè $H^{-1} = H^T$).

Assegnato $\mathbf{a}$, l'opportuna trasformazione di Householder si può costruire così; si definisce vettore di Householder $\mathbf{v}$ come

$$\mathbf{v} = \begin{pmatrix} a_1 \pm \|\mathbf{a}\|_2 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

quindi

$$H = I - \beta \mathbf{v}\mathbf{v}^T \quad \text{con} \quad \beta = \frac{2}{\|\mathbf{v}\|_2^2}.$$

Si può dimostrare che H è ortogonale [2], mentre la simmetria è ovvia dalla forma della H. Inoltre se applicata al vettore $\mathbf{a}$ si ha:

$$\begin{aligned} H\ \mathbf{a} &= (I - \beta \mathbf{v}\mathbf{v}^T)\mathbf{a} \\ &= \mathbf{a} - \beta \mathbf{v}\mathbf{v}^T\mathbf{a} \\ &= \mathbf{a} - \mathbf{v} \qquad \text{infatti} \qquad \beta \mathbf{v}^T\mathbf{a} = 1 \\ &= \mp \|\mathbf{a}\|_2 \mathbf{e}_1. \end{aligned}$$

Verifichiamo $\beta \mathbf{v}^T\mathbf{a} = 1$:

$$\begin{aligned} \mathbf{v}^T\mathbf{a} &= (a_1 \pm \|\mathbf{a}\|_2, a_2, \ldots, a_n) \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \\ &= a_1^2 \pm a_1\|\mathbf{a}\|_2 + a_2^2 + \ldots + a_n^2 \\ &= \|\mathbf{a}\|_2^2 \pm a_1\|\mathbf{a}\|_2 \\ &= \|\mathbf{a}\|_2(\|\mathbf{a}\|_2 \pm a_1). \end{aligned}$$

inoltre $\beta = \dfrac{2}{\|\mathbf{v}\|_2^2}$ e

$$\begin{aligned} \|\mathbf{v}\|_2^2 &= (a_1 \pm \|\mathbf{a}\|_2)^2 + a_2^2 + \ldots + a_n^2) \\ &= a_1^2 \pm 2a_1\|\mathbf{a}\|_2 + \|\mathbf{a}\|_2^2 + a_2^2 + \ldots + a_n^2 \\ &= 2\|\mathbf{a}\|_2^2 \pm 2a_1\|\mathbf{a}\|_2 \\ &= 2\|\mathbf{a}\|_2(\|\mathbf{a}\|_2 \pm a_1). \end{aligned}$$

e quindi

$$\beta \mathbf{v}^T\mathbf{a} = 1.$$

[2] $H^T H = (I - \beta\mathbf{v}\mathbf{v}^T)(I - \beta\mathbf{v}\mathbf{v}^T) = I - 2\beta\mathbf{v}\mathbf{v}^T + \beta^2\mathbf{v}\mathbf{v}^T\mathbf{v}\mathbf{v}^T = I - 2\beta\mathbf{v}\mathbf{v}^T + \beta\frac{2}{\|\mathbf{v}\|_2^2}\mathbf{v}\|\mathbf{v}\|_2^2\mathbf{v}^T = I$ *c.v.d.*

Esempio 9.5 *Dato il vettore* $\mathbf{a} = (1,1,1)^T$*, si determini la matrice di Householder che azzera la seconda e terza componente.*

$$\mathbf{v} = \begin{pmatrix} a_1 - \|\mathbf{a}\|_2 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1-\sqrt{3} \\ 1 \\ 1 \end{pmatrix} \qquad \beta = \frac{1}{\|\mathbf{a}\|_2(\|\mathbf{a}\|_2 - a_1)} = \frac{1}{\sqrt{3}(\sqrt{3}-1)}.$$

$$\begin{aligned} H = I - \beta\mathbf{v}\mathbf{v}^T &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{\sqrt{3}(\sqrt{3}-1)} \begin{pmatrix} 1-\sqrt{3} \\ 1 \\ 1 \end{pmatrix} (1-\sqrt{3},1,1) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{\sqrt{3}(\sqrt{3}-1)} \begin{pmatrix} (1-\sqrt{3})^2 & (1-\sqrt{3}) & (1-\sqrt{3}) \\ (1-\sqrt{3}) & 1 & 1 \\ (1-\sqrt{3}) & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{\sqrt{3}-1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}(\sqrt{3}-1)} & \frac{1}{\sqrt{3}(\sqrt{3}-1)} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}(\sqrt{3}-1)} & \frac{1}{\sqrt{3}(\sqrt{3}-1)} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1-\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \\ -\frac{\sqrt{3}}{3} & \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{2}+\frac{\sqrt{3}}{6} \\ -\frac{\sqrt{3}}{3} & \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{2}+\frac{\sqrt{3}}{6} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} & \frac{1}{2}-\frac{\sqrt{3}}{6} & -\frac{1}{2}-\frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{3} & -\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{2}-\frac{\sqrt{3}}{6} \end{pmatrix} \end{aligned}$$

Ovviamente $H\mathbf{a} = (\sqrt{3},0,0)^T$.

9.2.2 Metodo di Householder

Procediamo ad illustrare come funziona l'algoritmo per fattorizzare $Q\ R$ una matrice $A_{n\times n} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ (il procedimento in oggetto può essere applicato anche a matrici rettangolari $A_{m\times n}$ con $m \geq n$).

Si pone $A_1 := A$ e si costruisce una matrice elementare di Householder H_1 tale che

$$H_1\ A_1 = \begin{pmatrix} \|\mathbf{a}_1^{(1)}\| & \times & \ldots & \times \\ 0 & \vdots & & \vdots \\ \vdots & & & \\ 0 & \times & \ldots & \times \end{pmatrix} = A_2$$

quindi una H_2 tale che

$$H_2\ A_2 = \begin{pmatrix} \|\mathbf{a}_1^{(1)}\| & \times & \times & \dots & \times \\ 0 & \|\mathbf{a}_2^{(2)}\| & \times & \dots & \times \\ \vdots & 0 & \vdots & & \\ & \vdots & & & \\ 0 & 0 & \times & \dots & \times \end{pmatrix} = A_3$$

e così via fino ad una H_{n-1} tale che

$$H_{n-1}\ A_{n-1} = \begin{pmatrix} \|\mathbf{a}_1^{(1)}\| & \times & \times & \dots & \times \\ 0 & \|\mathbf{a}_2^{(2)}\| & \times & \dots & \times \\ \vdots & 0 & \ddots & & \vdots \\ & \vdots & & \ddots & \times \\ 0 & 0 & \dots & & \|\mathbf{a}_n^{(n-1)}\| \end{pmatrix} = R$$

così che

$$H_{n-1}\ H_{n-2}\ \dots H_1\ A = R.$$

Ma le H_k sono tutte ortogonali ed il prodotto di matrici ortogonali è ancora una matrice ortogonale, da cui

$$A = H_1^T\ H_2^T\ \dots H_{n-1}^T\ R = Q\ R.$$

Vediamo alcuni dettagli sulla costruzione delle matrici H_k:
al primo passo, sia $\mathbf{a}_1^{(1)}$ il vettore formato dagli elementi della prima colonna di $A_1 = A$ e sia

$$\theta_1 = \begin{cases} +1 & \text{se} \quad a_{1,1}^{(1)} \geq 0 \\ -1 & \text{se} \quad a_{1,1}^{(1)} < 0 \end{cases}$$

posto

$$\beta_1 = \frac{1}{\|\mathbf{a}_1^{(1)}\|_2(\|\mathbf{a}_1^{(1)}\|_2 + |a_{1,1}^{(1)}|)}$$

e

$$\mathbf{v}_1 = \begin{pmatrix} \theta_1(\|\mathbf{a}_1^{(1)}\|_2 + |a_{1,1}^{(1)}|) \\ a_{2,1}^{(1)} \\ a_{3,1}^{(1)} \\ \vdots \\ a_{n,1}^{(1)} \end{pmatrix}$$

allora la prima matrice elementare di Householder è data da

$$H_1 = I - \beta_1 \mathbf{v}_1 \mathbf{v}_1^T;$$

al $k-$esimo passo, sia $\mathbf{a}_k^{(k)}$ il vettore di ordine $n-k+1$ formato dagli elementi della $k-$esima colonna di A_k con indice di riga maggiore e uguale a k, e sia

$$\theta_k = \begin{cases} +1 & \text{se} \quad a_{k,k}^{(k)} \geq 0 \\ -1 & \text{se} \quad a_{k,k}^{(k)} < 0 \end{cases}$$

posto

$$\beta_k = \frac{1}{\|\mathbf{a}_k^{(k)}\|_2(\|\mathbf{a}_k^{(k)}\|_2 + |a_{k,k}^{(k)}|)}$$

e

$$\mathbf{v}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \theta_k(\|\mathbf{a}_k^{(k)}\|_2 + |a_{k,k}^{(k)}|) \\ a_{k+1,k}^{(k)} \\ \vdots \\ a_{n,k}^{(k)} \end{pmatrix}$$

la $k-$esima matrice elementare di Householder è data da

$$H_k = I - \beta_k \mathbf{v}_k \mathbf{v}_k^T.$$

Osservazione 9.2 *Se al $k-$esimo passo si ha $\mathbf{a}_k^{(k)} = \mathbf{0}$, si pone $H_k = I$, cioè il $k-$esimo passo non comporta alcuna operazione.*

Esempio 9.6 *Si calcoli la fattorizzazione $Q\ R$ della matrice*

$$A_1 = A = \begin{pmatrix} 72 & -144 & -144 \\ -144 & -36 & -360 \\ -144 & -360 & 450 \end{pmatrix}.$$

Al primo passo si ha

$$\beta_1 = \frac{1}{62208}, \qquad \mathbf{v}_1 = \begin{pmatrix} 288 \\ -144 \\ -144 \end{pmatrix}$$

per cui

$$H_1 = I - \beta_1 \mathbf{v}_1 \mathbf{v}_1^T = \frac{1}{6} \begin{pmatrix} -2 & 4 & 4 \\ 4 & 4 & -2 \\ 4 & -2 & 4 \end{pmatrix}$$

e

$$A_2 = \begin{pmatrix} -216 & -216 & 108 \\ 0 & 0 & 486 \\ 0 & -324 & 324 \end{pmatrix}.$$

