Appunti dalle lezioni di Calcolo Numerico

A.A. 2013/2014

Indice

Elenco delle figure

Capitolo 1

Come sono fatti gli elaboratori

1.1 Le componenti principali di un calcolatore elettronico

Un calcolatore elettronico è formato da un grande numero di componenti integrate tra di loro in maniera assai complessa. Come sono fatte e come interagiscono le diverse componenti, pur essendo una materia importante, non è per‘ rilevante ai fini del funzionamento numerico dei sistemi di calcolo. Per cercare di semplificazione si sceglie qui di dare una descrizione assai sommaria e superficiale delle componenti principali di importanza nelle applicazioni numeriche. Dala Figura 1.1, che mostra questa descrizione schematica, possiamo individuare tre componenti fondamentali: la *Memoria*, il *Processore*, e l'unità *Input-Output* o *I/O*. Tali componenti sono collegate da frecce che indicano lo scambio di informazioni sotto forma di *Dati e Istruzioni*, *Segnali di controllo*, *Indirizzi*.

1.1.1 Il processore

Il processore (o CPU, Central Processing Unit) è il cuore del computer, ovvero la componente che controlla tutte le altre unità e che esegue i calcoli numerici; la memoria è il dispositivo che contiene tutti i dati e le istruzioni; il sistema di I/O serve per poter colloquiare con la macchina. Bisogna pensare alla CPU come un sistema che esegue ciclicamente tre passi: 1. prende dati e istruzioni dalla memoria (fase di $fetch$); 2. li interpreta secondo un linguaggio prestabilito (fase di *decode*); 3. esegue istruzioni sui dati presi (fase di *execute*). Questo insieme di passi è ripetuto a intervalli regolari scanditi da una componente chiamata il *clock* (Fig. 1.2, sinistra). Nei computer moderni il clock lavora a diversi GHz, per cui i tre passi vengono ripetuti diversi milioni di volte al secondo (1 GHz $= 1 \times 10^6$ cicli/secondo), ogni volta però con dati e istruzioni che possono essere diversi.

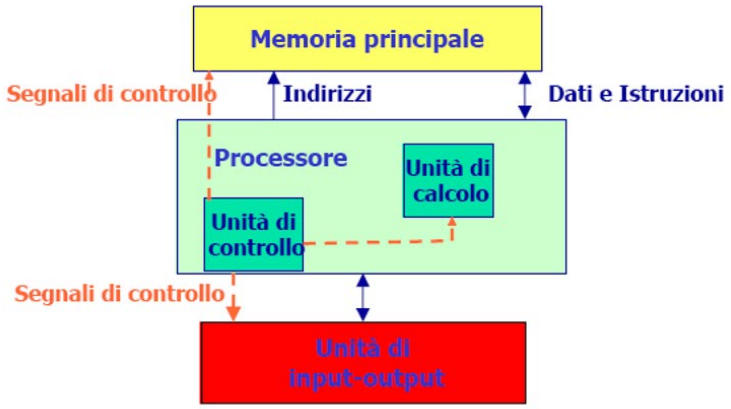

Figura 1.1: Descrizione schematica delle componenti di un computer dal punto di vista del Calcolo Numerico

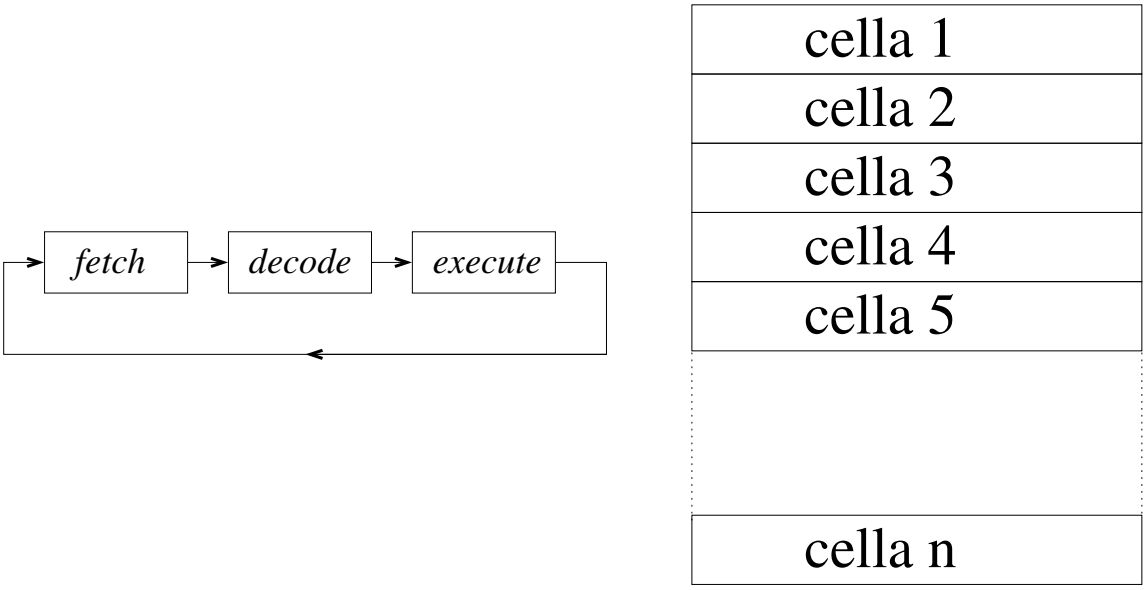

Figura 1.2: Organizzazione sequenziale del ciclo di una CPU (sinistra) e della memoria di un computer (destra)

.

1.1.2 La memoria

La memoria serve, come dice la parola, per memorizzare e rendere disponibile in qualsiasi istante dati e istruzioni[1]. Essa può essere pensata come un insieme di celle di dimensione costante che servono per memorizzare un singolo dato o istruzione. Ogni cella è *indirizzabile*, cioè è individuata univocamente e il suo contenuto può essere preso *fetched* e usato nella CPU oppure può essere variato per immagazzinare un nuovo dato. La rappresentazione in Figura 1.2 a destra può essere usata per immaginare come la memoria possa essere organizzata logicamente, anche se la realtà elettronica è diversa.

L'unità di misura dell'informazione è il cosidetto *bit*, che è definito come la più piccola parte di informazione che viene usato per formare il dato. Un bit può essere pensato come una cifra binaria, il cui utilizzo concreto sarà visto più avanti, che può assumere i valori binari zero o uno (0,1). Per questioni tecnologiche, i bit si raggruppano a 8 a 8 formando così il *byte* (= 8 bit). Le unità di misura successive cercano di seguire lo standard del Sistema Internazionale sostituendo però il fattore 1000 con 1024, cioè la potenza di 2 più vicina a 1000. In questo modo 1 kB (1 kilobyte) = 1024 byte, 1 MB (1 megabyte) = 1024 kB, 1 GB (1 gigabyte) = 1024 MB, 1 TB (1 Terabyte) = 1024 GB, eccetera. Bisogna stare attenti perchè per questioni commerciali alcune unità di misura sono in bit e non in bytes. Ad esempio, la velocità di trasmissione dei dati attraverso una rete si misura in Mbit = 1/8 MBytes! Bisogna quindi utilizzare le parole "bit" e "bytes" per evitare ambiguità.

Allo stesso modo, in alcuni settori (ad esempio costruttori di hard-disk) usano i prefissi "kilo", "mega", etc, come multipli di 1000 e non di 1024. Questo è il motivo per cui un disco fisso da 100 GB

Una cella di memoria è anche chiamata un *word*, o parola, ed è formata, nei computer moderni, da 64 bit, numerati da destra a sinistra da 0 a 63. Computer meno recenti hanno *word* a 32 bit, ma nella realtà si tende a dimenticare che ci sono moltissimi computer di tipo diverso in circolazione. Basti pensare ai telefoni cellulari, che altro non sono che dei computer con processori relativamente potenti che possono avere *word* anche a 32 o addiritura 16 bit. Per non parlare poi di applicazioni specializzate, quali data-loggers, eccetera. Per quello che ci riguarda, però, noi faremo sempre riferimento a computer che utilizzano *word* a 64 bit.

Ovviamente, ci manca ora un meccanismo con cui memorizzare le informazioni all'interno di ogni singola cella. Tale meccanismo verrà discusso parzialmente nel paragrafo successivo dedicato alla rappresentazione interna dei numeri.

[1]Il primo computer moderno viene generalmente identificato dal fatto che appunto sia i dati che il programma erano memorizzati internamente. Tale idea, originale di John von Neumann [11] scaturite dalle idee teoriche di Alan Turing [10] sono alla base dell'architettura di calcolatori attuali, chiamata architettura di von Neumann.

1.1.3 Unità di Input/Output

Ai fini della nostra comprensione dell'architettura di un computer, le unità di Input/Output non sono altro che le strutture hardware necessarie per interloquire con la macchina, ad esempio lo schermo e la tastiera. Nella realtà informatica si intende per I/O anche tutte le apparecchiatura per memorizzare in maniera permanente le informazioni, e qundi i cosidetti dischi fissi, i supporti magnetici, i supporti ottici (CDROM e DVDROM), eccetera. Noi però adottiamo una visione leggermente diversa, più aderente al dettato di von Neumann, che verrà spiegata nel paragrafo successivo a quello della rappresentazione dei numeri all'elaboratora. Pertanto, continuiamo a pensare alle unità I/O come quelle unità che ci permettono di colloquiare con il computer.

1.2 Le numerazioni nondecimali

Prima di passare alla rappresentazione interna dei numeri all'elaboratore bisogna introdurre la possibilità di rappresentare numeri con basi diverse. In tutta generalità un numero "decimale" a può essere rappresentato con la seguente notazione:

$$a = a_n N^n + a_{n-1} N^{n-1} + \ldots + a_2 N^2 + a_1 N + a_0 + a_{-1} N^{-1} + \ldots + a_{-r} N^{-r}, \quad (1.1)$$

con a_j un numero intero tale che $0 \leq a_j \leq N-1$, e con $n > 0$ e $r > 0$, quest'ultimo possibilmente infinito (numero con infinite cifre decimali. Il numero intero $N > 1$ è detta la base della rappresentazione. Si può dimostrare che tale rappresentazione è univoca e può rappresentare teoricamente qualsiasi numero. Ci basti però fare un esempio per convincerci della ragionevolezza della frase precedente.

Esempio 1.2.1. Il numero $a = 1234.5678$ si può scrivere come:

$$a = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10 + 4 + 5 \times 10^{-1} + 6 \times 10^{-2} + 7 \times 10^{-3} + 8 \times 10^{-4},$$

dove si è utilizzata la (1.1) con $N = 10$.

E' intuitibile dall'esempio precedente l'unicità della rappresentazione (1.1), come è immediato pensare che noi siamo abituati a usare e a far di conto con i numeri "decimali", cioè espressi tramite la base $N = 10$. E peró altrettanto intuibile come sia possibile usare una base diversa da quella decimale senza difficoltà teoriche [2]. Da quanto abbiamo visto in precedenza, i calcolatori usano il bit binario questioni

[2] Ci sarebbero ovviamente difficoltà importanti per noi a utilizzare una base diversa dalla decimale, alla quale siamo stati abitutati fin da bambini. Qualcuon sostiene che il motivo della facilità dell'uso della base 10 deriva dal fatto cha abbiamo 10 dita!

puramente tecnologiche, per cui dobbiamo abituarci ad usare la base $N = 2$ per poter apprezzare meglio le proprietà numeriche di un elaboratore elettronico. Nel caso $N = 2$ i coefficienti del polinomio (1.1) assumono i valori $\{0, 1\}$ e quindi è ovvio pensare di usare la base binaria per la rappresentazione dei numeri all'elaboratore.

Vediamo ora, con qualche esempio, come è possibile passare da una base ad un'altra. Dapprima notiamo di nuovo che noi siamo abituati a fare i conti in base $N = 10$, passare dalla base binaria alla base decimale risulta facile.

Esempio 1.2.2. Si trasformi il numero $(10011010010.1)_2$ in base decimale. Usiamo la (1.1) per ottenere:

$$2^{10} + 2^7 + 2^6 + 2^4 + 2^2 + 2^{-1} = 1024 + 128 + 64 + 16 + 2 + 0.5 = 1234.5.$$

Invece è più complicato passare dalla base 10 alla base 2. Vogliamo quindi ricavare una procedura per passare dalla base N alla base $M \neq N$, e cioè

$$\begin{aligned}(a)_N &= a_n N^n + a_{n-1}N^{n-1} + \ldots + a_2 N^2 + a_1 N + a_0 \\ &\quad + a_{-1}N^{-1} + \ldots + a_{-r}N^{-r} = \\ (a)_M &= b_p M^p + b_{p-1}M^{p-1} + \ldots + b_2 M^2 + b_1 M + b_0 \\ &\quad + b_{-1}M^{-1} + \ldots + b_{-s}M^{-s}\end{aligned}$$

Si noti innanzitutto che se $(a)_N$ è esprimibile in base N con un numero finito di cifre, non è detto che sia così per lo stesso numero $(a)_M$ espresso in base M. Riscriviamo ora $(a)_N$ separando la parte intera dalla parte frazionaria [3]:

$$a_{int} = a_n N^n + a_{n-1}N^{n-1} + \ldots + a_2 N^2 + a_1 N + a_0, \tag{1.2}$$

$$a_{fraz} = a_{-1}N^{-1} + \ldots + a_{-r}N^{-r}. \tag{1.3}$$

Concentriamoci dapprima su a_{int}, pensando di lavorare con un'aritmetica in base n. Dividendo a_{int} per il numero M otteniamo:

$$\begin{aligned}a_{int}/M &= (a_n N^n + a_{n-1}N^{n-1} + \ldots + a_2 N^2 + a_1 N + a_0)/M \\ &= a_n/MN^n + a_{n-1}/MN^{n-1} + \ldots + a_2/MN^2 + a_1/MN + a_0/M \\ &= a'_n N^n + a'_{n-1}N^{n-1} + \ldots + a'_2 N^2 + a'_1 N + a'_0 + \mod(a_0, M)),\end{aligned}$$

dove $\mod(a_0, M)$ indica la funzione che restituisce il resto della divisione intera tra a_0 e M, che ovviamente soddisfa alla proprietà $0 \leq \mod(a_0, M) < M$. E' quindi chiaro che tale cifra è, una volta trasformata in binario, la cifra b_0 che cerchiamo.

[3] Si noti che useremo sempre il "punto decimale" per separare la parte intera da quella frazionaria, in accordo con lo standard internazionale. Non useremo invece mai la maldestra abitudine di adattarsi alle convenzioni nazionali.

Continuando a dividere per M si ottengon le altre cifre $b_1, b_2, \ldots, b_p$ in modo analogo. Si noti che generalmente $p \neq n$.

Al contrario, per la parte frazionaria si vede immediatamente che le cifre $b_{-1}, b_{-2}, \ldots$ si possono ottenere moltiplicando successivamente per M la parte frazionaria.

Esempio 1.2.3. Calcolare la rappresentazione in base 2 del numero $a = 1234.5$. Dividiamo la parte intera dalla parte frazionaria. Per la parte intera costruiamo la seguente tabella ottenuta dividendo ogni volta il numero di sinistra per 2:

parte intera	
$12345 : 2 = 6172$	1
$6172 : 2 = 3086$	0
$3086 : 2 = 1543$	0
$1543 : 2 = 771$	1
$771 : 2 = 385$	1
$385 : 2 = 192$	1
$192 : 2 = 96$	0
$96 : 2 = 48$	0
$48 : 2 = 24$	0
$24 : 2 = 12$	0
$12 : 2 = 6$	0
$6 : 2 = 3$	0
$3 : 2 = 1$	1
$1 : 2 = 0$	1

Parte frazionaria

$0.5 \times 2 = 1.0$	1

Il numero in base due viene costruito leggendo le cifre binarie (dal basso per la parte intera, e dall'alto per la parte frazionaria) ottenendo:

$$(a)_2 = 11000000111001.1$$

Esempio 1.2.4. trasformiamo il numero $a = 0.1$ da base 10 a base 2.

$0.1 \times 2 = 0.2$	0
$0.2 \times 2 = 0.4$	0
$0.4 \times 2 = 0.8$	0
$0.8 \times 2 = 1.6$	1
$0.6 \times 2 = 1.2$	1
$0.2 \times 2 = 0.4$	0
$0.4 \times 2 = 0.8$	0
$0.8 \times 2 = 1.6$	1
$0.6 \times 2 = 1.2$	1
$\ldots$	$\ldots$

Il numero $(a)_2$ è dunque un numero periodico pari a $(a)_2 = 0.000\overline{1100}$. Questo è un esempio di un numero che in base 10 (0.1) è caratterizzato da un numero finito di cifre, mentre in base 2 ha infinite cifre.

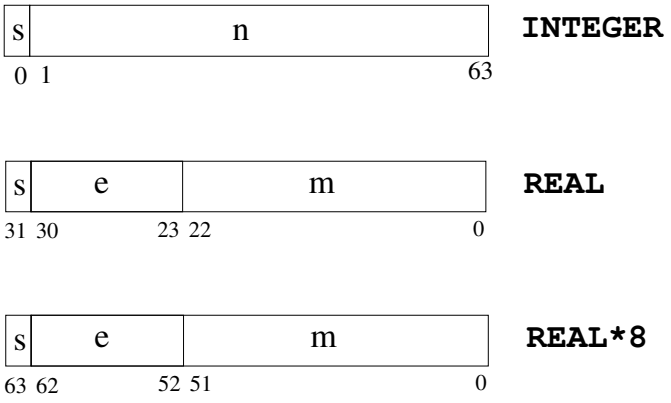

Figura 1.3: Struttura della rappresentazione dei numeri interi (`INTEGER`) e reali (`REAL`, 32 bit=4 byte, o `REAL*8`, 64 bit = 8 byte) secondo lo standard IEEE 754.

1.3 La rappresentazione interna dei numeri all'elaboratore

Questo capitolo è preso dal manuale della SUN Microsystems intitolato Numerical Computation Guide [8], che costituisce una utile e chiara guida di riferimento per chi volesse approfondire l'argomento.

Come abbiamo visto in precedenza, un *word* può essere a 32 o 64 bits, con i 32 bit ormai in uso solo per compatibilità retroattiva. Talchè è necessario predisporre un modello di rappresentazione dei numeri (interi e reali) che usufruisca al meglio della lunghezza di un *word*. Ogni costruttore implementa i propri modelli e non esistono standards universali. Tra tutte le possibilità praticamente usate, lo standard IEEE 754 [1] è considerato da tutte le case e adottato quantomeno su richiesta dell'utente (tramite opportuni flags del compilatore). Per questo motivo descriviamo qui velocemente questo modello ricordando che eventuali variazioni e loro effetti sulla precisione numerica dell'elaboratore in uso vanno verificati di volta in volta.

1.3.1 IEEE 754: numeri interi

Un numero intero è generalmente rappresentato da b=32 o b=64 bits[4] Il numero è quindi una sequenza di bits così fatta. Il primo bit (indicato col simbolo s) è riservato al segno del numero, per cui s=0 indica un numero positivo, s=1 un numero negativo. I rimanenti n (n=31 o n=63) bits sono dedicati al numero stesso (vedi Figura 1.3). E' facile quindi calcolare il valore massimo (in modulo) rappresentabile da questo modello. Tale valore si raggiunge quando i bits da 1 a 31 sono tutti 1, e vale:

$$1 \times 2^0 + 1 \times 2^1 + \ldots + 1 \times 2^{n-1} = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = 2^n - 1 \qquad (1.4)$$

che per n=31 fornisce il valore $I_{max,32} = \pm 2.147.483.648$. E' quindi impossibile la rappresentazione di valori interi maggiori di $I_{max,32}$. Un problema che potrebbe sorgere durante le operazioni con questa rappresentazione è denominato "integer overflow" e purtroppo non dà luogo a segnalazione di errori da parte del computer: l'addizione di due numeri la cui somma è maggiore di $I_{max,32}$ fornisce un risultato sbagliato e imprevedibile. Si consideri per esempio un'aritmetica a 4 bit ($s = 1$ e $n = 3$). Effettuiamo la somma tra il numero 7+2=9 in notazione binaria:

$$\begin{array}{ll} 0111 & + \\ 0010 & = \\ \hline 1001 & \end{array}$$

il cui risultato è -1 secondo la rappresentazione "integer" descritta sopra ($(1001)_2 = (-1)_{10}$). E' quindi opportuno avere sempre presente il valore massimo (o minimo) rappresentabile e lavorare sempre con valori lontani da esso. E' spesso utile lavore con interi a 64 bits (`long long` oppure `integer*8`) quando ci si avvicina al valore $I_{max,32}$ in considerazione del fatto che $I_{max,64} \approx \mathrm{int}(9.2 \times 10^{18})$.

