

ARITMETICA DI MACCHINA E ANALISI DEGLI ERRORI

Calcolo Numerico
corso A a.a.2021/22
F. lavernaro



Considerazioni preliminari

Per la rappresentazione dei numeri reali si utilizza, solitamente, una notazione posizionale. Ad esempio:

$$123.45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Più in generale, lavorando con una base $\beta \geq 2$:

$$d_m d_{m-1} \dots d_0 . d_1 d_2 \dots d_m$$

$$= d_m \cdot \beta^m + d_{m-1} \cdot \beta^{m-1} + \dots + d_0 \cdot \beta^0 + d_1 \beta^{-1} + \dots + d_m \beta^{-m}$$

Consideriamo il seguente esempio.

Esempio

Supponiamo che, per la memorizzazione dei numeri reali abbiamo a disposizione un campo di memoria costituito da 5 celle, ciascuna delle quali può ospitare una cifra.

Consideriamo il seguente numero:

$$x = 0.00123456$$

Nel campo di memoria assegnato, il numero x verrà rappresentato dalla sua approssimazione

$$x_1 = 0.0012 \approx x$$

Considerando invece la rappresentazione normalizzata
di x :

$$x = 1.23456 \cdot 10^{-3}$$

Utilizzando questa notazione, potremo dedicare
una cella per la memorizzazione dell'esponente,
ottenendo così l'approssimazione

$$x_2 = 1.234 \cdot 10^{-3}$$

È evidente che x_2 risulta una migliore approssima-
zione di x rispetto a x_1 . Quindi, per

La rappresentazione di numeri reali consente
utilizzando la notazione normalizzata:

$$x = \pm d_0 \cdot d_1 d_2 \dots \beta^P$$

Diagramma delle componenti della rappresentazione normalizzata:

- Segno** (indicato da un segno + o -)
- mantissa** (corrispondente al numero reale compreso tra 1 e β)
- esponente** (individuato da P)
- punto decimale** (separatore fra la mantissa e l'esponente)
- cifre della rappresentazione** (d0, d1, d2, ...)
- base** (individuata da β)

il cui significato è:

$$x = \pm \beta^P \sum_{i=0}^{\infty} d_i \beta^{-i}$$

con $d_0 \neq 0$ (Condizione di normalizzazione)

Teorema (della rappresentazione in base)

Sia $\beta \geq 2$, β intero. Allora, ogni numero reale $x \in \mathbb{R}$, $x \neq 0$ può essere espresso in maniera univoca come:

$$x = \pm \beta^P \sum_{i=0}^{\infty} d_i \beta^{-i}$$

dove:

- β è la base della rappresentazione
- $p \in \mathbb{Z}$ è l'esponente
- \pm è il segno
- $0 \leq d_i \leq \beta - 1$, $i = 0, 1, \dots$
- $d_0 \neq 0$ (condizione di normalizzazione)
- le cifre di m_n sono definitivamente uguali a $\beta - 1$

Osservazioni

- 1) Abbiamo escluso lo zero poiché questo non ammette una notazione normalizzata.
- 2) L'ultima condizione impone che le cifre di m_n siano uguali a $\beta - 1$ da un certo indice in poi. Tale condizione è importante per

avere l'unicità della rappresentazione.

Esempio

In base $\beta=10$, scegliamo il numero

$$x = 0.\overline{9} = 9 \cdot 10^{-1}$$

$$= 10^{-1} \cdot \sum_{i=0}^{\infty} 9 \cdot 10^{-i} = \frac{9}{10} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{10}\right)^i$$

$$= \frac{9}{10} \cdot \frac{1}{1 - \frac{1}{10}} = 1$$

In conclusione $0.\overline{9}$ e 1 rappresentano lo stesso numero reale. Analogamente:

$$1.2\overline{3} = 1.24$$

Gli calcolatori, disponendo di memoria finite, non può rappresentare correttamente gli infiniti numeri reali. Le difficoltà nascono da due impedimenti:

- il numero arbitrariamente elevato delle cifre di (si pensi ad esempio ai numeri irrazionali oppure ai numeri razionali che ammettono una rappresentazione periodica)
- l'esponente p può variare da $-\infty$ a $+\infty$.

Dovremo quindi fissare un numero massimo di cifre di e una tollerabilità finita per l'esponente.

Numeri di meccalma

Sono quei numeri reali che il calcolatore è in grado di rappresentare esattamente.