Al secondo passo si ha

$$\beta_2 = \frac{1}{104976}, \qquad \mathbf{v}_2 = \begin{pmatrix} 0 \\ 324 \\ -324 \end{pmatrix}$$

per cui

$$H_2 = I - \beta_2 \mathbf{v}_2 \mathbf{v}_2^T == \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

e

$$R = A_3 = \begin{pmatrix} -216 & -216 & 108 \\ 0 & -324 & 324 \\ 0 & 0 & -486 \end{pmatrix}.$$

Inoltre

$$Q = H_1\ H_2 = \frac{1}{6} \begin{pmatrix} -2 & 4 & 4 \\ 4 & -2 & 4 \\ 4 & 4 & -2 \end{pmatrix}.$$

9.2.3 Implementazione del metodo di Householder

Il metodo di Householder, per risolvere il sistema lineare $A\mathbf{x} = \mathbf{b}$, può essere implementato senza calcolare effettivamente le matrici H_k. Si procede nel seguente modo: si considera la matrice

$$T_1 = [A_1|\mathbf{b}_1] = [A|\mathbf{b}]$$

e si costruiscono β_1, $\mathbf{v}_1$ ed il vettore di $n+1$ componenti

$$\mathbf{y}_1^T = \mathbf{v}_1^T T_1 = (\theta_1(\|\mathbf{a}_1^{(1)}\|_2 + |a_{1,1}^{(1)}|), a_{2,1}^{(1)}, \ldots, a_{n,1}^{(1)}) \left(\begin{array}{cccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \ldots & a_{1,n}^{(1)} & b_1^{(1)} \\ a_{2,1}^{(1)} & & \ldots & & \vdots \\ \vdots & & \ldots & & \vdots \\ a_{n,1}^{(1)} & \ldots & & a_{n,n}^{(1)} & b_n^{(1)} \end{array}\right).$$

Allora è

$$\begin{aligned} T_2 &= H_1\ T_1 \\ &= (I - \beta_1 \mathbf{v}_1 \mathbf{v}_1^T)\ T_1 \\ &= T_1 - \beta_1 \mathbf{v}_1 \mathbf{y}_1^T \end{aligned}$$

Al $k-$esimo passo si costruisce

$$\mathbf{y}_k^T = \mathbf{v}_k^T T_k$$

e

$$T_{k+1} = T_k - \beta_k \mathbf{v}_k \mathbf{y}_k^T.$$

Dopo $n-1$ passi si ottiene la matrice

$$T_n = [A_n|\mathbf{b}_n] = [R|\mathbf{b}_n]$$

e quindi il sistema $R\mathbf{x} = \mathbf{b}_n$ con matrice dei coefficienti triangolare superiore equivalente al sistema $A\mathbf{x} = \mathbf{b}$.

9.2.4 Costo Computazionale per risolvere $A\mathbf{x} = \mathbf{b}$ tramite Q R

Si esamina il costo del metodo di soluzione proposto che non comporta la determinazione esplicita delle metrici H_k e della matrice ortogonale Q. Al $k-$esimo passo, per determinare $\mathbf{v}_k$ e β_k, assunte le prime $k-1$ componenti di $\mathbf{v}_k$ nulle e quindi con al più $n-k+1$ componenti non nulle, sarà:

$$\mathbf{v}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \theta_k(\|\mathbf{a}_k^{(k)}\|_2 + |a_{k,k}^{(k)}|) \\ a_{k+1,k}^{(k)} \\ \vdots \\ a_{n,k}^{(k)} \end{pmatrix} \qquad \beta_k = \frac{2}{\|\mathbf{v}_k\|_2^2}$$

e servono: $n-k+1$ moltiplicazioni ed una radice quadrata per calcolare $\|\mathbf{a}_k^{(k)}\|_2$, una moltiplicazione per $\|\mathbf{v}_k\|_2^2$ ed una divisione per β_k per un totale di $n-k+3$ molt./div.
Per $\mathbf{y}_k$ servono $(n-k+1)^2 + n - k + 1$ moltiplicazioni, mentre per $\beta_k \mathbf{v}_k \mathbf{y}_k^T$ ne servono $(n-k+1)^2 + n - k + 1$. In totale

$$n - k + 3 + 2(n-k+1)^2 + 2(n-k+1) =$$

$$2(n-k)^2 + 7(n-k) + 7.$$

Poiché i passi sono $k = 1, \ldots, (n-1)$ si ha

$$\frac{2(n-1)n(2n-1)}{6} + 7\frac{(n-1)n}{2} + 7(n-1) =$$

$$\frac{2}{3}(n-1)n(n+5.5) + 8(n-1) \approx \frac{2}{3}n^3.$$

9.2.5 Stabilità della Fattorizzazione $Q\ R$

Procedendo con l'analisi all'indietro, esattamente come già fatto nel caso della fattorizzazione $L\ U$ si ha che

$$\delta A \sim R \cdot \delta Q + Q \cdot \delta R$$

da cui si cerca una limitazione superiore per gli elementi delle matrici Q ed R. Si trova che, essendo Q unitaria, vale

$$\max_{i,j} |q_{i,j}| \leq \|Q\|_2 = 1;$$

inoltre si ha che

$$\max_{i,j} |r_{i,j}| \leq \sqrt{n}\ \max_{i,j} |a_{i,j}|.$$

Da cui risulta che l'algoritmo di fattorizzazione con matrici elementari di Householder ($Q\ R$) è stabile in senso debole con estremo $\sqrt{n}$, che risulta comunque più stabile dell'algoritmo $L\ U$ essendo $\sqrt{n} \ll 2^{n-1}$.

Capitolo 10

Sistemi lineari: metodi iterativi

Per risolvere un sistema lineare $A\mathbf{x} = \mathbf{b}$, oltre ai metodi diretti, si possono utilizzare anche i metodi iterativi, che risultano particolarmente convenienti se la matrice è sparsa, cioè se il numero degli elementi non nulli di A è dell'ordine della dimensione della matrice. Infatti quando si usa un metodo diretto può accadere che nelle matrici intermedie vengano generati molti elementi diversi da zero in corrispondenza di elementi nulli della matrice iniziale. Poiché i metodi diretti non sfruttano la sparsità della matrice, soprattutto poi se A è anche di grandi dimensioni, può essere conveniente utilizzare un metodo iterativo. Esistono comunque dei casi nei quali la matrice A è sparsa, ma è più conveniente usare dei metodi diretti che sfruttano specifiche proprietà di struttura della matrice. Cominciamo con alcuni richiami su autovalori e autovettori e su successioni convergenti di vettori e matrici.

10.1 Richiami su Autovalori e Autovettori

Definizione 10.1 *data una matrice $A \in \mathbb{R}^{n\times n}$, viene definito autovalore di A un numero $\lambda \in \mathbb{C}$ per cui valga la relazione*

$$A\,\mathbf{x} = \lambda\mathbf{x} \qquad con \quad \mathbf{x} \neq \mathbf{0}. \tag{10.1}$$

$\mathbf{x}$ è detto autovettore corrispondente a λ.

L'insieme degli autovalori di una matrice A costituisce lo **spettro** di A e l'autovalore massimo in modulo è detto **raggio spettrale** di A ed è indicato con $\rho(A)$.

La relazione (10.1) data può essere vista come un sistema lineare omogeneo

$$(A - \lambda I)\,\mathbf{x} = \mathbf{0}$$

che ammette soluzioni non nulle se e solo se

$$det(A - \lambda I) = 0. \tag{10.2}$$

Sviluppando la (10.2) risulta

$$det(A - \lambda I) = p(\lambda) = a_0 + a_1\lambda + \ldots + a_n\lambda^n$$

in cui

$$a_n = (-1)^n \qquad a_0 = det(A) \qquad a_{n-1} = (-1)^{n-1}tr(A)$$

dove con $tr(A)$ si indica la somma degli elementi diagonali di A detta traccia di A.
Dalle relazioni che legano i coefficienti e le radici di una equazione algebrica risulta che

$$\sum_{i=1}^{n} \lambda_i = tr(A) \qquad \text{e} \qquad \prod_{i=1}^{n} \lambda_i = det(A).$$

Il polinomio $p(\lambda)$ è detto **polinomio caratteristico** di A e l'equazione $p(\lambda) = 0$ è detta **equazione caratteristica** di A.
Per il teorema fondamentale dell'algebra l'equazione caratteristica ha, in campo complesso, n radici. Quindi una matrice di ordine n ha n autovalori nel campo complesso.
Poiché gli autovalori sono soluzioni non nulle del sistema lineare omogeneo visto, un autovettore corrispondente ad un autovalore λ risulta determinato a meno di una costante moltiplicativa $\alpha \neq 0$, cioè se $\mathbf{x}$ è un autovettore di A, anche $\alpha\mathbf{x}$ è un autovettore di A, corrispondente allo stesso autovalore.

Esempio 10.1 *Il polinomio caratteristico della matrice*

$$A = \begin{pmatrix} 1 & 3 \\ 3 & 1 \end{pmatrix}$$

si ricava dal determinante

$$det(A - \lambda I) = det \begin{pmatrix} 1-\lambda & 3 \\ 3 & 1-\lambda \end{pmatrix} = (1-\lambda)^2 - 9 = \lambda^2 - 2\lambda - 8.$$

L'equazione caratteristica corrispondente è:

$$\lambda^2 - 2\lambda - 8 = 0$$

ed ha come radici $\lambda_1 = -2$ e $\lambda_2 = 4$ che sono gli autovalori della matrice A. L'autovettore corrispondente a $\lambda_1 = -2$ si calcola risolvendo il sistema

$$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathbf{0};$$

dalla prima equazione si ottiene

$$x_1 + x_2 = 0$$

da cui $x_1 = -x_2$ *e qualunque vettore*

$$\mathbf{x} = \alpha \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

con $\alpha \neq 0$*, è autovettore corrispondente all'autovalore* $\lambda_1 = -2$*.*
L'autovettore corrispondente a $\lambda_2 = 4$ *si determina risolvendo il sistema*

$$\begin{pmatrix} -3 & 3 \\ 3 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathbf{0};$$

dalla prima equazione si ottiene

$$-x_1 + x_2 = 0$$

da cui $x_1 = x_2$ *e qualunque vettore*

$$\mathbf{x} = \alpha \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

con $\alpha \neq 0$*, è autovettore corrispondente all'autovalore* $\lambda_2 = 4$*.*

Osservazione 10.1 *Dato un polinomio si può costruire una matrice i cui autovalori sono gli zeri del polinomio; tale matrice si dice* **companion** *o matrice di* **Frobenius**. *Dato*

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

la matrice in questione è così definita:

$$F = \begin{pmatrix} 0 & \dots & & & 0 & -\frac{a_0}{a_n} \\ 1 & \ddots & & & \vdots & -\frac{a_1}{a_n} \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \\ & & & \ddots & 0 & \\ 0 & \dots & & 0 & 1 & -\frac{a_{n-1}}{a_n} \end{pmatrix}$$

Esempio 10.2 *Sia* $p(\lambda) = \lambda^2 - 2\lambda - 8$*, allora*

$$F = \begin{pmatrix} 0 & 8 \\ 1 & 2 \end{pmatrix}$$

verifichiamolo:

$$det(F - \lambda I) = det \begin{pmatrix} -\lambda & -8 \\ -1 & 2 - \lambda \end{pmatrix} = \lambda(\lambda - 2) - 8 = \lambda^2 - 2\lambda - 8.$$

10.1.1 Proprietà degli Autovalori

- Gli autovalori di una matrice diagonale o triangolare sono uguali agli elementi diagonali.

