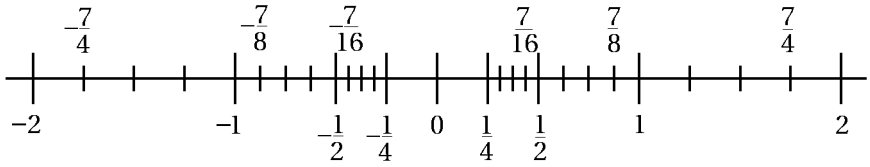
ARGOMENTI DEL CORSO CALCOLO NUMERICO

A.A. 2009/10



I Numeri Finiti e l'Aritmetica Floating Point

Giulio Casciola

(settembre 2003, rivista e corretta settembre 2008)

Indice

1 L'aritmetica computazionale **1**
1.1 Rappresentazione dei numeri reali 3
1.1.1 Errori di rappresentazione 7
1.2 Aritmetica floating-point 8
1.2.1 Standard IEEE 10
1.3 Propagazione degli errori 11
1.3.1 Analisi in avanti degli errori 11
1.3.2 Analisi all'indietro degli errori 12
1.3.3 Errore nelle operazioni di macchina 13
1.3.4 Errori numerici più comuni 14
1.3.5 Errore Inerente, Algoritmico e Analitico 15
1.3.6 Alcuni esempi 18
1.3.7 Limiti 20
1.4 L'indecidibilità dell'eguaglianza dei numeri floating-point . 20

2 Un esempio concreto **23**
2.1 Rette nel piano 2D 24
2.1.1 Equazione parametrica di una retta 24
2.1.2 Equazione implicita di una retta 25
2.1.3 Conversione fra equazione parametrica e implicita . 25
2.2 Distanza di un punto da una retta 26
2.2.1 Confronto algoritmi distanza punto/retta 28
2.3 Intersezione di segmenti: condizionamento 32
2.4 Intersezione di segmenti: metodi e stabilità 34
2.4.1 Intersezione rette parametrica/implicita 34
2.4.2 Intersezione rette parametrica/parametrica 34
2.4.3 Intersezione rette implicita/implicita 36
2.4.4 Intersezione di segmenti 36
2.5 Intersezione di segmenti: test numerici 40

Bibliografia **45**

Capitolo 1

L'aritmetica computazionale

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

- *dal modello reale a quello matematico*: in tutte le discipline scientifiche, ogni volta che si introduce un modello matematico per descrivere un oggetto o fenomeno reale, si introduce un'approssimazione necessaria a limitare il numero di fattori da considerare altrimenti ingestibili; per esempio, in un software CAD le curve e le superfici utilizzate non sono che un'approssimazione di quelle degli oggetti reali; nei modellatori poligonali le superfici continue che rappresentano, ad esempio, la carrozzeria di un'automobile, sono approssimate mediante una serie di piccole facce piane unite tra loro (vedi Fig. 1.1))
- *dal modello matematico alla sua rappresentazione computazionale*: i modelli matematici descritti teoricamente non sempre sono rappresentabili esattamente e i problemi relativi risolubili in modo esplicito ed esatto mediante un computer; scopo del *Calcolo Numerico* è proprio studiare gli effetti delle approssimazioni introdotte in questi passaggi. Nella presente trattazione porremo particolare attenzione a modelli e problemi che derivano dal settore della CAGD (Computer Aided Geometric Design).

La precisione con cui un modello matematico può essere implementato dipende dalle sue particolarità: più precisamente, essendo il computer una

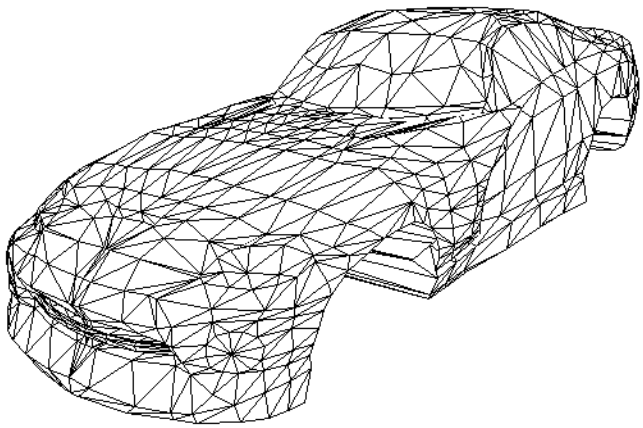


Figura 1.1: Modello poligonale

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

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

1.1 Rappresentazione dei numeri reali

Alla radice dell'impossibiltità di implementare in modo esatto i modelli matematici basati su un insieme continuo di simboli, sta la rappresentazione dei numeri reali utilizzata nei computer.
Questi, essendo macchine "finite", devono operare su numeri rappresentati mediante una sequenza finita di cifre; ad esempio, numeri come π o $\sqrt{2}$, la cui rappresentazione richiede infinite cifre, non possono che essere rappresentati se non in modo approssimato, commettendo cioè un errore. Inoltre, poiché l'insieme dei numeri rappresentabili è finito e quindi limitato, non potranno essere rappresentati numeri arbitrariamente piccoli o arbitrariamente grandi.
Un ulteriore problema nasce dalla base di rappresentazione; generalmente l'inserimento di valori numerici attraverso una interfaccia o shell avviene utilizzando la base decimale, usata naturalmente dall'uomo, mentre la rappresentazione interna al computer è binaria. Questo porta all'introduzione di approssimazioni in modo subdolo; ad esempio inserendo il numero decimale 0.1 questo viene memorizzato internamente in modo approssimato perché la sua rappresentazione binaria $0.00110011\ldots$ è periodica.

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

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

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

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

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

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

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

Gli n bit (o posizioni) disponibili per la memorizzazione di un numero finito vengono suddivisi tra le t cifre della mantissa e l'esponente p che può assumere $\omega - \lambda + 1$ configurazioni diverse, più un bit per il segno del numero. Nel caso della rappresentazione binaria, sarà sempre $\alpha_1 = 1$, per cui può essere sottointeso senza mai essere fisicamente rappresentato.
Alcune tipiche rappresentazioni sono:

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

In precisione singola vengono destinati 24 bit alla mantissa (in realtà solo 23) e 8 all'esponente ($2^8 = 256 = \omega - \lambda + 1$, con $\lambda = -128$ e $\omega = 127$), mentre in precisione doppia le cifre della mantissa sono 53 (rappresentati 52 bit) e dell'esponente 11 ($2^{11} = 2048 = \omega - \lambda + 1$, con $\lambda = -1024$ e $\omega = 1023$).

Si noti che la normalizzazione della mantissa permette di sfruttare al me-

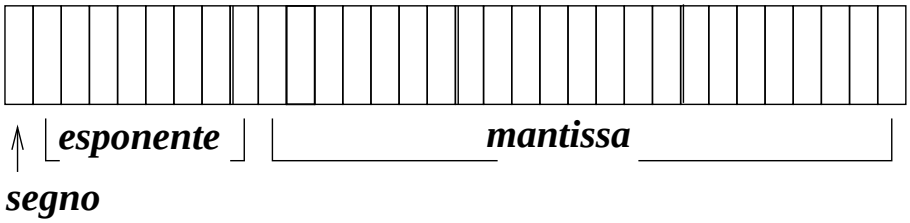


Figura 1.2: Rappresentazione binaria dei numeri finiti

glio le cifre disponibili, evitando di rappresentare esplicitamente gli 0 non significativi. Ad esempio 0.000124 viene rappresentato come 0.124×10^{-3}. La virgola quindi non ha una posizione fissa tra la parte intera e quella non, ma varia; per questo motivo tale rappresentazione è detta *floating-point* o a virgola mobile.

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

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

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

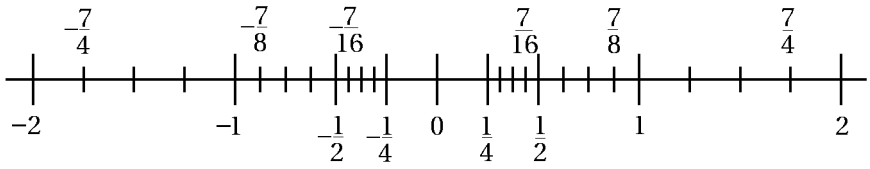


Figura 1.3: Discretizzazione della retta reale

Esempio 1.1 La figura 1.3 è ricavata impostando i valori $\beta = 2,\ t = 3$, da cui deriva il seguente insieme di numeri:

$$\{0.100 \times 2^p,\ 0.101 \times 2^p,\ 0.110 \times 2^p,\ 0.111 \times 2^p\}$$

dove $0.100,\ 0.101,\ 0.110,\ 0.111$ sono tutte le possibili mantisse e p il valore dell'esponente.
Sono rappresentati gli intervalli per $p = -1 :\ [\frac{1}{4}, \frac{7}{16}],\ p = 0 :\ [\frac{1}{2}, \frac{7}{8}],\ p = 1 :\ [1, \frac{7}{4}],\ \ldots$.
Risulta evidente l'aumento dell'ampiezza degli intervalli definiti dal valore p, in ognuno dei quali vengono localizzati i 4 valori della mantissa; ne risulta di conseguenza una diradazione delle suddivisioni verso gli estremi sinistro e destro della retta reale.