1.3.2 IEEE 754: numeri reali

Nel caso di numeri reali, il modello di rappresentazione (sempre in base binaria) utilizza la notazione in virgola mobile normalizzata. Secondo questa notazione, un numero in base decimale si può scrivere sempre come:

$$(a)_{10} = \pm 0.m \times 10^n$$

[4] Il valore b=32 è spesso indicato `int` nei linguaggi `C` e `C++` , é indicato `integer` in linguaggio `Fortran` . Purtroppo non esiste un vero standard e ogni produttore hardware o software può scegliere liberamente. Per evitare problemi di interoperabilitià, è quindi opportuno dichiarare ogni variabile esplicitamente a seconda che si voglia b=32 o b=64 con le istruzioni `int32` o `int64` in `C` o `C++` e le istruzioni `integer*4` e `integer*8` in `Fortran` .

dove $0.1 \leq m < 1$ e il valore di n viene aggiustato in modo da soddisfare la condizione su m. Le cifre che compongono m individuano la *mantissa* e costituiscono le cifre significative della rappresentazione.

Esempio 1.3.1.

$$(1234.5)_{10} = 0.12345 \times 10^4 \qquad (0.000012345)_{10} = 0.12345 \times 10^{-4}$$

Usando la notazione binaria, un numero $\neq 0$ può essere scritto come:

$$(a)_2 = (-1)^s \times 2^{e-b} \times 1.f$$

dove s è il bit del segno del numero, $e-b$ è l'esponente con deviazione (o *bias*) b, che discuteremo più avanti, e f è la mantissa. Si noti che non si fa l'ipotesi che il numero sia sempre diverso da zero per cui si può evitare di memorizzare la cifra 1 della parte intera, e si aggiusta l'esponente opportunamente. Il bit che rappresenta la cifra 1 non memorizzata si chiama *hidden bit*. I numeri e e f sono memorizzati con formato intero senza segno (sono sempre positivi o nulli) con un numero di cifre diverso tra loro. Facciamo riferimento sempre alla Figura 1.3 per la localizzazione di s, e e f all'interno di una cella di memoria. Mentre ovviamente s vale sempre o zero o 1, bisogna distinguere i casi di *word* a 32 o 64 bit. Nel caso a 32 bit (chiamato anche *singola precisione* o `real*4` per indicare che le celle di memoria sono a 4 bytes) e è costituito da 8 bit (30÷23) e f da 23 bit (22÷0). Nel caso a 64 bit (chiamato *doppia precisione* o `real*8`) e è costituito da 11 bit e f da 53 bit.

Per avere sempre numeri positivi, e non memorizzare quindi il segno dell'esponente, si usa il cosidetto *bias*, che mi rappresenta il centro dell'intervallo di variazione di e. Facendo i conti, nel caso a 32 bit $0 < e < 255$ e $b = 127$, mentre nel caso a 64 bit $0 < e < 2047$ e $b = 1023$. La tabella 1.1 mostra un riassunto schematico di quanto abbiamo descritto finora.

1.3.3 La precisione di macchina

Per completare la descrizione della rappresentazione di un numero all'elaboratore, bisogna considerare che il fatto di avere un numero finito di cifre per memorizzare la mantissa porta necessariamente ad approssimazioni non sempre predicibili in maniera esatta. E' quindi di fondamentale importanza studiare l'errore massimo che si può commettere nella rappresntazione, per poi andare a verificare come questo errore si propaga nelle operazioni.

Abbiamo già visto che le cifre memorizzate formano le "cifre significative". Dobbiamo ora definire la *precisione di macchina* ovvero la *machine epsilon*. Nell'aritmetica dell'IEEE si assume che il numero viene "arrotondato" alla cifra significativa più vicina. La cifra soggetta ad arrotondamento è nella m-esima posizione della

32 bit		64 bit	
intervallo di variazione di e, f, b	Rappresentazione del numero	intervallo di variazione di e, f, b	Rappresentazione del numero
$0 < e < 255$ $b = 127$ $f > 0$	$(-1)^s \times 1.f \times 2^{e-b}$	$0 < e < 2047$ $b = 1023$	$(-1)^s \times 1.f \times 2^{e-b}$
$e = 0$ $f = 0$	$(-1)^s \times 0.0$ (zero con segno)	$e = 0$ $f = 0$	$(-1)^s \times 0.0$
$e = 255$ $s = f = 0$	+INF (infinito positivo)	$e = 2047$ $s = f = 0$	+INF
$e = 255$ $s = 1, f = 0$	-INF (infinito negativo)	$e = 2047$ $s = 1, f = 0$	-INF
$e = 255$ $s = 0, 1, f > 0$	NaN (Not-a-Number)	$e = 2047$ $s = 0, 1, f > 0$	NaN

Numeri "reali" a 32 bit			Numeri "reali" a 64 bit	
Nome	Rapp. interna	decimale	Rapp. interna	decimale
+0	0000 0...0	+0.0	0000 0...0	+0.0
-0	100 0...0	-0.0	1000 0...0	-0.0
1	0100 1111 1000 0...0	1.0	0100 1111 1111 0...0	1.0
2	0100 0...0	2.0	0100 0...0	2.0
N_{max}	0110 1111 0110 1...1	3.402E+38	0110 1111 1110 1...1	1.798e+308
N_{min}	0000 0000 0100 0...0	1.175E-38	0000 0000 0001 0...0	2.225e-308
N_{min}	0000 0000 0100 0...0	1.175E-38	0000 0000 0001 0...0	2.225e-308
$+\infty$	0011 1111 0100 0...0	Infinity	0011 1111 1111 0...0	Infinity
$-\infty$	1111 1111 0100 0...0	Infinity	1111 1111 1111 0...0	Infinity
Not-A-Number	0011 1111 1100 0...0	NaN	0011 1111 1111 0100 0...0	NaN

Tabella 1.1: Tabella in alto: schema riassuntivo dei valori che possono assumere e f e b in diversi casi. Tabella in basso: schema riassuntivo della sequenza binaria per numeri particolari secondo lo standard IEEE. Si veda Figura 1.3 per una rappresentazione grafica.

mantissa e verrà arrotondata a 1 se la $m+1$-esima cifra è 1 o a 0 se la $m+1$-sima cifra è zero. Ne consegue che l'errore massimo che si commette è:

$$\epsilon = \frac{1}{2}2^{-(f-1)}.$$

Usando questa formula, nel caso di singola precisione la precisione di macchina è $\epsilon_{32} = 2^{-24} \approx 5.96e{-}08$ mentre per la doppia abbiamo $\epsilon_{64} = 2^{-53} \approx 1.11e{-}16$. Si noti che nello scrivere questi numeri si è utilizzata la notazione tipicamente informatica detta EXP, per cui il numero $5.96e-08$ equivale a 5.96×10^{-08}. Si dice quindi che in singola precisione si hanno circa 8 cifre decimali significative, mentre in doppia precisione le cifre significative diventano 16.

Per completare la trattazione, bisogna capire quali sono i numeri massimi e minimi rappresentabili in singola e doppia precisione. Per fare questo, dobbiamo calcolare i massimi e i minimi valori assumibili dalla mantissa e dall'esponente nei due casi. Con facili conti si ottiene:

$$\begin{aligned}
|A_{max,32}| &= (2-2^{-23}) \times 2^{-126} \approx 3.40282E+38 \\
|A_{min,32}| &= 2 \times 2^{-127} \approx 1.17549E-38 \\
|A_{max,64}| &= (2-2^{-52}) \times 2^{-1023} \approx 1.7977E+308 \\
|A_{min,64}| &= 2 \times 2^{-1023} \approx 2.2251E-308
\end{aligned}$$

1.4 Algoritmo e schema numerico

Si vuole puntualizzare in questo paragrafo la nomenclatura che si utilizza normalmente in calcolo numerico. In particolare, quando si parla di *schema* numerico si intende un'equazione o un insieme di equazioni che descrivono matematicamente in maniera compiuta le operazioni che devono essere fatte dall'elaboratore elettronico per risolvere un problema matematico. Un *algoritmo* è invece un insieme di istruzioni descritte da formule matematiche che mostrano come uno schema numerico può essere "implementato" in un linguaggio di programmazione.

Per esemplificare meglio cosa intendiamo, si consideri il seguente:

1.4.1 Problema matematico.

Si vuole calcolare l'integrale:

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \, dx, \tag{1.5}$$

per ogni valore di n. Applichiamo il il teorema di integrazione per parti ottenendo:

$$I_n = \frac{1}{e}\left(x^n e^x \|_0^1 - \int_0^1 n x^{n-1} e^x \, dx \right) = 1 - n\frac{1}{e}\int_0^1 x^{n-1} e^x \, dx$$

da cui risulta:

$$I_n = 1 - nI_{n-1} \tag{1.6}$$

Quindi possiamo rappresentare lo schema numerico per il calcolo dell'integrale I_n (eq. (1.5)) con la seguente equazione:

$$\tilde{I}_0 = \frac{e-1}{e} \tag{1.7}$$

$$\tilde{I}_n = 1 - n\tilde{I}_{n-1} \qquad n = 1, 2, \ldots N \tag{1.8}$$

Si noti che in (1.7) abbiamo usato il simbolo $\tilde{I}_n$ per distinguere l'approssimazione numerica dal valore vero I_n.

1.4.2 Algoritmo numerico

Si può quindi pensare il seguente algoritmo:

ALGORITHM INT_INSTABILE

1. Porre $\tilde{I}_0 = (e-1)/e$; inizializzare $NMAX$ (ad es. 20).
2. FOR $k = 1, \ldots, NMAX$

$$\tilde{I}_k = 1 - k\tilde{I}_{k-1}$$

END FOR

1.5 Instabilità e malcondizionamento

Rappresentiamo astrattamente un problema matematico come:

$$\mathcal{F}(x, d) = 0$$

dove $\mathcal{F}$ rappresenta l'insieme di equazioni che rappresentano formalmente il problema, x rappresenta la variabile incognita che deve essere trovata, e d rappresenta l'insieme dei dati.

Uno schema numerico lo rappresentiamo nello stesso modo con:

$$\mathcal{F}_n(\tilde{x}, \tilde{d}) = 0$$

dove $\mathcal{F}_n$ rappresenta l'insieme di equazioni che rappresentano formalmente lo schema, $\tilde{x}$ rappresenta l'approssimazione della soluzione del problema matematico, e $\tilde{d}$ è la rappresentazione numerica dell'insieme dei dati.

Definizione 1.5.1. Si dice che il problema matematico è *ben posto* se ad una variazione dei dati corrisponde una piccola variazione della soluzione

In termini matematici, il problema perturbato si traduce in:

$$\mathcal{F}(x + \delta_x, d + \delta_d) = 0$$

che è quindi *ben posto* se esiste un numero positivo ϵ tale per cui, data una variazione δ_d dei dati, la variazione della soluzione non è maggiore, in valore assoluto, di ϵ, e cioè:

$$\forall \epsilon > 0, \exists \delta(\epsilon) \text{ such that if } |\delta_d| < \delta \text{ then } |\delta_x| < \epsilon$$

1.5.1 Instabilità di uno schema

Definizione 1.5.2. Uno **s**chema si dice *stabile* se gli errori (di rappresentazione o di arrotondamento o di qualsiasi altro tipo) che gli errori commessi all'aumentare del numero delle operazioni dello schema rimangono limitati, e quindi non si accumulano in maniera distruttiva per l'accuratezza del calcolo.

L'analisi numerica si occupa, tra l'altro, di studiare il comportamento degli schemi in modo da verificane le condizioni di stabilità e di accuratezza, in maniera tale da poter avere un controllo sulla bontà della soluzione numerica.

Esempio di schema instabile. Utilizzando lo schema (1.7) in un elaboratore a 32 bit (quindi con rappresentazione in singola precisione con 8 cifre decimali significative), si scopre che l'accuratezza del risultato risulta compromessa già a partire da piccoli valori di n (ad es. con n=6 l'errore è maggiore di 10^-2) e anche con un elaboratore a 64 bit (quindi con rappresentazione in doppia precisione con 16 cifre decimali significative) l'accuratezza degrada fortemente a partire da n=15 (si veda la Tabella 1.2.

E' possibile migliorare la situazione utilizzando uno schema diverso. Esprimiamo I_{n-1} in funzione di I_n ottenendo:

$$I_{n-1} = \frac{1 - I_n}{n} \tag{1.9}$$

Ossevando che

$$\lim_{n\to\infty} I_n = 0$$

si può pensare all'algoritmo seguente:

	I_n	Errore	I_n	Errore	I_n a partire da I_{20}=0	Errore
0	0.63212056	1.17144E-09	0.632120559	9.99201E-16	0.632120559	6.66134E-16
1	0.3679	2.05588E-05	0.367879441	9.99201E-16	0.367879441	6.66134E-16
2	0.2642	4.11177E-05	0.264241118	9.99201E-16	0.264241118	3.33067E-16
3	0.2074	0.000123353	0.207276647	1.9984E-15	0.207276647	5.55112E-17
4	0.1704	0.000493412	0.170893412	7.99361E-15	0.170893412	2.498E-16
5	0.148	0.002467059	0.145532941	3.89966E-14	0.145532941	3.88578E-16
6	0.112	0.014802357	0.126802357	2.32009E-13	0.126802357	4.44089E-16
7	0.216	0.103616496	0.112383504	1.62101E-12	0.112383504	6.93889E-17
8	-0.728	0.828931967	0.100931967	1.2967E-11	0.100931967	4.85723E-16
9	7.552	7.460387707	0.091612293	1.16701E-10	0.091612293	6.77236E-15
10	-74.52	74.60387707	0.083877071	1.16701E-09	0.08387707	6.79595E-14
11	820.72	820.6426478	0.077352216	1.28371E-08	0.077352229	7.47263E-13
12	-9847.64	9847.711773	0.071773408	1.54045E-07	0.071773254	8.96713E-12
13	128020.32	128020.2531	0.0669457	2.00258E-06	0.066947703	1.16572E-10
14	-1792283.48	1792283.543	0.0627602	2.80362E-05	0.062732162	1.63201E-09
15	26884253.2	26884253.14	0.058596998	0.000420543	0.059017565	2.44802E-08
16	-430148050.2	430148050.3	0.062448027	0.006728681	0.055718954	3.91683E-07
17	7312516854	7312516854	-0.061616465	0.114387584	0.052777778	6.65861E-06
18	-1.31625E+11	1.31625E+11	2.109096362	2.058976507	0.05	0.000119855
19	2.50088E+12	2.50088E+12	-39.07283088	39.12055363	0.05	0.002277244
20	-5.00176E+13	5.00176E+13	782.4566175	782.4110726	0	0.04554489

Tabella 1.2: Calcolo dell'integrale (1.6) con lo schema (1.7) con calcoli in singola e doppia precisione e con lo schema (1.9) con calcoli in doppia precisione

ALGORITHM INT_STABILE

1. Porre $I_0 = (e-1)/e$; inizializzare N (ad es. 20); porre $I_N = 0$.
2. FOR $k = N, \ldots, 1$

$$\tilde{I}_{k-1} = \frac{1 - \tilde{I}_k}{k} \tag{1.10}$$

END FOR

3. IF $\left| I_0 - \tilde{I}_0 \right| > \tau$
   aumentare il valore di N
   ripetere il ciclo 2

Per studiare il motivo dell'instabilità dello schema si deve studiare il comportamento dell'errore ad ogni "iterazione", seguendo la definizione di stabilità data in 1.5.2. Per fare questo, definiamo l'errore come la differenza tra la soluzione numerica e la soluzione vera (chiamata anche analitica)[5]:

$$\epsilon_n = \tilde{I}_n - I_n.$$

Chiaramente risulterà

$$\tilde{I}_n = I_n + \epsilon_n,$$

che sostituita nello schema instabile (rappresentato da (1.9)) fornisce:

$$I_n + \epsilon_n = 1 - n(I_{n-1} + \epsilon_{n-1}).$$

Notando quindi che vale la (1.6), la precedente equazione si semplifica in:

$$\epsilon_n = -n\epsilon_{n-1}$$

che fornisce una relazione per l'errore alle diverse iterazioni. Partendo da $n = 0$, si ottiene per induzione:

$$\epsilon_n = (-1)^n n! \epsilon_0,$$

da cui si vede che qualsiasi errore iniziale all'iterazione n viene amplificato di un fattore $n!$, dimostrando quindi che lo schema non verifica la condizione di stabilità 1.5.2.

[5] L'errore sarà sempre definito così, a meno di un segno che però non ci interessa perché a noi interesserà in realtà il valore assoluto dell'errore.

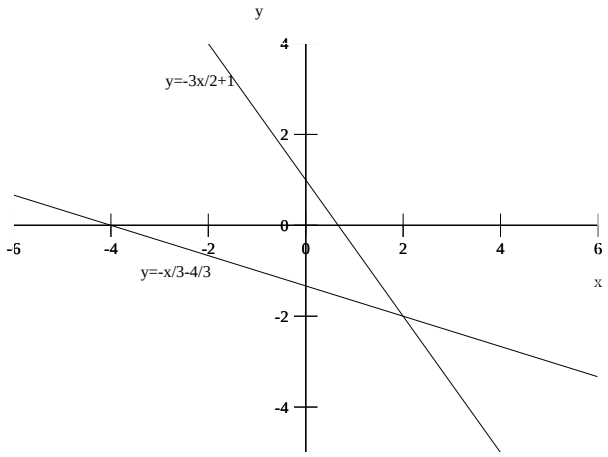

Figura 1.4: Interpretazione geometrica del sistema lineare (1.11) e (1.12)

Lo schema (1.10) risulta invece stabile. Infatti, procedendo nello stesso modo, si ottiene subito la seguente sequenza di errori:

$$\begin{aligned} \epsilon_N &= I_N \\ \epsilon_{N-1} &= -\frac{\epsilon_N}{N} \\ \epsilon_{N-2} &= \frac{\epsilon_N}{N(N-1)} \\ \epsilon_{N-3} &= -\frac{-\epsilon_N}{N(N-1)(N-2)} \\ &\dots \end{aligned}$$

da cui si capisce immediatamente che l'errore iniziale ϵ_N diminuisce all'aumentare delle iterazioni, e quindi soddisfacendo alla definizione di schema stabile.

1.5.2 Problema malcondizionato

Definizione 1.5.3. Un **p**roblema si dice *malcondizionato* se piccole perturbazioni dei dati causano grandi variazioni dei risultati.

Per prima cosa, vogliamo sottolineare che mantre il concetto di stabilità si applica ad uno schema numerico, il concetto di malcondizionamento è una caratteristica di un problema matematico, non di uno schema. Per vedere bene cosa succede ad un

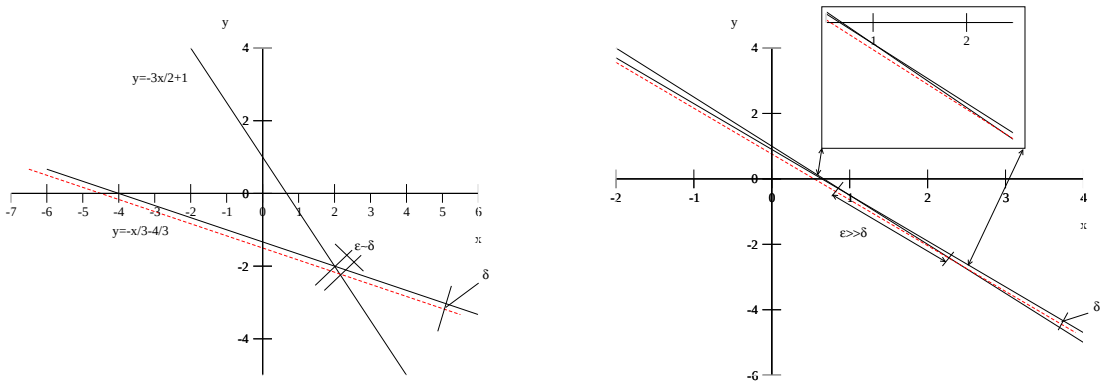

Figura 1.5: Interpretazione geometrica di un sistema lineare ben condizionato (grafico di sinistra) e di uno malcondizionato (grafico di destra)

problema malcondizonato, si pensi ad un sistema lineare di due equazioni in due incognite, ad esempio:

$$3x + 2y = 2 \tag{1.11}$$
$$2x + 6y = -8. \tag{1.12}$$

Il problema matematico è quindi quello di trovare il punto (x, y) tale che le equazioni (1.11) e (1.12) sono simultanemante soddisfatte. La soluzione di tale problema è $(x, y) = (2, -2)$. Questo sistema si può riscrivere dividendo la prima equazione per 2 e la seconda per 6:

$$y = -\frac{3}{2}x + 1 \tag{1.13}$$
$$y = -\frac{1}{3}x - \frac{4}{3}. \tag{1.14}$$

Questo sistema può essere interpretato geometricamente come il problema di trovare il punto di intersezione delle due rette rappresentate dalle equazioni (1.13) e (1.14), come si vede in Figura 1.4.

Proviamo ora a dare una perturbazione ai dati del nostro problema e vediamo come varia la soluzione. Nel piano (x, y) questo si traduce nel perturbare per esempio il termine noto della seconda equazione di un valore δ, quindi ottenendo una traslazione rigida verso il basso della retta, e vediamo che il punto di intersezione delle due rette si sposta di un valore $\epsilon \approx \delta$. Invece, se le due rette hanno pendenze non molto diverse tra di loro, rappresentate da sistema lineare ovviamente diverso, si vede che ad una perturbazione δ corrisponde uno spostamento della soluzione molto

grande ($\epsilon \gg \delta$). Questo comportamento è tipico dei problemi malcondizionati, ed è visualizzato in figure 1.5.