- Se λ è un autovalore di una matrice A non singolare e $\mathbf{x}$ un autovettore corrispondente, allora risulta $\lambda \neq 0$ e $1/\lambda$ è un autovalore di A^{-1} con $\mathbf{x}$ autovettore corrispondente. Infatti da $A\mathbf{x} = \lambda\mathbf{x}$ si ha $\mathbf{x} = \lambda A^{-1}\mathbf{x}$ e quindi $\lambda \neq 0$ e $A^{-1}\mathbf{x} = \frac{1}{\lambda}\mathbf{x}$.

- Se λ è autovalore di una matrice ortogonale A, cioè $A^TA = AA^T = I$, allora risulta $|\lambda| = 1$. Infatti dalla relazione $A\mathbf{x} = \lambda\mathbf{x}$ si ha $(A\mathbf{x})^T = (\lambda\mathbf{x})^T$ e quindi $\mathbf{x}^TA^T = \lambda\mathbf{x}^T$ da cui si ha

  $$\mathbf{x}^TA^TA\mathbf{x} = \lambda\mathbf{x}^T\lambda\mathbf{x}$$

  e poiché A è ortogonale risulta

  $$\mathbf{x}^T\mathbf{x} = \lambda^2\mathbf{x}^T\mathbf{x} \qquad \text{e} \quad |\lambda| = 1.$$

10.1.2 Proprietà degli Autovettori

Teorema 10.1 *Autovettori corrispondenti ad autovalori distinti sono linearmente indipendenti.*

Osservazione 10.2 *Dal precedente teorema risulta che se una matrice A di ordine n ha n autovalori tutti distinti, allora A ha n autovettori linearmente indipendenti. Segue che se A non ha autovalori distinti, A può avere, ma anche non avere, n autovettori linearmente indipendenti.*

Definizione 10.2 *La molteplicità di un autovalore λ come radice dell'equazione caratteristica, è indicata con $\sigma(\lambda)$, ed è detta* **molteplicità algebrica** *di λ. Il massimo numero di autovettori linearmente indipendenti corrispondenti a λ è indicato con $\tau(\lambda)$ ed è detta* **molteplicità geometrica** *di λ.*

Osservazione 10.3 *È evidente che*

$$1 \leq \sigma(\lambda) \leq n \qquad e \qquad 1 \leq \tau(\lambda) \leq n.$$

Teorema 10.2 *Vale la seguente disuguaglianza*

$$\tau(\lambda) \leq \sigma(\lambda).$$

10.2 Richiami su Successioni di Vettori e Convergenza

Definizione 10.3 *Una successione* $\{\mathbf{x}^{(k)}\}$ *di vettori di* $\mathbb{R}^n$ *si dice convergente al vettore* $\mathbf{x}^* \in \mathbb{R}^n$ *se esiste una norma vettoriale per cui*

$$\lim_{k\to\infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = \mathbf{0};$$

in tal caso si pone

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*.$$

Per il teorema di equivalenza delle norme, la definizione appena data non dipende da una particolare norma. La condizione di convergenza data si traduce in una condizione di convergenza delle successioni formate dalle singole componenti. Infatti, considerando la norma ∞ o del max

$$|x_i^{(k)} - x_i^*| \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_\infty \qquad i = 1, \ldots, n$$

e quindi

$$\lim_{k\to\infty} |x_i^{(k)} - x_i^*| = 0$$

da cui

$$\lim_{k\to\infty} x_i^{(k)} = x_i^*.$$

Viceversa se vale quest'ultima è ovviamente verificata la condizione di convergenza data nella definizione, per la norma ∞.

Per le successioni di matrici $\{A^{(k)}\}$ si può dare una definizione di convergenza analoga a quella data per vettori. Il seguente teorema è di fondamentale importanza nello studio dei metodi iterativi per la soluzione di sistemi lineari.

Teorema 10.3 *Sia* $A \in \mathbb{R}^{n\times n}$*, allora*

$$\lim_{k\to\infty} A^k = 0 \quad \textit{se e solo se} \quad \rho(A) < 1$$

con $\rho(A)$ *il raggio spettrale della matrice* A*.*

Teorema 10.4 *Sia* $A \in \mathbb{R}^{n\times n}$*, allora*

$$det(I - A) \neq 0$$

e

$$\lim_{k\to\infty} \sum_{i=0}^{k} A^i = (I - A)^{-1} \quad \textit{se e solo se} \quad \rho(P) < 1.$$

Osservazione 10.4 *Come per le serie numeriche, si usa scrivere*

$$\sum_{i=0}^{\infty} A^i = (I - A)^{-1}.$$

10.3 Decomposizione della matrice

Sia $A \in \mathbb{R}^{n\times n}$ una matrice non singolare e si consideri la decomposizione di A nella forma

$$A = M \ - \ N$$

dove M è una matrice non singolare. Sostituendo tale decomposizione nel sistema $A\mathbf{x} = \mathbf{b}$ si ha

$$(M \ - \ N)\mathbf{x} = \mathbf{b}$$

$$M\mathbf{x} \ - \ N\mathbf{x} = \mathbf{b}$$

ed essendo M non singolare

$$\mathbf{x} = M^{-1} \ N\mathbf{x} \ + \ M^{-1}\mathbf{b}.$$

Posto $P = M^{-1} \ N$ e $\mathbf{q} = M^{-1}\mathbf{b}$ si ottiene il seguente sistema

$$\mathbf{x} = P\mathbf{x} + \mathbf{q}$$

equivalente ad $A\mathbf{x} = \mathbf{b}$.
Dato un vettore iniziale $\mathbf{x}^{(0)}$, si considera la successione $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ così definita

$$\mathbf{x}^{(k)} = P\mathbf{x}^{(k-1)} + \mathbf{q} \quad k = 1, 2, \ldots \tag{10.3}$$

Se la successione $\{\mathbf{x}^{(k)}\}$ è convergente, cioè

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

allora passando al limite nella (10.3) risulta

$$\mathbf{x}^* = P\mathbf{x}^* + \mathbf{q} \tag{10.4}$$

cioè $\mathbf{x}^*$ è la soluzione del sistema $\mathbf{x} = P\mathbf{x} + \mathbf{q}$ e quindi del sistema $A\mathbf{x} = \mathbf{b}$.
La relazione (10.3) individua un metodo iterativo in cui, partendo da un vettore iniziale $\mathbf{x}^{(0)}$, la soluzione viene approssimata utilizzando una successione $\{\mathbf{x}^{(k)}\}$ di vettori. La matrice P si dice **matrice di iterazione** del metodo.

Osservazione 10.5 *Al variare del vettore iniziale $\mathbf{x}^{(0)}$ si ottengono dalla (10.3) diverse successioni $\{\mathbf{x}^{(k)}\}$, alcune delle quali possono essere convergenti ed altre no. Un metodo iterativo è detto convergente se, qualunque sia il vettore iniziale $\mathbf{x}^{(0)}$, la successione $\{\mathbf{x}^{(k)}\}$ è convergente.*

Teorema 10.5 *Il metodo iterativo (10.3) risulta convergente se e solo se $\rho(A) < 1$.*

dim. Sottraendo dalla (10.4) la (10.3) si ha

$$\mathbf{x}^* - \mathbf{x}^{(k)} = P(\mathbf{x}^* - \mathbf{x}^{(k-1)}) \quad k = 1, 2, \ldots$$

Indicato con

$$\mathbf{e}^{(k)} = \mathbf{x}^* - \mathbf{x}^{(k)}$$

il vettore **errore** alla $k-$esima iterazione, si ha

$$\mathbf{e}^{(k)} = P\mathbf{e}^{(k-1)} \quad k = 1, 2, \ldots$$

e quindi

$$\mathbf{e}^{(k)} = P\mathbf{e}^{(k-1)} = P^2\mathbf{e}^{(k-2)} = \ldots = P^k\mathbf{e}^{(0)}.$$

(Condizione sufficiente) Se $\rho(P) < 1$, per il teorema visto, risulta

$$\lim_{k\to\infty} P^k = 0$$

e segue che per ogni vettore $\mathbf{e}^{(0)}$ si ha

$$\lim_{k\to\infty} \mathbf{e}^{(k)} = \mathbf{0}.$$

(Condizione necessaria) Se il metodo è convergente il $\lim_{k\to\infty} \mathbf{e}^{(k)} = \mathbf{0}$ vale per ogni $\mathbf{x}^{(0)}$, e in particolare deve valere se $\mathbf{x}^{(0)}$ è tale che il vettote $\mathbf{e}^{(0)} = \mathbf{x}^* - \mathbf{x}^{(0)}$ è un autovettore di P corrispondente ad un autovalore λ di modulo massimo, cioè $|\lambda| = \rho(P)$.
In questo caso risulta

$$P\mathbf{e}^{(0)} = \lambda\mathbf{e}^{(0)}$$

e quindi

$$\mathbf{e}^{(k)} = P^k\mathbf{e}^{(0)} = \lambda^k\mathbf{e}^{(0)}.$$

Segue che

$$\lim_{k\to\infty} [\rho(P)]^k = 0$$

e quindi $\rho(P) < 1$.