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

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

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

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

  Troncamento di α alla t-esima cifra

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

  Arrotondamento di α alla t-esima cifra

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

Quindi

$$\begin{array}{lll} \text{se } \alpha_{t+1} < \frac{\beta}{2}, & \text{allora si ha} & fl_A(\alpha) = fl_T(\alpha) \\ \text{se invece } \alpha_{t+1} \geq \frac{\beta}{2}, & \text{allora si ha} & fl_A(\alpha) = fl_T(\alpha) + \beta^{p-t} \end{array}$$

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

$$x \leq \alpha < y$$

allora

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

e risulta

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

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

Esempio 1.2 Definito l'insieme dei numeri finiti $F_{(10,5,-50,49)}$ resta definito il numero di posizioni (bit nel caso di base 2) necessarie per rappresentare in memoria un numero finito dell'insieme. Nel caso specifico saranno 1 per il segno (0 se positivo e $\beta - 1$ se negativo), 2 per l'esponente (si usa la tecnica di memorizzazione per traslazione, cioè nei due campi

per l'esponente si memorizzano i valori da 00 a 99 intendendo gli esponenti da -50 a 49) e 5 posizioni per la mantissa (si memorizza a partire da sinistra). Vediamo qualche esempio numerico:

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

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

$\alpha = 0.05302$	04953020
$\alpha = -237.141$	95323714
$\alpha = -0.00321665$	94832167

1.1.1 Errori di rappresentazione

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

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

$$\left|\frac{\tilde{\alpha} - \alpha}{\alpha}\right| \text{ se } \alpha \neq 0 \quad \text{errore relativo}$$

La rappresentazione discreta della retta reale descritta in precedenza (vedi Figura 1.3) è tale che fornisce un errore relativo di rappresentazione massimo costante per ogni α, mentre quello assoluto, di conseguenza, aumenta proporzionalmente al valore di α.
Banalmente si deduce che l'entità più informativa è l'errore relativo in quanto l'errore assoluto dipende da α; per cui, nella pratica, quando possibile si userà sempre l'errore relativo.

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

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

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

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

Teorema 1.3 *Per ogni* $\alpha \in \mathbb{R}$ *e* $\alpha \neq 0$ *vale* [1]*:*

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

La quantità u a seconda della modalità di approssimazione usata, limita superiormente il modulo dell'errore relativo, ed è anche detta *precisione di macchina* nel fissato sistema floating-point. La sua importanza numerica è data dalla seguente caratterizzazione:

u è il più piccolo numero finito positivo tale che

$$fl(u + 1) > 1$$

Questo implica che per ogni numero finito $v < u$ sarà $fl(v + 1) = 1$.
L'errore relativo che si commette nel rappresentare il numero reale α con un numero finito $fl(\alpha)$ lo indicheremo con ϵ e lo chiameremo *errore di rappresentazione*:

$$\epsilon = \frac{fl(\alpha) - \alpha}{\alpha} \tag{1.4}$$

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

$$fl(\alpha) = \alpha(1 + \epsilon), \qquad con\ |\epsilon| < u \tag{1.5}$$

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

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

da questo si ha poi che definendo

$$\epsilon = \frac{\alpha - fl(\alpha)}{fl(\alpha)} \qquad vale \qquad fl(\alpha) = \frac{\alpha}{1 + \epsilon} \qquad con\ |\epsilon| < u. \tag{1.6}$$

1.2 Aritmetica floating-point

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

[1] Con la funzione $fl()$, d'ora in poi, s'indende una generica approssimazione del suo argomento secondo uno dei due criteri definiti nelle 1.2 e 1.3

Esempio 1.3 Siano 0.1×10^{-3} e $0.1 \times 10^{3} \in F_{(10,3,\lambda,\omega)}$. Eseguendo la somma si ha:

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

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

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

> se $\tilde{op}$ è l'operazione di macchina che approssima l'operazione esatta op, per tutti i numeri finiti x, y per cui l'operazione non dia luogo a condizioni di overflow o di underflow, risulti:
>
> $$x\tilde{op}\, y = fl(x op\, y)$$
>
> o equivalentemente usando l'espressione 1.5
>
> $$x\tilde{op}\, y = (x op\, y)(1+\epsilon), \qquad |\epsilon| < u \qquad (1.7)$$
>
> dove u è la precisione di macchina introdotta.

Questo equivale a dire che il risultato dell'operazione macchina deve essere uguale all'approssimazione con un numero finito del risultato dell'operazione esatta. ϵ è l'errore relativo commesso nell'operazione.

Si noti che per tali operazioni non valgono molte delle proprietà fondamentali dell'aritmetica come l'associativa, di cancellazione, ecc...

Esempio 1.4 Siano $F_{(10,2,\lambda,\omega)}$, $a = 0.11 \times 10^{0}$, $b = 0.13 \times 10^{-1}$, $c = 0.14 \times 10^{-1}$
In aritmetica floating-point con arrotondamento:

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

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

per cui:

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

1.2.1 Standard IEEE

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

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

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

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

d. se $p = 0$ e $m \neq 0$ allora $v = (-1)^s (0.m) \times 2^{p-128}$ (Not Normalized)

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

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

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

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

I dettagli del formato **extended** precisone **single** sono lasciati all'implementatore, e viene solo fatta la richiesta minima di 1 bit per il segno, almeno 32 bit per la mantissa e un esponenete che può essere BIASED e deve soddisfare $\lambda \leq -1023$ ed $\omega \geq 1024$.
Il formato **basic** precisone double è analogo alla precisione **single**. 1 bit per il segno, 11 bit per l'esponente e 52 bit per la mantissa per un totale di 64 bit.
Nel formato **extended** precisione **double** si usano almeno 63 bit per la mantissa.
Un'implementazione dello standard fornisce le operazioni di addizione, sottrazione, moltiplicazione, divisione e radice quadrata, così come conversioni binario-decimale e viceversa. Tranne che per le conversioni, tutte le operazioni forniscono un risultato corrispondente all'approssimazione floating-point del risultato teorico o a precisione infinita.
La modalità standard di arrotondamento coincide con quella precedentemente descritta con la particolarità dell'**arrotondamento ai pari** quando un reale α è esattamente equidistante dai due numeri finiti x ed y consecutivi (questo si verifica quando $\alpha_{t+1} = \beta/2$ e $\alpha_i = 0$ per $i > t+1$); in questa situazione l'arrotondamento comporta l'incremento di α_t di 1 se α_t è dispari, ma di non incrementarlo se è pari.
Nel caso di arrotondamento ai pari u viene caratterizzato come il più grande numero finito positivo tale che $fl(u+1) = 1$.

1.3 Propagazione degli errori

Al di là degli errori introdotti dalle singole operazioni i problemi maggiori nascono nell'esecuzione di una sequenza di queste nella quale, ad ogni passo, vengono utilizzati come input i valori precedentemente calcolati. In questo caso si verifica una *propagazione degli errori* la cui entità molto spesso è tutt'altro che trascurabile.
Il controllo e la gestione di tale fenomeno risulta fondamentale in tutte le applicazioni informatiche che implementano in modo approssimato modelli matematici teorici, al fine di determinare l'attendibilità dei risultati ottenuti. La teoria della propagazione degli errori fornisce alcuni strumenti di base per questo scopo.

1.3.1 Analisi in avanti degli errori

Per valutare l'entità della propagazione degli errori in una sequenza di operazioni si considera l'errore totale dato dalla somma dei singoli errori. Data una funzione esatta $f(x)$ per ogni suo coefficiente ed operazione si

introduce un fattore d'errore relativo che rappresenta l'approssimazione introdotta dall'aritmetica finita (espressioni 1.5, 1.7) e si ricava una corrispondete funzione approssimata $\tilde{f}(x)$.
La quantità

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

fornisce un'indicazione sull'entità dell'errore relativo totale che affligge la valutazione di $f(x)$ (vedi Fig. 1.4).

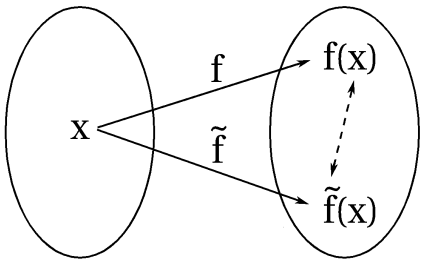


Figura 1.4: Analisi in avanti dell'errore

Esempio 1.5 Come esempio per chiarire questo tipo di analisi degli errori si consideri l'addizione di due numeri x ed y finiti; avremo

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

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

1.3.2 Analisi all'indietro degli errori

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

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

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

quindi:

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

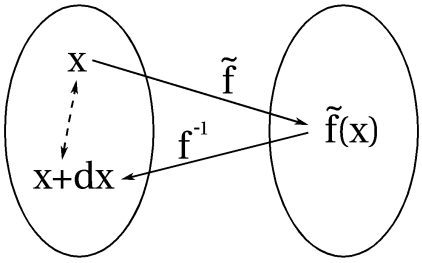


Figura 1.5: Analisi all'indietro dell'errore

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

1.3.3 Errore nelle operazioni di macchina

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

moltiplicazione

$$\begin{aligned}\frac{fl(fl(x)fl(y)) - xy}{xy} &= \frac{x(1+\epsilon_1)y(1+\epsilon_2)(1+\epsilon_3) - xy}{xy} \\ &= \frac{xy + xy(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_1\epsilon_2 + \epsilon_1\epsilon_3 + \epsilon_2\epsilon_3 + \epsilon_1\epsilon_2\epsilon_3) - xy}{xy} \\ &\simeq \epsilon_1 + \epsilon_2 + \epsilon_3 < 3u\end{aligned}$$

dove $|\epsilon_1|, |\epsilon_2|, |\epsilon_3| < u$ e le semplificazioni sono ottenute conducendo un'analisi del primo ordine[2] .
Risulta una propagazione lineare degli errori, per cui l'operazione di moltiplicazione è stabile.

divisione

$$\begin{aligned}\frac{fl(fl(x)/fl(y)) - x/y}{x/y} &= \frac{x(1+\epsilon_1)\frac{(1+\epsilon_2)}{y}(1+\epsilon_3) - x/y}{x/y} \\ &= (1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3) - 1 \\ &\simeq \epsilon_1 + \epsilon_2 + \epsilon_3 < 3u\end{aligned}$$

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

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

addizione e sottrazione

$$\begin{aligned}&\frac{fl(fl(x) \pm fl(y)) - (x \pm y)}{x \pm y}\\ &= \frac{[x(1+\epsilon_1) \pm y(1+\epsilon_2)](1+\epsilon_3) - (x \pm y)}{x \pm y}\\ &\simeq \frac{x}{x \pm y}\epsilon_1 \pm \frac{y}{x \pm y}\epsilon_2 + \epsilon_3\end{aligned}$$

Nel caso dell'addizione (sottrazione) di due numeri x, y con segno concorde (discorde) l'errore finale è maggiorato da $3u$.
Altrimenti non è possibile dare una maggiorazione dell'errore relativo indipendente da x, y. In particolare se $(x \pm y) \to 0$ la limitazione dell'errore risulta molto elevata.
In questi casi, l'elevato errore che si può ritrovare nel risultato non è generato dall'operazione aritmetica in floating-point (infatti l'errore locale dell'operazione ϵ_3 ha modulo minore di u), ma è dovuto alla presenza degli errori relativi non nulli ϵ_1, ϵ_2 su x, y che sono amplificati dalle operazioni aritmetiche di addizione e sottrazione.
Tale fenomeno viene definito di *cancellazione numerica* (vedi anche paragrafo 1.3.4) ed è senza dubbio la conseguenza più grave della rappresentazione con precisione finita dei numeri reali.