Capitolo 2

La soluzione di equazioni nonlineari

Data una funzione algebrica o trascendente f che per semplicità consideriamo assuma valori in $\mathbb{R}$, un'equazione nonlineare prende la forma di

$$f(x) = 0 \tag{2.1}$$

Il numero ξ tale che $f(\xi) = 0$ si dice soluzione o radice di questa equazione. Ci possono essere più radici di una equazione, cioè dati $\xi_1, \xi_2, \ldots, \xi_n$, si avrà $f(\xi_1) = 0; f(\xi_2) = 0; \ldots; f(\xi_n) = 0$. In generale, la radice di una equazione, anche nelle condizioni più favorevoli di continuità della funzione f, non può essere trovata analiticamente se non in casi particolari (si pensi alle radici di polinomi di grado maggiore di 4). E' quindi necessario ricorrere a tecniche numeriche. In questo caso però non sarà possibile trovare la radice esatta, ma a causa della limitatezza della rappresentabilità dei numeri reali all'elaboratore, e a causa degli errori di arrotondamento si può pensare di trovare solamente un'approssimazione alla soluzione vera ξ. Si dovrà quindi cercare di trovare quel numero $\hat{x}$ che più si avvicina a ξ. In altri termini, fissata l'accuratezza desiderata, e cioè fissata una *tolleranza tol*, vogliamo trovare quel numero $\hat{x}$ che approssimi la radice a meno di *tol*:

$$|\xi - \hat{x}| < tol$$

La grandezza $\epsilon = |\xi - \hat{x}|$ è chiamata l'*errore*. La conoscenza dell'errore richiede conoscenza della radice ξ. L'errore non ha quindi valore pratico, ma è molto utile nello studio delle proripetà degli schemi che andremo a studiare. Al posto della condizione sull'errore potremmo richiedere che:

$$|f(\hat{x})| < tol$$

e cioè che il *residuo nonlineare* sia minore della tolleranza. Ovviamente quest'ultima condizione non garantisce che la soluzione numerica $\hat{x}$ approssimi effettivamente la

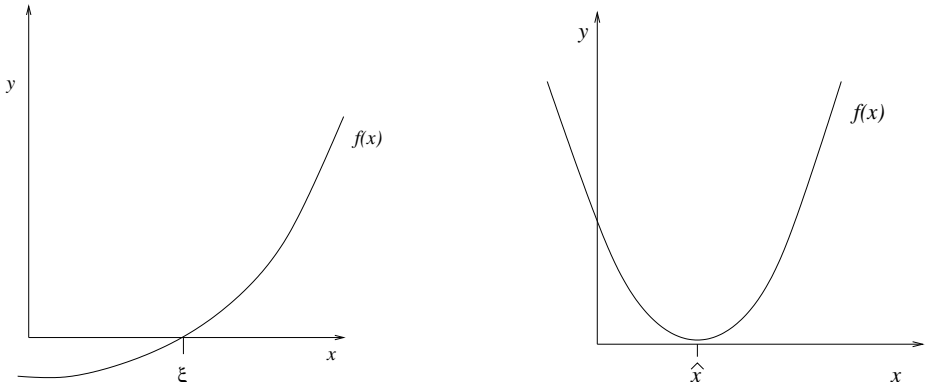

Figura 2.1: Grafico di un una funzione $f(x)$ che ammette radice ξ (sinistra) e che non ammette alcuna radice (destra).

radice, ma rappresenta solamente una condizione necessaria. Si veda per esempio la figura 2.1 in cui nel punto $\hat{x}$ la condizione precedente potrabbe essere verificata per opportuni valori di *tol* ma la $f(x)$ non ammette alcuna radice in $\mathbb{R}$. Specificheremo meglio nei paragrafi che seguono questi concetti. In questa trattazione assumiamo che $f : \mathbb{R} \to \mathbb{R}$ sia una funzione continua e opportunamente derivabile.

2.1 Prime prove

PROBLEMA: trovare un'approssimazione numerica delle radici dell'equazione

$$x^2 - 3x + 2 = 0 \tag{2.2}$$

Le radici vere di tale equazione sono facilmente calcolabili:

$$\xi_{1,2} = \frac{3 \pm \sqrt{9-8}}{2} \qquad \xi_1 = 1 \qquad \xi_2 = 2$$

Per calcolare un'approssimazione della radice ξ_2 con un metodo numerico procediamo come segue. Esplicitiamo la x dalla (2.2), per esempio:

$$x = g_1(x) = \sqrt{3x-2} \tag{2.3}$$

Tale equazione ha evidentemente le stesse radici. Utilizzando una calcolatrice tascabile è facile quindi costruire una tabella in cui dato un valore di partenza x_0 il nuovo valore x_1 è calcolato valutando la funzione g_1 nel punto x_0 ($x_1 = g_1(x_0)$). Nella tabella seguente si riportano i risultati del procedimento partendo da $x_0 = 3$ e ripetendo il procedimento per 7 volte:

	x	$g_1(x)$
1	3	$\sqrt{7}$=2.6458
2	2.6458	2.4367
3	2.4367	2.3043
4	2.3043	2.2165
5	2.2165	2.1563
6	2.1563	2.1140
7	2.1140	2.0837

Si noti che nella prima colonna vi sono i valori della $g_1(x)$ calcolati nella riga precedente. I valori della seconda colonna approssimano via via sempre meglio il valore della radice $\xi_2 = 2$. Si dice che questo procedimento **converge** alla soluzione vera ξ_2, o in altri termini $x_{k+1}(= g(x_k)) \to \xi_2$, ove il pedice k indica le varie righe della tabella, chiamate anche **iterazioni**.

Proviamo ora a cambiare la funzione $g(x)$ esplicitando il termine lineare della (2.2) anzichè il termine quadratico. Otteniamo:

$$x = g_2(x) = \frac{x^2 + 2}{3} \tag{2.4}$$

e costruiamo la stessa tabella a partire ancora da $x_0 = 3$ ma usando ora la funzione $g_2(x)$:

	x	$g_2(x)$
1	3	3.6667
2	3.6667	5.1481
3	5.1481	9.5011

Come si verifica immediatamente il procedimento questa volta costruisce un'approssimazione che si allontana sempre più dalla soluzione ξ_2. Si dice in questo caso che il procedimento **diverge**.

Proviamo infine ad usare una terza funzione $g_3(x)$ data da:

$$x = g_3(x) = \frac{x^2 - 2}{2x - 3} \tag{2.5}$$

Si noti che tale equazione ha come radici esattamente ξ_1 e ξ_2, come si può facilmente vedere sostituendo ad ambo i membri una delle due radici arrivando così all'identità. Costruiamo la stessa tabella di prima:

	x	$g_3(x)$
1	3	2.333333
2	2.333333	2.066667
3	2.066667	2.003922
4	2.003922	2.000015

e otteniamo una ottima approssimazione già alla quarta iterazione. Ci si domanda quindi come si fa a costruire uno schema che converga alla soluzione esatta, e,

una volta costruito lo schema, come si fa a vedere quanto velocemente converge. Collegato a questo c'è da domandarsi come si fa a decidere quante iterazioni fare, o in altri termini, qual'è l'accuratezza dei conti alla quale ci fermiamo.

Per dare un senso pratico a queste domande, costruiamo quella che si chiama la tabella degli errori ϵ, ove l'errore è definito intuitivamente dalla differenza tra la soluzione approssimata x_k e la soluzione vera ξ_2:

	ϵ	$g_1(\epsilon)$
1	1	0.6458
2	0.6458	0.4367
3	0.4367	0.3043
4	0.3043	0.2165
5	0.2165	0.1563
6	0.1563	0.1140
7	0.1140	0.0837

	ϵ	$g_2(\epsilon)$
1	1	1.6667
2	1.6667	3.1481
3	3.1481	7.5011

	ϵ	$g_3(\epsilon)$
1	1	0.333333
2	0.333333	0.066667
3	0.066667	0.003922
4	0.003922	0.000015

Dalle tabelle si nota che gli errori tendono a zero nello stesso modo con cui la soluzione approssimata tende alla soluzione vera. Nel primo caso (g_1) possiamo facilmente verificare che il rapporto di riduzione dell'errore ad ogni iterazione è pari a circa 0.73 (i.e. $\epsilon_7/\epsilon_6 \approx \epsilon_6/\epsilon_5 \approx 0.73$) e cioè:

$$\epsilon_6 \approx 0.73\epsilon_5 \qquad \text{e} \qquad \epsilon_7 \approx 0.73\epsilon_6$$

Nel caso di g_3 invece tale rapporto tende a zero mentre è una costante il rapporto fra l'errore corrente e il quadrato dell'errore precedente: $\epsilon_4/\epsilon_3^2 \approx \epsilon_3/\epsilon_2^2 \approx 1$, che significa:

$$\epsilon_3 \approx \epsilon_2^2 \qquad \text{e} \qquad \epsilon_4 \approx \epsilon_3^2$$

Per vedere meglio perché succede così calcoliamo la derivata prima della funzione g e la valutiamo nella radice:

$$\begin{aligned} g_1'(x) &= \left.\frac{3}{2\sqrt{3x-2}}\right|_{x=2} = 3/4 \qquad (<1) \\ g_2'(x) &= \left.\frac{2}{3}x\right|_{x=2} = 4/3 \qquad (>1) \\ g_1'(x) &= \left.\frac{2x^2-6x+4}{2x-3}\right|_{x=2} = 0 \qquad (<1) \end{aligned}$$

Empiricamente possiamo quindi affermare che ove la derivata prima di g è minore di 1 lo schema converge. Inoltre, $g_1'(\xi_2)$ è una buona approssimazione del fattore di riduzione dell'errore mentre ciò non è vero nel caso di g_3, ove si ha una riduzione dell'errore quadratica e una convergenza molto più veloce.

Per quanto riguarda invece il numero di iterazioni dopo il quale fermarsi, bisogna anche qui ragionare sulle tabelle degli errori. In un problema pratico uno ha già in

mente qual'è l'errore che è disposto ad accettare. In questo caso basta fissare un limite (tolleranza) per l'errore e iterare fin a che l'errore (in valore assoluto) diventa inferiore a questo limite. Per esempio se fissiamo una tolleranza $TOLL = 0.001$, pochi conti con la calcolatrice tascabile mostrano che con la g_1 ci vogliono almeno 23 iterazioni per soddisfare tale criterio di arresto ($|\epsilon_{23}| < TOLL$), mentre per g_3 il criterio è soddisfatto già alla quarta iterazione ($|\epsilon_4| < TOLL$).

Un altro modo di porsi il problema è quello di chiedersi qual'è il valore della differenza tra due soluzioni successive al di sotto del quale le due approssimazioni possono essere considerate equivalenti. In questo caso il controllo non è più fatto utilizzando l'errore ma lo scarto (la differenza tra due approssimazioni successive). Questo modo di lavorare è in realtà l'unico disponibile nella pratica, poiché la conoscenza dell'errore richiederebbe la conoscenza della soluzione vera.

2.2 Lo schema delle iterazioni successive (o di Picard)

Lo schema visto in precedenza è un ben noto algoritmo che prende il nome di schema delle iterazioni successive o di Picard. Si costruisce una successione di approssimanti della radice utilizzando la seguente formula ricorrente:

$$x_{k+1} = g(x_k) \tag{2.6}$$

dove l'indice k viene chiamato iterazione.

Lo schema precedente risolve il classico problema del "punto fisso", e cioè trovare il valore della $x \in \mathbb{R}$ tale che

$$x = g(x) \tag{2.7}$$

Graficamente questo problema consiste nel trovare il punto di intersezione ξ tra la retta $y = x$ e la curva $y = g(x)$, come evidenziato in Figura 2.2, ove si vede anche la convergenza dello schema di Picard.

Un algoritmo per lo schema di Picard implementabile all'elaboratore può essere descritto nel modo seguente. Si noti che nello scrivere un algoritmo utilizzeremo un linguaggio simile ad un linguaggio di programmazione ma utile per una descrizione "matematica" dell'algoritmo. Alcune notazioni sono tipiche della scrittura degli algoritmi. Per esempio il simbolo := denota il concetto di assegnazione, per cui prima si valuta il valore che l'espressione a destra del simbolo assume e solo successivamente tale valore viene assegnato alla variabile a sinistra del simbolo. La "keyword" FINCHÉ individua quello che in gergo informatico si chiama ciclo WHILE, per cui si ripete in successione il gruppo di righe comprese tra la "keyword" FINCHÉ e quella

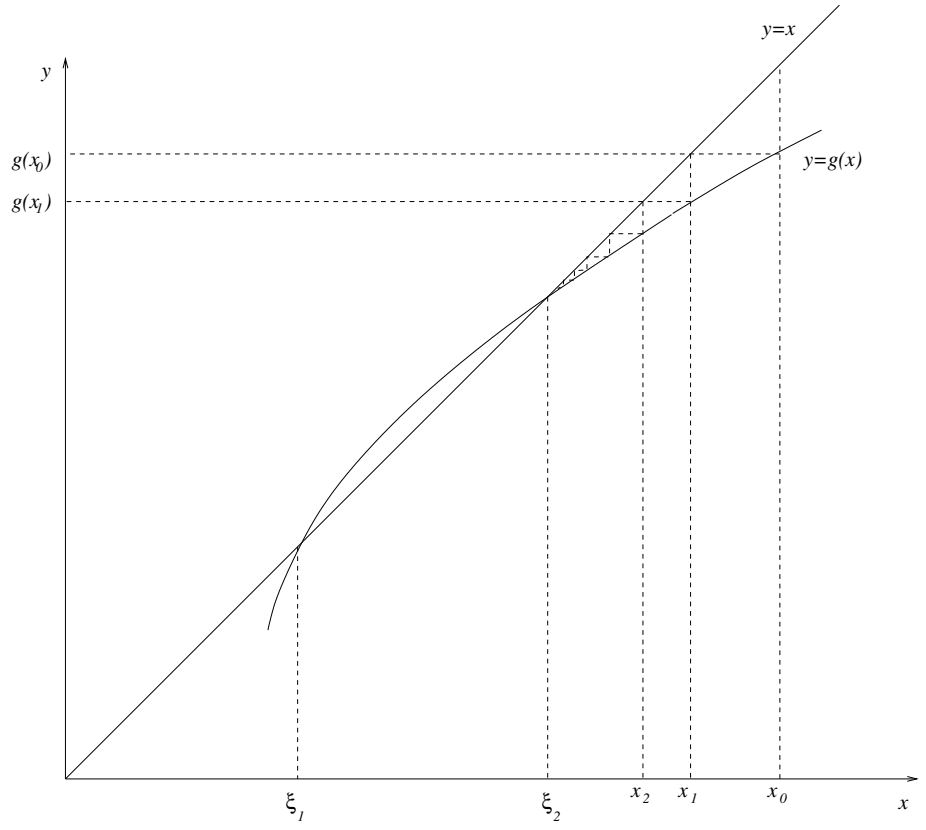

Figura 2.2: Rappresentazione grafica del problema del punto fisso e dello schema di Picard per la soluzione dell'equazione $x^2 - 3x + 2 = 0$ con funzione di punto fisso $g_1(x) = \sqrt{3x - 2}$.

FINE FINCHÉ se la condizione scritta tra parentesi subito a destra del FINCHÉ è soddisfatta. Non appena tale condizione non è verificata, si esce dal ciclo e si continua con le istruzioni che seguono la "keyword" di chiusura ciclo (FINE FINCHÉ).

ALGORITMO DI PICARD I:

data una soluzione iniziale x_0;

PER $k = 1, 2, \ldots, s$ fino a convergenza:

1. $x_{k+1} = g(x_k)$

Questo algoritmo non è però completamente implementabile, nel senso che bisogna specificare più dettagliatamente cosa significa la frase "fino a convergenza". Per fare ciò, come abbiamo visto, abbiamo bisogno di introdurre il concetto di scarto e di tolleranza. L'algoritmo seguente implementa lo schema di Picard in modo del tuto simile a quanto si fa con un linguaggio di programmazione:

ALGORITMO DI PICARD II:

data una soluzione iniziale x_0 e una tolleranza $TOLL$;

$XK := x_0$; $SCARTO := 2 * TOLL$

FINCHÉ $SCARTO > TOLL$ esegui:

1. $XKP1 = g(XK)$
2. $SCARTO = |XKP1 - XP|$
3. $XK = XKP1$

FINE FINCHÉ.

Si noti che senza l'istruzione 3 l'algoritmo non uscirebbe mai dal ciclo FINCHÉ. Infatti $XKP1$ sarebbe sempre uguale a $g(XK)$ visto che $XK = x_0$ non verrebbe mai aggiornata. Un ulteriore controllo è però necessario: nel caso in cui il metodo diverge, l'algoritmo implementa un ciclo infinito visto che $SCARTO$ aumenta ad ogni iterazione. In questo caso il computer eseguirebbe le iterazioni per un numero molto grande di iterazioni fino a che probabilmente il valore di $SCARTO$ o di $XKP1$ non diventi talmente grande da superare la capacità di rappresentazione dei numeri reali dell'elaboratore (in gergo informatico si avrebbe un "overflow"). Per guardarsi da una tale evenienza, bisogna contare le iterazioni che vengono fatte e verificare che

esse non superino un numero massimo ($ITMAX$) prestabilito. Il seguente algoritmo implementa anche questo controllo:

ALGORITMO DI PICARD III:

data una soluzione iniziale x_0, una tolleranza $TOLL$ e un numero massimo di iterazioni $ITMAX$;

$XK := x_0$; $SCARTO := 2 * TOLL$; $ITER = 0$;

FINCHÉ ($SCARTO > TOLL$ e $ITER < ITMAX$) esegui:

1. $ITER := ITER + 1$;
2. $XKP1 := g(XK)$;
3. $SCARTO := |XKP1 - XP|$;
4. $XK := XKP1$;

FINE FINCHÉ.

2.3 Convergenza dei metodi iterativi

Lo studio della convergenza degli schemi iterativi è un capitolo molto importante del Calcolo Numerico e serve per trovare le condizioni che garantiscono un funzionamento corretto degli algoritmi in maniera tale da ottenere in un tempo ragionevole una buona approssimazione numerica della soluzione del problema considerato.

Nel caso di metodi iterativi, come quello di Picard precedentemente descritto, lo studio della convergenza riguarda essenzialmente lo studio della propagazione dell'errore da un'iterazione all'altra. E' utile cercare di rendere i concetti di convergenza in termini matematici, anche se non si pretende qui di riportare una trattazione formale dell'argomento, ritenendo sufficiente dare alcune spiegazioni intuitive che però sono fondamentali per la comprensione dei metodi numerici.

2.3.1 Studio della convergenza dello schema di Picard

Uno schema iterativo può essere sempre pensato come un insieme di regole per costruire una successione di numeri reali che tende alla soluzione vera del problema. In termini matematici, indichiamo la convergenza con:

$$\{x_k\} \to \xi$$

dove x_k denota la successione e ξ la soluzione analitica del problema matematico. Definiamo l'errore come la differenza tra la soluzione vera e la soluzione approssimata:

$$\epsilon_k = \xi - x_k \tag{2.8}$$

La successione $\{x_k\}$ converge alla radice ξ se $\{\epsilon_k\}$ tende a zero, e cioè:

$$\lim_{k\to\infty} |\epsilon_k| = 0 \tag{2.9}$$

Capire quanto velocemente questa successione converge a zero, significa anche capire quanto velocemente la successione x_k converge alla soluzione vera ξ. Idealmente si vorrebbe che ad ogni iterazione k l'errore diminuisse, ma questo non sempre è vero. Se l'errore diminuisce di un fattore costante ad ogni iterazione, si può scrivere:

$$|\epsilon_{k+1}| \leq M|\epsilon_k| \qquad k = 1, 2, \ldots \qquad\qquad M < 1$$

Condizioni per la convergenza

Per studiare le condizioni di convergenza, bisogna quindi studiare in quali condizioni la condizione (2.9) è verificata. Dall'equazione (2.6) e dal fatto che la soluzione vera ξ soddisfa la (2.7), si ottiene immediatamente:

$$\xi - x_1 = g(\xi) - g(x_0) = g'(\xi_0)(\xi - x_0)$$

ove l'ultima espressione deriva dall'applicazione del teorema del valor medio. Continuando, possiamo evidentemente scrivere:

$$\begin{aligned} \xi - x_1 &= g(\xi) - g(x_0) = g'(\xi_0)(\xi - x_0) \\ \xi - x_2 &= g(\xi) - g(x_1) = g'(\xi_1)(\xi - x_1) = g'(\xi_1)g'(\xi_0)(\xi - x_0) \end{aligned}$$

e quindi si verifica facilmente che:

$$\epsilon_k = g'(\xi_{k-1})g'(\xi_{k-2}) \cdots g'(\xi_1)g'(\xi_0)\epsilon_0 \tag{2.10}$$

Ponendo

$$m = \max_j |g'(\xi_j)|$$

otteniamo la seguente maggiorazione per l'errore alla k-esima iterazione:

$$|\epsilon_k| \leq m|\epsilon_{k-1}| \leq m^k|\epsilon_0| \tag{2.11}$$

da cui si ricava che se $m < 1$ la condizione (2.9) è verificata. Bisogna stare attenti perché questa dimostrazione mi dice che se tutte le derivate nei punti opportuni ξ_k interni all'intervallo $[x_0, \xi]$ sono in valore assoluto minori di uno certamente lo schema

converge (condizione sufficiente). E' peraltro vero che se una di queste derivate risulta avere modulo maggiore di 1, nondimeno lo schema può ancora convergere. Inoltre, il punto x_0 non è stato specificato, per cui la condizione di convergenza per lo schema di Picard risulta essere:

$$|g'(x)| < 1 \qquad \text{per} \quad x \in I_\xi \tag{2.12}$$

ove I_ξ indica un intorno del punto ξ. Se la g' è una funzione continua, e

$$|g'(\xi)| < 1, \tag{2.13}$$

esiste certamente un intorno di ξ per cui la (2.13) è verificata. Dalla Figura 2.2 si nota $|g'(\xi_2)| < 1$ mentre $|g'(\xi_1)| > 1$, come si può facilmente vedere confrontando le pendenze delle tangenti nei punti ξ_1 e ξ_2 con la pendenza della retta $y = x$.