Definizione

Siamo $\beta \geq 2$, $t \geq 0$, M_1, M_2 interi.

Si definisce insieme dei numeri di meccalma

in base β , con $t+1$ cifre significative e
range per l'esponente $p \in (\mathbb{M}_1, \mathbb{M}_2)$, il
seguente sottoinsieme di \mathbb{R} :

$$F = \left\{ x \in \mathbb{R} \mid x = \pm \beta^p \sum_{i=0}^t d_i \beta^{-i} \right\} \cup \{0\}$$

dove:

- β è la base della rappresentazione
- \pm è il segno
- $0 \leq d_i \leq \beta-1$, $d_0 \neq 0$
- $p \in (\mathbb{M}_1, \mathbb{M}_2)$

Proprietà e osservazioni

1) $\text{card}(F) < +\infty$, F è un sottoinsieme
finito di \mathbb{R} . Si calcoli, per esercizio, $\text{card}(F)$,
numero di elementi di F .

- 2) Al fine di memorizzare numeri piccoli (con esponente negativo) e grandi (con esponente positivo), sceglieremo $M_1 < 0$ e $M_2 > 0$
- 3) Per la base β , i calcolatori utilizzano generalmente il sistema binario ($\beta = 2$). Questo scelto è suggerito dalla natura fisica dei dispositivi elettronici che memorizzano ed elaborano i dati. Essi possono assumere due stati fisici. Ad esempio:
- paraggio o meno di corrente attraverso una resistenza
 - magnetizzazione o meno di un dispositivo magnetico.

4) Un vantaggio dell'uso del sistema binario

è che, ovendo essere $d_0 = 0, 1$,

dalle condizione di normalizzazione:

$$d_0 \neq 0 \implies d_0 = 1$$

quindi è inutile memoizzare la cifra d_0
risparmiando così 1 bit.

- 5) I numeri di macchina sono disposti
simmetricamente rispetto allo zero, tuttavia
la loro distribuzione non è uniforme.

Più in particolare consideriamo un generico
numero di macchina positivo

$$a = \beta^p \sum_{i=0}^t d_i \cdot \bar{\beta}^{-i}$$

Il numero di macchina successivo, lo
denotiamo con b , lo si ottiene incrementando
di una unità l'ultima cifre d_t :

$$b = a + \beta^p \cdot 0.0\ldots 1$$

$$= a + \beta^p \cdot \beta^{-t} = a + \beta^{p-t}$$

da cui, la distanza fra due numeri di macchine consecutive è:

$$b-a = \beta^{p-t}$$

Notiamo che questa distanza dipende dalla grandezza del numero di macchine considerato per la presenza dell'esponente p . Ne consegue che i numeri di macchine sono più densi vicino all'origine (lo zero) e si dilatano man mano che ci allontaniamo dall'origine.

Si vede l'esempio n° 7 della dispensa.

6) Utilizzando la notazione posizionale

Un numero di macchina non nullo

ha l'espressione:

$$x = \pm \beta^p (d_0.d_1 \dots d_t)$$

Determiniamo il $\min \{ \overline{F}_+^* \}$, $\max \{ \overline{F}_+^* \}$

dove \overline{F}_+^* denota l'insieme dei numeri di macchina positivi.

Siamo $w = \min \{ \overline{F}_+^* \}$, $\Sigma = \max \{ \overline{F}_+^* \}$

Per si ottiene minimizzando i valori dell'esponente p e delle mantisse. Pertanto:

$$w = \beta^{M_1+1} \cdot 1.00 \dots 0 = \beta^{M_1+1}$$

Per determinare Σ , premettiamo la seguente proprietà. Sia $z \neq 1$; risulta

$$1+z+\dots+z^t = \frac{(1-z)(1+z+\dots+z^t)}{1-z}$$

$$= \frac{1 + z + z^2 + \dots + z^t - z - z^2 - \dots - z^{t+1}}{1-z}$$

$$= \frac{1 - z^{t+1}}{1 - z}$$

Queste sono le formule della di una progressione geometrica di ragione z . Passiamo al calcolo di Σ :

$$\begin{aligned} \Sigma &= \beta^{N_2-1} \cdot \sum_{i=0}^t (\beta-1) \beta^{-i} = \beta \cdot (\beta-1) \cdot \sum_{i=0}^t (\beta^{-1})^i \\ &= \beta^{N_2-1} (\beta-1) \cdot \frac{1 - \beta^{-t-1}}{1 - \beta^{-1}} \\ &= \beta^{N_2-1} \cdot (\cancel{\beta-1}) \cdot \beta \cdot \frac{1 - \beta^{-t-1}}{\cancel{\beta-1}} \\ &= \nearrow \beta^{N_2-1} \cdot (\beta - \beta^{-t}) \quad \text{sul calcolatore} \\ &\searrow \beta^{N_2} \cdot (1 - \beta^{-t-1}) \quad \text{teorico} \end{aligned}$$