La condizione $\rho(P) < 1$, necessaria e sufficiente per la convergenza della (10.3), non è in generale di agevole verifica. Conviene utilizzare, quando è possibile, delle condizioni sufficienti di convergenza di più facile verifica. Una tale condizione è data dal seguente teorema.

Teorema 10.6 *Se esiste una norma matriciale indotta $\|\cdot\|$, per cui $\|P\| < 1$, il metodo iterativo (10.3) è convergente.*

dim. La tesi segue dal teorema di convergenza visto e dalla proprietà che

$$\rho(P) < \|P\|.$$

Osservazione 10.6 *Il raggio spettrale $\rho(P)$ di una matrice viene a volte anche definito come l'estremo inferiore di tutte le norme di P.*

Osservazione 10.7 *Poiché il determinante di una matrice è uguale al prodotto degli autovalori, se $|det(P)| \geq 1$, almeno uno degli autovalori di P è in modulo maggiore o uguale ad 1 e quindi il metodo (10.3) non è convergente.*
Poiché la traccia di una matrice è uguale alla somma degli autovalori, se $|tr(P)| > n$, almeno uno degli autovalori di P è in modulo maggiore o uguale ad 1 e il metodo (10.3) non è convergente.
Quindi le condizioni:

$$|det(P)| < 1 \qquad e \qquad |tr(P)| < n$$

sono condizioni necessarie affinché il metodo (10.3) sia convergente.

10.4 Controllo della Convergenza

Se $\mathbf{e}^{(k-1)} \neq 0$, la quantità

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(k-1)}\|}$$

esprime la **riduzione dell'errore al $k-$esimo passo**.
la media geometrica delle riduzioni dell'errore sui primi k passi

$$\sigma_k = \left(\frac{\|\mathbf{e}^{(1)}\|}{\|\mathbf{e}^{(0)}\|} \cdot \frac{\|\mathbf{e}^{(2)}\|}{\|\mathbf{e}^{(1)}\|} \cdots \frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(k-1)}\|}\right)^{\frac{1}{k}} = \left(\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|}\right)^{\frac{1}{k}}$$

esprime la **riduzione media per passo** dell'errore relativo ai primi k passi. Dalla

$$\|\mathbf{e}^{(k)}\| \leq \|P^k\| \cdot \|\mathbf{e}^{(0)}\|$$

risulta

$$\sigma_k \leq (\|P^k\|)^{\frac{1}{k}}.$$

La quantità che si ottiene facendo tendere k all'infinito esprime la **riduzione asintotica media per passo** e, come risulta dal seguente teorema, è indipendente dalla particolare norma utilizzata.

Teorema 10.7 *Sia $A \in \mathbb{R}^{n\times n}$ e sia $\|\cdot\|$ una qualunque norma matriciale indotta. Allora*

$$\lim_{k\to\infty} (\|P^k\|)^{\frac{1}{k}} = \rho(P).$$

Osservazione 10.8 *La quantità $\rho(P)$, indipendente dalla norma utilizzata e dall'indice di iterazione k, viene quindi assunta come* **misura della velocità di convergenza** *del metodo (10.3).*

10.5 Test di Arresto

Poiché con un metodo iterativo non è possibile calcolare la soluzione con un numero finito di iterazioni, occorre individuare dei criteri per l'arresto del procedimento.
I criteri più comunemente usati, fissata una tolleranza Tol, che tiene conto anche della precisione utilizzata nei calcoli, sono i seguenti

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \leq Tol$$

oppure, se $\mathbf{x}^{(k)} \neq \mathbf{0}$

$$\frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|}{\|\mathbf{x}^{(k)}\|} \leq Tol.$$

Si noti che queste due condizioni non garantiscono che la soluzione sia stata approssimata con la precisione Tol. Infatti da

$$\mathbf{e}^{(k)} = P\mathbf{e}^{(k-1)}$$

è

$$\begin{aligned} \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} &= (\mathbf{x}^* - \mathbf{x}^{(k-1)}) - (\mathbf{x}^* - \mathbf{x}^{(k)}) \\ &= \mathbf{e}^{(k-1)} - \mathbf{e}^{(k)} \\ &= (I - P)\mathbf{e}^{(k-1)} \end{aligned}$$

e passando alle norme, se $\|P\| < 1$, per il teorema seguente si ha

$$\begin{aligned} \|\mathbf{e}^{(k-1)}\| \leq & \|(I-P)^{-1}\| \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \\ \leq & \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|}{1 - \|P\|}. \end{aligned}$$

Per cui può accadere che $\|\mathbf{e}^{(k-1)}\|$ sia elevata anche se le condizioni di arresto date sono verificate.

Teorema 10.8 *Sia $A \in \mathbb{R}^{n\times n}$ e sia $\|\cdot\|$ una qualunque norma matriciale indotta tale che $\|A\| < 1$. Allora la matrice $I \pm A$ è non singolare e vale la disuguaglianza*

$$\|(I \pm A)^{-1}\| \leq \frac{1}{1 - \|A\|}.$$

- In un programma che implementa un tale metodo iterativo deve essere comunque previsto un controllo per interrompere l'esecuzione quando il numero delle iterazioni diventa troppo elevato.

- Può anche accadere che un metodo iterativo la cui matrice di iterazione P è tale che $\rho(P) < 1$, per gli effetti indotti dagli errori di arrotondamento non converga in pratica. Questo accade, in particolare, quando la matrice A è fortemente mal condizionata e $\rho(P)$ è molto vicino ad 1.

- Si deve rilevare che un metodo iterativo, rispetto ad un metodo diretto è in generale meno sensibile alla propagazione degli errori. Infatti il vettore $\mathbf{x}^{(k)}$ può essere considerato come il vettore generato con una sola iterazione a partire dal vettore iniziale $\mathbf{x}^{(k-1)}$, e quindi risulta affetto dagli errori di arrotondamento generati dalla sola ultima iterazione.

- In un metodo iterativo, ad ogni iterazione, il costo computazionale è principalmente determinato dalle operazioni di moltiplicazione della matrice P per un vettore, che richiede n^2 operazioni moltiplicative. Se A è sparsa, il numero di moltiplicazioni è dell'ordine di n. In questo caso i metodi iterativi possono risultare competitivi con quelli diretti.

10.6 Metodi di Jacobi e Gauss-Seidel

Tra i metodi iterativi individuati da una particolare scelta della decomposizione $A = M - N$ sono particolarmente importanti il metodo di Jacobi e il metodo di Gauss-Seidel, per i quali è possibile dare delle condizioni sufficienti di convergenza. Si consideri la decomposizione della matrice A

$$A = D - L - U$$

con

$$D = \begin{pmatrix} a_{1,1} & & & \\ & a_{2,2} & & \\ & & \ddots & \\ & & & a_{n,n} \end{pmatrix},$$

$$L = \begin{pmatrix} 0 & \dots & & 0 \\ -a_{2,1} & \ddots & & \vdots \\ \vdots & \ddots & & \\ -a_{n,1} & \dots & -a_{n,n-1} & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & -a_{1,2} & \dots & -a_{1,n} \\ \vdots & \ddots & \ddots & \\ & & & -a_{n-1,n} \\ 0 & \dots & & 0 \end{pmatrix}.$$

Scegliendo

$$M = D \qquad N = L + U,$$

si ottiene il metodo di Jacobi, mentre scegliendo

$$M = D - L \qquad N = U,$$

si ottiene il metodo di Gauss-Seidel.
Per queste decomposizioni risulta $det(M) \neq 0$ se e solo se tutti gli elementi diagonali di A sono non nulli.

Indicando con J la matrice di iterazione del metodo di Jacobi, dalla $P = M^{-1}N$ si ha

$$J = D^{-1}(L + U)$$

per cui la (10.3) diviene

$$\mathbf{x}^{(k)} = J\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b} = D^{-1}[\mathbf{b} + (L + U)\mathbf{x}^{(k-1)}]$$

e in termini di componenti

$$x_i^{(k)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1 \; j \neq i}^{n} a_{i,j} x_j^{(k-1)} \right) \quad i = 1, \ldots, n.$$

Il metodo di Jacobi è detto anche metodo degli spostamenti simultanei, in quanto le componenti del vettore $\mathbf{x}^{(k)}$ sostituiscono simultaneamente, al termine dell'iterazione, le componenti di $\mathbf{x}^{(k-1)}$.

Indicando con G la matrice di iterazione di Gauss-Seidel, dalla $P = M^{-1}N$ si ha

$$G = (D - L)^{-1}U$$

per cui la (10.3) diviene

$$\mathbf{x}^{(k)} = G\mathbf{x}^{(k-1)} + (D - L)^{-1}\mathbf{b}.$$

Per descrivere questo metodo in termini di componenti, conviene prima trasformarla nel modo seguente:

$$(D - L)\mathbf{x}^{(k)} = U\mathbf{x}^{(k-1)} + \mathbf{b}$$

$$D\mathbf{x}^{(k)} = L\mathbf{x}^{(k)} + U\mathbf{x}^{(k-1)} + \mathbf{b}$$

$$\mathbf{x}^{(k)} = D^{-1}\left(L\mathbf{x}^{(k)} + U\mathbf{x}^{(k-1)} + \mathbf{b}\right)$$

ottenendo quindi

$$x_i^{(k)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k-1)} \right) \quad i = 1, \ldots, n.$$

Confrontando questa con quella di Jacobi, risulta che per calcolare le componenti del vettore $\mathbf{x}^{(k)}$ sono utilizzate componenti già calcolate dello stesso vettore. Per questo motivo il metodo di Gauss-Seidel è detto anche metodo degli spostamenti successivi.

Nell'implementare Jacobi è necessario disporre, contemporaneamente di entrambe i vettori $\mathbf{x}^{(k-1)}$ e $\mathbf{x}^{(k)}$, mentre per Gauss-Seidel è sufficiente disporre di un solo vettore.

Osservazione 10.9 *In molte applicazioni il metodo di Gauss-Seidel, che utilizza immediatamente i valori calcolati nella iterazione corrente, risulta più veloce del metodo di Jacobi. Però esistono casi in cui risulta non solo che il metodo di Jacobi sia più veloce di Gauss-Seidel, ma anche che Jacobi sia convergente e Gauss-seidel no.*

Per i metodi di Jacobi e Gauss-Seidel si possono ricavare delle condizioni sufficienti di convergenza di facile verifica sul sistema lineare.