Relazione tra scarto e errore

Possiamo ora dare una giustificazione all'uso dello scarto al posto dell'errore nel controllo del ciclo WHILE dell'algoritmo di Picard. Definiamo quindi lo scarto come la differenza tra le soluzioni a due iterate successive:

$$e_k = x_k - x_{k-1}. \tag{2.14}$$

Possiamo quindi scrivere:

$$\epsilon_k = x_k - \xi = x_k - x_{k+1} + x_{k+1} - \xi,$$

da cui, prendendo i valori assoluti, si ottiene:

$$|\epsilon_k| \leq |x_{k+1} - x_k| + |\xi - x_{k+1}| = |e_k| + |\epsilon_{k+1}|. \tag{2.15}$$

Dalla (2.10) possiam esprimere ϵ_k in funzione di ϵ_{k+1}, ottenendo quindi:

$$|\epsilon_k| \geq \frac{1}{m}|\epsilon_{k+1}|,$$

che sostituita in (2.15), fornisce:

$$\frac{1}{m}|\epsilon_{k+1}| \leq |\epsilon_k| \leq |\epsilon_{k+1}| + |e_{k+1}|,$$

che è valida per ogni valore di k, e quindi:

$$|\epsilon_k| \leq \frac{m}{1-m}|e_k|. \tag{2.16}$$

Questa relazione mi dice che lo scarto si comporta asintoticamente (per k grande) come l'errore e quindi può essere usato come surrogato dell'errore nell'algoritmo di Picard.

Se $m \leq 1/2$, allora lo scarto è sempre una maggiorazione dell'errore.

Ordine di convergenza

E' facile constatare, guardando la (2.10), che l'errore alla k iterazione tenderà a zero tanto più velocemente quanto più il prodotto delle derivate sarà vicino allo zero. Per quantificare in maniera più precisa questo concetto, prendiamo l'errore cambiato di segno[1] per cui

$$x_k = \xi + \epsilon_k \tag{2.17}$$

che inserita nella (2.6) fornisce:

$$\xi + \epsilon_{k+1} = g(\xi + \epsilon_k)$$

Se lo schema converge $\{\epsilon_k\} \to 0$ per cui possiamo espandere in serie di Taylor la g attorno a ξ, ottenendo:

$$\xi + \epsilon_{k+1} = g(\xi) + \epsilon_k g'(\xi) + \frac{\epsilon_k^2}{2} g''(\xi) + \frac{\epsilon_k^3}{6} g'''(\xi) + \cdots$$

da cui, ricordandosi ancora una volta che $\xi = g(\xi)$:

$$\epsilon_{k+1} = \epsilon_k g'(\xi) + \frac{\epsilon_k^2}{2} g''(\xi) + \frac{\epsilon_k^3}{6} g'''(\xi) + \cdots \tag{2.18}$$

Si vede subito che, essendo ϵ_k^2 e ϵ_k^3 infinitesimi di ordine superiore, il termine dominante è proporzionale a $|\epsilon_k|$ con costante di proporzionalità pari proprio a $g'(\xi)$. E' chiaro inoltre che perché $|\epsilon_{k+1}| < |\epsilon_k|$ dovrà essere $|g'(\xi)| < 1$. Se $g'(\xi) \neq 0$, si ha immediatamente:

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|} = \lim_{k\to\infty} \left| g'(\xi) + \frac{\epsilon_k}{2} g''(\xi) + \frac{\epsilon_k^2}{6} g'''(\xi) + \cdots \right| = |g'(\xi)|$$

mentre se $g'(\xi) = 0$ si potrà scrivere:

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k^2|} = \lim_{k\to\infty} \left| \frac{1}{2} g''(\xi) + \frac{\epsilon_k}{6} g'''(\xi) + \cdots \right| = \left| \frac{1}{2} g''(\xi) \right|$$

e in questo caso sono i termini di secondo ordine a comandare la convergenza, che risulta essere quindi più veloce. E' quindi naturale definire l'ordine p di convergenza e la costante M asintotica di convergenza (o dell'errore) con la relazione:

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^p} = M \tag{2.19}$$

Per $p = 1$ la convergenza è anche detta lineare e necessariamente $M < 1$; per $p = 2$ la convergenza è detta quadratica, e così via. Si noti nel caso della $g_1(x)$ studiata in precedenza si ha $p = 1$ e $M = |g_1'(2)| = 0.75$, mentre per la $g_3(x)$ si ha $p = 2$ e $M = g_3''(\xi)/2 = 1$.

[1] in questo caso ci interessano esclusivamente quantità prese in valore assoluto

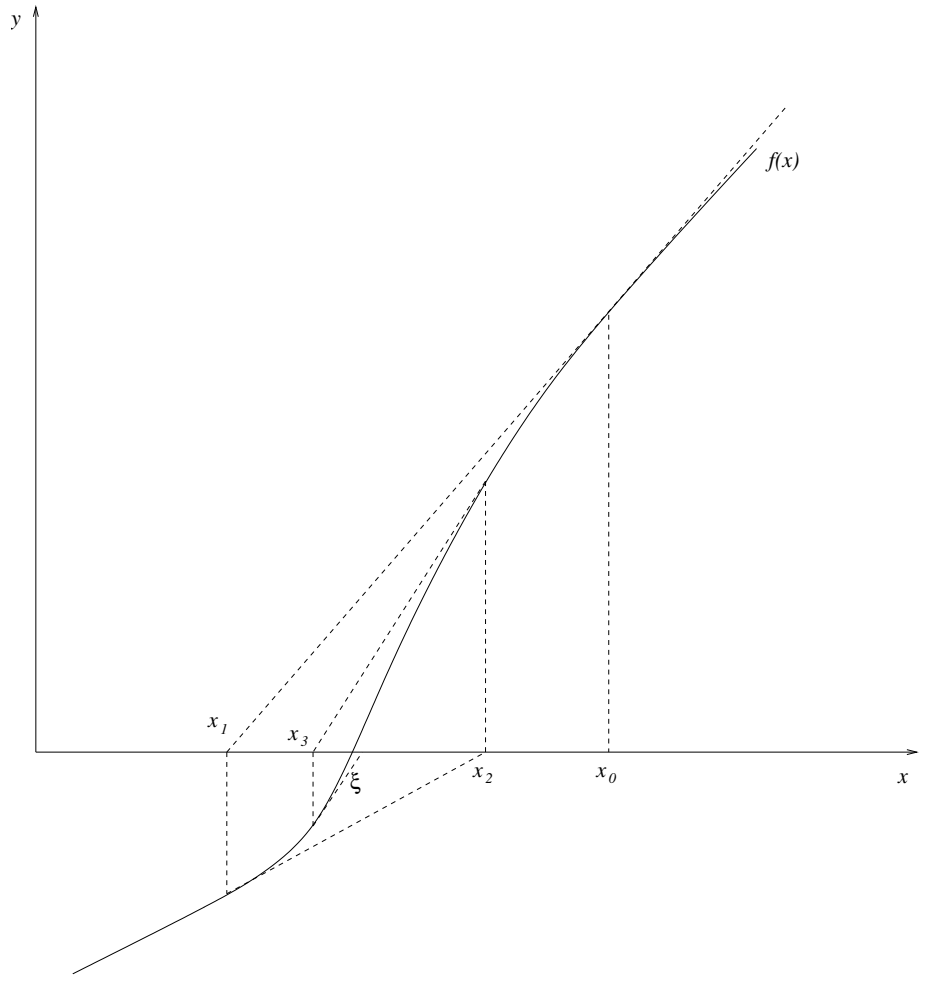

Figura 2.3: Rappresentazione grafica del funzionamento del metodo di Newton-Raphson per la soluzione dell'equazione $f(x) = 0$.

2.3.2 Lo schema di Newton-Raphson

Abbiamo visto che se $|g'(\xi)| = 0$ lo schema di Picard ha ordine di convergenza $p = 2$. Cerchiamo quindi di trovare un modo per ricavare tale schema. Partiamo dal problema (2.1) e cerchiamo di costruire una successione definita da:

$$x_{k+1} = x_k + h \tag{2.20}$$

Idealmente noi vorremmo che, partendo da una soluzione di tentativo x_k, il valore di h ci fornisca la soluzione esatta, o in formule:

$$f(x_{k+1}) = f(x_k + h) = 0$$

Espandendo in serie di Taylor la precedente equazione attorno a x_k si ottiene:

$$0 = f(x_k + h) = f(x_k) + hf'(x_k) + \frac{h^2}{2}f''(x_k) + \frac{h^3}{3!}f'''(x_k) + \cdots$$

Trascurando i termini di ordine superiore al primo, si ottiene un'equazione lineare in h la cui soluzione, inserita nella (2.20) definisce lo schema iterativo di Newton-Raphson, che può essere visto come uno schema di punto fisso in cui si utilizza una $g(x)$ particolare:

$$x_{k+1} = g_{nr}(x_k) = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2.21}$$

Si vede facilmente che il valore approssimato x_{k+1} alla nuova terazione di Newton-Raphson è dato dall'intersezione della retta tangente alla $f(x)$ nel punto $(x_k, f(x_k))$, che ha dunque pendenza pari a $f'(x_k)$, con l'asse x, come evidenziato in Figura 2.3.

L'algoritmo per l'implementazione dello schema di Newton-Raphson è molto simile a quello di Picard, con la sostituzione della generica $g(x)$ con la $g_{nr}(x)$:

ALGORITMO DI NEWTON-RAPHSON:

data una soluzione iniziale x_0, una tolleranza $TOLL$ e un numero massimo di iterazioni $ITMAX$;

$XK := x_0$; $SCARTO := 2 * TOLL$; $ITER = 0$;

FINCHÉ ($SCARTO > TOLL$ e $ITER < ITMAX$) esegui:

1. $ITER := ITER + 1$;
2. $XKP1 := g_{nr}(XK)$;
3. $SCARTO := |XKP1 - XP|$;
4. $XK := XKP1$;

FINE FINCHÉ.

...

FUNCTION $g_{nr}(X)$;

$$g_{nr}(X) = X - \frac{f(X)}{f'(X)}$$

FINE FUNCTION g_{nr}

Condizioni di convergenza

Consideriamo la funzione di Newton-Raphson g_{nr} e imponiamo le condizioni di convergenza dello schema di punto fisso, e, osservando che se ξ è lo zero della funzione allora necessariamente $f(\xi) = 0$ si ottiene:

$$|g'_{nr}(\xi)| = \left|1 - \frac{[f'(\xi)]^2 - f(\xi)f''(\xi)}{[f'(\xi)]^2}\right| = \left|\frac{f(\xi)f''(\xi)}{[f'(\xi)]^2}\right| = 0$$

da cui segue immediatamente che la (2.12) è sempre verificata se si assume la continuità della f e delle sue prime due derivate. Si dice quindi che lo schema di Newton-Raphson è *generalmente* convergente. Si noti, tuttavia, che bisogna stare attenti a queste affermazioni, perché il punto iniziale x_0 non è stato specificato e la (2.12) vale in condizioni asintotiche. Si può affermare quindi che lo schema di Newton-Raphson è generalmente convergente per una soluzione iniziale x_0 sufficientemente vicina.

Nel caso di radice doppia ($f(\xi) = 0; f'(\xi) = 0; f''(\xi) \neq 0$) la condizione di convergenza è ancora soddisfatta, anche se la g'_{nr} non è piú nulla. Infatti si ha:

$$\begin{aligned}\lim_{x\to\xi} |g'_{nr}(x)| &= \lim_{x\to\xi}\left|\frac{f(x)f''(\xi)}{[f'(x)]^2}\right| = \text{(usando l'Hospital)} = \\ &= |f''(\xi)| \lim_{x\to\xi}\left|\frac{f'(x)}{2f'(x)f''(x)}\right| = \frac{1}{2}\end{aligned}$$

E' quindi chiaro dalla (2.18) che in questo caso l'ordine di convergenza si riduce a lineare con costante asintotica dell'errore $M = 1/2$. Possiamo dunque concludere che lo schema di Newton-Raphson è generalmente convergente, sempre pero' in relazione ad una soluzione iniziale x_0 sufficientemente vicina alla soluzione finale ξ.

Ordine di convergenza

Come abbiamo già notato indirettamente, nel caso generale l'ordine di convergenza dello schema di Newton-Raphson è $p = 2$ con costante asintotica dell'errore $M = \frac{1}{2}|g''_{nr}(\xi)|$. E' possibile però ricavare l'ordine direttamente dall'espressione (2.21) con la stessa procedura usata per lo schema di punto fisso. A tal fine, sostituiamo la (2.17) nella (2.21):

$$\xi + \epsilon_{k+1} = \xi + \epsilon_k - \frac{f(\xi + \epsilon_k)}{f'(\xi + \epsilon_k)}$$

Espandendo sia la $f(\xi + \epsilon_k))$ del numeratore e la $f'(\xi + \epsilon_k)$ del denominatore in serie di Taylor attorno ad x_k, notando che $f(\xi) = 0$ e assumendo che la radice sia singola,

si ottiene:

$$\begin{aligned}\epsilon_{k+1} &= \epsilon_k - \frac{f(\xi) + \epsilon_k f'(\xi) + \epsilon_k^2/2 f''(\xi) + \epsilon_k^3/3 f'''(\xi) \cdots}{f'(\xi) + \epsilon_k f''(\xi) \cdots} \\ &= \frac{\epsilon_k^2/2 f''(\xi) + \epsilon_k^3/3 f'''(\xi) \cdots}{f'(\xi) + \epsilon_k f''(\xi) \cdots} \end{aligned} \tag{2.22}$$

per cui la definizione (2.19) è soddisfatta con $p = 2$ e $M = 1/2|f''(\xi)/f'(\xi)|$:

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^2} = \frac{1}{2}\left|\frac{f''(\xi)}{f'(\xi)}\right|$$

che conferma l'ordine "quadratico" di Newton-Raphson e dalla quale si può ricavare indirettamente l'espressione della g''_{nr}.

Nel caso di radice doppia, e cioè $f(\xi) = f'(\xi) = 0$, risulta immediatamente dalla

2.3.3 Altri schemi "Newton-like"

Possiamo riscrivere il metodo di Newton nel seguente modo:

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)}{C_k} \\ C_k &= f'(x_k) \end{aligned}$$

E' possibile utilizzare diverse espressioni per C_k, cercando sempre che tali espressioni si avvicinino il più possibile a $f'(x_k)$, che abbiamo visto garantisce convergenze "ottimale" quadratica. In particolare si ha la seguente tabella che riassume i principali schemi:

C_k	Nome schema	Ordine
$f'(x_0)$	tangente fissa	1
$f'(x_k)$	Newton-Raphson (tangente variabile)	2
$\frac{f(x_1)-f(x0)}{x_1-x_0}$	secante fissa	1
$\frac{f(x_k)-f(xk-1)}{x_k-x_{k-1}}$	Regula Falsi (secante variabile)	1.618

Verifichiamo per esercizio l'ordine e la costante asintotica dell'errore per il metodo della tangente fissa. Tale schema approssima la derivata con la derivata prima calcolata nel punto iniziale e poi la mantiene costante. Si avrà quindi:

$$\xi + \epsilon_{k+1} = \xi + \epsilon_k - \frac{f(\xi + \epsilon_k)}{f(x_0)}$$

Espandendo $f(\xi + \epsilon_k)$ in serie di Taylor attorno alla radice ξ si ha:

$$\begin{aligned}\epsilon_{k+1} &= \epsilon_k - \frac{f(\xi) + \epsilon_k f'(\xi) + \epsilon_k^2/2 f''(\xi) + \epsilon_k^3/3 f'''(\xi) \cdots}{f'(x_0)} \\ &= \frac{\epsilon_k(f'(x_0) - f'(\xi)) - \epsilon_k^2/2 f''(\xi) \cdots}{f'(x_0)}\end{aligned}$$

da cui si vede che lo schema della tangente fissa converge con $p = 1$ e $M = (f'(x_0) - f'(\xi))/f'(x_0)$.

Veniamo ora allo schema della Regula-Falsi. Tale schema approssima la derivata prima con un rapporto incrementale ottenuto con le informazioni più recentemente calcolate (x_k, x_{k-1}, $f(x_k)$ e $f(x_{k-1})$). La formula iterativa del metodo della Regula-Falsi si può scrivere nel modo seguente:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}.$$

L'algoritmo è innescato da due soluzioni iniziali che devono essere fornite esternamente. La prima, x_0, è come al solito arbitraria. La seconda viene generalmente calcolata con una iterazione del metodo di Newton-Raphson. Una volta note x_0 e x_1, si possono calcolare le approssimazioni successive.

L'analisi di convergenza richiede qualche calcolo in più rispetto al metodo di Newton-Raphson, ma la tecnica è la stessa. Si ha infatti, ricordando sempre che $f(\xi) = 0$:

$$\begin{aligned}\epsilon_{k+1} &= \frac{\epsilon_{k-1} f(\xi + \epsilon_k) - \epsilon_k f(\xi + \epsilon_{k-1})}{f(\xi + \epsilon_k) - f(\xi + \epsilon_{k-1})} \\ &= \frac{\epsilon_{k-1}\left[\epsilon_k f'(\xi) + \frac{1}{2}\epsilon_k^2 f''(\xi) + \cdots\right] - \epsilon_k\left[\epsilon_{k-1} f'(\xi) + \frac{1}{2}\epsilon_{k-1}^2 f''(\xi) + \cdots\right]}{\epsilon_k f'(\xi) + \frac{1}{2}\epsilon_k^2 f''(\xi) + \cdots - \epsilon_{k-1} f'(\xi) - \frac{1}{2}\epsilon_{k-1}^2 f''(\xi) - \cdots} \\ &= \frac{\epsilon_{k-1}\epsilon_k f'(\xi) + \epsilon_{k-1}\frac{1}{2}\epsilon_k^2 f''(\xi) + \cdots - \epsilon_k\epsilon_{k-1} f'(\xi) - \epsilon_k\frac{1}{2}\epsilon_{k-1}^2 f''(\xi) - \cdots}{\epsilon_k f'(\xi) + \frac{1}{2}\epsilon_k^2 f''(\xi) + \cdots - \epsilon_{k-1} f'(\xi) - \frac{1}{2}\epsilon_{k-1}^2 f''(\xi) - \cdots} \\ &= \frac{\frac{1}{2} f''(\xi)\epsilon_k\epsilon_{k-1}(\epsilon_k - \epsilon_{k-1}) + \cdots}{f'(\xi)(\epsilon_k - \epsilon_{k-1}) + \frac{1}{2} f''(\xi)(\epsilon_k^2 - \epsilon_{k-1}^2)\cdots} \\ &= \frac{1}{2}\frac{f''(\xi)}{f'(\xi)}\epsilon_k\epsilon_{k-1} + \ldots\end{aligned}$$

per cui risulta infine

$$\epsilon_{k+1} = \frac{1}{2}\frac{f''(\xi)}{f'(\xi)}\epsilon_k\epsilon_{k-1} + \ldots = A\epsilon_k\epsilon_{k-1} + \ldots \tag{2.23}$$

Prendendo i moduli e ricordando la definizione (2.19), in condizioni asintotiche si ha in tuta generalità:

$$\frac{|\epsilon_{k+1}|}{|\epsilon_k|^p} = M \tag{2.24}$$

da cui evidentemente si ricava:

$$|\epsilon_{k-1}| = \left(\frac{|\epsilon_k|)}{M}\right)^{1/p}$$

che sostituita nella (2.23) fornisce:

$$|\epsilon_{k-1}| = AM^{-1/p}\epsilon_k^{(p+1)/p} \tag{2.25}$$

dove la costante $A = |f''(\xi)|/(2|f'(\xi)|)$ coincide con la costante asintotica dell'errore del metodo di Newton-Raphson. La coincidenza della (2.24) con la (2.25) si ha quando:

$$AM^{-1/p} = M \qquad\qquad \epsilon_k^{(p+1)/p} = \epsilon_k^p$$

che fornise $M = A^{(p+1)/p}$ e $p = (1 \pm \sqrt{(}5))/2$. Escludendo la radice negativa, la Regula Falsi risulta avere quindi convergenza superlineare pari a $p = 1.618\cdots$ e costante asintotica pari a $M = A^{0.618\cdots}$.