Osservazione: fai in MATLAB w e Σ

sono due variabili predefinite, indicate con
realmin e realmax

STANDARD IEEE - 754

Nel 1985 l'Institute of Electric an
Electronic Engineers formulò uno standard
per l'implementazione dell'aritmetica di macchine
su calcolatori, al fine di rendere compatibili
i codici sulle varie piattaforme. Questo
standard prevede le seguenti due modalità
fondamentali:

- Semplice (o single) precision
Single precision
- Doppia precision
Double precision

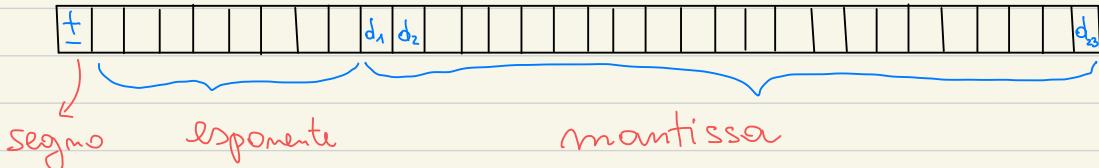
La doppia precisione è la modalità più

frequentemente utilizzate (è il default per MATLAB e Python). In ogni caso la base utilizzata è $B = 2$.

Semplice precisione

In semplice precisione, per la memorizzazione di un numero di macchina, viene predisposto un campo di memoria costituito da 32 bit così suddivisi:

- 1 bit per il segno;
- 8 bit per l'esponente;
- 23 bit per la mantissa.



Segno: 0 per il più e 1 per il meno.

EspONENTE: con 8 bit possiamo realizzare

$2^8 = 256$ configurazioni distinte. Si sceglie:

$$M_1 = -127; \quad M_2 = 128.$$

Oltreché memorizzare l'esponente con il suo segno, si applica una tecnica di traslazione (bias) in base alla quale viene memorizzata

la cosiddette characteristic p^* che è relazionata all'esponente p mediante l'equazione

$$p^* = p - M_1 \iff p = p^* + M_1$$

La seguente tabella chiarisce meglio i termini di questa relazione.

| P^* | P |
|-----------------|------|
| 0 0 0 0 0 0 0 0 | |
| 0 0 0 0 0 0 0 1 | -126 |
| { | { |
| 0 1 1 1 1 1 1 1 | 0 |
| } | { |
| 1 1 1 1 1 1 1 0 | 127 |
| 1 1 1 1 1 1 1 1 | |

La configurazione snata ($p^* = 0$) e quelle piene ($p^* = 255$) sono state lasciate libere. Esse sono utilizzate per altri scopi:

- lo zero si rappresenta con caratteristica nulla e mantissa nulla.
- la caratteristica piena ($p^* = 255$) serve per gestire le cosiddette eccezioni

preiste dello standard IEEE :

- $p^*=255$ e mantissa nulla identificano il simbolo $\pm \infty$
- $p^*=255$ e mantissa non nulla identificano il simbolo NaN

Questi simboli sono l'output di alcune operazioni non consentite in aritmetica reale e il loro scopo è evitare l'arresto dell'esecuzione di un codice in corrispondenza di queste operazioni. Ad esempio:

$$\frac{1}{0} = +\infty \quad \tan \frac{\pi}{2} = +\infty$$

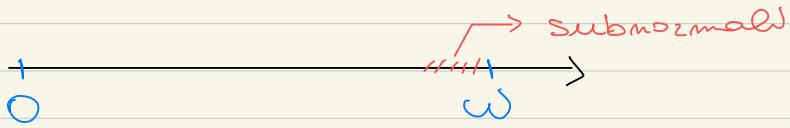
$$\arctan(+\infty) = \frac{\pi}{2}$$

$$\frac{0}{0} = \text{NaN}; \quad \infty - \infty = \text{NaN}$$

$$1^{\text{fInf}} = \text{NaN}; \quad \frac{2^{\text{fInf}}}{2^{\text{fInf}}} = \text{NaN}$$

- Scegliendo caratteristiche nulle ($p^* = 0$) e mantissa non nulla, si identificano i cosiddetti numeri **subnormali** (o non normalizzati o denormali)

I numeri subnormali sono numeri non normalizzati, quindi con $d_0 = 0$, più piccoli ma molto prossimi a w



Sono utilizzati per memorizzare numeri reali più piccoli ma molto prossimi ad w meglio di quanto non facciano lo zero e lo stesso w .