Esempio 1.7 Sia $F_{(10,6,\lambda,\omega)}$ con rappresentazione per arrotondamento e siano dati i numeri reali $a = 0.147554326$ ed $b = -0.147251742$. La loro approssimazione nell'insieme F sarà: $fl(a) = 0.147554$ e $fl(b) = 0.147252$. L'addizione esatta darà: $a + b = 0.302584 \times 10^{-3}$, mentre in aritmetica finita darà: $fl(fl(a)+fl(b)) = 0.302000 \times 10^{-3}$. L'errore relativo commesso sarà: $\epsilon \simeq 0.2 \times 10^{-2}$, mentre $u = 0.5 \times 10^{-5}$ comportando un grave errore.

1.3.4 Errori numerici più comuni

Nei paragrafi precedenti sono state descritte la rappresentazione dei numeri finiti e l'aritmetica floating-poing.
Da queste approssimazioni derivano una serie di errori che vanno tenuti in considerazione per valutare l'attendibilità dei numeri floating-point risultato di un problema. Nell'elenco che segue, si cerca di dare una descrizione

di quelli più comuni(negli esempi si sottointende l'insieme dei numeri finiti $F_{(10,5,\lambda,\omega)}$):

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

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

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

$$\begin{array}{r} 100.0 \quad + \\ 0.0003 \\ \hline 100.0003 \end{array}$$

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

Errore di cancellazione numerica : la differenza di due numeri a e b quasi uguali ha meno cifre significative rispetto sia ad a che b. Per esempio $0.90905 \times 10^2 - 0.90903 \times 10^2 = 0.2 \times 10^{-2}$. La mantissa risultante 0.20000 , a meno che a e b siano esatti, contiene quattro zeri non significativi. Infatti essi non derivano da un calcolo, ma sono dovuti alla perdita di precisione (vedi anche paragrafo 1.3.3).

1.3.5 Errore Inerente, Algoritmico e Analitico

In questo paragrafo si formalizza ulteriormente l'esame della propagazione dell'errore esaminando il caso generico della valutazione di una funzione. Le funzioni effettivamente calcolabili su un computer sono solo le *funzioni razionali*, il cui valore è ottenuto mediante un numero finito di operazioni

aritmetiche. Le funzioni non razionali, come ad esempio le trigonometriche, possono solo essere approssimate da opportune funzioni razionali.
In generale, data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ ed un vettore $x = (x_1, x_2, \ldots x_n)$ di dati in input, comprendente le variabili pure e i coefficenti numerici che definiscono f [3] , il valore effettivamente calcolato può essere affetto da errori indotti da diversi fenomeni:

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

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

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

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

Teorema 1.4 *Siano gli $x_i \neq 0$, $i = 1, 2, \ldots, n$, e sia $f(x)$ una funzione razionale tale che $f(x) \neq 0$ e $f(\tilde{x}) \neq 0$. Indicati con*

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

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

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

*l'*errore inerente *e con*

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

*l'*errore algoritmico*, risulta*

$$\epsilon_{tot} = \epsilon_{alg}(1 + \epsilon_{in}) + \epsilon_{in}. \tag{1.8}$$

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

Applicando l'analisi del primo ordine (vedi nota a pagina 13) la 1.8 risulta così semplificata:

$$\epsilon_{tot} \simeq \epsilon_{alg} + \epsilon_{in}. \tag{1.9}$$

Se $f(x)$ è differenziabile in un intorno di x si ha che

$$\epsilon_{in} \simeq \sum_{1=1}^{n} c_i \epsilon_i \tag{1.10}$$

dove

$$c_i = \frac{x_i}{f(x)} \frac{\partial f(x)}{\partial x_i} \tag{1.11}$$

con $\epsilon_i = \frac{\tilde{x}_i - x_i}{x_i}$, e $|\epsilon_i| < u$, per $i = 1, 2, \ldots, n$.

I coefficienti c_i sono detti *coefficienti di amplificazione* e danno una misura di quanto influisce l'errore relativo ϵ_i, da cui è affetto il dato x_i, sul risultato: se i coefficienti c_i sono di modulo elevato, anche piccoli errori ϵ_i inducono grossi errori su $f(x)$; in questo caso il calcolo di $f(x)$ è detto *mal condizionato.*
Tale errore, come evidente dalla formula 1.11, dipende unicamente dalla funzione $f(x)$ ed in particolare dalla sua rappresentazione simbolica, mentre *i coefficenti associati alle variabili pure sono costanti al variare dell'espressione usata per la f.* L'errore algoritmico ϵ_{alg}, invece, è generato dal calcolo della funzione $\tilde{f}(\tilde{x})$ come composizione di un numero finito di operazioni di macchina. Poiché gli errori delle singole operazioni sono in modulo minori della precisione di calcolo, al primo ordine si può stimare

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

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

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

Con procedimenti equivalenti a quelli usati in precedenza risulta

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

1.3.6 Alcuni esempi

Si applica la stima 1.10 nei casi già visti di moltiplicazione, divisione e addizione/sottrazione fra numeri reali:

moltiplicazione $f(x_1, x_2) = x_1 x_2$; applicando la 1.11 si ha

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

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

$$\epsilon_{in} = \epsilon_1 + \epsilon_2 \quad \text{mentre} \quad \epsilon_{alg} = \epsilon_3.$$

divisione $f(x_1, x_2) = x_1/x_2$; applicando la 1.11 si ha

$$c_1 = \frac{x_1}{x_1/x_2} \frac{1}{x_2} = 1 \qquad c_2 = \frac{x_2}{x_1/x_2} \frac{-x_1}{x_2^2} = -11$$

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

$$\epsilon_{in} = \epsilon_1 - \epsilon_2 \quad \text{mentre} \quad \epsilon_{alg} = \epsilon_3.$$

addizione/sottrazione $f(x_1, x_2) = x_1 + x_2$; applicando la 1.11 si ha

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

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

$$\epsilon_{in} = \frac{x_1}{x_1 + x_2} \epsilon_1 + \frac{x_2}{x_1 + x_2} \epsilon_2 \quad \text{mentre} \quad \epsilon_{alg} = \epsilon_3.$$

Si esamina il problema della valutazione della seguente espressione:

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

Si vede immediatamente che per $|x_2| << |x_1|$ si incorre in un errore di cancellazione numerica e quindi di instabilità del procedimento (infatti anche se x_1 e x_2 sono numeri finiti, non lo saranno $\sqrt{x_1 + x_2}$ e $\sqrt{x_1}$. In

tal caso si cerca un algoritmo differente che eviti il problema. Nel caso in questione, razionalizzando si ha:

$$y = (\sqrt{x_1 + x_2} - \sqrt{x_1}) \frac{\sqrt{x_1 + x_2} + \sqrt{x_1}}{\sqrt{x_1 + x_2} + \sqrt{x_1}}$$
$$= \frac{x_2}{\sqrt{x_1 + x_2} + \sqrt{x_1}}.$$

L'espressione trovata è esente dal possibile errore segnalato; infatti a denominatore si effettua una addizione fra quantità non negative; l'operazione pericolosa è stata eseguita analiticamente.
Si osserva però che per $x_1 + x_2 \to 0$ anche la versione più stabile dà risultati scadenti; qual è la causa? analizziamo l'errore inerente utilizzando la stima 1.10: nel caso specifico sarà $\epsilon_{in} = c_1\epsilon_1 + c_2\epsilon_2$ con c_1 e c_2 calcolati tramite la 1.11; facendo i conti si ottiene:

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

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

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

  $$y = 0.4999875 \times 10{-}6$$
  $$y_1 = 0.4994869 \times 10{-}6$$
  $$y_2 = 0.4999875 \times 10{-}6$$

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

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

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

  che mostra come entrambi i risultati calcolati siano scadenti.

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

1.3.7 Limiti

La teoria sulla propagazione dell'errore, pur essendo fondamentale per capire l'essenza del problema, presenta molti limiti che la rendono inutilizzabile nella maggior parte dei casi concreti.
Primo fra tutti, la pesantezza del procedimento con cui si ricava l'espressione dell'errore.
Si è visto nei vari esempi come l'analisi anche di semplici problemi, comporti una notevole quantità di calcoli. Pensando ad un software composto da varie centinaia di funzioni che dipendono in modo complesso l'una dall'altra si capisce come una analisi del genere sia impraticabile (per una trattazione completa dei procedimenti per ricavare le espressioni dell'errore si consulti [BBCM92], paragrafi 2.8 - 2.10).
Un ulteriore limite è dovuto al fatto che le espressioni ricavate rappresentano una maggiorazione dell'errore, e quindi una previsione pessimistica. Gli errori che si verificano nell'esecuzione reale delle varie operazioni possono risultare molto più limitati, e quindi essere mal rappresentati dai valori massimi calcolati.
Per ottenere valutazioni dell'errore più aderenti ai calcoli reali esistono metodi che seguono i vari passi di esecuzione a run-time, considerando i valori numerici effettivamente in gioco.

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

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

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

Esempio 1.8

- il calcolo della soluzione di un'equazione

$$f(x) = 0$$

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

$$|f(x_i)| < \epsilon \tag{1.12}$$

dove ϵ *non può essere scelto arbitrariamente piccolo*, perchè, a causa degli errori di arrotondamento, la condizione 1.12 potrebbe non essere mai soddisfatta (per maggiori dettagli si rimanda a [BBCM92] capitolo 3).

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

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

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

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

I valori approssimati a 3 cifre decimali sono

$$\sin 28 = 0.469 \qquad cos28 = 0.883$$

Si ottiene quindi

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

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

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

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

questo non è più perfettamente uguale a zero.