2.4 Localizzazione delle radici

Finora abbiamo assunto l'unicità oltra all'esistenza della radice ξ. In realtà questa assunzione può considerarsi valida a patto di avere preliminarmente individuato un intervallo I all'interno del quale esista una e una sola soluzione. Vediamo quindi ora come possiamo agire.

2.4.1 Condizioni necessarie e sufficienti per l'esistenza di una unica radice

Ricordiamo che il problema che stiamo affrontando è quello di trovare una radice della funzione $f(x)$ localizzata in un dato intervallo:

Problema 2.4.1. Trovare $x \in \mathbb{R}$ tale che:

$$f(x) = 0$$

con $f : \mathbb{R} \to \mathbb{R}$ una funzione continua.

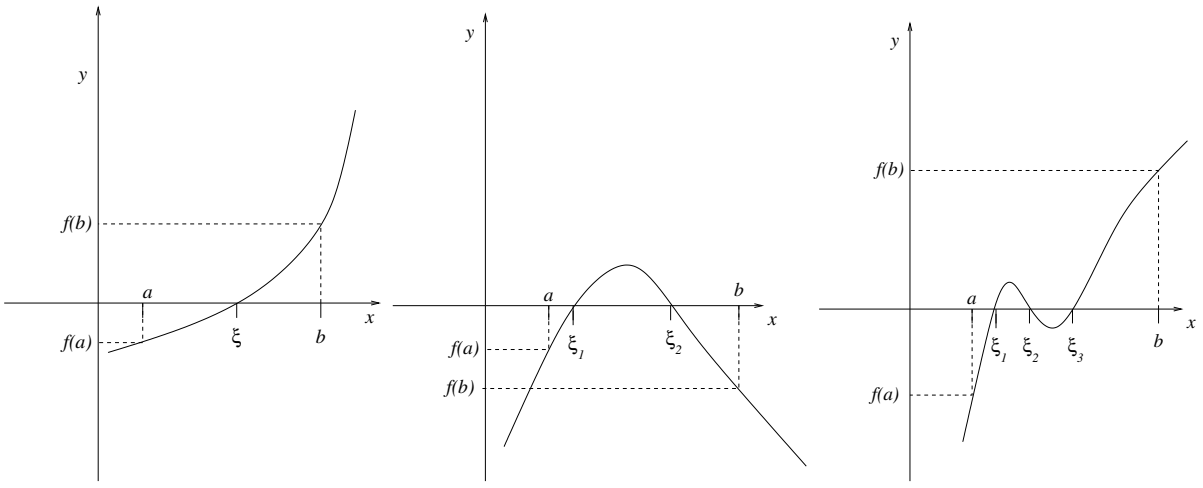

Figura 2.4: Problema di trovare la radice $x = \xi$ della funzione $f(x)$ all'interno dell'intervallo $I = [a, b]$. Figura di sinistra: caso di una unica radice; figura centrale: due radici; figura di destra: 3 radici.

Indichiamo la soluzione vera (teorica) di tale problema con la lettera greca ξ[2], per cui si ha che $f(\xi) = 0$. Ricordando che il problema è quello di cercare il punto di intersezione tra la funzione $y = f(x)$ e l'asse x (cioè la funzione $y = 0$), possiamo dire intuitivamente che esiste almeno una radice se la funzione interseca l'asse x in almeno un punto interno all'intervallo I. Avendo fatto l'ipotesi di continuità della $f(x)$, una condizione sufficiente per avere almeno una radice in I è che la funzione assuma valori opposti agli estremi dell'intervallo, e cioè:

$$\begin{array}{ccc} f(a) < 0 & \text{e} & f(b) > 0 \\ & \text{oppure} & \\ f(a) > 0 & \text{e} & f(b) < 0. \end{array}$$

Ovviamente questa condizione non può essere necessaria e non implica l'unicità della radice, come si può facilmente vedere dalla figura 2.4. Una condizione necessaria per l'unicità è la monotonia della $f(x)$, e quindi la permanenza del segno della $f'(x)$.

Nel caso del metodo di punto fisso, possiamo fare gli stessi ragionamenti andando a verificare le condizioni precedenti per la funzione $h(x) = x - g(x)$. Facendo i conti si trova che la condizione sufficiente per l'esistenza di un punto fisso ξ nell'intervallo

[2]Si legge "csi".

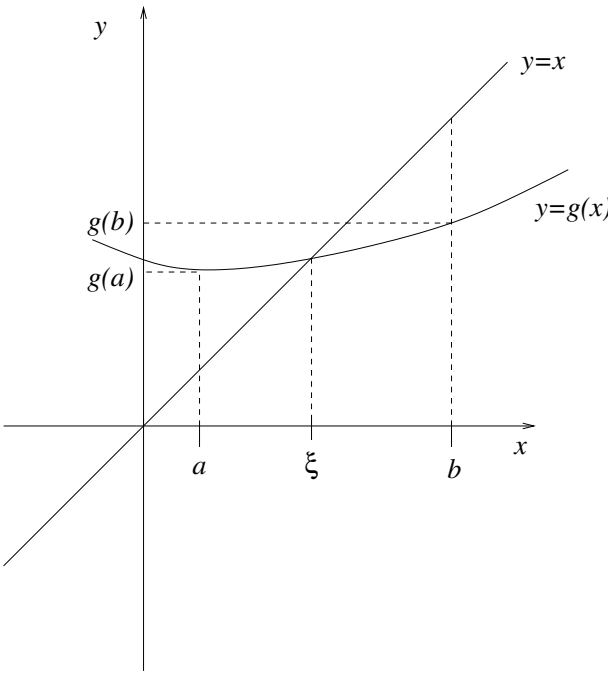

Figura 2.5: Problema di punto fisso nell'intervallo $I = [a, b]$.

$I = [a, b]$ è:

$$\begin{array}{ccc} h(a) > 0 & \text{e} & h(b) < 0 \\ & \text{oppure} & \\ h(a) < 0 & \text{e} & h(b) > 0, \end{array}$$

con l'unicità derivante dalla permanenza del segno della $h'(x) = 1 - g'(x)$. Utilizzando la continuità della funzione di punto fisso $g(x)$, e quindi della $h(x)$, si ottengono allora le seguenti condizioni sufficienti per l'esistenza e l'unicità del punto fisso:

$$\begin{cases} g(a) > a \quad \text{e} \quad g(b) < b \\ g'(x) < 1 \quad \text{e} \quad g'(x) > -1, \quad \forall x \in I, \end{cases}$$

ovvero

$$\begin{cases} g(a) > a \qquad \text{e} \quad g(b) < b \\ |g'(x)| < 1, \qquad \forall x \in I. \end{cases}$$

Le equazioni precedenti, insieme alla continuità, implicano che $g(I) \subset I$, per cui si dice che la funzione g è una "contrazione". In Figura 2.5 vengono visualizzati questi concetti. In questo caso, si ha che $g(a) > a$ e $g(b) < b$, e inoltre $|g'(x)| < 1$. Si nota subito che quest'ultima condizione significa che la funzione di punto fisso in ogni

punto di I potrà crescere (o decrescere) al massimo quanto la retta $y = x$, per cui non potrà esistere un secondo punto fisso.

Esempio 2.4.2. Data l'equazione:

$$f(x) = x^3 - x - 1 = 0$$

mostrare che essa ammette una e una sola soluzione ξ nell'intervallo $I = [1, 2]$.

La funzione f è continua, qundi applichiamo le considerazioni fatte sopra:

$$f(1) = -1 < 0, \qquad f(2) = 8 - 2 - 1 = 5 > 0,$$

quindi esiste una soluzione $\xi \in I$. La derivata prima di f vale:

$$f'(x) = 3x^2 - 1,$$

ed è positiva per $x > \sqrt{3}/3$ e quindi per tutti gli x in $I = [1, 2]$. Pertanto la radice ξ è unica.

Esempio 2.4.3. Data la funzione di punto fisso:

$$x = g(x) = \sqrt[3]{1 + x}$$

si dimostri che il punto fisso coincide con la radice della funzione dell'esempio precedente, e si dimostri esistenza e unicità del punto fisso.

Dalla equazione di punto fisso sopra scritta, con semplici passaggi si ottiene:

$$x^3 - x - 1 = 0,$$

che è proprio la $f(x)$ dell'esempio precedente.

L'esistenza del punto fisso ξ deriva dal fatto che:

$$g(1) = \sqrt[3]{2} > 1, \qquad g(2) = \sqrt[3]{3} < 2.$$

Per analizzare l'unicità di ξ andiamo a vedere la derivata prima di g. Risulta:

$$g'(x) = \frac{1}{3}(1 + x)^{-2/3}.$$

Si vede immediatamente che la $g'(x)$ è una funzione positiva e sempre decrescente per $x > 0$. Il suo valore massimo in I si ottiene in $x = 1$ e vale $g'(1) = 1/(3\sqrt[3]{4}) \approx 0.21 < 1$. Quindi ξ è l'unico punto fisso in I.

2.4.2 Il metodo dicotomico

Passiamo ora a discutere il metodo "dicotomico" o di "bisezione", un metodo molto usato per localizzare un intervallo iniziale sufficientemente piccolo con cui inizializzare gli schemi di puto fisso o di Newton-Raphson. Il metodo dicotomico è in realtà un vero e proprio metodo iterativo per la soluzione di equazioni nonlineari, e si ricava intuitivamente sfruttando le considerazioni fatte in precedenza.

Dato un intervallo $I_0 = [a, b]$ che contiene la radice, che assumiamo unica, si costruiscono due successioni $\{s_k\}$ e $\{d_k\}$ che convergono a ξ rispettivamente da sinistra e da destra, e che individuano quindi una successione di intervalli $I_i = [s_i, t_i]$ che tende a zero mantenendo la condizione $\xi \in I_k$. Si procede con il seguente algoritmo:

ALGORITMO DICOTOMICO:

dato un intervallo iniziale $I_0 = [a, b]$ con $\xi \in I_0$ e una tolleranza $TOLL$;

$sk := a$; $dk := b$; $=SCARTO := 2 * TOLL$

FINCHÉ $SCARTO > TOLL$ esegui:

1. $xk = 0.5 * (sk + dk)$

2. IF $f(xk) * f(dk) < 0$: $sk = xk$; dk invariato;
   ELSE $dk = sk$; sk invariato

FINE FINCHÉ.

Si vede che ad ogni iterazione si prende come estremi del nuovo intervallo il punto medio e un estremo dell'intervallo precedente in modo tale da garantire la permanenza della radice ξ all'interno del nuovo intervallo.

Capitolo 3

Approssimazione e interpolazione di dati

In questo capitolo ci occuperemo di risolvere il problema dell'approssimazione di dati. Nella pratica ingegneristica succede molto spesso che sia nota una serie di osservazioni e si voglia trovare una forma funzionale che leghi queste osservazioni in qualche modo. As esempio, sia nota la serie della temperatura oraria misurata in un dato posto. La serie è quindi formata da n coppie ordinate [ora, temperatura] che possiamo chiamare $(x_i, y_i), i = 0, \ldots, n$. Supponiamo di scegliere come forma funzionale un polinomio di grado opportuno m. La scelta della forma polinomiale è suggerita dal fatto che qualsiasi funzione "analitica" può essere approssimata con sufficiente accuratezza da un polinomio di grado sufficientemente elevato. La classe (spazio) dei polinomi di grado m la chiamiamo $\mathcal{P}_m$, per cui:

$$\mathcal{P}_m = \left\{ P_m(x) : \mathbb{R} \to \mathbb{R} : \; P_m(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x + a_0. \right\}. \quad (3.1)$$

Un polinomio di grado m è definito da m coefficienti incogniti e la loro determinazione richiede m condizioni (equazioni) linearmente indipendenti.

Ci sono due strade alternative per scegliere il grado m e definire i coefficienti. La prima, chiamata "interpolazione polinomiale", sfrutta il fatto che esiste uno e un solo polinomio di grado n che passa per $n + 1$ punti, e cioè che soddisfa alle cosidette relazioni di interpolazione: dati $n+1$ punti di appoggio $(x_i, y_i), i = 0, \ldots, n$, il polinomio soddisfa alle seguenti condizioni di interpolazione, e cioè che passa per tutti i punti di appoggio, vale a dire che per ogni i si ha $P_n(x_i) = y_i$. Ci sono quindi $n+1$ condizioni indipendenti e il polinomio interpolatore è determinato univocamente. Si noti che il polinomio interpolatore "passa" per tutti i punti di appoggio, come si vede dall'esempio di figura 3.1. Questo fatto, in particolare quando le ascisse dei punti di appoggio sono equidistanti, è molto restrittivo e, come si vedrà più avanti, porta a problemi "mal condizionati".

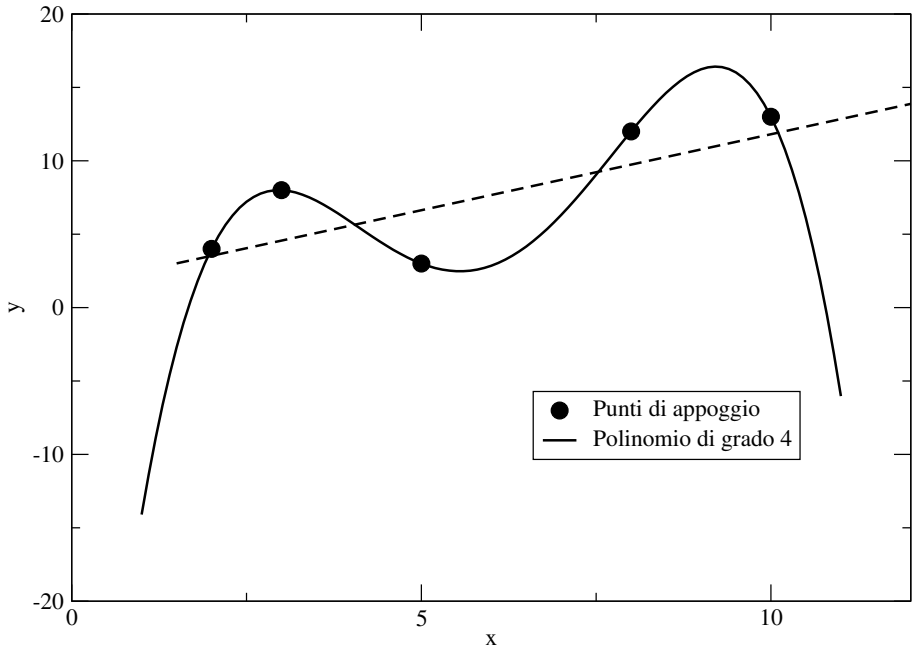

Figura 3.1: Esempio di polinomio di interpolatore e approssimatore. In questo esempio, ci sono 5 punti di appoggio, e quindi il polinomio interpolatore è di grado 5, mentre il polinomio approssimatore è di grado 1

La seconda strada determina un polinomio di grado $m < n$, chiamato polinomio approssimatore, i cui coefficienti sono determinati in modo da minimizzare qualche misura della differenza tra i valori approssimati e i dati di appoggio.

3.1 Interpolazione polinomiale

Il problema che vogliamo risolvere è il seguente:

Problema 3.1.1 (Interpolazione Polinomiale)**.** Dati $n+1$ punti di appoggio (x_i, y_i), $i = 0, \ldots, n$, trovare il polinomio di grado n tale che:

$$P_n(x_i) = y_i \qquad\qquad i = 0, \ldots, n. \tag{3.2}$$

Si scriva dunque un polinomio di grado n come:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$$

Imponendo le condizioni di interpolazione (3.2) si arriva al seguente sistema lineare, detto sistema di van der Monde:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^n \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

Purtroppo, questo sistema risulta molto malcondizionato e la sua soluzione numerica diventa assai inaccurata anche per piccoli valori di n, dell'ordine di 5 o 6. Per questo motivo tale tecnica non si usa quasi mai.

3.1.1 Polinomio interpolatore di Lagrange

Notiamo innanzitutto che tutti i polinomi di grado $m = n$ dell'insieme defininto dalla (3.1) è uno spazio vettoriale di dimensione $n + 1$ e che una base di tale spazio è formata dalle potenze di x:

$$\mathcal{P}_n = \text{span}\{1, x, x^2, \ldots, x^n\}.$$

Questa peraltro è proprio la base definita dalla matrice di van der Monde, per cui non va bene per i calcoli. Cerchiamo quindi una base diversa. A tal fine scriviamo il nostro polinomio come una combinazione di $n+1$ polinomi $L_i(x)$ tutti di grado n:

$$P_n(x) = \sum_{i=0}^{n} \alpha_i L_i(x). \tag{3.3}$$

Si tratta ora di determinare la base $L_i(x)$ e i coefficienti α_i che risolvono il problema (3.1.1).

Si noti che perché le condizioni di interpolazione (3.2) siano soddisfatte da un polinomio della forma (3.3), basta richiedere che:

$$L_j(x_i) = \begin{cases} 1, & \text{se } i = j, \\ 0, & \text{se } i \neq j. \end{cases}$$

In questo caso si vede immediatamente che $\alpha_i = y_i$, per cui:

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x). \tag{3.4}$$

Possiamo calcolare i polinomi di base a partire dai valori x_i notando che la funzione formata da tutti i monomi di x fatti con gli x_i si azzera in corrispondenza di ogni punto di appoggio:

$$F(x) = (x - x_0)(x - x_1)\dots(x - x_n) = 0 \qquad \forall x = x_i, i = 0, \dots, n.$$

La $F(x)$ è un polinomio di grado $n + 1$, e quindi dobbiamo tirare via un monomio se vogliamo avere un polinomio di grado n. Definiamo quindi il polinomio di grado n escludendo dalla precedente il monomio $(x - x_k)$:

$$F_k(x) = (x - x_0)(x - x_1)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_n),$$

Il polinomio che cerchiamo è dunque:

$$L_i(x) = \frac{F_k(x)}{F_k(x_k)}.$$

Esempio 3.1.2. Si vuole trovare il polinomio che interpola i seguenti punti:

x_i	1	2	3	4
y_i	3	1	5	2

Notiamo subito che il grado del polinomio sarà $n = 3$. Dovremo quindi calcolare $L_0(x), L_1(x), L_2(x), L_3(x)$:

$$\begin{aligned} L_0(x) &= \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} \\ L_1(x) &= \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\ L_2(x) &= \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} \\ L_3(x) &= \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}. \end{aligned}$$

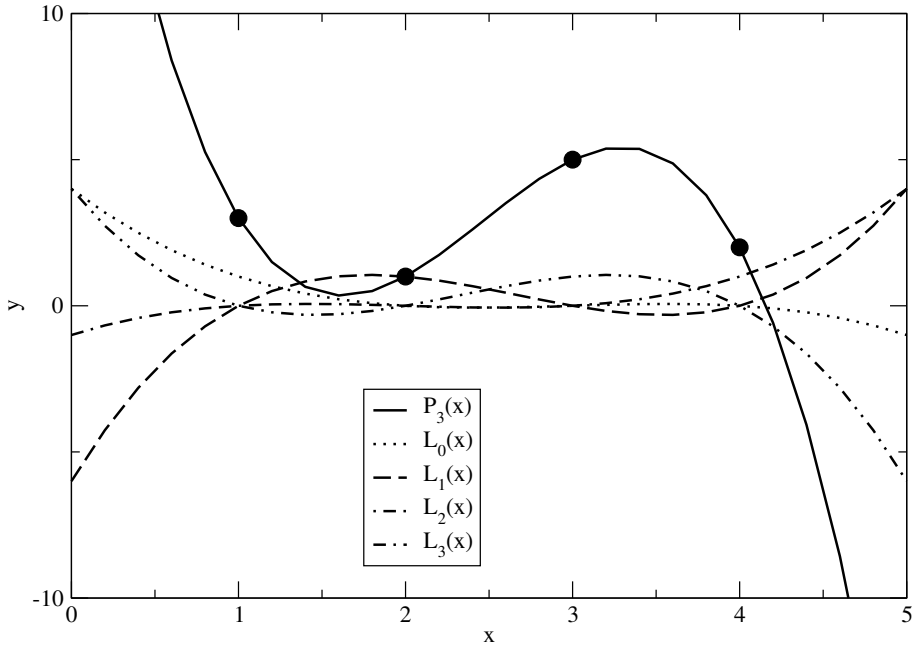

Figura 3.2: Punti di appoggio, polinomio interpolatore, e polinomi di Lagrange per l'esempio (3.1.2).

Il polinomio che si ricava applicando la (3.4) è:

$$P_3(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x) = 24 - 34.8333x + 16x^2 - 2.16667x^3.$$

In figura 3.2 sono riportati i punti di appoggio assieme al polinomio interpolatore $P_3(x)$ e ai polinomi di base $L_0(x)$, $L_1(x)$, $L_2(x)$ e $L_3(x)$.