L'espressione generica di un numero

subnormale è:

$$\pm \beta^{M_1+1} \cdot 0.d_1 d_2 \dots d_t$$

Ad esempio, il più grande numero positivo subnormale lo si ottiene scegliendo

$$d_1 = d_2 = \dots = d_t = \beta^{-1}$$

Il più piccolo numero subnormale positivo è: β^{M_1+1-t}

Doppia precisione

A ogni numero di macchine viene riservato un campo costituito da 64 bit così suddivisi:

- 1 bit per il segno;
- 11 bit per l'esponente;
- 52 bit per la mantissa.

Per l'esponente p si sceglie il seguente range:

$$M_1 = -1023; \quad M_2 = 1024$$

Ne consegue che, in doppie precisione:

$$\omega = 2^{-1022} \approx 2.2 \cdot 10^{-308}$$

$$\underline{\omega} = 2^{1024} \left(1 - 2^{-53}\right) \approx 1.7 \cdot 10^{308}$$

In doppie precisione, i numeri subnormali sono (partendo da ω a scendere)

$$\omega = 2^{-1022} \cdot 1.0 \dots \quad 0$$

$$\text{Subnormali } 2^{-1022} \cdot 0.1 \dots \quad 1$$

{

}

$$2^{-1022} \cdot 0.0 \dots \quad .01$$

!!

$$2^{-1074}$$

Osservazione: i numeri subnormali

hanno un numero di cifre significative minore di $M+1$. In effetti le presenti degli zeri all'inizio delle mantisse servono per perfezionare l'esponente permettendogli di assumere valori minori di $M+1$ a discapito di una perdita di accuratezza in termini di cifre significative.

Rappresentazione dei numeri reali mediante i numeri di meccchina

Ogni numero reale servé approssimato sul calcolatore mediante un opportuno numero di meccchina. Occorre definire una funzione

$$f_E : \mathbb{R} \longrightarrow \mathbb{F}$$

Distinguiamo i seguenti casi. Sia $x \in \mathbb{R}$ e, senza perdere la generalità, supponiamo $x > 0$, sfruttando le simmetrie di \mathbb{F} . Possiamo scrivere:

$$x = \beta^p \sum_{i=0}^{\infty} \text{di} \beta^{-i}$$

1) $x \in \mathbb{F}$. In tal caso, ovviamente, poniamo

$$\text{fl}(x) = x$$

In altri termini,

$$\text{fl}_{\mathbb{F}} = \text{identità}$$

2) $p \geq M_2$.



L'esponente di x è più grande del massimo esponente consentito in \mathbb{F} ($M_2 - 1$). Si

genera una condizione di **overflow** e si pone

$$f(x) = +\infty$$

3) $p \leq M_1$.



L'esponente di x è più piccolo del minimo esponente consentito in \mathbb{F} (M_1+1). Si genera una condizione di underflow e si pone

$$f(x) = \begin{cases} \text{subnormale, se } M_1-t+1 \leq p \leq M_1 \\ 0 \quad , \text{ se } p \leq M_1-t \end{cases}$$

Ora analizziamo il caso più frequente.

4) $M_1 < p < M_2$, $x \notin \mathbb{F}$. Ciò significa che

$\exists i > t$ tale che $d_i \neq 0$.

Si possono presentare due sotto casi'

(il secondo è eccezionale)

ω Σ β

- $x \in (\omega, \Sigma)$

- $p = M_{z-1}$, $d_0 = d_1 = \dots = d_t = \beta^{-1}$ e
 $\exists i > t$ t.c. $d_i \neq 0$

Nel secondo sottocaso x è subito a destra di Σ . Nel seguito analizzeremo il primo sottocaso: nel secondo si utilizzerà analogo ragionamento.