Teorema 10.9 *Se la matrice A è a predominanza diagonale in senso stretto, allora i metodi di Jacobi e di Gauss-Seidel sono convergenti.*

Osservazione 10.10 *Una matrice $A \in \mathbb{R}^{n\times n}$ si dice a predominanza diagonale in senso stretto, se vale*

$$|a_{i,j}| > \sum_{j=1\ j\neq i}^{n} |a_{i,j}| \qquad i = 1,\dots,n \quad \textit{per righe}$$

$$|a_{i,j}| > \sum_{i=1\ i\neq j}^{n} |a_{i,j}| \qquad j = 1,\dots,n \quad \textit{per colonne}$$

Esempio 10.3 *Applichiamo il metodo di Jacobi al sistema lineare*

$$A\mathbf{x} = \mathbf{b} \qquad \textit{con} \qquad A = \begin{pmatrix} 20 & 2 & -1 \\ 2 & 13 & -2 \\ 1 & 1 & 1 \end{pmatrix} \qquad \textit{e} \qquad \mathbf{b} = \begin{pmatrix} 25 \\ 30 \\ 2 \end{pmatrix}.$$

La decomposizione di Jacobi è

$$M = \begin{pmatrix} 20 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad N = \begin{pmatrix} 0 & -2 & 1 \\ -2 & 0 & 2 \\ -1 & -1 & 0 \end{pmatrix}$$

La matrice di iterazione J ed il vettore $\mathbf{q}$ sono

$$J = \begin{pmatrix} 0 & -1/10 & 1/20 \\ -2/13 & 0 & 2/13 \\ -1 & -1 & 0 \end{pmatrix}, \quad \mathbf{q} = \begin{pmatrix} 5/4 \\ 30/13 \\ 2 \end{pmatrix}$$

Risulta $\rho(J) = 0.44896 < 1$, da cui il metodo converge e converge alla soluzione $\mathbf{x} = (1, 2, -1)^T$. Sia $\mathbf{x}^{(0)} = (0, 0, 0)^T$, allora:

$$\begin{aligned}
\mathbf{x}^{(1)} &= (1.25, 2.3\dots, 2)^T & \|\mathbf{e}^{(1)}\|_2 &\approx 3.026 \\
\mathbf{x}^{(2)} &= (1.12\dots, 2.42\dots, -1.55\dots)^T & \|\mathbf{e}^{(2)}\|_2 &\approx 0.71 \\
\mathbf{x}^{(3)} &= (0.93\dots, 1.89\dots, -1.54\dots)^T & \|\mathbf{e}^{(3)}\|_2 &\approx 0.55 \\
\mathbf{x}^{(4)} &= (0.98\dots, 1.92\dots, -0.82\dots)^T & \|\mathbf{e}^{(4)}\|_2 &\approx 0.19 \\
\mathbf{x}^{(5)} &= (1.01\dots, 2.02\dots, -0.91\dots)^T & \|\mathbf{e}^{(5)}\|_2 &\approx 0.09 \\
\mathbf{x}^{(6)} &= (1.00\dots, 2.01\dots, -1.04\dots)^T & \|\mathbf{e}^{(6)}\|_2 &\approx 0.046
\end{aligned}$$

Esempio 10.4 *Applichiamo il metodo di Gauss-Seidel al sistema lineare*

$$A\mathbf{x} = \mathbf{b} \qquad con \qquad A = \begin{pmatrix} 20 & 2 & -1 \\ 2 & 13 & -2 \\ 1 & 1 & 1 \end{pmatrix} \qquad e \qquad \mathbf{b} = \begin{pmatrix} 25 \\ 30 \\ 2 \end{pmatrix}.$$

La decomposizione di Gauss-Seidel è

$$M = \begin{pmatrix} 20 & 0 & 0 \\ 2 & 13 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \quad N = \begin{pmatrix} 0 & -2 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}.$$

La matrice di iterazione G ed il vettore $\mathbf{q}$ sono

$$G = \begin{pmatrix} 0 & -1/10 & 1/20 \\ 0 & 1/65 & 9/130 \\ 0 & 11/130 & -51/260 \end{pmatrix}, \quad \mathbf{q} = \begin{pmatrix} 5/4 \\ 55/26 \\ -71/52 \end{pmatrix}$$

Risulta $\rho(G) = 0.243858 < 1$, da cui il metodo converge e converge alla soluzione $\mathbf{x} = (1, 2, -1)^T$. Sia $\mathbf{x}^{(0)} = (0, 0, 0)^T$, allora:

$$\begin{aligned} \mathbf{x}^{(1)} &= (1.25, 2.11\ldots, -1.36)^T & \|\mathbf{e}^{(1)}\|_2 &\approx 0.45 \\ \mathbf{x}^{(2)} &= (0.97\ldots, 2.05\ldots, -0.91\ldots)^T & \|\mathbf{e}^{(2)}\|_2 &\approx 0.101 \\ \mathbf{x}^{(3)} &= (0.998\ldots, 2.08\ldots, -1.011\ldots)^T & \|\mathbf{e}^{(3)}\|_2 &\approx 0.084 \\ \mathbf{x}^{(4)} &= (0.991\ldots, 2.07\ldots, -0.99\ldots)^T & \|\mathbf{e}^{(4)}\|_2 &\approx 0.078 \end{aligned}$$

Capitolo 11

Metodi per Autovalori e Autovettori

Tra i diversi metodi numerici esistenti per calcolare gli autovalori e gli autovettori di una matrice, alcuni sono di tipo generale e sono applicabili a matrici dense e senza struttura, altri utilizzano in modo specifico eventuali proprietà di struttura o sparsità della matrice, permettendo di trattare problemi anche con dimensioni molto grandi. Alcuni metodi possono essere utilizzati per calcolare tutti gli autovalori e autovettori di una matrice, altri invece servono per calcolare solo alcuni autovalori, per esempio quelli che si trovano all'estremità dello spettro, ed i corrispondenti autovettori, come è richiesto in molte applicazioni. Iniziamo con alcuni richiami.

11.1 Similitudine fra Matrici

Definizione 11.1 *Due matrici A e $B \in \mathbb{R}^{n\times n}$ si dicono* **simili** *se esiste una matrice non singolare S per cui*

$$A = S\ B\ S^{-1}.$$

Osservazione 11.1 *La trasformazione che associa la matrice A alla matrice B viene detta trasformazione per similitudine; se la matrice S è ortogonale, la trasformazione viene detta trasformazione per similitudine ortogonale.*

Teorema 11.1 *Due matrici simili hanno gli stessi autovalori con le stesse molteplicità algebriche e geometriche.*

dim. Siano A e B simili, cioè tali che

$$A = S\ B\ S^{-1}.$$

Si ha che

$$\begin{aligned} det(A - \lambda I) &= det(S\ B\ S^{-1} - \lambda S\ S^{-1}) \\ &= det(S(B - \lambda I)S^{-1}) \\ &= det(S)\ det(B - \lambda I)\ det(S^{-1}) \\ &= det(B - \lambda I) \end{aligned}$$

per cui le due matrici hanno lo stesso polinomio caratteristico e quindi hanno gli stessi autovalori con le stesse molteplicità algebriche.
Se $\mathbf{x}$ è autovettore di A corrispondente all'autovalore λ, risulta

$$S\ B\ S^{-1}\mathbf{x} = \lambda\mathbf{x}$$

e quindi

$$B\ S^{-1}\mathbf{x} = \lambda S^{-1}\mathbf{x}$$

perciò il vettore $\mathbf{y} = S^{-1}\mathbf{x}$ è autovettore di B corrispondente a λ.
Inoltre, essendo S^{-1} non singolare, se $\mathbf{x}_i, i = 1, \ldots, \tau(\lambda)$ sono autovettori linearmente indipendenti di A, anche $\mathbf{y}_i = S^{-1}\mathbf{x}_i, i = 1, \ldots, \tau(\lambda)$ sono linearmente indipendenti.

Definizione 11.2 *Una matrice A simile ad una matrice diagonale D si dice* **diagonalizzabile.**

Teorema 11.2 *Una matrice A di ordine n è diagonalizzabile se e solo se ha n autovettori linearmente indipendenti. Inoltre le colonne della matrice S, per cui $S^{-1}\ A\ S$ è diagonale, sono gli autovettori di A.*

dim. Siano $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ n autovettori linearmente indipendenti di A corrispondenti agli autovalori $\lambda_1, \lambda_2, \ldots, \lambda_n$. Siano D la matrice diagonale avente λ_i come $i-$esimo elemento diagonale, ed S la matrice la cui $i-$esima colonna è uguale a $\mathbf{x}_i$. Dalla relazione

$$A\mathbf{x}_i = \lambda_i\mathbf{x}_i \qquad i = 1, \ldots n$$

si ha che

$$A\ S = S\ D.$$

Essendo S non singolare, perché formata da colonne linearmente indipendenti, esiste S^{-1}; quindi si ha

$$A = S\ D\ S^{-1}.$$

Viceversa, sia $A = S\ D\ S^{-1}$, con D matrice diagonale con gli autovalori di A come elementi diagonali. Allora risulta

$$A\ S = S\ D.$$

Indicando con $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ le colonne di S, si ha

$$A[\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n] = [\lambda_1 \mathbf{s}_1, \lambda_2 \mathbf{s}_2, \ldots, \lambda_n \mathbf{s}_n].$$

Perciò le colonne di S sono n autovettori di A che risultano linearmente indipendenti.

Esempio 11.1 *Sia*

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Questa ha come autovalori

$$\lambda_1 = 1 - \sqrt{2} \qquad \lambda_2 = 1 \qquad \lambda_3 = 1 + \sqrt{2}$$

e i corrispondenti autovettori sono

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix}.$$

Allora A è diagonalizzabile, cioè esiste una matrice S, non singolare, tale che

$$A = S\ D\ S^{-1}.$$

Infatti per quanto visto sarà:

$$S = \begin{pmatrix} 1 & 1 & 0 \\ -\sqrt{2} & 0 & \sqrt{2} \\ 0 & 1 & 1 \end{pmatrix}$$

da cui risulta

$$S^{-1} = \begin{pmatrix} 1/4 & -\sqrt{2}/4 & 1/4 \\ 1/2 & 0 & -1/2 \\ 1/4 & \sqrt{2}/4 & 1/4 \end{pmatrix}$$

e

$$A = S \begin{pmatrix} 1 - \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 + \sqrt{2} \end{pmatrix} S^{-1}.$$

Osservazione 11.2 *Fra le trasformazioni per similitudine che associano alla matrice B la matrice $A = S\ B\ S^{-1}$, hanno particolare importanza quelle per cui S è ortogonale, cioè $S^T\ S = S\ S^T = I$. Il teorema che segue mostra come sia possibile, mediante una trasformazione per similitudine ortogonale, ricondurre una qualsiasi matrice ad una forma triangolare superiore.*

Teorema 11.3 (Forma Normale di Schur) *Sia $A \in \mathbb{C}^n$ e siano $\lambda_1, \ldots, \lambda_n$ i suoi autovalori. Allora esiste una matrice ortogonale/unitaria Q e una matrice triangolare superiore T i cui elementi diagonali sono i λ_i, tali che*

$$A = Q\ T\ Q^T.$$

se la matrice A ha elementi reali esiste la forma normale reale di Schur.