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

dati due numeri floating-point a e b, pur disponendo di una maggiorazione dell'errore assoluto ϵ su di essi, risulta che:

- se $|a - b| > \epsilon$ si può affermare con sicurezza che $a \neq b$
- se $|a - b| \leq \epsilon$ non si può affermare con sicurezza né che $a = b$ né che $a \neq b$.

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

Dati due numeri floating-point a e b si definisce

$a = b$ se $|a - b| < tol$
dove tol è una tolleranza assoluta

oppure

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

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

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

In questo modo si ottengono dei miglioramenti immediati, infatti settando la tolleranza alcuni ordini di grandezza maggiore della precisione di macchina disponibile (vedi paragrafo 1.1.1), gli algoritmi risultano, in molti casi, insensibili agli errori di arrotondamento.
In molti software vengono definite più tolleranze ϵ, una per le misure di lunghezza, una per le misure angolari, una per le misure di aree, etc. Questa differenziazione evidenzia una delle difficoltà d'implementazione di questo approccio: *definire a priori un valore per la tolleranza ϵ che porti ad un comportamento ottimale degli algoritmi.*
Quindi, non esistendo il valore ottimale per ogni situazione, si ricorre ad un insieme di valori definiti arbitrariamente, di solito in base ad una serie di test che verificano il comportamento del programma al variare di questi (processo di tuning).

Capitolo 2

Un esempio concreto: l'intersezione di segmenti

Trovare il punto comune fra due segmenti è un calcolo che viene eseguito numerose volte in un pacchetto CAD (Computer Aided Design) o in Computer Graphics. Si consideri per esempio la Fig.2.1 a sinistra dove sono mostrati molti segmenti che si intersecano. Allo scopo di colorare alcune delle regioni da essi definiti, come mostra la Fig.2.1 a destra, è necessario conoscere i punti di intersezione. Problemi di intersezioni sono presenti in molte altre applicazioni e non solo fra segmenti [FaPr87]; questo capitolo si limiterà a trattare questo problema base per far capire che numericamente anche un problema all'apparenza semplice può essere complesso e come si debba procedere per progettare un algoritmo stabile.
Iniziamo con alcuni richiami di geometria elementare che, se già noti, possono essere velocemente scorsi o addirittura saltati.

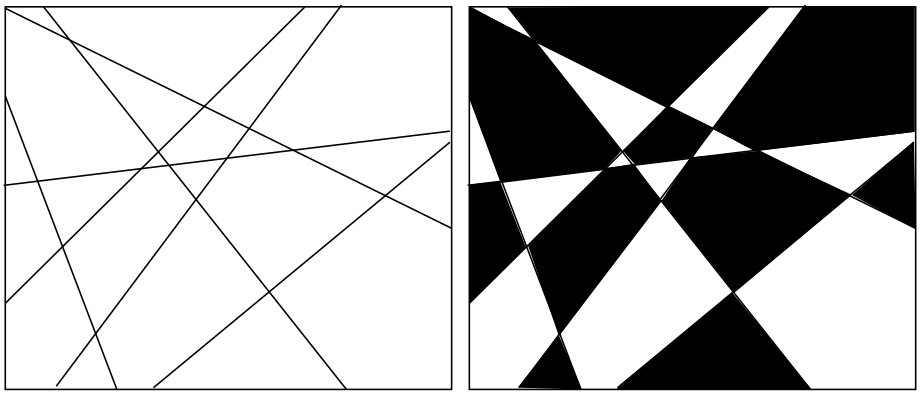


Figura 2.1: Esempio di intersezione

2.1 Rette nel piano 2D

Una retta del piano può essere data o definita in differenti modi (vedi Fig.2.2):

1. due punti;
2. un punto ed un vettore unitario parallelo alla retta;
3. un punto ed un vettore unitario perpendicolare alla retta.

Un vettore unitario perpendicolare alla retta è detto normale alla retta. Questi tre differenti modi di dare una retta implicano la sua definizione mediante tre diverse rappresentazioni.

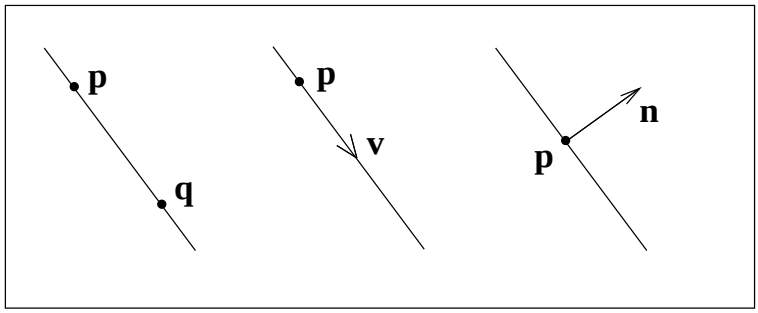


Figura 2.2: Rappresentazioni rette

2.1.1 Equazione parametrica di una retta

L'equazione parametrica di una retta $\mathbf{r}(t)$ ha la forma

$$\mathbf{r}(t) = \mathbf{p} + t\mathbf{v} \tag{2.1}$$

dove $\mathbf{p} \in E^2$ piano Euclideo e $\mathbf{v} \in \mathbb{R}^2$ piano reale. Il valore t è il parametro. La valutazione della (2.1) per uno specifico parametro $t = \hat{t}$, genera un punto sulla retta. Interpretando $\mathbf{v}$ come la differenza di due punti, $\mathbf{v} = \mathbf{q} - \mathbf{p}$, questa equazione può essere riformulata come

$$\mathbf{r}(t) = (1 - t)\mathbf{p} + t\mathbf{q}. \tag{2.2}$$

L'equazione parametrica di una retta può quindi essere della forma (2.1) o (2.2); la (2.2) è molto utile per definire l'equazione di un segmento di estremi $\mathbf{p}$ e $\mathbf{q}$ limitando il parametro $t \in [0, 1]$. Si osservi che per $t \in [0, 1]$

la $\mathbf{r}(t)$ è una combinazione convessa. Come già detto, la forma parametrica è molto comoda per calcolare punti della retta, infatti è sufficiente valutarla in corrispondenza di un assegnato parametro t per averne un suo punto. Questo comporta dei calcoli floating point e quindi ci si deve attendere di ottenere un punto che sia solo un'aprossimazione di un punto della retta.

2.1.2 Equazione implicita di una retta

Un altro modo per rappresentare la stessa retta è usare un'equazione implicita. Per questa rappresentazione, si parte da un punto $\mathbf{p}$ e un vettore $\mathbf{n}$ normale alla retta; allora per ogni $\mathbf{x}$ sulla retta vale

$$\mathbf{n} \cdot (\mathbf{x} - \mathbf{p}) = 0 \tag{2.3}$$

Infatti se $\mathbf{x}$ soddisfa questa equazione, significa che $\mathbf{n}$ ed il vettore $\mathbf{x} - \mathbf{p}$ sono perpendicolari. Se $\mathbf{n}$ è unitario la (2.3) è detta forma normale della retta. Sviluppando questa equazione, si ottiene

$$n_1 x_1 + n_2 x_2 - n_1 p_1 - n_2 p_2 = 0.$$

Questa si trova comunemente scritta come

$$a x_1 + b x_2 + c = 0 \tag{2.4}$$

dove

$$a = n_1 \qquad b = n_2 \qquad c = -n_1 p_1 - n_2 p_2. \tag{2.5}$$

L'equazione (2.4) è detta equazione implicita della retta. La forma implicita è molto utile per decidere se un punto appartiene alla retta. Per trovare se un punto $\mathbf{x}$ è sulla retta, basta sostituire le sue coordinate nella (2.4); se il valore dell'equazione è zero, il punto appartiene alla retta. Questo calcolo verrà effettuato in aritmetica finita con numeri finiti, per cui per l'indecidibilità dell'uguaglianza dei numeri floating point questo non avverrà quasi mai e si dovrà prevedere un test di uguaglianza a meno di una prefissata tolleranza.

2.1.3 Conversione fra equazione parametrica e implicita

Entrambe le rappresentazioni presentano vantaggi e svantaggi. A seconda dell' algoritmo geometrico, può essere più conveniente usare una forma piuttosto che l'altra. Vediamo prima la conversione da parametrica ad

implicita.
Data la retta $\mathbf{r}(t)$ in forma parametrica (2.1) o (2.2) con $\mathbf{v} = \mathbf{q} - \mathbf{p}$ si vogliono trovare i coefficienti a, b e c che definiscono l'equazione implicita (2.4).
Si determina un vettore $\mathbf{n}$ che sia perpendicolare al vettore $\mathbf{v}$. Scegliendo

$$\mathbf{n} = \begin{pmatrix} -v_2 \\ v_1 \end{pmatrix}$$

questo permette di calcolare, come già visto, a, b e c dalle (2.5).
Vediamo ora la conversione da implicita a parametrica.
Abbiamo bisogno di un punto sulla retta e di un vettore parallelo alla retta; il vettore si trova semplicemente come

$$\mathbf{v} = \begin{pmatrix} b \\ -a \end{pmatrix}$$

per trovare un punto sulla retta, si può cercare fra i punti di intersezione della retta con gli assi:

$$\begin{pmatrix} -c/a \\ 0 \end{pmatrix} \quad \textit{oppure} \quad \begin{pmatrix} 0 \\ -c/b \end{pmatrix}.$$

Per stabilità numerica, si sceglie l'intersezione più vicina all'origine. Così si scelga il primo se $|a| > |b|$, altrimenti il secondo.