Per studiare la bontà della interpolazione polinomiale così ottenuta, bisogna guardare l'andamento dell'errore del modello matematico. Si assume quindi che i punti di appoggio (x_i, y_i) provengano da una funzione incognita $f(x)$. Si assume quindi che esista una funzione tale che $y_i = f(x_i)$ per $i = 0, \ldots, n$ e si vuole studiare come si comporta l'errore definito da:

$$E(x) = f(x) - P_n(x).$$

Sia $I_x = [\min_i x_i, \max_i x_i]$ l'intervallo di interpolazione, e prendiamo un punto t interno a I_x e diverso da tutti gli x_i. Se t conicide con uno degli x_i immediatamente abbiamo che $E(t) = 0$ e abbiamo finito. Altrimenti, cerchiamo una funzione $G(x)$ che si annulla in tutti i punti x_i e anche in t. Osserviamo che la (3.2) implica che $E(x)$ si annulla in tutti i punti x_i. Anche la funzione $F(x)$, per definizione, si annulla in tali punti. Costruiamo quindi la funzione:

$$G(x) = f(x) - P_n(x) + S_0(t)F(x).$$

Tale funzione, per quanto detto prima, si annulla in tutti gli x_i, e calcoliamo S_0 in maniera tale che $G(t) = 0$. Otteniamo:

$$S_0(t) = \frac{P_n(t) - f(t)}{F(t)}.$$

Siccome funzione $G(x)$ si annulla in $n + 2$ punti di appoggio ed è infintamente derivabile e quindi continua, il teorema di Rolle assicura che esistono $n + 1$ punti η_i, $i = 0, \ldots, n$ dove la derivata prima di $G(x)$ si annulla: $G'(\eta_i) = 0$. Esiste quindi n punti dove si annulla la derivata seconda, e quindi esiste un punto $\eta \in I_x$ dove si annulla la derivata $n + 1$-esima di $G(x)$. tale punto sarà ovviamente funzione del punto t, per cui scriviamo $\eta(t)$. Un semplice calcolo mostra che:

$$G^{(n+1)}(x) = f^{(n+1)}(x) - S_0(t)(n + 1)!.$$

Imponendo ora $G^{(n+1)}(\eta(t)) = 0$, si identifica il valore di $S_0(t)$ in funzione della derivata $n + 1$-esima della $f(x)$:

$$S_0(t) = \frac{f^{(n+1)}(\eta(t))}{(n + 1)!}.$$

Notando che possiamo far variare $t \in I_x$ con continuità, e riscrivendo quindi x al posto di t, si ha:

$$f(x) - P_n(x) = F(x)\frac{f^{(n+1)}(\eta)}{(n + 1)!}. \tag{3.5}$$

Tale formula, chiamata formula del resto di Lagrange, ci dice che sostituendo alla funzione (incognita) f il suo polinomio interpolatore si incorre in un errore massimo che è proporzionale alla derivata $n + 1$-esima di f.

3.1.2 Interpolazione di Newton

Il polinomio di Lagrange è di difficile valutazione dal punto di vista numerico. Si ricorre quindi ad una tecnica (detta di Newton) che permette una valutazione agevole di tale polinomio. Tale tecnica è basata su quantità chiamate *differenze divise* che si possono definire ricorsivamente:

$$f(x_0, x_1, \ldots, x_n) = \frac{f(x_1, \ldots, x_n) - f(x_0, \ldots, x_{n-1})}{x_n - x_0}, \tag{3.6}$$

con $f_(x_i) = y_i$. Dati gli $n+1$ punti di appoggio (x_i, y_i), formiamo quindi la tabella (matrice):

$$
\begin{array}{llllll}
x_0 & y_0 = f(x_0) & & & & \\
 & & f(x_0, x_1) & & & \\
x_1 & y_1 = f(x_1) & & f(x_0, x_1, x_2) & & \\
 & & f(x_1, x_2) & & & \\
x_2 & y_2 = f(x_2) & & & & \\
\dots & & & & & \\
\dots & \dots & \dots & \dots & f(x_0, x_1, \dots, x_{n-1}, x_n) & \\
\dots & & & & & \\
x_{n-1} & y_{n-1} = f(x_{n-1}) & & & & \\
 & & f(x_{n-1}, x_n) & & & \\
x_n & y_n = f(x_n) & & f(x_{n-1}, x_n, x) & & \\
 & & f(x_n, x) & & & \\
x & y = f(x) & & & &
\end{array}
$$

Ad esempio, per n=2, otteniamo:

$$
\begin{array}{llll}
x_0 & y_0 = f(x_0) & & \\
 & & f(x_0, x_1) & \\
x_1 & y_1 = f(x_1) & & f(x_0, x_1, x_2) \\
 & & f(x_1, x_2) & f(x_0, x_1, x_2, x) \\
x_2 & y_2 = f(x_2) & & f(x_1, x_2, x) \\
 & & f(x_2, x) & \\
x & y = f(x) & &
\end{array}
$$

Dalla definizione di differenza divisa (eq. (3.6)), si ottiene:

$$
\begin{aligned}
f(x_1) &= f(x_0) + f(x_0, x_1)(x_1 - x_0) \\
f(x_2) &= f(x_1) + f(x_1, x_2)(x_2 - x_1) \\
f(x) &= f(x_2) + f(x_2, x)(x - x_2) \\
f(x_1, x_2) &= f(x_0, x_1) + f(x_0, x_1, x_2)(x_2 - x_0) \\
f(x_2, x) &= f(x_,x_1) + f(x_0, x_1, x_2)(x_2 - x_0) \\
f(x_1, x_2, x) &= f(x_0, x_1, x_2) + f(x_0, x_1, x_2, x)(x - x_0)
\end{aligned}
$$

Usando queste espressioni nelle differenze divise di ordine crescente e semplificando opportunamente, otteniamo:

$$
\begin{aligned}
f(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) \\
+ f(x_0, x_1, x_2, x)(x - x_0)(x - x_1)(x - x_2).
\end{aligned}
$$

Guardando a questa espressione, si vede che i primi 3 termini formano in polinomio di grado 2 che passa per i punti di appoggio $P_2(x_i) = y_i = f(x_i)$, per cui coincide con il polinomio interpolatore (perché è unico). Quindi possiamo scrivere:

$$f(x) = P_2(x) + RF(x),$$

e confrontando quest'ultima con la (3.5), si vede subito che $R = f^{(3)}(\eta)/(3)!$. Generalizzando per n qualsiasi si avrà:

$$\begin{aligned} P_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \\ + \ldots + f(x_0, x_1, x_2, \ldots, x_n)(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \end{aligned}$$

Osservazione 3.1.3. Si osservi che la differenza divisa di ordine 1 altro non è che un rapporto incrementale. E' quindi ovvio che:

$$f(x_0, x_0) = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0)$$

e più in generale:

$$f(\overset{k+1 \text{ volte}}{x_0, x_0, \ldots, x_0}) = \frac{f^{(k)}(x_0)}{k!}.$$

E' possibile quindi includere come punti di appoggio alcuni valori delle derivate e costruire in maniera appropriata la tabella di Newton per ottenere il polinomio interpolatore che considera anche i valori dati delle derivate di f.

Esempio 3.1.4. Siano dati i seguenti punti di appoggio:

x_i	0.1	0.5	1.1	1.8	2.5
y_i	11.0702	3.43895	3.37089	6.21211	20.3560

1. scrivere la tabella delle differenze divise di Newton;

2. determinare il polinomio di grado 4 che interpola i punti dati;

3. Sapendo che i punti di appoggio sono relativi alla funzione $f(x) = \exp(x)/\sin(x)$, dare una stima dell'errore massimo che si può commettere utilizzando il polinomio interpolatore al posto della funzione.

Svolgimento. In questo esercizio si ha $I_x = [0.1, 2.5]$.

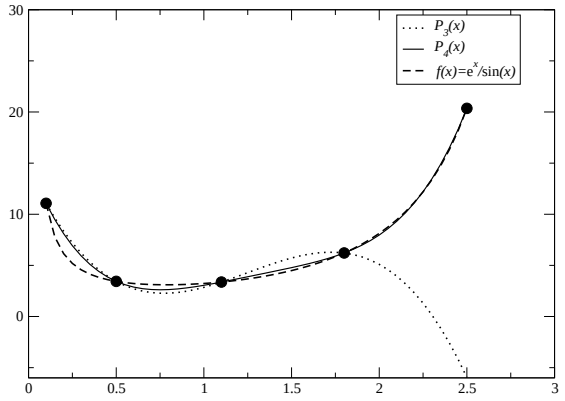

Figura 3.3:

1. La tabella delle differenze divise di Newton è:

$$\begin{array}{lllll}
0.1 & 11.0702 & & & \\
 & & -19.0780 & & \\
0.5 & 3.43895 & & 18.9646 & \\
 & & -11.3429 & & -9.26771 \\
1.1 & 3.37089 & & 3.20947 & & 5.59568 \\
 & & 4.05888 & & 4.16193 & \\
1.8 & 6.21211 & & 11.5333 & & \\
 & & 20.2055 & & & \\
2.5 & 20.3560 & & & &
\end{array}$$

2. Il polinomio interpolatore diventa quindi:

$$\begin{aligned}
P_4(x) &= 11.0702 \\
&\quad -19.0780(x-0.1) \\
&\quad +18.9646(x-0.1)(x-0.5) \\
&\quad -9.26771(x-0.1)(x-0.5)(x-1.1) \\
&\quad +5.59568(x-0.1)(x-0.5)(x-1.1)(x-1.8) \\
&= 14.9899 - 44.4959x + 55.8154x^2 - 28.8526x^3 + 5.59568x^4
\end{aligned}$$

E' agevole trovare il polinomio di grado 3 che interpola i primi 4 punti della

tabella: coincide con i primi 4 termini dell'equazione precedente, e vale:

$$\begin{aligned} P_3(x) &= 11.0702 \\ &\quad -19.0780(x-0.1) \\ &\quad +18.9646(x-0.1)(x-0.5) \\ &\quad -9.26771(x-0.1)(x-0.5)(x-1.1) \\ &= 14.436 - 37.0368x + 34.7197x^2 - 9.26771x^3 \end{aligned}$$

La figura 3.3 mostra la funzione originale, i punti di appoggio e i polinomi $P_3(x)$ e $P_4(x)$.

3. La stima dell'errore massimo che si può commettere usa la formula del resto di Lagrange (eq. (3.5)). Possiamo scrivere infatti:

$$|E_n(x)| \leq \max_{x \in I_x} \left[|F(x)| \frac{\left|f^{(n+1)}(x)\right|}{(n+1)!} \right].$$

In questo esercizio $n = 4$, e quindi si ha:

$$|E_4(x)| \leq \max_{x \in I_x} \left[|F(x)| \frac{\left|f^{(5)}(x)\right|}{5!} \right].$$

Con conti lunghi ma semplici si ottiene che la derivata quinta della $f(x)$ vale:

$$f^{(5)}(x) = \frac{e^x}{\sin^6(x)} \left[-478\cos(x) - 3\cos(3x) + \cos(5x) + \right.$$
$$\left. 230\sin(x) + 85\sin(3x) - \sin(5x) \right],$$

e risulta essere una funzione crescente per $x \in I_x$, per cui il suo valore massimo si ottiene per $x = 2.5$ e vale $f^(5)(2.5) = 39793.9$. Usando Newton Raphson per trovare gli zeri di $F'(x)$ si trova che la $F(x) = (x-0.1)(x-0.5)(x-1.1)(x-1.8)(x-2.5)$ assume valore massimo in $x* = 2.25569$ che vale $|F(x^*)| = 0.486952$. Mettendo tutto insieme otteniamo:

$$|E_4(x)| \leq |F(x^*)| f^{(5)}(2.5)/5! = 0.486952 * 39793.9/120 = 161.481.$$

Si noti che questo valore è assai pessimistico. Analizzando l'errore vero $E_4(x) = P_4(x) - f(x)$, e usando il metodo di Newton Raphson per trovare gli zeri della sua derivata prima, si scopre che la funzione $E_4'(x)$ ha 5 zeri in I_x che valgono $x_1 = 0.193095$, $x_2 = 0.734279$, $x_3 = 1.42398$, $x_4 = 2.03462$, $x_5 = 2.40515$, e il valore massimo è molto più piccolo e vale:

$$\max_{x \in I_x} |E_4(x)| = E_4(x_1) = 1.95803.$$

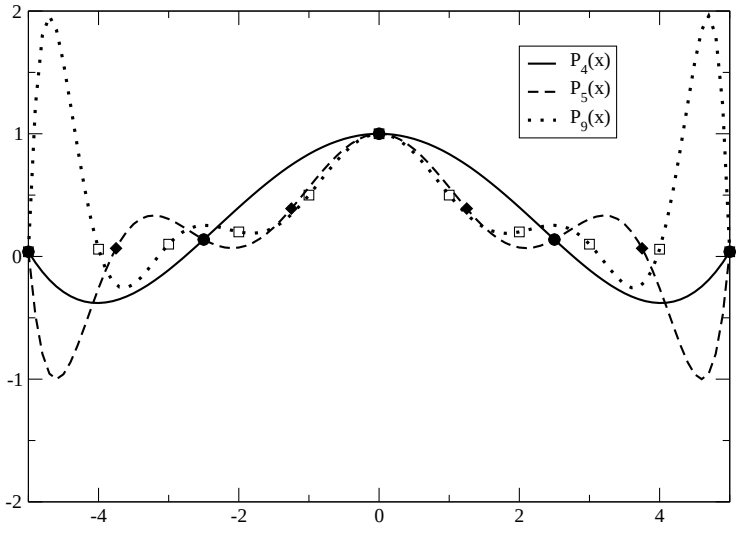

Figura 3.4: Polinomi interpolanti la funzione $f(x) = 1/(1+x^2)$ in $I = [-5, 5]$ con n punti di appoggio equispaziati per valori di n pari a 5, 6, 10.

3.1.3 Fenomeno di Runge e la stabilità dell'interpolazione polinomiale

L'interpolazione polinomiale con punti equispaziati ($x_i - x_{i-1} = h = \text{cost}$, $i = 1, \ldots, n$) può diventare instabile all'aumentare di n. Questa stabilità si manifesta, in funzione della regolarità della funzione interpolanda, anche per valori piccoli di n. Prima di studiare la stabilità vediamo un esempio famoso, noto col nome di *fenomeno di Runge*.

-5	0.038462				
-2.5	0.13793	0.0397878			
0	1	0.34483	0.061008		
2.5	0.13793	-0.34483	-0.13793	-0.026525	
5	0.038462	-0.039788	0.061008	0.026525	0.0053050

-5	$3.846E{-2}$										
-4	$5.882E{-2}$	$2.036E{-2}$									
-3	0.1	$4.118E{-2}$	$1.041E{-2}$								
-2	0.2	0.1	$2.941E{-2}$	$6.335E{-3}$							
-1	0.5	0.3	0.1	$2.353E{-2}$	$4.299E{-3}$						
0	1	0.5	0.1	0	$-5.882E{-3}$	$-2.036E{-3}$					
1	0.5	-0.5	-0.5	-0.2	$-5E{-2}$	$-8.824E{-3}$	$-1.131E{-3}$				
2	0.2	-0.3	0.1	0.2	0.1	$3E{-2}$	$6.471E{-3}$	$1.086E{-3}$			
3	0.1	-0.1	0.1	0	$-5E{-2}$	$-3E{-2}$	$-1E{-2}$	$-2.353E{-3}$	$-4.299E{-4}$		
4	$5.882E{-2}$	$-4.118E{-2}$	$2.941E{-2}$	$-2.353E{-2}$	$-5.882E{-3}$	$8.824E{-3}$	$6.471E{-3}$	$2.353E{-3}$	$5.882E{-4}$	$1.131E{-4}$	
5	$3.846E{-2}$	$-2.036E{-2}$	$1.041E{-2}$	$-6.335E{-3}$	$4.299E{-3}$	$2.036E{-3}$	$-1.131E{-3}$	$-1.086E{-4}$	$-4.299E{-4}$	$-1.131E{-4}$	$-2.262E{-5}$

Tabella 3.1: Tabelle delle differenze divise di Newton relative all'Esempio 3.1.5, per $n = 5$ punti di appoggio (pannello superiore) e per $n = 10$ punti di appoggio (pannello inferiore)

Esempio 3.1.5 (Fenomeno di Runge)**.** Proviamo ad interpolare la funzione:

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

con una suddivisione equispaziata dei punti di appoggio. Quindi definiamo $x_i = x_0 + ih$ con $x_0 = -5$ e $h = 10/n$. Studiamo i casi $n = 5$ e $n = 10$. Le tabelle delle differenze divise di Newton sono riprtate in 3.1, mentre l'andamento dei polinomi $P_4(x)$, $P_7(x)$ e $P_9(x)$ sono riportati in Figura 3.4. Si vede dai risultati che i polinomi interpolatori di grado crescente tendono a mostrare oscillazioni sempre più grandi, sintomo di instabilitià numerica.

Ovviamente, si potrebbe in teoria pensare di far tendere n all'infinito, in modo da coprire uniformemememente l'intervallo I_x. Il risultato fondamentale di Runge è la dimostrazione che, se le ascisse dei punti di appoggio sono equispaziate, esisterà sempre qualche punto $x \in I_x$ in cui il resto di Lagrange è diverso da zero, e cioè::

$$\lim_{n\to\infty} |f(x) - P_n(x)| \neq 0.$$

Questo fatto ci dice che il resto di Lagrange in generale non può convergere uniformemente a zero se i punti di apppoggio sono equispaziati.

Per studiare la stabilità del problema dell'interpolazione analizziamo il resto di Lagrange (eq. (3.5)) al variare del numero di punti di appoggio $n+1$. Ovviamente in generale si possono scegliere infinte combinazioni di $n+1$ punti di appoggio $(x_i, f(x_i))$, $x_i \in I_x$. Raccogliamo le ascisse degli $n+1$ punti di appoggio, comunque prese, in una matrice X (la matrice di interpolazione) di dimensione $n+1 \times n+1$ in cui ogni riga corrisponde ad una precisa scelta x_i, $i = 0, \ldots, n$. Indichiamo quindi con $P_{n,X}(x)$ un polinomio di grado n ottenuto utilizzando una riga della matrice X come ascisse dei punti di appoggio, e indichiamo con $\mathcal{P}_{n,X}$ la famiglia di tali polinomi. Chiamiamo con $P^*_{n,X}(x)$ il miglior polinomio interpolatore in $\mathcal{P}_{n,X}$, cioè $P^*_{n,X}(x)$ indica quel polinomio che minimizza la differenza (massima) tra la funzione $f(x)$ e il polinomio stesso, al variare dei punti di appoggio x_i, cioè della matrice X. In altre parole, $P^*_{n,X}(x)$ soddisfa:

$$E^*_n(X) = \max_{x\in I_x} \left|f(x) - P^*_{n,X}(x)\right| \leq \max_{x\in I_x} |f(x) - P_{n,X}(x)| = E_n(X) \qquad \forall P_{n,X}(x) \in \mathcal{P}_{n,X}.$$

Esiste il seguente risultato:

Proposizione 3.1.1. *Sia $f \in C^0(I_x)$ e X una matrice di interpolazione in I_X. Allora:*

$$E_n(X) \leq E^*_n(X)(1 + \Lambda_n(X)), \qquad n = 0, 1, \ldots,$$

dove $\Lambda_n(X)$ è detta costante di Lebesgue di X.

Nel caso dell'esempio di Runge, si può far vedere che la costante di Lebesgue tende a infinito all'aumentare di n.

Per studiare la stabilità dell'algoritmo di interpolazionae e il malcondizionamento del problema collegato, assumiamo le ascisse dei punti di appoggio esatte e consideriamo un insieme di valori perturbati delle ordinate dei punti di appoggio (i nostri dati), che indichiamo con $(x_i, \tilde{f}(x_i))$, $i = 0, \ldots, n$. indichiamo con $\tilde{P}_n(x)$ il polinomio interpolante tali dati. Allora si ha il seguente risultato:

$$\begin{aligned}\max_{x\in I_x}\left|P_n(x)-\tilde{P}_n(x)\right| &= \max_{x\in I_x}\left|\sum_{j=0}^{n}\left(f(x_j)-\tilde{f}(x_j)\right)L_j(x)\right| \\ &\leq \Lambda_n(X)\max_{j=0,\ldots,n}\left|f(x_j)-\tilde{f}(x_j)\right|.\end{aligned}$$

La costante di Lebesgue è dunque anche il numero di condizionameno del problema di interpolazione, per cui tale problema risulta ben condizionato solo se Λ_n è piccola. Purtroppo, si può dimostrare che in generale Λ_n aumenta all'aumentare di n, dando quindi luogo a potenziali instabilità per grandi valori di n, come visto nell'esempio 3.1.5.