Teorema 11.4 (Forma Normale Reale di Schur) *se $A \in \mathbb{R}^{n\times n}$, esiste una matrice ortogonale $Q \in \mathbb{R}^{n\times n}$ e una matrice $T \in \mathbb{R}^{n\times n}$ triangolare superiore a blocchi della forma*

$$T = \begin{pmatrix} R_{1,1} & R_{1,2} & \ldots & R_{1,m} \\ & R_{2,2} & & \vdots \\ & & \ddots & \\ 0 & & & R_{m,m} \end{pmatrix}.$$

dove i blocchi $R_{j,j}$ per $j = 1, \ldots, m$ hanno ordine 1 o 2. Se λ_j è autovalore reale di A, allora $R_{j,j}$ ha ordine 1 e coincide con λ_j, se invece è complesso, allora il blocco $R_{j,j}$ ha ordine 2 ed ha come autovalori λ_j e $\bar{\lambda}_j$ (il complesso coniugato). La somma delle dimensioni dei blocchi $R_{j,j}$ $j = 1, \ldots, m$ è pari ad n.

Osservazione 11.3 *Una classe particolarmente importante di matrici è quella delle matrici normali, cioè tali che*

$$A\ A^H = A^H\ A$$

dove H sta per Hermitiana ed equivale ad una matrice simmetrica in campo complesso. Questa classe è importante perché comprende tutte e sole le matrici diagonalizzabili con trasformazioni per similitudine unitarie. Vale infatti il seguente teorema.

Teorema 11.5 *Una matrice $A \in \mathbb{C}^{n\times n}$ è normale (cioè $A\ A^H = A^H\ A$) se e solo se esiste una matrice unitaria Q tale che*

$$A = Q \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} Q^T$$

in cui $\lambda_1, \lambda_2, \ldots, \lambda_n$ sono gli autovalori di A. La matrice Q ha per colonne gli autovettori di A, che quindi sono a due a due ortonormali.

11.2 Ricerca di Autovalori

Tutti i metodi per la ricerca degli autovalori di una matrice possono essere divisi in due classi:

1. Metodi in cui il calcolo viene effettuato in due fasi

   - riduzione con metodi diretti della matrice A in una matrice simile B, di cui sia più agevole calcolare gli autovalori;
   - calcolo degli autovalori di B con un metodo iterativo.

   Questi metodi si applicano in generale a problemi di piccole dimensioni, per i quali tutti i dati su cui si opera possono essere contenuti nella memoria centrale del calcolatore.

2. Metodi completamente iterativi, che richiedono ad ogni passo la moltiplicazione di una matrice per un vettore, o la risoluzione di un sistema lineare. Questi metodi si applicano in generale a problemi di grandi dimensioni, anche nel caso in cui non sia possibile contenere tutti i dati nella memoria del calcolatore.

Nei metodi della prima classe, per la riduzione della matrice A nella matrice B, si utilizzano metodi diretti analoghi a quelli descritti per la fattorizzazione delle matrici. Nel caso più generale la matrice B che si ottiene è tale che

$$b_{i,j} = 0 \quad \text{per} \quad i > j + 1 \quad i, j = 1, \ldots, n.$$

Una matrice B con questa proprietà è detta essere in **forma di Hessenberg superiore**. La trasformazione per similitudine della matrice A nella matrice B è fatta per passi successivi

$$A_{k+1} = S_k \; A_k \; S_k^{-1} \quad k = 1, \ldots, m-1$$

dove

$$A_1 := A \quad \text{e alla fine} \quad B := A_m$$

per cui posto $S := S_{m-1} \; S_{m-2} \; \ldots S_1$, risulta

$$B = S \; A \; S^{-1}$$

e se $\mathbf{x}$ è autovettore di B, $S\mathbf{x}$ è autovettore di A.
Le matrici S_k sono di solito matrici elementari di Gauss o di Householder. Se S_k è una matrice di Householder, risulta

$$\|S_k\|_2 \cdot \|S_k^{-1}\|_2 = 1$$

I metodi iterativi per il calcolo degli autovalori di B potrebbero essere anche applicati direttamente alla matrice A. Trasformando però la matrice A nella matrice B, si abbassa notevolmente la complessità computazionale (per esempio per il metodo QR che vedremo si passa da $O(n^3)$ a $O(n^2)$). Per il calcolo degli autovalori della matrice B, due sono le tecniche più usate:

- se sono richiesti solo pochi autovalori rispetto alla dimensione della matrice (non più del 25%), conviene usare un metodo iterativo che calcoli un singolo autovalore per volta, come ad esempio un metodo di iterazione funzionale applicato all'equazione caratteristica o il metodo delle potenze inverse.

- se sono richiesti tutti o molti degli autovalori, il metodo migliore è in generale il metodo QR che vedremo.

Osservazione 11.4 *Se la matrice A è hermitiana, e la trasformazione viene eseguita con matrici unitarie, la matrice B risulta hermitiana tridiagonale.*

11.2.1 Riduzione di una matrice in forma di Hessenberg

Se si applicano ad una matrice $A_{n\times n}$ matrici elementari di Householder con l'obiettivo di azzerare gli elementi della colonna $j-$esima di indici $i = j+2, \ldots, n$ si ottiene una matrice di Hessenberg superiore ad un costo computazionale di $\frac{5}{3}n^3$ operazioni moltiplicative.

Esempio 11.2 *Si consideri la matrice $A \in \mathbb{R}^{4\times 4}$*

$$A = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 1 & 4 & 3 & 2 \\ 1 & 1 & 4 & 3 \\ 1 & 1 & 1 & 4 \end{pmatrix}$$

Applicando il metodo di Householder, al primo passo si ottiene

$$\mathbf{v}_1 = (0.0, 2.732051, 1.0, 1.0)^T \qquad \beta_1 = 0.2113248$$

e quindi $H_1 = I - \beta_1 \mathbf{v}_1 \mathbf{v}_1^T$ e $A_2 = H_1 A H_1^T$

$$A_2 = \begin{pmatrix} 4 & -3.464098 & -0.366024 & -1.366024 \\ -1.73205 & 7.666641 & -0.5446615 & -1.12209 \\ 0 & 0.08931351 & 1.877991 & 1.032692 \\ 0 & 1.244013 & -0.6993591 & 2.455341 \end{pmatrix}.$$

Al secondo passo si ottiene

$$\mathbf{v}_2 = (0.0, 0.0, 1.336528, 1.244013)^T \qquad \beta_2 = 0.5999027$$

e quindi $H_2 = I - \beta_2 \mathbf{v}_2 \mathbf{v}_2^T$ *e* $A_3 = H_2 A_2 H_2^T$

$$A_3 = \begin{pmatrix} 4 & -3.464098 & 1.388726 & 0.2672625 \\ -1.73205 & 7.666641 & 1.158131 & 0.4629154 \\ 0 & -1.247213 & 2.476189 & -0.7423077 \\ 0 & 0 & 0.9897442 & 1.857142 \end{pmatrix}$$

che è in forma di Hessenberg superiore.

11.2.2 Metodo Q R per il calcolo degli autovalori

Il metodo QR è il metodo più usato per calcolare tutti gli autovalori di una matrice, in quanto il più efficiente.
Il metodo è molto complicato, sia come descrizione che come implementazione, anche se il principio su cui si basa è semplice. Il metodo richiede tutta una serie di accorgimenti, senza i quali non potrebbe essere efficiente: riduzione preliminare della matrice in forma di Hessenberg superiore (o triangolare nel caso di matrici hermitiana) per ridurre il costo computazionale ad ogni iterazione, utilizzazione di una tecnica di traslazione per aumentare la velocità di convergenza; riduzione dell'ordine della matrice quando un autovalore è stato approssimato con sufficiente precisione, per calcolare un altro autovalore.
Il metodo QR, che è stato descritto da Francis nel 1961, utilizza la fattorizzazione QR di una matrice; esso deriva da un precedente metodo, detto metodo LR, proposto da Rutishauser nel 1958, che utilizza la fattorizzazione LU di una matrice.

Algoritmo di base

Nel metodo QR viene generata una successione $\{A_k\}$ di matrici nel modo seguente: posto

$$A_1 := A$$

per $k = 1, 2, \ldots$ si calcola una fattorizzazione QR di A_k

$$A_k = Q_k R_k$$

dove Q_k è unitaria e R_k è triangolare superiore, e si definisce la matrice A_{k+1} per mezzo della relazione

$$A_{k+1} := R_k Q_k.$$

per le precedenti risulta che

$$A_{k+1} = Q_k^T A_k Q_k$$

e quindi le matrici della successione $\{A_k\}$ sono tutte simili fra loro. Sotto opportune ipotesi la successione converge ad una matrice triangolare superiore (diagonale se A è hermitiana) che ha come elementi diagonali gli autovalori di A.

Risultati di convergenza

Il seguente teorema di convergenza, viene dato in ipotesi piuttosto restrittive in cui è più facile la sua dimostrazione, ma la sua validità può essere provata anche in ipotesi più deboli.

Teorema 11.6 *Sia $A \in \mathbb{C}^{n\times n}$ tale che i suoi autovalori λ_i, $i = 1, \ldots, n$ abbiano moduli tutti distinti, cioè*

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0.$$

Indicata con X la matrice degli autovettori di A, tale che

$$A = XDX^{-1},$$

in cui D è la matrice diagonale degli autovalori, si supponga che la matrice X^{-1} ammetta la fattorizzazione LU. Allora esistono delle matrici S_k tali che

$$\lim_{k\to\infty} S_k^H R_k S_{k-1} = \lim_{k\to\infty} S_{k-1}^H A_k S_{k-1} = T$$

e

$$\lim_{k\to\infty} S_{k-1}^H Q_k S_k = I$$

dove T è triangolare superiore con gli elementi diagonali uguali a $\lambda_1, \lambda_2, \ldots, \lambda_n$. Quindi gli elementi diagonali di A_k tendono agli autovalori di A.