2.2 Distanza di un punto da una retta

In questo paragrafo vogliamo analizzare un problema molto frequente in pratica e che sarà utile per affrontare il problema dell'intersezione fra due segmenti.
Data una retta r in forma implicita (2.4) e cioè definita dai coefficienti a, b e c e un punto $\mathbf{x}$, si vuole trovare $ds(r, \mathbf{x})$, o per brevità ds, distanza con segno del punto dalla retta. Sarà

$$ds = \frac{ax_1 + bx_2 + c}{\sqrt{a^2 + b^2}}$$

o a partire dall'equazione vettoriale (2.3)

$$ds = \frac{\mathbf{n} \cdot (\mathbf{x} - \mathbf{p})}{\|\mathbf{n}\|}.$$

Si osserva che se deve essere calcolata la distanza di molti punti rispetto ad una stessa retta o comunque controllare se molti punti appartengono alla retta o meno, risulta vantaggioso memorizzare la retta in forma normale; questo significa che $\sqrt{a^2 + b^2} = \|\mathbf{n}\| = 1$, evitando quindi la divisione.
Si vuole ora affrontare lo stesso problema, ma a partire dalla retta in forma parametrica (2.1) che risulta la situazione più frequente nelle applicazioni pratiche.
Una prima soluzione, che chiameremo **alg1**, può essere descritto dai seguenti passi (vedi Fig. 2.3) dove $\mathbf{v}$ è unitario;

1. determinare $\mathbf{w} := \mathbf{x} - \mathbf{p}$;
2. calcolare il prodotto scalare $t := \mathbf{w} \cdot \mathbf{v}$ (in questo modo si ottiene la norma della proiezione del vettore $\mathbf{w}$ sulla retta a partire da $\mathbf{p}$);
3. calcolare d come

$$d := \sqrt{\|\mathbf{w}\|^2 - t^2} = \sqrt{w_1^2 + w_2^2 - t^2}.$$

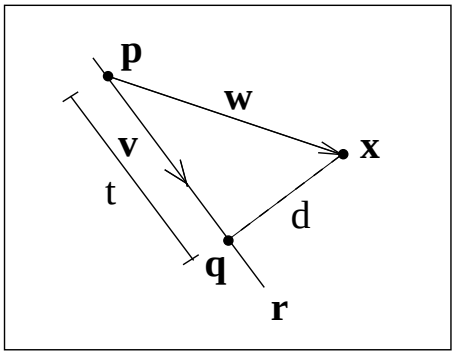


Figura 2.3: Distanza di un punto da una retta

Alternativamente al passo 3. indicato si possono eseguire i due seguenti passi, ottenendo un procedimento che chiameremo **alg2**:

3. calcolare il vettore $\mathbf{q} := t\mathbf{v}$ (cioè il vettore proiezione di $\mathbf{w}$ sulla retta $\mathbf{r}$);
4. calcolare d come

$$d := \|\mathbf{x} - \mathbf{q}\| = \sqrt{(x_1 - q_1)^2 + (x_2 - q_2)^2}.$$

Si noti che in questi ultimi casi resta determinata la distanza del punto dalla retta, ma senza segno. Un modo per attribuire il segno alla distanza può essere il seguente: verificato che $d > \epsilon_{dis}$ (dove con ϵ_{dis} intendiamo una prefissata tolleranza che discrimina se il punto appartiene o meno alla retta), si calcola il prodotto scalare fra $\mathbf{w}$ ed $\mathbf{n} := (-v_2, v_1)^T$; se questo risulta positivo $ds := +d$, se negativo $ds := -d$.
Nel prossimo paragrafo confronteremo, su esempi test, questi due differenti algoritmi in termini di stabilià numerica.

2.2.1 Confronto algoritmi distanza punto/retta

I due algoritmi proposti nella sezione precedente sono *perfettamente equivalenti dal punto di vista geometrico*, ma, come evidenzierà il seguente confronto, le pur lievi differenze nella sequenza dei calcoli eseguiti, producono valori numerici finali differenti.
Il confronto è basato su un semplicissimo test: si applicano ripetutamente i due algoritmi, calcolando la distanza tra una retta di riferimento costante $\mathbf{r}$ e un insieme di punti $\mathbf{x}$, creati volutamente vicini a questa. Si ritiene infatti tale condizione critica rispetto a dei punti lontani (si noti infatti che più $\mathbf{x}$ è vicino ad $\mathbf{r}$ più i vettori $\mathbf{v}$ e $\mathbf{w}$ tendono al parallelismo).
Le posizioni reciproche tra $\mathbf{r}$ ed i punti $\mathbf{x}$ sono state scelte in modo da poter calcolare la distanza in modo preciso ed alternativo ad **alg1** e **alg2**, condizione necessaria per poter valutare l'affidabiltà dei due algoritmi.
Le tabelle successive riportano le seguenti informazioni ottenute elaborando in $F_{(2,24,-1024,1023)}$:

- *Distanza Es.*: la distanza esatta
- *Distanza Calc.*: la distanza calcolata
- *Err. Rel.*: l'errore relativo.

Nel **Test 1** si è impostata $\mathbf{r}$ coincidente con l'asse x e i punti $\mathbf{x}$ sono generati fissando un valore per x_1 e variando x_2, da un valore iniziale, fino a zero, condizione quest'ultima per cui $\mathbf{x} \in \mathbf{r}$.
Banalmente x_2 rappresenta la distanza esatta d, come evidente in Fig. 2.4.
I risultati riportati nelle tabelle 2.1 e 2.2 sono un sottoinsieme significativo dei risultati ottenuti eseguendo questo test.
Nel **Test 2** la retta $\mathbf{r}$ corrisponde all'asse di simmetria del quadrante xy.
I punti $\mathbf{x}$ sono calcolati come nel test precedente, cioè fissando un valore x_1 e variando x_2 da un valore iniziale, fino a $\bar{x_2} \equiv x_1$, condizione per cui

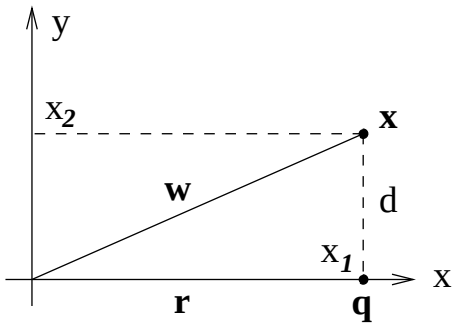


Figura 2.4: Distanza punto/retta: Test 1

Distanza Es.	Distanza Calc.	Err. Rel.
0.30517578125000e-05	0.30517199772859e-05	0.12397842966555e-04
0.15258789062500e-05	0.15258963684484e-05	0.11444026313950e-04
0.76293945312500e-06	0.76272987676082e-06	0.27469593205265e-03
0.38146972656250e-06	0.38107370890809e-06	0.10381365199700e-02
0.19073486328125e-06	0.19082797281244e-06	0.48816209888253e+00
0.95367431640625e-07	0.94243218307745e-07	0.11788231197380e-01
0.47683715820313e-07	0.47121609153872e-07	0.11788231197380e-01
0.23841857910156e-07	0.21073424255447e-07	0.11611652351682e+00
0.11920928955078e-07	0.00000000000000e+00	1.00000000000000e+00
0.59604644775391e-08	0.00000000000000e+00	1.00000000000000e+00

Tabella 2.1: Distanza punto/retta: risultati di **alg1** sul Test1

Distanza Es.	Distanza Calc.	Err. Rel.
0.30517578125000e-05	0.30517578125000e-05	0.00000000000000e+00
0.15258789062500e-05	0.15258789062500e-05	0.00000000000000e+00
0.76293945312500e-06	0.76293945312500e-06	0.00000000000000e+00
0.38146972656250e-06	0.38146972656250e-06	0.00000000000000e+00
0.19073486328125e-06	0.19073486328125e-06	0.00000000000000e+00
0.95367431640625e-07	0.95367431640625e-07	0.00000000000000e+00
0.47683715820313e-07	0.47683715820313e-07	0.00000000000000e+00
0.23841857910156e-07	0.23841857910156e-07	0.00000000000000e+00
0.11920928955078e-07	0.11920928955078e-07	0.00000000000000e+00
0.59604644775391e-08	0.59604644775391e-08	0.00000000000000e+00

Tabella 2.2: Distanza punto/retta: risultati di **alg2** su Test1.

$\mathbf{x} \in \mathbf{r}$, come evidente in figura 2.5.
La distanza d è calcolata con precisione come

$$d = (x_2 - \bar{x_2})/\sqrt{2}.$$

I risultati riportati nelle tabelle 2.3 e 2.4 sono un sottoinsieme significativo dei risultati ottenuti eseguendo questo test.

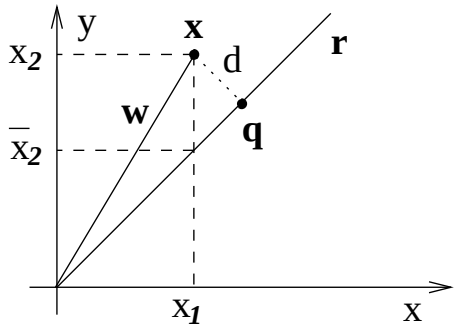


Figura 2.5: Distanza punto/retta: Test 2

Distanza Es.	Distanza Calc.	Err. Rel.
0.53947966093944e-05	0.53948789268299e-05	0.15258672649057e-04
0.26973983046972e-05	0.26974806215047e-05	0.30517112478053e-04
0.13486991523486e-05	0.13488637784282e-05	0.12206286282879e-03
0.67434957617430e-06	0.67500779944099e-06	0.97608612793553e-03
0.33717478808715e-06	0.33848931466343e-06	0.38986502630600e-02
0.16858739404358e-06	0.16989937798666e-06	0.77822185373200e-02
0.84293697021788e-07	0.91856926723852e-07	0.89724735885170e-01
0.42146848510894e-07	0.42146848510894e-07	0.15700900000000e-15
0.21073424255447e-07	0.00000000000000e+00	1.00000000000000e+00
0.10536712127724e-07	0.21073424255447e-07	1.00000000000000e+00

Tabella 2.3: Distanza punto/retta: risultati di **alg1** su Test2.

Da tutti i risultati numerici riportati nelle tabelle *appare evidente la maggior precisione offerta dal secondo algoritmo* in entrambi i test.
In particolare si nota che le differenze dipendono dagli errori che si accumulano diversamente a seconda delle specifiche operazioni aritmetiche utilizzate.
I primi due passi sono identici, per cui la differenza nasce sicuramente nei passi 3 e 4.