Quindi supponiamo $x \in (\omega, \Sigma)$

In tal caso x sarà compreso fra due numeri di macchine consecutivi, che denotiamo con a e b . Evidentemente:



$$a = \beta^p \sum_{i=0}^t d_i \beta^{-i}; \quad b = a + \beta^{p-t}$$

Lo standard IEEE contempla due tecniche.

Tecnica del truncamento

$$f_t(x) = \text{trunc}(x) = a$$

Tecnica dell'arotondamento

$$f_t(x) = \begin{cases} a, & \text{se } x < \frac{a+b}{2} \\ b, & \text{se } x > \frac{a+b}{2} \end{cases}$$

Nel caso in cui $x = \frac{a+b}{2}$

$$f_t(x) = \begin{cases} a, & \text{se } d_t \text{ è pari} \\ b, & \text{se } d_t \text{ è dispari} \end{cases}$$

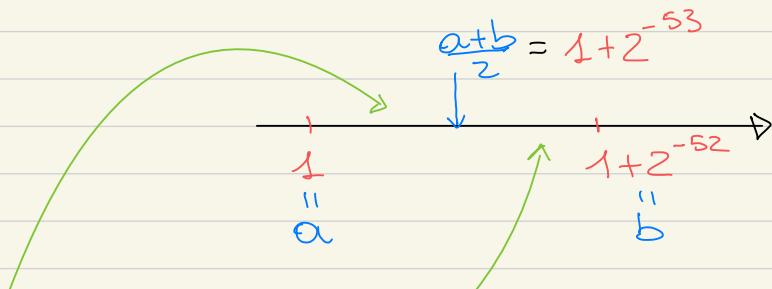
In questo modo si evita di creare una direzione privilegiata (polarizzazione), mantenendo la simmetria della procedura. In particolare, in base $\beta=2$, l'ultima relazione diventa

$$f(x) = \begin{cases} a, & \text{se } d_t = 0 \\ b, & \text{se } d_t = 1 \end{cases}$$

Esempio

In doppia precisione, consideriamo i numeri

reali :



$$f\left(1 + 2^{-54}\right) = 1$$

$$f\left(1 + (2^{-53} + 2^{-54})\right) = 1 + 2^{-52}$$

$$f\left(1 + 2^{-53}\right) = f\left(\frac{a+b}{2}\right) = 1$$

dove abbiamo utilizzato le tecniche dell'

arrotondamento. In particolare, nell'ultimo

caso abbiamo :

$$x = \frac{a+b}{2} = 2^0 \cdot 1.0 \underbrace{0 \dots}_{52} \overset{d_t \quad d_{t+1}}{0} 1$$

Osservazione Nel secondo sottocaso, si

regione in modo analogo scegliendo

$$a = \Sigma; \quad b = a + \beta^{M_2-1-t}$$

Se $x > \frac{a+b}{2}$, si pone $f(x) = +\infty$

Se $x < \frac{a+b}{2}$, si pone $f(x) = \Sigma$

Se $x = \frac{a+b}{2}$ e β è pari: $f(x) = +\infty$

Errori di rappresentazione

Sono gli errori che si generano quando

un numero reale viene approssimato da

un numero di macchina. Anche qui

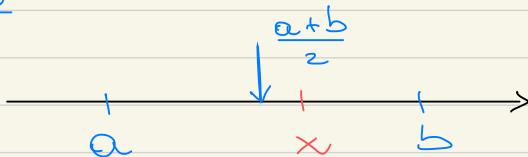
analizzeremo il caso $x > 0$. Supponiamo

che non si incorra in condizioni di

overflow o underflow e in particolare che

$x \in [\omega, \Omega]$. Analizziamo, qui di seguito, gli errori assoluto e relativo in corrispondenza delle due tecniche di approssimazione: troncamento e arrotondamento.

Errore assoluto



Tecnica del troncamento

$$E_a = |f(x) - x| = |a - x| < b - a = \beta^{p-t}$$

Tecnica dell'arrotondamento

$$E_a = |\text{arr}(x) - x| \leq \frac{1}{2} \beta^{p-t}$$

Osservazione

L'errore assoluto dipende dalle grandezze di x (cioè dal suo esponente p).