Se X^{-1} non ammette fattorizzazione LU, si può dimostrare che il metodo QR è ancora convergente. In questo caso gli elementi diagonali di T coincidono ancora con i λ_i, ma non sono più in ordine di modulo decrescente.
Se l'ipotesi che tutti gli autovalori abbiano modulo distinto, non è verificata, la successione formata dagli elementi diagonali di A_k non converge. Questa ipotesi è troppo restrittiva, e non consente di utilizzare il metodo QR in casi particolarmente importanti nelle applicazioni, come quelli in cui la matrice A ha elementi reali e autovalori non reali. Però anche in questo caso il metodo QR può essere applicato con opportune varianti. Sia ad esempio

$$|\lambda_1| > \ldots > |\lambda_r| = |\lambda_{r+1}| > \ldots > |\lambda_n| > 0$$

dove λ_r e λ_{r+1} sono due numeri complessi coniugati, oppure due numeri reali. Allora la successione delle $\{A_k\}$ o meglio degli elementi diagonali non converge agli autovalori di stesso modulo, ma gli autovalori dei blocchi diagonali convergono a λ_r e λ_{r+1}. Come detto, gli elementi diagonali di A_k di indice diverso da r ed $r+1$ invece convergono agli altri autovalori.
Situazioni analoghe si presentano quando la matrice A ha più autovalori di modulo uguale e in questo caso il metodo QR genera matrici R_k con struttura triangolare a blocchi, in cui gli autovalori dei blocchi diagonali convergono ad autovalori di A.

Costo computazionale e stabilità

Il metodo QR applicato a una matrice di ordine n ha ad ogni passo un costo computazionale dell'ordine di n^3 operazioni moltiplicative (per calcolare la fattorizzazione $A_k = Q_k R_k$ e per moltiplicare la matrice triangolare R_k per le matrici elementari di Householder).
Per abbassare il costo computazionale globale conviene prima trasformare la matrice A in forma di Hessenberg superiore. Questa trasformazione viene eseguita una sola volta perché il metodo QR, applicato a matrici in forma di Hessenberg superiore produce matrici A_k in forma di Hessenberg superiore.
Infatti se A_k è in forma di Hessenberg superiore, la matrice Q_k è data dal prodotto di $n-1$ matrici elementari di Hoseholder che sono in forma di Hessenberg superiore e quindi la matrice A_{k+1}, prodotto di una matrice triangolare superiore R_k per una matrice Q_k in forma di Hessenberg superiore, risulta ancora in forma di Hessenberg superiore.
Se la matrice A è hermitiana, la matrice in forma di Hessenberg superiore, ottenuta applicando ad A il metodo di Householder, è ancora hermitiana, e quindi risulta tridiagonale. Inoltre anche tutte le matrici A_k generate dal metodo QR sono hermitiane e quindi tridiagonali.
Il metodo QR applicato ad una matrice A in forma di Hessenberg superiore ha ad ogni passo un costo di $2n^2$ operazioni moltiplicative (che è il costo computazionale per calcolare la fattorizzazone $A_k = Q_k R_k$).
Per quel che riguarda la stablità si può dimostrare che il metodo QR per il calcolo degli autovalori gode delle stesse proprietà di stabilità di cui gode la fattorizzazione QR di una matrice.

11.2.3 Metodo delle Potenze

Il metodo delle potenze è un metodo iterativo per approssimare l'autovalore di modulo massimo di una matrice e il corrispondente autovettore. Vediamolo nel semplice caso che la matrice sia diagonalizzabile e abbia un solo autovalore di

modulo massimo. Sia $A_{n\times n}$ con n autovettori $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$, linearmente dindipendenti e autovalori $\lambda_1, \lambda_2, \ldots, \lambda_n$ tali che

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|.$$

Fissato il vettore $\mathbf{y}^{(0)}$ si genera la successione $\{\mathbf{y}^{(k)}\}$ $k = 1, 2, \ldots$ così definita

$$\mathbf{y}^{(k)} = A\mathbf{y}^{(k-1)} \qquad k = 1, 2, \ldots.$$

Poiché gli autovettori sono linearmente indipendenti, il vettore $\mathbf{y}^{(0)}$ può essere espresso come una loro opportuna combinazione lineare

$$\mathbf{y}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{x}^{(i)}$$

e sia scelto in modo che $\alpha_1 \neq 0$; allora

$$\begin{aligned} \mathbf{y}^{(k)} &= A^k\mathbf{y}^{(0)} = \textstyle\sum_{i=1}^{n} \alpha_i A^k \mathbf{x}^{(i)} = \sum_{i=1}^{n} \alpha_i \lambda_i^k \mathbf{x}^{(i)} \\ &= \lambda_1^k \left[\alpha_1 \mathbf{x}^{(1)} + \textstyle\sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \mathbf{x}^{(i)}\right]. \end{aligned} \tag{11.1}$$

Indicate con $y_r^{(k)}$ e $x_r^{(i)}$ le r-esime componenti dei vettori $\mathbf{y}^{(k)}$ e $\mathbf{x}^{(i)}$, per gli indici j per cui $y_j^{(k)} \neq 0$ e $x_j^{(i)} \neq 0$ si ha

$$\frac{y_j^{(k+1)}}{y_j^{(k)}} = \lambda_1 \frac{\alpha_1 x_j^{(1)} + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k+1} x_j^{(i)}}{\alpha_1 x_j^{(1)} + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k} x_j^{(i)}} \tag{11.2}$$

e poiché $|\frac{\lambda_i}{\lambda_1}| < 1$ per $i \geq 2$ si ha

$$\lim_{k\to\infty} \frac{y_j^{(k+1)}}{y_j^{(k)}} = \lambda_1.$$

Quindi da un certo indice k in poi l'autovalore λ_1 può essere approssimato mediante uno dei rapporti $\frac{y_j^{(k+1)}}{y_j^{(k)}}$.

Con questo metodo si può approssimare anche l'autovettore $\mathbf{x}^{(1)}$. Dalla (11.1) risulta

$$\lim_{k\to\infty} \frac{\mathbf{y}^{(k)}}{\lambda_1^k} = \alpha_1 \mathbf{x}^{(1)}$$

e quindi per $j = 1, 2, \ldots, n$ è

$$\lim_{k\to\infty} \frac{y_j^{(k)}}{\lambda_1^k} = \alpha_1 x_j^{(1)}$$

e facendo il rapporo delle due precedenti

$$\lim_{k\to\infty} \frac{\mathbf{y}^{(k)}}{y_j^{(k)}} = \frac{\mathbf{x}^{(1)}}{x_j^{(1)}}$$

per tutti gli indici j per cui $x_j^{(1)} \neq 0$.

Questo metodo richiede ad ogni passo il prodotto di una matrice A per un vettore (vedi (11.1)); operando in aritmetica finita, dopo pochi passi della (11.1) si possono presentare condizioni di overflow o underflow. Per evitare queste situazioni è necessario eseguire ad ogni passo una normalizzazione del vettore ottenuto, costruendo una successione $\mathbf{y}^{(k)}$ $k = 1, 2, \ldots$ così fatta:

$$\mathbf{u}^{(k)} = A\mathbf{y}^{(k-1)}$$

$$\mathbf{y}^{(k)} = \frac{\mathbf{u}^{(k)}}{||\mathbf{u}^{(k)}||}$$

così ché $||\mathbf{y}^{(k)}|| = 1$.

Si osserva che

$$\mathbf{y}^{(k)} = \frac{A\mathbf{y}^{(k-1)}}{||\mathbf{u}^{(k)}||} = \frac{A\mathbf{y}^{(\mathbf{k-2})}}{||\mathbf{u}^{(k)}||||\mathbf{u}^{(k-1)}||} = \cdots = \frac{A\mathbf{y}^{(0)}}{\gamma_k}$$

con $\gamma_k = \prod_{i=1}^{k} ||\mathbf{u}^{(i)}||$ e ragionando come nella (11.1) si ha che il rapporto fra le j-esime componenti di $\mathbf{u}^{(k+1)}$ e $\mathbf{y}^{(k)}$ per gli indici j per cui $y_j^{(k)} \neq 0$ e $x_j^{(1)} \neq 0$ è dato come nella (11.2) e quindi

$$\lim_{k\to\infty} \frac{u_j^{(k+1)}}{y_j^{(k)}} = \lambda_1.$$

Osservazione 11.5 *Fissata una tolleranza tol, come condizione di arresto del metodo iterativo si può utilizzare una delle condizioni seguenti*

$$\left| ||\mathbf{u}^{(k+1)}|| - ||\mathbf{u}^{(k)}|| \right| < tol$$

$$\left| \frac{||\mathbf{u}^{(k+1)}|| - ||\mathbf{u}^{(k)}||}{||\mathbf{u}^{(k+1)}||} \right| < tol.$$

11.2.4 Condizionamento del calcolo degli autovalori

Una matriice può avere alcuni autovalori molto sensibili alle perturbazioni dei suoi elementi mentre altri insensibili. È conveniente avere un numero che definisce il condizionamento di una matrice rispetto al problema degli autovalori;

esso sarà il **numero di condizione spettrale**.
È evidente che tale numero può fornire una informazione insoddisfacente. Infatti se qualche autovalore è molto sensibile allora il numero di condizione spettrale è necessariamente grande, anche se altri autovalori sono insensibili. Si può ovviare a questo inconveniente introducendo numeri di condizione che evidenziano la sensibilità dei singoli autovalori e che si chiamano **numeri di condizione della matrice** rispetto al problema degli autovalori.

Definizione 11.3 *Si dice norma assoluta $\|\cdot\|$ una norma matriciale indotta che verifichi le seguenti proprietà*

$$\|D\| = \max_{i=1,\dots,n} |d_{i,i}|$$

per ogni matrice diagonale $D \in \mathbb{C}^{n\times n}$.