3.2 Approssimazione polinomiale

Passiamo ora al problema dell'approssimazione polinomiale. Il problema è il seguente:

Problema 3.2.1 (Approssimazione Polinomiale)**.** Dati $n + 1$ punti di appoggio (x_i, y_i), $i = 0, \ldots, n$, trovare il polinomio di grado $m < n$ tale che una certa misura della differenza tra i valori osservati e il polinomio sia minimo:

$$\min_{P_m(x)\in\mathcal{P}_m}\left[\mathrm{d}(P_m(x_i)-y_i)_{i=0,\ldots,n}\right],$$

dove $\mathrm{d}(P_m(x_i) - y_i)_{i=0,\ldots,n}$ rappresenta la misura accennata sopra.

L'opportuna definizione della misura d distingue i vari metodi. In questo capitolo ci occuperemo dell'approssimazione "ai minimi quadrati" nella quale la misura d è definita usando gli scarti quadratici. Distingueremo tra scarti verticali e scarti orizzontali.

3.2.1 Retta ai minimi quadrati

Siano date le $n+1$ osservazioni (x_i, y_i). Vogliamo determinare un polinomio di grado $m = 1$ $P_1(x) = a_0 + a_1x$ che minimizza la somma degli scarti quadratici S definita

come:

$$\mathrm{d} = S(a_0, a_1) = \sum_{i=0}^{n} [P_1(x_i) - y_i]^2 = \sum_{i=0}^{n} [a_0 + a_1 x_i - y_i]^2 .$$

Si noti che la somma degli scarti quadratici è funzione dei coefficienti del polinomio a_0, a_1. Una condizione necessaria perché la somma sia minima è che si annullino tutte le derivate parziali. L'esistenza e l'unicità del punto di minimo derivano dall'analisi della funzione S, che essendo quadratica garantisce tale risultato. Usando la regola della derivata di funzione composta si ottiene:

$$\begin{aligned} \frac{\partial S}{\partial a_0} &= 2\sum_{i=0}^{n} [a_0 + a_1 x_i - y_i] = 0 \\ \frac{\partial S}{\partial a_1} &= 2\sum_{i=0}^{n} [a_0 + a_1 x_i - y_i]\, x_i = 0 \end{aligned}$$

Semplificando il 2 e applicando la proprietà distributiva si ottiene il seguente sistema lineare:

$$\begin{cases} (n+1)a_0 + \left(\sum_{i=0}^{n} x_i\right) a_1 = \sum_{i=0}^{n} y_i \\ \left(\sum_{i=0}^{n} x_i\right) a_0 + \left(\sum_{i=0}^{n} x_i^2\right) a_1 = \sum_{i=0}^{n} x_i y_i. \end{cases}$$

La soluzione di tale problema fornisce i coefficienti della retta ai minimi quadrati ricercata.

Esempio 3.2.2. Siano dati i seguenti punti di appoggio (gli stessi dell'es. 3.1.4)::

x_i	0.1	0.5	1.1	1.8	2.5
y_i	11.0702	3.43895	3.37089	6.21211	20.3560

1. determinare i coefficienti della retta ai minimi quadrati che minimizza gli scarti verticali;

2. determinare i coefficienti della retta ai minimi quadrati che minimizza gli scarti orizzontali;

3. individuare quale delle due rette approssima meglio i dati nel senso dei minimi quadrati.

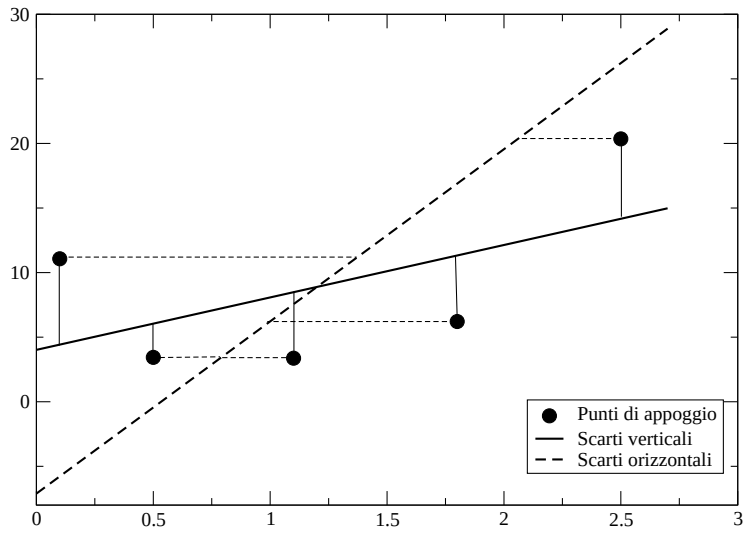

Figura 3.5: Punti di appoggio e rette ai minimi quadrati che minimizzano gli scarti verticali e gli scarti orizzontali per l'esempio 3.2.2.

Svolgimento. Vogliamo quindi determinare il polinomio di grado 1 $P_1(x) = a_0 + a_1x$ che minimizza la somma degli scarti quadratici verticali e successivamente il polinomio di grado 1 $P_1'(x) = b_0 + b_1x$ che minimizza la somma degl iscarti quadratici orizzontali. Impostiamo la seguente tabella:

i	x_i	x_i^2	y_i	x_iy_i	y_i^2
0	0.1	11.0702	0.01	1.10702	122.549
1	0.5	3.43895	0.25	1.71948	11.8264
2	1.1	3.37089	1.21	3.70798	11.3629
3	1.8	6.21211	3.24	11.1818	38.5903
4	2.5	20.3560	6.25	50.8900	414.367
Tot	6	44.4482	10.96	68.6063	598.696

1. la somma degli scarti quadratici verticali è data da:

$$S_v(a_0, a_1) = \sum_{i=0}^{4} \left[a_0 + a_1x_i - y_i\right]^2 .$$

Ripetendo gli sviluppi riportati sopra si trova che il sistema da risolvere per calcolare a_0 e a_1 è:

$$\begin{bmatrix} 5 & 6 \\ 6 & 10.96 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 44.4482 \\ 68.6063 \end{bmatrix}$$

la cui soluzione è $a_0 = 4.01671$ e $a_1 = 4.06077$, per cui si ha:

$$P_1(x) = 4.01671 + 4.06077x$$

2. Per minimizzare gli scarti quadratici orizzontali, osserviamo che basta procedere come nel modo precedente ma scambiando il ruolo di x e y. Quindi scriviamo il polinomio come $P_1^*(y) = b_0^* + b_1^*y$ e andiamo a minimizzare la la somma degli scarti quadratici orizzontali data da:

$$S_o^*(b_0^*, b_1^*) = \sum_{i=0}^{4} \left[b_0^* + b_1^*y_i - x_i\right]^2 .$$

Ripetendo gli sviluppi riportati sopra si trova che il sistema da risolvere per calcolare b_0^* e b_1^* è:

$$\begin{bmatrix} n+1 & \sum y_i \\ \sum y_i & \sum y_i^2 \end{bmatrix} \begin{bmatrix} b_0^* \\ b_1^* \end{bmatrix} = \begin{bmatrix} \sum x_i \\ \sum y_ix_i \end{bmatrix}$$

e quindi

$$\begin{bmatrix} 5 & 44.4482 \\ 44.4482 & 598.696 \end{bmatrix} \begin{bmatrix} b_0^* \\ b_1^* \end{bmatrix} = \begin{bmatrix} 6 \\ 68.6063 \end{bmatrix}$$

la cui soluzione è $b_0^* = 0.533239$ e $a_1 = 0.750044 \times 10^{-1}$, per cui si ha:

$$P_1^*(y) = 0.533239 + 0.750044 \times 10^{-1}\, y$$

e scrivendo il polinomio in forma canonica in funzione di x otteniamo:

$$P_1'(x) = -7.10944 + 13.3326\, x.$$

I grafici dei due polinomi con i punti di appoggio sono mostrati in Figura 3.5.

3. Per verificare quale delle due rette ha migliori proprietà di approssimazione, andiamo a verificare le somme degli scarti quadratici. Otteniamo:

$$S_v = \sum_{i=0}^{4} \left[4.01671 + 4.06077\, x_i - y_i\right]^2 = 141.566$$

$$S_o = \sum_{i=0}^{4} \left[-7.10944 + 13.3326\, x_i - y_i\right]^2 = 464.799$$

da cui si deduce che l'approssimazione migliore si ottiene con la retta che minimizza gli scarti verticali. Si noti che per questo confronto di ottimalità abbiamo usato il S_o e non S_o^*. Infatti il valore di S_o sopra indicato non è uguale a quello di S_o^*, mentre il punto di minimo delle due somme è lo stesso ma rappresentato in sistemi di riferimento diversi. E' quindi importante usare lo stesso sistema di riferimento quando si fanno i confronti di ottimalità, e quindi usare le somme degli scarti quadratici riferite allo stesso sistema $x - y$.

3.2.2 Polinomio ai minimi quadrati

Siano date le $n+1$ osservazioni (x_i, y_i). Vogliamo quindi determinare un polinomio di grado $m < n$ $P_m(x) = a_0 + a_1 x + \ldots + a_m x^m$ che minimizza la somma degli scarti quadratici S definita come:

$$\mathrm{d} = S(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} \left[P_m(x_i) - y_i\right]^2.$$

Si noti che la somma degli scarti quadratici è funzione dei coefficienti del polinomio $a_0, a_1, \ldots, a_m$. Una condizione necessaria perché la somma sia minima è che si annullino tutte le derivate parziali. L'esistenza e l'unicità del punto di minimo derivano dall'analisi della funzione S, che essendo quadratica garantisce tale risultato.

Usando la regola della derivata di funzione composta si ottiene:

$$\begin{aligned}
\frac{\partial S}{\partial a_0} &= 2\sum_{i=1}^{n}\left[P_m(x_i)-y_i\right]=0\\
\frac{\partial S}{\partial a_1} &= 2\sum_{i=1}^{n}\left[P_m(x_i)-y_i\right]x_i=0\\
\frac{\partial S}{\partial a_2} &= 2\sum_{i=1}^{n}\left[P_m(x_i)-y_i\right]x_i^2=0\\
&\dots\\
\frac{\partial S}{\partial a_m} &= 2\sum_{i=1}^{n}\left[P_m(x_i)-y_i\right]x_i^m=0.
\end{aligned}$$

Semplificando il 2, utilizzando l'espressione completa di $P_m(x)$ e applicando la proprietà distributiva si ottiene il seguente sistema lineare:

$$\begin{cases}
ma_0 + \left(\sum_{i=0}^{n} x_i\right)a_1 + \left(\sum_{i=0}^{n} x_i^2\right)a_2 + \dots + \left(\sum_{i=0}^{n} x_i^m\right)a_m = \sum_{i=0}^{n} y_i\\
\left(\sum_{i=0}^{n} x_i\right)a_0 + \left(\sum_{i=0}^{n} x_i^2\right)a_1 + \left(\sum_{i=0}^{n} x_i^3\right)a_2 + \dots + \left(\sum_{i=0}^{n} x_i^{m+1}\right)a_m = \sum_{i=0}^{n} x_i y_i\\
\left(\sum_{i=0}^{n} x_i^2\right)a_0 + \left(\sum_{i=0}^{n} x_i^3\right)a_1 + \left(\sum_{i=0}^{n} x_i^4\right)a_2 + \dots + \left(\sum_{i=0}^{n} x_i^{m+2}\right)a_m = \sum_{i=0}^{n} x_i^2 y_i\\
\dots\\
\left(\sum_{i=0}^{n} x_i^m\right)a_0 + \left(\sum_{i=0}^{n} x_i^{m+1}\right)a_1 + \left(\sum_{i=0}^{n} x_i^{m+2}\right)a_2 + \dots + \left(\sum_{i=0}^{n} x_i^{2m}\right)a_m = \sum_{i=0}^{n} x_i^m y_i
\end{cases}$$

che è certamente simmetrico e definito positivo. Definendo la matrice A come:

$$A=\begin{bmatrix}
1 & x_0 & x_0^2 & \dots & x_0^m\\
1 & x_1 & x_1^2 & \dots & x_1^m\\
1 & x_2 & x_2^2 & \dots & x_2^m\\
\dots & & & & \\
1 & x_n & x_n^2 & \dots & x_n^m
\end{bmatrix}$$

si verifica che il sistema dei minimi quadrati si può scrivere in notazione matriciale come:

$$A^T A a = b$$

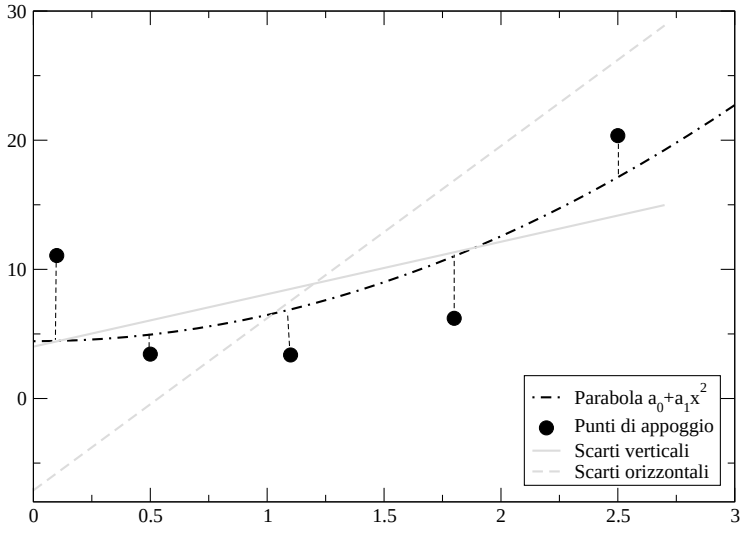

Figura 3.6: Punti di appoggio e parabola ai minimi quadrati che minimizza gli scarti verticali. In grigio sono mostrate per confronto le rette ai minimi quadrati dell'esempio 3.2.2.

dove il vettore incognito è $a = (a_0, a_1, \ldots, a_m)^T$ e il vettore termini noti è dato da $b = A^T y$ con $y = (y_0, \ldots, y_n)^T$. Si può notare che la matrice $A^T A$ è non singolare. Infatti, se fosse singolare esisterebbe un vettore a tale per cui $A^T Aa = 0$. Questo è vero se $Aa = 0$. Quest'ultimo sistema si può scrivere come:

$$\begin{cases} a_0 + a_1 x_0 + \ldots + a_m x_0^m = 0 \\ a_0 + a_1 x_1 + \ldots + a_m x_1^m = 0 \\ a_0 + a_1 x_2 + \ldots + a_m x_2^m = 0 \\ \\ \ldots \\ \\ a_0 + a_1 x_n + \ldots + a_m x_n^m = 0, \end{cases}$$

Questo sistema ha l'unica soluzione $a = 0$, perché altrimenti si avrebbe un polinomio di grado m con $n > m$ radici, e questo è impossibile.

Esempio 3.2.3. Siano dati i seguenti punti di appoggio (gli stessi dell'es. 3.2.3)::

x_i	0.1	0.5	1.1	1.8	2.5
y_i	11.0702	3.43895	3.37089	6.21211	20.3560

1. determinare i coefficienti della parabola della forma:

$$P_2(x) = a_0 + a_1\, x^2$$

che minimizza gli scarti verticali nel senso dei minimi quadrati;

2. verificare che tale parabola raggiunge un grado di approssimazione migliore della retta ai minimi quadrati calcolata nell'esempio 3.2.2.

Svolgimento.

1. Vogliamo determinare il polinomio di grado 2 $P_2(x) = a_0 + a_1 x^2$ che minimizza la somma degli scarti quadratici verticali. Notiamo che la somma è ancora funzione di due incognite a_0 e a_1:

$$S_2(a_0, a_1) = \sum_{i=0}^{4} \left[a_0 + a_1\, x_i^2 - y_i\right]^2 .$$

Imponendo l'annullarsi delle due derivate parziali, otteniamo il seguente sistema:

$$\begin{cases} 5a_0 & + \left(\displaystyle\sum_{i=0}^{n} x_i^2\right) a_1 & = \displaystyle\sum_{i=0}^{n} y_i \\ \left(\displaystyle\sum_{i=0}^{n} x_i^2\right) a_0 & + \left(\displaystyle\sum_{i=0}^{n} x_i^4\right) a_1 & = \displaystyle\sum_{i=0}^{n} x_i^2 y_i. \end{cases}$$

i	x_i^2	x_i^4	y_i	$x_i^4 y_i$
0	0.01	0.0001	11.0702	0.110702
1	0.25	0.0625	3.43895	0.859737
2	1.21	1.4641	3.37089	4.07878
3	3.24	10.498	6.21211	20.1272
4	6.25	39.063	20.3560	127.225
Tot	10.96	51.087	44.4482	152.401

Il sistema lineare diventa quindi:

$$\begin{bmatrix} 5 & 10.9600 \\ 10.9600 & 51.087 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 44.4482 \\ 152.401 \end{bmatrix}$$

la cui soluzione è $a_0 = 4.43709$ e $a_1 = 2.03127$, per cui si ha:

$$P_2(x) = 4.43709 + 2.03127\, x^2$$

La parabola è disegnata in Figura 3.6, dove sono anche disegnate per confronto e rette dell'esercizio precedente.

Capitolo 4

Quadratura numerica

In questo capitolo ci occupiamo di trovare il valore dell'integrale definito:

$$I = \int_a^b f(x)\, dx. \tag{4.1}$$

Si noti che il calcolo numerico non si occupa di analisi simbolica, per cui non sarà possibile trovare primitive, ma solamente trovare approssimazioni dell'integrale definito descritto in (4.1). Ci limiteremo a considerare integrali monodimensionali per semplicità. L'estensione a più dimensioni richiede concetti di interpolazione multivariata che non abbiamo affrontato.

L'idea fondamentale che vogliamo sfruttare parte dall'osservazione che in generale conosciamo la funzione integranda in tutti i punti, per cui possiamo procedere a interpolare la funzione con un polinomio opportuno e calcolare il valore di I come integrale del polinomio interpolatore. Una seconda idea molto utile è di sfruttare la linearità dell'operatore di integrale per arrivare alla cosidette formula composte. Si tratta di sudividere l'intervallo $[a, b]$ in n sottointervalli di ampiezza h_j tali che $x_j = x_0 + jh_j$, $j = 0, 1, \ldots, n$, con $x_0 = a$ e $x_n = b$; se i sottointervalli hanno ampiezza costante $h_j = h$, allora $h = (b - a)/n$. Con queste definizioni si ha immediatamente:

$$I = \int_a^b f(x)\, dx = \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(s)\, ds. \tag{4.2}$$

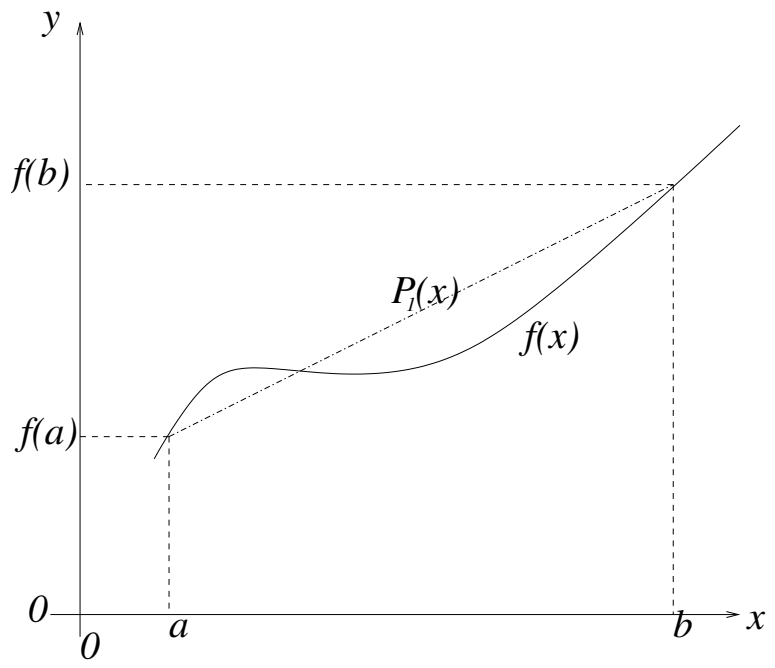

Figura 4.1: Interpretazione geometrica del metodo dei trapezi

4.1 Formule di quadratura con punti di appoggio equispaziati

4.1.1 Il metodo dei trapezi

Vogliamo approssimare il valore di I dell'Eq. (4.1). Scegliamo di usare un polinomio interpolatore di grado 1 (una retta) e scegliamo come punti di appoggio gli estremi dell'intervallo: $\{(a, f(a)), (b, f(b))\}$. Scriviamo la tabella del polinomio di Newton:

$$\begin{array}{cc} a & f(a) \\ b & f(b) \end{array} \quad \frac{f(b) - f(a)}{b - a}$$

e scriviamo il polinomio di grado 1 come:

$$P_1(x) = f(a) + \frac{f(b) - f(a)}{b - a} x.$$

Figura 4.2: Suddivisione dell'intervallo $[a, b] \in \mathbb{R}$ in punti equispaziati.