Distanza Es.	Distanza Calc.	Err. Rel.
0.53947966093944e-05	0.53947966093944e-05	0.00000000000000e+00
0.26973983046972e-05	0.26973983046972e-05	0.00000000000000e+00
0.13486991523486e-05	0.13486991523486e-05	0.00000000000000e+00
0.67434957617430e-06	0.67434957617430e-06	0.00000000000000e+00
0.33717478808715e-06	0.33717478808715e-06	0.00000000000000e+00
0.16858739404358e-06	0.16858739404358e-06	0.00000000000000e+00
0.84293697021788e-07	0.84293697021788e-07	0.00000000000000e+00
0.42146848510894e-07	0.42146848510894e-07	0.00000000000000e+00
0.21073424255447e-07	0.21073424255447e-07	0.00000000000000e+00
0.10536712127724e-07	0.10536712127724e-07	0.00000000000000e+00

Tabella 2.4: Distanza punto/retta di **alg2** su Test2.

alg1 la norma di $\mathbf{w}$ ed il valore t sono interpretati come l'ipotenusa ed un cateto di un triangolo rettangolo, quindi, in base al teorema di Pitagora, viene calcolato l'altro cateto corrispondente a d.
Si consideri il primo caso di test. Dai risultati numerici si osserva che il valore della distanza calcolato tende a zero più velocemente della distanza reale: ciò è dovuto al calcolo di $\|\mathbf{w}\|^2$ al passo 3. Infatti, $\|\mathbf{w}\|^2$ è ottenuto come la somma dei quadrati $w_1^2 + w_2^2$, dove $w_1 >> w_2$ (w_2 tende a zero come x_2). Elevando questi ultimi valori al quadrato, la differenza si accentua maggiormente, finché non si verifica un *errore di assorbimento* (paragrafo 1.3.4).
Risulta quindi che, nella somma finale non appare il contributo di w_2^2. Essendo tale valore equivalente alla distanza d calcolata, anch'essa risulta nulla.

alg2 a differenza dell'altro algoritmo, al passo 3, viene calcolato il vettore $\mathbf{q}$ proiezione di $\mathbf{x}$ su $\mathbf{r}$, quindi, al passo 4, la distanza d^2 è calcolata come $\|\mathbf{x} - \mathbf{q}\|^2$.
Le singole differenze, presenti in quest'ultima equazione coinvolgono direttamente i valori delle componenti x e y dei vettori che sono mantenute distinte fino all'ultimo passo. Il quadrato è calcolato solo alla fine, sulle singole differenze, che sono dimensionalmente simili (perchè nei test lo sono anche i vettori $\mathbf{w}$ e $\mathbf{v}$). Non si hanno quindi errori di assorbimento [1].

[1] Un esempio simile si può trovare in [BBCM92], dove si dimostra che è più stabile il calcolo $(x - y)(x + y)$ rispetto a $(x^2 - y^2)$

Riferendosi sempre al primo caso di test si noti che: la componente x_2, al passo 4, è l'unica che non viene annullata dalla relativa componente di $\mathbf{q}$ cioè q_2. Viene quindi rappresentata al massimo della precisione disponibile. Infatti, nella tabella 2.2 il calcolo della distanza risulta esatto.

2.3 Intersezione di segmenti: condizionamento del problema

In tutti i testi di Analisi Numerica, come semplice esempio di problema mal condizionato si riporta l'intersezione di due rette, e solitamente viene mostrato come l'amplificazione dell'errore inerente risulti proporzionale al loro grado di parallelismo. Difficilmente si trova invece l'osservazione che l'intersezione può essere calcolata con buona precisione anche se le due rette sono quasi parallele.
Infatti il punto di fondamentale importanza è che la valutazione delle due rette sia affidabile in prossimità del punto d'intersezione.
Nella Figura 2.6 si osserva come una piccola variazione del coefficiente

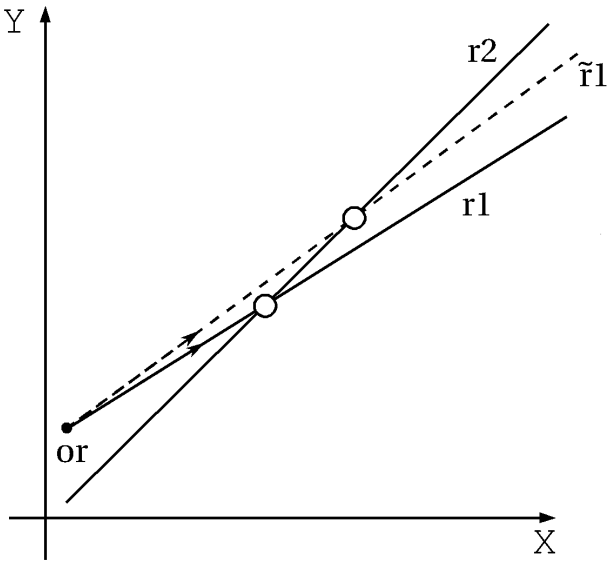


Figura 2.6: Intersezione mal condizionata

angolare provoca grossi errori nel calcolo dell'intersezione, mentre con le stesse rette, rappresentate con l'origine coincidente con il punto d'intersezione (Figura 2.7), non si ha nessun errore.

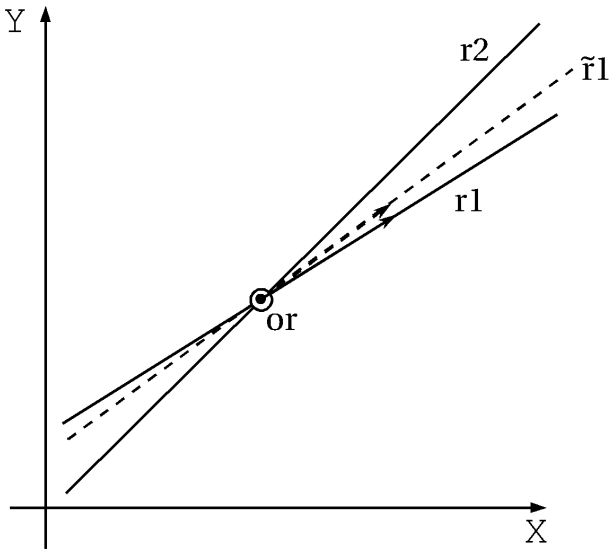


Figura 2.7: Intersezione ben condizionata

Si noti inoltre che le rette sono state volutamente impostate vicine al parallelismo, condizione che, come noto, amplifica gli errori di rappresentazione. Infatti, lo stesso esempio, con rette quasi-ortogonali produce un errore finale molto più contenuto.
Purtroppo, assegnate le rette, non è detto che le loro origini siano proprio nell'intorno del punto di intersezione, ed anche conoscendo a priori il punto di intersezione, passare dalla rappresentazione assegnata della retta ad una con l'origine traslata potrebbe non essere sufficiente. Dopo questa breve digressione sull'intersezione di due rette, torniamo al condizionamento del problema dell'intersezione fra due segmenti.

Assumiamo di possedere i due segmenti mediante i loro punti estremi, ossia di avere due espressione della forma (2.2). A partire da tale formulazione possiamo essere sufficientemente certi che il nostro problema sarà ben condizionato, nel senso che, individuato il caso in cui le due rette sono parallele entro una certa tolleranza, tal per cui non sarebbe possibile determinare un punto di intersezione, negli altri casi il punto verrà calcolato a partire da una definizione che garantisce che a piccole variazione dei dati (gli estremi dei due segmenti) si ottengano piccole variazione sul risultato (il punto di intersezione) in quanto i dati sono locali al risultato, ossia la valutazione dei due segmenti è affidabile in prossimità del punto di intersezione.

2.4 Intersezione di segmenti: metodi e stabilità

Cominciamo questo paragrafo con un breve richiamo sui metodi più semplici per determinare l'intersezione fra due rette. Questo perché, una volta stimato che fra i due segmenti c'è intersezione, questa verrà determinata calcolando l'intersezione fra le rette dei due segmenti.

2.4.1 Intersezione rette parametrica/implicita

Le due rette siano date rispettivamente nelle forme (2.1) e (2.4) e cioè:

$$r_1 : \mathbf{r}(t) = \mathbf{p} + t\mathbf{v} \qquad r_2 : ax_1 + bx_2 + c = 0. \tag{2.6}$$

Il problema può essere risolto trovando lo specifico parametro $\hat{t}$, rispetto alla retta $\mathbf{r}(t)$ del punto di intersezione. Questo punto, quando inserito nella seconda equazione delle (2.6), porta ad una equazione in una incognita.

$$a(p_1 + \hat{t}v_1) + b(p_2 + \hat{t}v_2) + c = 0$$

e risolvendo per $\hat{t}$,

$$\hat{t} = \frac{-c - ap_1 - bp_2}{av_1 + bv_2};$$

allora il punto di intersezione $\mathbf{x}$ cercato sarà dato dat $\mathbf{r}(\hat{t})$.
Si noti che nel calcolare $\hat{t}$ mediante la formula data, il denominatore non solo non deve essere nullo, ma neppure più piccolo di una certa toleranza, per evitare problemi numerici. Si noti che il denominatore altro non è che il prodotto scalare fra i vettori $(a, b)^T$ e $\mathbf{v}$ dove $(a, b)^T$ rappresenta un vettore ortogonale alla retta in forma implicita e $\mathbf{v}$ un vettore parallelo alla retta in forma parametrica; per cui denominatore nullo vuol dire che le due rete sono parallele, caso che deve essere già stato escluso prima di arrivare a questa fase.