Osservazione 11.6 *Le norme $\|\cdot\|_1, \|\cdot\|_2$ e $\|\cdot\|_\infty$ sono assolute.*

Teorema 11.7 (di Bauer-Fike) *Sia $A \in \mathbb{C}^{n\times n}$ una matrice diagonalizzabile, cioè tale che*

$$A = TDT^{-1}$$

con D diagonale e T non singolare. Se $\delta A \in \mathbb{C}^{n\times n}$ e η è un autovalore di $A+\delta A$, allora esiste almeno un autovalore λ di A tale che

$$|\lambda - \eta| \leq \mu(T)\|\delta A\|$$

dove $\mu(T) = \|T\| \cdot \|T^{-1}\|$ per una norma assoluta $\|\cdot\|$.

Introduciamo un numero di condizione che dia una misura dell'effetto della perturbazione su un solo autovalore di A.

Teorema 11.8 *Sia $A \in \mathbb{C}^{n\times n}$, λ un autovalore di A di molteplicità algebrica 1, $\mathbf{x}, \mathbf{y} \in \mathbb{C}^{n\times n}$, $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$, tali che*

$$A\mathbf{x} = \lambda\mathbf{x}$$

$$\mathbf{y}^H A = \lambda\mathbf{y}^H.$$

Allora è $\mathbf{y}^H\mathbf{x} \neq 0$ ed inoltre esiste nel piano complesso un intorno V dello zero e una funzione $\lambda(\epsilon) : V \to \mathbb{C}$, analitica, tale che

1. *$\lambda(\epsilon)$ è autovalore con molteplicità 1 di $A + \epsilon F$, $F \in \mathbb{C}^{n\times n}$*
2. *$\lambda(0) = \lambda$*

3. $\lambda'(0) = \frac{\mathbf{y}^H F \mathbf{x}}{\mathbf{y}^H \mathbf{x}}$

4. *a meno di termini di ordine superiore in* ϵ *è*

$$\lambda(\epsilon) - \lambda = \epsilon \frac{\mathbf{y}^H F \mathbf{x}}{\mathbf{y}^H \mathbf{x}}.$$

da questo risultato si ha che la variazione nell'autovalore dovuta alla perturbazione ϵF di A è proporzionale ad ϵ.
Inoltre il condizionamento del problema dipende dalla quantità

$$\left| \frac{\mathbf{y}^H F \mathbf{x}}{\mathbf{y}^H \mathbf{x}} \right|$$

che, data F, è tanto più grande quanto più piccolo è $|\mathbf{y}^H \mathbf{x}|$.
Nel caso di autovalore λ di molteplicità algebrica $\sigma(\lambda) > 1$ e molteplicità geometrica $\tau(\lambda)$ si può arrivare ad una relazione del tipo

$$|\lambda_i(\epsilon) - \lambda| \leq \gamma |\epsilon|^{1/\mu}$$

per $i = 1, \ldots, \sigma(\lambda)$, γ costante positiva; se $\mu > 1$ il problema del calcolo di λ può essere fortemente malcondizionato.

Capitolo 12

Il problema dei Minimi Quadrati

12.1 Le equazioni normali

Sia

$$A\mathbf{x} = \mathbf{b} \tag{12.1}$$

un sistema lineare con $A \in \mathbb{R}^{m\times n}$ tale che $m \geq n$. Se $m > n$, il sistema (12.1) ha più equazioni che incognite e si dice **sovradeterminato**. Se il sistema (12.1) non ha soluzione, fissata una norma vettoriale $\|\cdot\|$, si cercano i vettori $\mathbf{x} \in \mathbb{R}^n$ che minimizzano la quantità $\|A\mathbf{x} - \mathbf{b}\|$. In norma 2, il problema diventa quello di determinare un vettore $\mathbf{x} \in \mathbb{R}^n$ tale che

$$\|A\mathbf{x} - \mathbf{b}\|_2 = \min_{\mathbf{y}\in\mathbb{R}^n} \|A\mathbf{y} - \mathbf{b}\|_2 = \mu. \tag{12.2}$$

Questo problema viene detto **problema dei minimi quadrati**.

Teorema 12.1 *Se A ha rango massimo la soluzione del problema (12.2) esiste, è unica e coincide con la soluzione del sistema lineare*

$$A^TA\mathbf{x} = A^T\mathbf{b}. \tag{12.3}$$

Tale sistema viene detto sistema delle **equazioni normali**.

dim. Siano

$$S(A) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = A\mathbf{x},\ \mathbf{x} \in \mathbb{R}^n\}$$

e

$$S(A)^\perp = \{\mathbf{z} \in \mathbb{R}^m : \mathbf{z}^T\mathbf{y} = 0,\ \forall \mathbf{y} \in S(A)\}$$

il sottospazio di $\mathbb{R}^m$ immagine di A, e il sottospazio ortogonale a $S(A)$. Il vettore $\mathbf{b}$ può essere così decomposto

$$\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2, \quad \text{dove} \quad \mathbf{b}_1 \in S(A) \quad \text{e} \quad \mathbf{b}_2 \in S(A)^\perp$$

per cui per il residuo

$$\mathbf{r} = \mathbf{b}_1 - A\mathbf{x} + \mathbf{b}_2 = \mathbf{y} + \mathbf{b}_2,$$

dove

$$\mathbf{y} = \mathbf{b}_1 - A\mathbf{x} \in S(A) \quad \text{e} \quad \mathbf{b}_2 \in S(A)^\perp$$

vale

$$\|\mathbf{r}\|_2^2 = (\mathbf{y} + \mathbf{b}_2)^T(\mathbf{y} + \mathbf{b}_2) = \|\mathbf{y}\|_2^2 + \|\mathbf{b}_2\|_2^2,$$

in quanto $\mathbf{y}^T\mathbf{b}_2 = \mathbf{b}_2^T\mathbf{y} = 0$. Poiché solo $\mathbf{y}$ dipende da $\mathbf{x}$, si ha che $\|\mathbf{r}\|_2^2$ è minimo se e solo se $\mathbf{b}_1 = A\mathbf{x}$, cioè se e solo se il vettore $\mathbf{r} \in S(A)^\perp$ ed è quindi ortogonale alle colonne di A, cioè

$$A^T\mathbf{r} = A^T(\mathbf{b} - A\mathbf{x}) = \mathbf{0}.$$

Ne segue quindi che $\mathbf{x}$ è soluzione di (12.2) se e solo se è soluzione di (12.3).

12.2 Metodo QR per i minimi quadrati

Sia A di rango massimo e si applichi il metodo di Householder per ottenere la fattorizzazione $Q\ R$ di A; sarà $Q \in \mathbb{R}^{m\times m}$ ortogonale ed $R \in \mathbb{R}^{m\times n}$ con la seguente particolare struttura

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

con $R_1 \in \mathbb{R}^{n\times n}$ triangolare superiore e non singolare essendo H di rango massimo.
Dalla fattorizzazione $A = Q\ R$ si ha

$$\begin{aligned} \|A\mathbf{x} - \mathbf{b}\|_2 &= \|QR\mathbf{x} - \mathbf{b}\|_2 \\ &= \|Q(R\mathbf{x} - Q^T\mathbf{b})\|_2 \\ &= \|R\mathbf{x} - Q^T\mathbf{b}\|_2 \\ &= \|R\mathbf{x} - \mathbf{c}\|_2 \end{aligned}$$

dove $\mathbf{c} = Q^T\mathbf{b}$.
Partizionando il vettore $\mathbf{c}$ nel modo seguente

$$\mathbf{c} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix}$$

con $\mathbf{c}_1 \in \mathbb{R}^n$ e $\mathbf{c}_2 \in \mathbb{R}^{m-n}$, si ha che

$$R\mathbf{x} - \mathbf{c} = \begin{pmatrix} R_1\mathbf{x} - \mathbf{c}_1 \\ -\mathbf{c}_2 \end{pmatrix}$$

da cui

$$\begin{aligned} \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2 = & \min_{\mathbf{x}} \|R\mathbf{x} - \mathbf{c}\|_2^2 \\ = & \min_{\mathbf{x}} (\|R_1\mathbf{x} - \mathbf{c}_1\|_2^2 + \|\mathbf{c}_2\|_2^2) \\ = & \|\mathbf{c}_2\|_2^2 + \min_{\mathbf{x}} (\|R_1\mathbf{x} - \mathbf{c}_1\|_2^2) \end{aligned}$$

Poicé R_1 è non singolare, esiste una ed una sola soluzione per il sistema

$$R_1\mathbf{x} = \mathbf{c}_1,$$

sia questa $\mathbf{x}^*$ e sarà

$$\min_{\mathbf{x}} \left(\|R_1\mathbf{x} - \mathbf{c}_1\|_2^2\right) = \|R_1\mathbf{x}^* - \mathbf{c}_1\|_2^2 = 0$$

Segue che $\mathbf{x}^*$ è la soluzione del problema dei minimi quadrati e

$$\|\mathbf{c}_2\|_2 = \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2.$$

Elenco delle figure

Bibliografia

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

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

[Ove01] M. L. Overton. Numerical Computing with IEEE floating Point Arithmetic. *Siam*, 2001.

[Higham2002] N. Higham. Accuracy and Stability of Numerical Algorithms. *Siam*, 2002.

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

[Bau88] D. Bau, N. Trefethen. *Numerical Linear Algebra*. SIAM, 1988.

[Bini88] D. Bini, M. Capovani, O. Menchi. *Metodi Numerici per l'Algebra Lineare*. Zanichelli, 1988.

[PRE75] P.M. Prenter. *Splines and Variational Methods*. John Wiley & Sons, 1975.

[PreSha85] F.P. Preparata, M.I. Shamos. *Computational Geometry*. Springer, 1985.

[Gol91] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 1991. Una ristampa è disponibile presso: `<http://download.oracle.com/docs/cd/E19957-01/806-3568/ncg_goldberg.html>`

[Mic02] D. Michelucci. The Robustness Issue. *Technical report of Ecole de Mines,F-42023 Saint-Étienne 2002*. Disponibile presso: `<http://www.emse.fr/~micheluc/robustness.ps.gz>`

[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.

[LaRi81] J.M.Lane, R.F.Riesenfeld, *"Bounds on a Polynomial"*, *BIT*, Vol. 21, (1981), pp. 112-117

[NisSedKak90] T.Nishita, T. W.Sederberg, M.Kakimoto. Ray Tracing Trimmed Rational Surface Patches; *Computer Graphics*, 24, 4 (1990), 337-345.