Sostiutiamo quindi $P_1(x)$ al posto di $f(x)$ in (4.1) ottenendo:

$$\begin{aligned} I \approx I_h &= \int_a^b P_1(x)\, dx = f(a)x|_a^b + \frac{f(b)-f(a)}{b-a}\, \frac{x^2}{2}\Big|_a^b \\ &= f(a)(b-a) + \frac{f(b)-f(a)}{b-a}\frac{b^2-a^2}{2} \\ &= (b-a)\frac{f(a)+f(b)}{2}. \end{aligned}$$

che è la cosidetta *formula dei trapezi*. Il nome deriva dall'interpretazione geometrica dell'integrale come il valore dell'area del trapezio di Figura 4.1.

L'errore che si commette è dato dall'integrale della formula del resto di Lagrange:

$$\begin{aligned} I - I_h &= \int_a^b (f(x) - P_1(x))\, dx = \int_a^b E(x)\, dx = \int_a^b F(x)\frac{f''(\eta)}{6}\, dx \\ &= \frac{f''(\xi)}{6}\int_a^b (x-a)(x-b)\, dx = -\frac{(b-a)^3}{12} f''(\xi). \end{aligned}$$

Riassumendo, la formula dei trapezi I_T e il suo errore E_T sono dati da:

$$I_T = (b-a)\frac{f(a)+f(b)}{2}; \tag{4.3}$$

$$E_T = -\frac{(b-a)^3}{12} f''(\xi). \tag{4.4}$$

Si osservi che, coerentemente con il grado del polinomio interpolatore, il metodo dei trapezi è esatto (errore nullo) se la funzione integranda fosse una retta. L'ordine del polinomio per cui la formula di quadratura è esatta è detto *ordine di accuratezza*.

4.1.2 Le Formule di Newton Cotes

Si può estendere il procedimento precedente a polinomi di ordine n. Ovviamente, si ha la libertà di scelta dei punti di appoggio: scegliendo punti equispaziati si ottengono le formule di Newton Cotes. Sia dunque $\{(x_j, f(x_j))\}$, $j = 0, \ldots, n$ l'insieme dei punti di appoggio distribuiti uniformemente nell'intervallo $[a, b]$, per cui $x_0 = a$, $x_n = b$, $x_j = x_0 + jh$, $h = (b-a)/n$ (si veda la Figura 4.2). Il generico punto

$x \in [a, b]$ può essere descritto da una coordinata s intrinseca all'intervallo $[0, n]$ per cui:

$$x = x_0 + sh \qquad\qquad s \in \mathbb{R} \text{ e } 0 \leq s \leq n.$$

Il polinomio di Lagrange che interpola i punti di appoggio è dato da:

$$P_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x),$$

e il valore approssimato dell'integrale è dato da:

$$I_h = \int_a^b P_n(x)\, dx.$$

Il calcolo dell'integrale è effettuato tramite il cambio di variabile da x a s. Per fare ciò osserviamo che per $x = a$ si ha $s = 0$, per $x = b$, $s = n$, e lo jacobiano della trasformazione è $|J| = h$. Otteniamo dunque:

$$I_h = \int_a^b P_n(x)\, dx = h \sum_{j=0}^{n} f(x_j) \int_0^n L_j^{(n)}(s)\, ds.$$

Essendo $h = (b - a)/n$, possiamo scrivere le formule di Newton-Cotes come:

$$I_h = (b - a) \sum_{j=0}^{n} C_j^{(n)} f(x_j), \qquad\qquad C_j^{(n)} = \frac{1}{n} \int_0^n L_j^{(n)}(s)\, ds, \tag{4.5}$$

dove $L_j^{(n)}(s)$ sono i polinomi di Lagrange di grado n e valgono:

$$L_j^{(n)}(s) = \frac{s(s-1)(s-2)\dots(s-j+1)(s-j-1)\dots(s-n+1)(s-n)}{j(j-1)(j-2)\dots(1)(-1)\dots(j-n+1)(j-n)}.$$

Possiamo quindi specializzare tali formule calcolando i valori dei coefficienti $C_j^{(n)}$ una volta scelto il valore di n.

Per $n = 1$ (polinomio di grado 1) otteniamo:

$$C_0^{(1)} = \int_0^1 L_0^{(1)}(s)\, ds = \int_0^1 \frac{s-1}{-1}\, ds = s - \frac{s^2}{2}\bigg|_0^1 = \frac{1}{2}$$
$$C_1^{(1)} = \int_0^1 L_1^{(1)}(s)\, ds = \int_0^1 s\, ds = \frac{s^2}{2}\bigg|_0^1 = \frac{1}{2}$$

e quindi si ottiene:

$$I_T = (b - a)\frac{f(a) + f(b)}{2},$$

che, come ci si aspettava, coincide con il metodo dei trapezi.

Per $n = 2$ si ottiene:

$$C_0^{(2)} = \frac{1}{2}\int_0^2 L_0^{(2)}(s)\,ds = \frac{1}{2}\int_0^2 \frac{(s-1)(s-2)}{-1(-2)}\,ds = \frac{1}{4}\left(\frac{s^3}{3} - \frac{3s^2}{2} + 2s\right)_0^2 = \frac{1}{6}$$
$$C_1^{(2)} = \frac{1}{2}\int_0^2 L_1^{(2)}(s)\,ds = \frac{1}{2}\int_0^2 \frac{s(s-2)}{1(-1)}\,ds = \frac{1}{2}\left(\frac{s^3}{3} - s^2\right)_0^2 = \frac{4}{6}$$
$$C_2^{(2)} = \frac{1}{2}\int_0^2 L_2^{(2)}(s)\,ds = \frac{1}{2}\int_0^2 \frac{s(s-1)}{2(1)}\,ds = \frac{1}{2}\left(\frac{s^3}{3} - \frac{s^2}{2}\right)_0^2 = \frac{1}{6},$$

da cui si ottiene la cosidetta formula di *Cavalieri-Simpson*:

$$I_{CS} = \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right], \tag{4.6}$$

e la conseguente formula per l'errore che riportiamo senza dimostrazione:

$$E_{CS} = -\frac{(b-a)^5}{2880} f^{iv}(\xi). \tag{4.7}$$

4.1.3 Errore delle formule di Newton-Cotes

Come si è visto per il caso della formula dei trapezi, è possibile esplicitare l'errore commesso dalle formule di Newton Cots a partire dal resto della Formula di Lagrange. I calcoli non sono però così agevoli come per il caso dei trapezi e non vengono riportati in queste note. Si riassumono soltanto i risultati per $n1 \div 4$ in tabella 4.1.

Dalla tabella si evidenzia un comportamento particolare: a parità di punti di appoggio le formule di ordine n pari sono più accurate. Per esempio, la formula dei trapezi è esatta per funzioni integrande che sono polinomi di grado al massimo 1 (rette), in coincidenza con il fatto che il polinomio interplatore è in questi casi esatto. La formula di Cavalieri Simpson, ottenuta sostituendo alla funzione integranda il polinomio interplatore di grado 2, è esatta fino a polinomi di grado 3, e così via. Questa osservazione apparentemente contraddittoria si spiega con l'uso del punto medio dell'intervallo $((a+b)/2)$ da parte delle formule di grado pari, e quindi con questioni di simmetria. In realtà, come vedremo più avanti, il punto centrale dell'intervallo è un punto *di Gauss* e in pratica l'uso di questo punto garantisce nelle formule di quadratura un'accuratezza più elevata, come se in quel punto avessimo utilizzato anche il valore della derivata prima della funzione integranda per calcolare il polinomio interpolatore. Se le formule di Newton-Cotes non utilizzassero punti equispaziati, non si avrebbe questo fenomeno, e gli errori seguirebbero l'esattezza del polinomio interpolatore.

$n = 1$ (Trapezi)	$C_0^1 = 1/2;\ C_1^1 = 1/2$	$-\frac{(b-a)^3}{12} f''(\xi)$
$n = 2$ (Cavalieri-Simpson)	$C_0^2 = 1/6;\ C_1^2 = 4/6$ $C_2^2 = 1/6$	$-\frac{(b-a)^5}{2880} f^{iv}(\xi)$
$n = 3$	$C_0^3 = 1/8;\ C_1^3 = 3/8$ $C_2^3 = 3/8\ C_3^3 = 1/8$	$-\frac{(b-a)^5}{c_3} f^{iv}(\xi)$
$n = 4$	$C_0^4 = 7/90;\ C_1^4 = 32/90$ $C_2^4 = 12/90\ C_3^4 = 32/90$ $C_4^4 = 7/90$	$-\frac{(b-a)^7}{c_4} f^{vi}(\xi)$

Tabella 4.1: Pesi ed errori per le formule di Newton-Cotes per $n = 1, 2, 3, 4$.

4.1.4 Formule composte

Nel capitolo precedente si è visto che l'interpolazione polinomiale con punti equispaziati è soggetta a problemi di stabilità e mal-condizionamento, come dimostrato nel fenomeno di Runge (esempio 3.1.5 nel paragrafo 3.1.3). Per porre rimedio a questo problema bisogna mantenere n piccolo, per esempio usando la proprietà di linearità dell'operatore di integrale e quindi la formula 4.2 in maniera sistematica.

Metodo dei trapezi composto

Partiamo con l'applicare la formula dei trapezi a ciascun intervallo $[x_{j-1}, x_j]$. Otteniamo:

$$I_{T,n} = \sum_{j=1}^{n} \frac{x_j - x_{j-1}}{2} \left[f(x_j) + f(x_{j-1}) \right].$$

Nel caso $h = x_j - x_{j-1} = cost = (b-a)/n$, la formula dei trapezi composta prende la forma semplificata:

$$\begin{aligned} I_{T,n} &= \frac{b-a}{2n} \left[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n) \right] \\ &= \frac{b-a}{2n} \sum_{j=1}^{n} \left[f(x_j) + f(x_{j-1}) \right]. \end{aligned} \tag{4.8}$$

Per analizzare l'errore commesso da tale formula si noti che la formula precedente commette ad ogni sottointervallo un errore dato da (4.4), per cui, indicando con

$\xi_j \in [x_{j-1}, x_j]$ un punto opportuno, si ottiene:

$$\begin{aligned} E_{T,n} &= -\sum_{j=1}^{n} \frac{(x_j - x_{j-1})^3}{12} f''(\xi_j) \\ &= -\frac{(b-a)^3}{12n^3} \sum_{j=1}^{n} f''(\xi_j) \\ &= -\frac{(b-a)^3}{12n^2} f''(\xi), \end{aligned} \tag{4.9}$$

dove l'ultima riga si è ottenuta osservando che, per il teorema del valor medio, esiste un punto $\xi \in [a, b]$ tale che:

$$f''(\xi) = \frac{1}{n} \sum_{j=1}^{n} f''(\xi_j).$$

La formula dei trapezi composta con suddivisione uniforme dell'intervallo $[a, b]$ ha quindi un errore (eq. 4.9) che tende a zero quadraticamente al tendere all'infinito del numero di sottointervalli, cioè per $h \to 0$, a patto che la derivata seconda della funzione integranda sia limitata.

Esempio 4.1.1. Calcolare l'integrale:

$$I = \int_1^2 x \log x \, dx.$$

con il metodo dei trapezi composto.

Usiamo la formula (4.8) con suddivisione uniforme dell'intervallo $[1, 2]$. La tabella seguente riporta i valori dell'integrale approssimato ($I_{T,n}$), l'errore $|I_{T,n} - I|$, e il rapporto $E_{T,n}/E_{T,2n}$ tra errori consecutivi al variare di h da 1 a 1/512 ($n = 1 \div 512$).

n	$I_{T,n}$	$\lvert I_{T,n} - I \rvert$	$E_{T,n}/E_{T,2n}$
1	6.9315E-01	5.6853E-02	1.7589E+01
2	6.5067E-01	1.4378E-02	3.9541E+00
4	6.3990E-01	3.6061E-03	3.9871E+00
8	6.3720E-01	9.0228E-04	3.9967E+00
16	6.3652E-01	2.2561E-04	3.9992E+00
32	6.3635E-01	5.6404E-05	4.0000E+00
64	6.3631E-01	1.4098E-05	4.0008E+00
128	6.3630E-01	3.5217E-06	4.0032E+00
256	6.3630E-01	8.7757E-07	4.0130E+00
512	6.3629E-01	2.1654E-07	4.0528E+00

Dalla tabella si vede che il valore calcolato dallo schema dei trapezi composto converge al valore vero quadraticamente (rapporto tra gli errori pari a 4). L'errore si comporta come previsto dalla teoria. Infatti, se $f''(x)$ non varia di molto, per cui $f''(\xi_j)$ è circa indipendente da j, si ha:

$$\frac{E_{T,n}}{E_{T,2n}} = \frac{(b-a)^3 f''(\xi_n)}{12n^2} \cdot \frac{12(2n)^2}{(b-a)^3 f''(\xi_{2n})} \approx 2^2 = 4$$

Ovviamente, dalla tabella si notano piccole oscillazioni del rapporto tra gli errori, come conseguenza delle variazioni, comunque piccole, della derivata seconda.

Esempio 4.1.2. In questo esempio verifichiamo come la mancanza di limitatezza nella derivata seconda possa prevenire la convergenza teorica del metodo.

Si calcoli il valore dell'integrale:

$$I = \int_0^5 f(x)\, dx = \int_0^5 (x-3)^{-2/3}\, dx.$$

In questo caso ci aspettiamo che l'errore sia comandato dal valore della derivata seconda in un intorno del punto di discontinuità della $f(x)$ e delle sue derivate. Procedendo nello stesso modo di prima calcoliamo la seguente tabella:

n	$I_{T,n}$	$\lvert I_{T,n} - I\rvert$	$E_{T,n}/E_{T,2n}$
1	2.7768E+00	5.3297E+00	1.8763E-01
2	5.3569E+00	2.7496E+00	1.9384E+00
4	5.0535E+00	3.0530E+00	9.0062E-01
8	6.4611E+00	1.6454E+00	1.8555E+00
16	6.2061E+00	1.9004E+00	8.6582E-01
32	7.0758E+00	1.0307E+00	1.8438E+00
64	6.9108E+00	1.1957E+00	8.6201E-01
128	7.4576E+00	6.4895E-01	1.8425E+00
256	7.3533E+00	7.5317E-01	8.6163E-01
512	7.6977E+00	4.0879E-01	1.8424E+00
1024	7.6320E+00	4.7446E-01	8.6159E-01
2048	7.8490E+00	2.5752E-01	1.8424E+00
4096	7.8076E+00	2.9889E-01	8.6159E-01
8192	7.9443E+00	1.6223E-01	1.8424E+00

La tabella mostra come anche con $h = 2^{-13}$ ($n = 8192$) l'errore è estremamente elevato. Questo è l'effetto dell'errore che si commette vicino alla discontinuità della funzione integranda, che tende a $+\infty$ per $x = 3$, e della sua derivata seconda.

Si noti che lo schema dei trapezi composto funziona e non dà problemi numerici (overflow) solo perché le suddivisioni dell'intervallo di integrazione sono scelte in modo tale che nessun punto di appoggio dove la formula dei trapezi richiede di calcolare la funzione coincide con $x = 3$.

Esempio 4.1.3. Calcolare l'integrale:

$$I = \int_0^1 x \log x \, dx.$$

con il metodo dei trapezi composto.

In questo caso, l'estremo $x = 0$ non è un punto di discontinuità per la funzione integranda, ma il calcolo all'elaboratore richiede il calcolo di $\log x$ per $x \to 0^+$. Un modo per procedere numericamente è quello di definire un intervallo di integrazione con estremo inferiore > 0, ad esempio $[0.1, 1]$, e diminuiamo l'estremo di sinistra progressivamente verificando come varia il valore numerico ottenuto. Operativamente, definiamo l'intervallo di integrazione $[a, b]$ con $a = 0.1/10^r$ e $b = 1$, e andiamo a vedere al variare di r come varia il valore dell'integrale. Nella tabella seguente si riportano i risultati numerici ottenuti per $n = 2048$ suddivisioni. Nelle diverse colonne si mostrano rispettivamente i diversi valori dell'estremo sinistro dell'intervallo (a), del valore dell'integrale ottenuto con il metodo dei trapezi ($I_{T,2048}$), l'errore vero ($|I_{T,2048} - I|$) e il rapporto tra errori consecutivi $E_{T,1024}/E_{T,2048}$.

a	$I_{T,2048}$	$\lvert I_{T,2048} - I\rvert$	$E_{T,1024}/E_{T,2048}$
1.0000E-01	-2.3599E-01	1.4125E-07	4.1480E+00
1.0000E-03	-2.5000E-01	5.4024E-07	4.0167E+00
1.0000E-05	-2.5000E-01	7.7013E-07	3.7513E+00
1.0000E-07	-2.5000E-01	7.8769E-07	3.7212E+00
1.0000E-09	-2.5000E-01	7.8809E-07	3.7204E+00
1.0000E-11	-2.5000E-01	7.8809E-07	3.7204E+00

Si nota che lo schema converge al valore teorico dell'integrale rispettando le previsioni teoriche dell'andamento dell'errore.

Metodo di Cavalieri-Simpson composto

Procedendo nello stesso modo, su si ricava il metodo di Cavalieri-Simpson composto. Applichiamo quindi lo schema (4.6) su ogni suddivisione dell'intervallo $[a, b]$. In questo caso, però, bisogna tenere conto che se si suddivide l'intervallo in n suddivisioni, il numero di punti di appoggio diventa è $m + 1$ con $m = 2n$. Sommando i termini uguali e semplificando, otteniamo:

$$\begin{aligned} I_{CS,n} &= \frac{b-a}{6n} \left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) \ldots + 2f(x_{m-2}) + 4f(x_{m-1}) + f(x_m) \right] \\ &= \frac{b-a}{6n} \left[f(x_0) + 2 \sum_{j=1}^{m/2-1} f(x_{2j}) + 4 \sum_{j=1}^{m/2} f(x_{2j-1}) + f(x_m) \right]. \end{aligned} \tag{4.10}$$

Nello stesso modo con cui si è ricavato l'errore del metodo dei trapezi composto, osservando che la (4.7) vale per ognuno degli n sottointervalli e applicando il teorema del valor medio, si arriva a:

$$E_{CS,n} = -\frac{(b-a)^5}{2880n^4} f^{iv}(\xi). \tag{4.11}$$

Si vede quindi che la convergenza del metodo di Cavalieri-Simpson è di ordine 4, e quindi raddoppia rispetto al metodo dei trapezi, sempre assumendo che le derivate quarte della funzione integranda siano limitate.

Metodo di estrapolazione di Richardson

Il metodo di estrapolazione di Richardson è una tecnica generale che sfrutta il calcolo di due approssimazioni diverse per dare una stima dell'errore. Nello specifico, sia I il valore vero dell'integrale, $I_{T,n}$ il valore approssimato di I calcolato con il metodo dei trapezi suddividendo l'intervallo di integrazione $[a, b]$ in n suddivisioni di passo h. Il valore dell'integrale vero sarà pari al valore approssimato più l'errore:

$$I = I_{T,n} + E_{T,n}.$$

Scrivendo la formula precedente per n e per $2n$ e sottraendo membro a membro otteniamo:

$$\begin{aligned} I &= I_{T,2n} + E_{T,2n} && (4.12)\\ I &= I_{T,n} + E_{T,n} && (4.13)\\ \hline 0 &= I_{T,2n} - I_{T,n} + E_{T,2n} - E_{T,n}. && (4.14) \end{aligned}$$

Dalla formula dell'errore del metodo dei trapezi composto (eq. (4.9)), assumendo le derivate seconde uguali per n e per $2n$, si ha:

$$E_{T,n} = 4E_{T,2n},$$

otteniene una stima dell'errore:

$$E_{T,2n} \approx \frac{I_{T,2n} - I_{T,n}}{3},$$

potendo quindi ottenere una approssimazione dell'errore quando sostituito in (4.12), fornisce una più accurata approssimazione dell'integrale:

$$I_{R,T} = \frac{4I_{T,2n} - I_{T,n}}{3}. \tag{4.15}$$

Con un po' di conti elementari si scopre che questa approssimazione coincide con la formula di Cavalieri-Simpson.

La stessa procedura si può utilizzare partendo dal metodo di Cavalieri-Simpson, ottenendo, con uso di notazioni simile al precedente:

$$I_{R,CS} = \frac{16 I_{CS,2n} - I_{CS,n}}{15}.$$

Bibliografia

[1] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985 (IEEE 754)*. New York, 1985.

[2] G. Gambolati. *Lezioni di metodi numerici per ingegneria e scienze applicate*. Cortina, Padova, Italy, 2 edition, 2002. 619 pp.

[3] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[4] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, Berlin, Heidelberg, second edition, 2007.

[5] Y. Saad. *Iterative Methods for Sparse Linear Systems. Second edition*. SIAM, Philadelphia, PA, 2003.

[6] Y. Saad. *Numerical Methods for large eigenvalue problems. Second edition*. SIAM, Philadelphia, PA, 2011.

[7] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[8] Sun Microsystems, Inc. *Numerical Computation Guide, Sun ONE Studio 8*, part no. 817-0932-10 edition, May 2003.

[9] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[10] A. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, volume 42, 1936.

[11] J. von Neumann. First draft of a report on the edvac. Technical report, Moore School of Electrical Engineering, University of Pennsylvania, 1945.