2.4.2 Intersezione rette parametrica/parametrica

Le due rette siano date rispettivamente nella forma (2.1) ossia

$$r_1 : \mathbf{r}_1(t) = \mathbf{p} + t\mathbf{v} \qquad r_2 : \mathbf{r}_2(s) = \mathbf{q} + s\mathbf{w}. \tag{2.7}$$

Il problema può essere risolto trovando i due valori parametrici $\hat{t}$ ed $\hat{s}$ tali che

$$\mathbf{p} + \hat{t}\mathbf{v} = \mathbf{q} + \hat{s}\mathbf{w}.$$

Questa può essere scritta come

$$\hat{t}\mathbf{v} - \hat{s}\mathbf{w} = \mathbf{q} - \mathbf{p}.$$

Si hanno due equazioni in due incognite $\hat{t}$ ed $\hat{s}$. Trovate queste, il punto di intersezione $\mathbf{x}$ viene calcolato sostituendo uno dei due parametri nella corrispondente espressione della retta. Nel risolvere esplicitamente il sistema delle due equazioni si trova una espressione il cui denominatore è $w_1v_2 - w_2v_1$ e che corrisponde al prodotto scalare fra i vettori $(-w_2, w_1)^T$ e $\mathbf{v}$ che sono rispettivamente nella direzione ortogonale della seconda e nella direzione della prima retta; tale denominatore indica quindi, se nullo, che le due rette sono parallele e su tale valore valgono le stesse considerazioni fatte nel paragrafo precedente.

Un algoritmo alternativo che presentiamo, sfrutta la primitiva distanza punto/retta. Assumiamo che le due rette siano definite in forma parametrica da due coppie di punti; più precisamente sia $\mathbf{r}_1(t)$ definita da $\mathbf{p}_1$ e $\mathbf{q}_1$ non coincidenti e $\mathbf{r}_2(s)$ definita da $\mathbf{p}_2$ e $\mathbf{q}_2$ non coincidenti. Mediante la primitiva distanza con segno punto/retta si determinano $ds_1 := ds(\mathbf{p}_1, \mathbf{r}_2)$ e $ds_2 := ds(\mathbf{q}_1, \mathbf{r}_2)$.

- Se una o entrambe le distanze, considerate in valore assoluto, sono inferiori ad una certa tolleranza, l'intersezione è uno dei due estremi o le due rette sono sovrapposte.

- Se le due distanze con segno sono uguali, a meno di una certa tolleranza, le due rette sono parallele.

Superati questi controlli preliminari e certi quindi dell'esistenza di un punto di intersezione, questo può essere determinato procedendo mediante una semplice proporzione sui triangoli di vertici $\mathbf{p}_1$, $\mathbf{q}_1$, C e $\mathbf{p}_1$, D, E (vedi Fig. 2.8 sinistra), cioè:

$$\|\mathbf{p}_1 - D\| : \|\mathbf{p}_1 - \mathbf{q}_1\| = |ds_1| : (|ds_1| + |ds_2|)$$

da cui

$$\|\mathbf{p}_1 - D\| = \frac{\|\mathbf{p}_1 - \mathbf{q}_1\||ds_1|}{|ds_1| + |ds_2|}.$$

Se definiamo la retta per $\mathbf{p}_1$ e $\mathbf{q}_1$ come:

$$\mathbf{r}_1(t) = \mathbf{p}_1 + t\frac{\mathbf{q}_1 - \mathbf{p}_1}{\|\mathbf{q}_1 - \mathbf{p}_1\|}$$

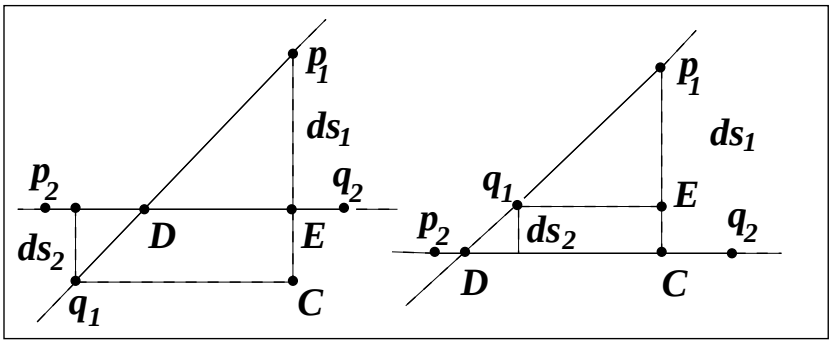


Figura 2.8: Intersezione rette

il punto di intersezione si troverà prendendo per t il valore $\|\mathbf{p}_1 - D\|$.
Si osservi che il procedimento funziona anche nel caso in cui l'intersezione non sia interna ai segmenti che definiscono le due rette; in questo caso ds_1 e ds_2 saranno dello stesso segno (vedi Fig. 2.8 destra) e in questo caso dovrà essere:

$$\|\mathbf{p}_1 - D\| = \frac{\|\mathbf{p}_1 - \mathbf{q}_1\| \, |ds_1|}{| \, |ds_1| - |ds_2| \, |}.$$

Se richiesto, si può calcolare il parametro s del punto di intersezione per la retta $\mathbf{r}_2(s)$, procedendo nello stesso modo.

2.4.3 Intersezione rette implicita/implicita

Le due rette siano date rispettivamente nella forma (2.4) ossia

$$r_1 : a_1x_1 + b_1x_2 + c_1 = 0 \qquad r_2 : a_2x_1 + b_2x_2 + c_2 = 0. \tag{2.8}$$

Il problema può essere risolto trovando il punto $\mathbf{x}$ che contemporaneamente soddisfa le equazioni delle due rette. Nel risolvere esplicitamente il sistema delle due equazioni si trova una espressione il cui denominatore è $a_1b_2 - a_2b_1$ e che corrisponde al prodotto scalare fra i vettori $(b_2, -a_2)$ e $(a_1, b_1)^T$ che sono rispettivamente nella direzione ortogonale alla prima e nella direzione della seconda retta; tale denominatore indica, se nullo, che le due rette sono parallele e su tale valore valgono le stesse considerazioni fatte precedentemente.

2.4.4 Intersezione di segmenti

Iniziamo questo paragrafo ricordando che la soluzione del problema in oggetto consiste nell'intersezione fra i due segmenti se questa esiste, ma

anche nell'informazione di non intersezione se non esiste ed eventualmente nel sottosegmento dei due se questi si sovrappongono ritornando gli estremi (vedi Fig. 2.9 per una casistica di posizioni dei due segmenti). Si

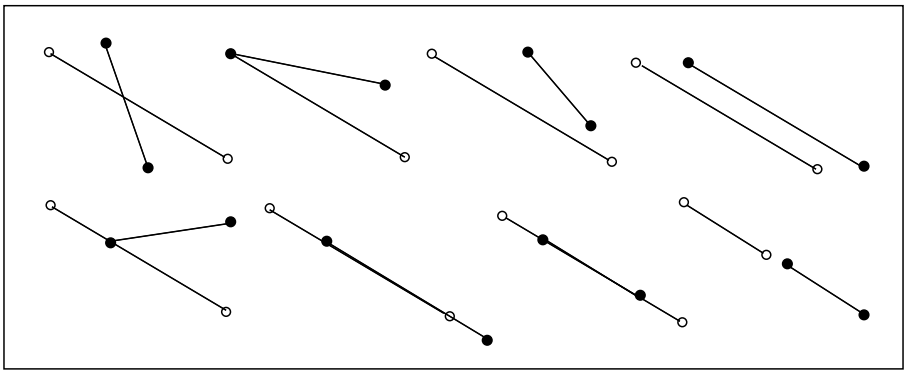


Figura 2.9: Differenti casi di intersezione fra segmenti

considera un primo metodo di intersezione fra due segmenti di estremi $\mathbf{p}_1$, $\mathbf{q}_1$ e $\mathbf{p}_2$, $\mathbf{q}_2$ rispettivamente. Si considerino le rette dei due segmenti date come

$$\mathbf{r}_1(t) = \mathbf{p}_1 + \mathbf{v}_1 t \quad \text{con} \quad \mathbf{v}_1 = \frac{\mathbf{q}_1 - \mathbf{p}_1}{\|\mathbf{q}_1 - \mathbf{p}_1\|}$$

$$\mathbf{r}_2(s) = \mathbf{p}_2 + \mathbf{v}_2 s \quad \text{con} \quad \mathbf{v}_2 = \frac{\mathbf{q}_2 - \mathbf{p}_2}{\|\mathbf{q}_2 - \mathbf{p}_2\|}.$$

Un possibile algoritmo, espresso in pseudo-codifica, è il seguente:

se $ds(\mathbf{p}_1, \mathbf{r}_2)ds(\mathbf{q}_1, \mathbf{r}_2) < \epsilon_{dis}^2$ **e** $ds(\mathbf{p}_2, \mathbf{r}_1)ds(\mathbf{q}_2, \mathbf{r}_1) < \epsilon_{dis}^2$ **allora**

- **se** $|\mathbf{v}_1 \cdot \mathbf{v}_2| < \epsilon_{ang}$ **allora** $\mathbf{r}_1$ è parallela a $\mathbf{r}_2$
  - **se** $d(\mathbf{p}_2, \mathbf{r}_1) < \epsilon_{dis}$ **allora ritorna** ($\mathbf{r}_1, \mathbf{r}_2$ sono coincidenti) perché parallele e con un punto in comune
  - **altrimenti ritorna** (nessuna intersezione) perché parallele e distinte
- **altrimenti** calcola il punto d'intersezione $\mathbf{x}$ e **ritorna**($\mathbf{x}$)

altrimenti ritorna (nessuna intersezione)

dove ϵ_{ang} e ϵ_{dis} sono rispettivamente la tolleranza per le misure angolari e di distanza.
Il primo test assicura che fra i due segmenti c'è intersezione, mentre il

secondo test sul parallelismo nasce dalla giusta considerazione di evitare il calcolo del punto d'intersezione tra rette quasi parallele (situazione chiamata in letteratura degenere). Purtroppo, anche se la sequenza di deduzioni seguita dall'algoritmo è perfettamente valida nel modello ideale, l'utilizzo di un'aritmetica approssimata introduce alcuni problemi.
Si consideri la figura 2.10 dove sono rappresentate le due rette $\mathbf{r}_1, \mathbf{r}_2$ con i rispettivi spessori di tolleranza trateggiati. Si supponga che il test $|\mathbf{v}_1 \cdot \mathbf{v}_2| < \epsilon_{ang}$ risulti verificato, a questo punto sono possibili due situazioni:

- l'ordine di intersezione è $\mathbf{r}_1, \mathbf{r}_2$: viene verificato il test $d(\mathbf{p}_2, \mathbf{r}_1) < \epsilon_{dis}$ che risulta vero, per cui il risultato finale è che le due linee sono coincidenti
- l'ordine di intersezione è $\mathbf{r}_2, \mathbf{r}_1$: viene verificato il test $d(\mathbf{p}_1, \mathbf{r}_2) < \epsilon_{dis}$ che risulta falso, per cui il risultato finale è che non ci sono intersezioni

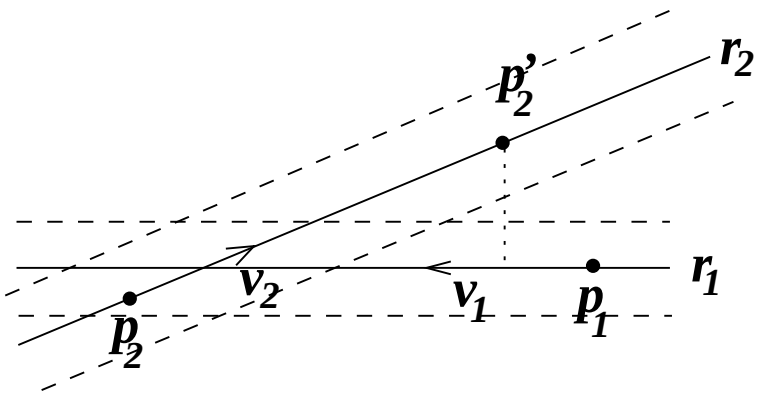


Figura 2.10: Intersezione di rette ϵ

Si ha quindi che invertendo l'ordine di intersezione si ottengono due risultati praticamente opposti, in particolare il secondo è evidentemente incoerente con la condizione reale delle due rette.
Ancora, se come origine della retta $\mathbf{r}_2$ si considera il punto $\mathbf{p}'_2$ al posto di $\mathbf{p}_2$, il problema non cambia, ma il risultato dell'intersezione $\mathbf{r}_1, \mathbf{r}_2$ invece risulta opposto (nessuna intersezione). Il motivo è lo stesso della seconda situazione descritta sopra, il test $d(\mathbf{p}'_2, \mathbf{r}_1) < \epsilon_{dis}$, secondo questa rappresentazione di $\mathbf{r}_2$, risulta non verificato.
La causa di queste inaccuratezze è da ricercarsi nell'uso di test approssimati. In particolare $|\mathbf{v}_1 \cdot \mathbf{v}_2| < \epsilon_{ang}$ verifica solamente una condizione di

quasi-parallelismo, per cui le deduzioni seguenti risultano inaffidabili in quanto basate sul concetto ideale di parallelismo.

Questi esempi mettono in luce la difficoltà di implementare algoritmi geometrici accurati valendosi di un'aritmetica approssimata. Se da un lato l'approccio di usare delle tolleranze per l'indecidibilità dell'uguaglianza li rende meno sensibili agli errori di arrotondamento, dall'altro annulla la validità di molte proprietà geometriche sfruttate naturalmente dai programmatori, le quali possono diventare fonte di errori.

Si propone allora un altro algoritmo il cui codice è riportato alla fine del capitolo. Brevemente, esso consiste dei seguenti passi:

1. vengono calcolate le distanze, senza segno, d_1, d_2, d_3 e d_4 degli estremi dei due segmenti dall'altro segmento e in base a queste si determinano gli estremi che incidono sull'altro segmento

2. se si ha 1 estremo incidente internamente sull'altro segmento, allora si ritorna tale punto come intersezione

3. se si hanno 2 estremi incidenti internamente sull'altro segmento allora si ritorna la relativa zona di sovrapposizione

4. stimato che i due segmenti si intersecano, si cerca il punto d'intersezione.
   Utilizzando quanto visto nel paragrafo 2.4.2, si calcola il parametro t mediante la seguente proporzione:

   $$t = \|\mathbf{p}_1 - \mathbf{q}_1\| d_1/(d_1 + d_2) \tag{2.9}$$

   dove d_1, d_2 sono le distanze senza segno dei due estremi $\mathbf{p}_1$ e $\mathbf{q}_1$ dall'altro segmento (eventualmente esteso) (vedi Figura 2.11 sinistra).

Calcolato allo stesso modo s, si ricavano i relativi punti $\mathbf{r}_1(t)$ ed $\mathbf{r}_2(s)$ sui due segmenti. Il punto restituito è una media dei due punti calcolati. Questo approccio presenta alcuni vantaggi:

- viene evitato l'utilizzo di tolleranze angolari, tipo ϵ_{ang}, per decidere il parallelismo e la condizione di sovrapposizione, che, come si è visto, provocano inaccuratezze.
  Si ha sovrapposizione se e soltanto se esistono due estremi dei segmenti che incidono sull'altro (passo 3).

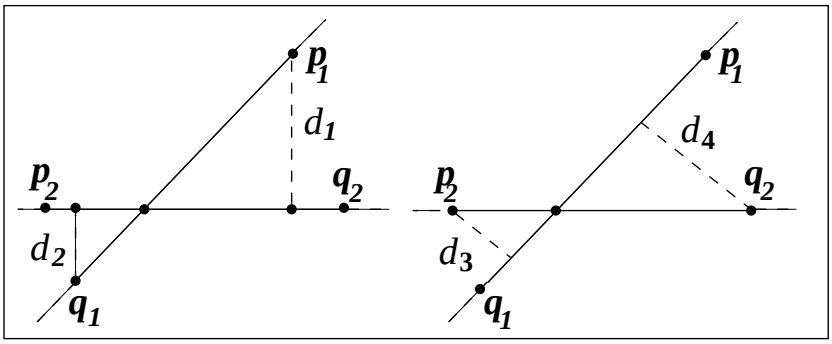


Figura 2.11: Intersezione segmento/segmento

- è *simmetrico*, perché la verifica della sovrapposizione dipende da tutti e quattro gli estremi dei due segmenti (passo 1). Nel precedente approccio ne veniva verificato invece solo uno.
- la proporzione 2.9, usata per il calcolo dei parametri associati ai potenziali punti d'intersezione, è *semplice e stabile*. Si noti che, grazie ai controlli effettuati nei passi 1 - 3, risulta sempre $d_1 > \epsilon_{dis}$ e $d_2 > \epsilon_{dis}$.

2.5 Intersezione di segmenti: test numerici

Impostare dei test significativi è uno dei problemi principali nel processo di correzione degli errori del software; in questo paragrafo si illustrano i risultati dei test sugli algoritmi di intersezione tra segmenti, visti nella sezione precedente.
I test eseguiti procedono come segue: si definisce un segmento di estremi $\mathbf{p}_1$ e $\mathbf{q}_1$. Esso viene ripetutamente intersecato con un insieme di segmenti di estremi $\mathbf{p}_2$ e $\mathbf{q}_2$ dove:

- $\mathbf{p}_1 = \mathbf{p}_2$
- $\mathbf{q}_2$ ha la stessa ascissa di $\mathbf{q}_1$, mentre l'ordinata è la stessa di $\mathbf{q}_1$, ma aumentata di d, dove d varia da un valore iniziale fino a 0, condizione quest'ultima per cui i due segmenti coincidono (vedi figura 2.12)

Per ogni iterazione si osserva il numero k di punti d'intersezione trovati e si calcolano, in funzione di d, gli intervalli in cui tale numero è costante. Il comportamento corretto che ci si attende è che si distinguano due intervalli:

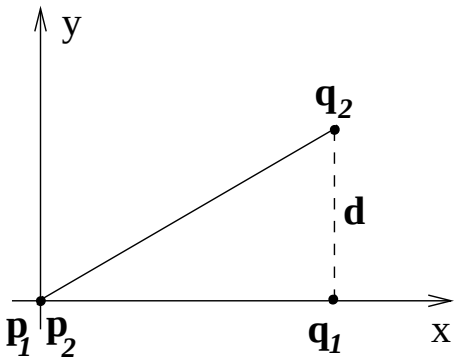


Figura 2.12: Intersezione di segmenti: test

1. per $d > \Delta$, $k = 1$, dove il punto d'intersezione corrisponde all'estremo comune dei due segmenti: $\mathbf{p}_1 = \mathbf{p}_2$

2. per $d \leq \Delta$, $k = 2$, cioè i due segmenti sono sovrapposti

dove Δ è un valore che dipende dalla tolleranza associata ai segmenti (nelle condizioni di test riportate a seguito $tol = 1.0e^{-15}$).
Questo test viene ripetuto in diverse condizioni, in cui *i segmenti intersecati sono sempre gli stessi*, ma varia l'ordine d'intersezione o la loro rappresentazione (ordine degli estremi): Nel caso del primo algoritmo i risultati ottenuti *risultano differenti nelle varie condizioni* come previsto in fase di analisi dell'algoritmo stesso.

Nel caso del secondo algoritmo invece i risultati dei test confermano le previsioni, infatti in tutte le condizioni si ottiene lo stesso risultato:

- per $d > 1.0e^{-15},\ k = 1$
- per $d \leq 1.0e^{-15},\ k = 2$

Elenco delle figure

1.1 Modello poligonale 2
1.2 Rappresentazione binaria dei numeri finiti 4
1.3 Discretizzazione della retta reale 5
1.4 Analisi in avanti dell'errore 12
1.5 Analisi all'indietro dell'errore 13

2.1 Esempio di intersezione 23
2.2 Rappresentazioni rette 24
2.3 Distanza di un punto da una retta 27
2.4 Distanza punto/retta: Test 1 29
2.5 Distanza punto/retta: Test 2 30
2.6 Intersezione mal condizionata 32
2.7 Intersezione ben condizionata 33
2.8 Intersezione rette 36
2.9 Differenti casi di intersezione fra segmenti 37
2.10 Intersezione di rette ϵ 38
2.11 Intersezione segmento/segmento 40
2.12 Intersezione di segmenti: test 41

Bibliografia

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

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

[Ove01] M. L. Overton. Numerical Computing with IEEE floating Point Arithmetic. *Siam*, 2001.

[Mic] D. Michelucci. The Robustness Issue. *Technical report of Ecole de Mines,F-42023 Saint-Étienne 02*. Disponibile presso: `<http://www.emse.fr/~micheluc/robustness.ps.gz>`.