Errore relativo

Osservazione

$$x = \beta^P \cdot d_0.d_1\dots \geq \beta^P \cdot 1.0\dots 0 = \beta^P$$

Pensando ai reciproci

$$\frac{1}{x} \leq \frac{1}{\beta^P} = \beta^{-P}$$

Tecnica del truncamento

$$E_r = \frac{|f(x) - x|}{|x|} < \beta^{P-t} \cdot \beta^{-P} = \beta^{-t}$$

Tecnica dell'arrotolamento

$$E_r = \frac{|\operatorname{arctan}(x) - x|}{|x|} \leq \frac{1}{2} \beta^{P-t} \cdot \beta^{-P} = \frac{1}{2} \beta^{-t}$$

Osserviamo che l'egualeggiante potrebbe

verificarsi quando, contemporaneamente

$$\left\{ \begin{array}{l} |\operatorname{arr}(x) - x| = \frac{1}{2} \beta^{p-t} \iff x = \frac{\alpha + b}{2} \\ x = \beta^P \end{array} \right.$$

Tuttavia, le due condizioni sono incompatibili perché :

$$\frac{\alpha + b}{2} \notin \mathbb{F}, \quad \beta^P \in \mathbb{F}$$

In conclusione, possiamo scrivere

$$E_r < \frac{1}{2} \beta^{-t} \quad (*)$$

Osservazione

Nel caso dell'errore relativo, la maggiorazione ottenuta è indipendente dalla grandezza del numero x . Ulteriormente, la maggiorazione (*) ci dice che $f(x)$ contiene

almeno t cifre significative corrette.

Definizione

lavorando con l'arotondamento, le quantità

$$u = \frac{1}{2} \bar{p}^t$$

è detta precisione di macchina. È la più piccola maggiorazione sull'errore relativo in fase di rappresentazione dei numeri reali mediante i numeri di macchina.

In doppia precisione:

$$u = \frac{1}{2} \cdot 2^{-52} = 2^{-53} \approx 1.1 \cdot 10^{-16}$$

Ciò significa che, operando in doppia precisione, ad esempio in Python o Matlab, possiamo confidare in 15/16 cifre significative corrette.

Osservazione

Un modo equivalente per esprimere il fatto che l'errore relativo è minore della precisione di macchina è il seguente.

Poniamo

$$\varepsilon = \frac{f(x) - x}{x} \quad (\text{quindi } |\varepsilon| < u)$$

Ricavando $f(x)$:

$$f(x) = x(1 + \varepsilon), \text{ con } |\varepsilon| < u$$

Operazioni di macchina

Consideriamo il seguente esempio

Esempio

Consideriamo un insieme di numeri di macchina in base $\beta=10$ con $t=1$

(due cifre significative). Dati i numeri
di macchina

$$x_1 = 1.2 \cdot 10^0; \quad x_2 = 3.4 \cdot 10^{-1}$$

determinarne la somma

$$\begin{array}{r} x_1 + x_2 : \\ 1.2 \cdot 10^0 + \\ 0.34 \cdot 10^0 \\ \hline 1.54 \cdot 10^0 \end{array}$$

Evidentemente $x_1 + x_2 \notin \mathbb{F}$

Questo esempio mostra che il calcolatore
non può implementare le quattro operazioni
elementari

$$\text{op} : +, -, *, /$$

in modo esatto, ma dovrà approssimarle
mediante altrettante operazioni di macchina

op : \oplus , \ominus , $*$, $/$

$$\text{op} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

$$\text{op} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{F}$$

Nell'esempio precedente, potremmo porre

$$\begin{aligned} x_1 \oplus x_2 &= \text{fl}(x_1 + x_2) = \text{ar}((x_1 + x_2)) \\ &= \text{ar}(1.5h \cdot 10^0) = 1.5 \cdot 10^0 \end{aligned}$$

In fase di ogni singola operazione elementare, si richiederà la medesima accuratezza che si ottiene in fase di rappresentazione:

$$\forall x_1, x_2 \in \mathbb{F} : x_1 \text{op} x_2 = (x_1 \text{op} x_2)(1 + \epsilon)$$

$$\text{con } |\epsilon| < \mu$$

Un'aritmetica che soddisfi queste proprietà
è detta aritmetica floating-point o
in lingua mobile.

Ad esempio, è un'aritmetica floating-point
quella da si ottiene ponendo:

$$x_1 \oplus x_2 = \text{fl}(x_1, op x_2)$$

Proprietà dell'aritmetica di macchina

Le seguenti proprietà dell'aritmetica reale continuano
a valere per l'aritmetica di macchina

Permanenza degli elementi neutri

$$\forall x \in \mathbb{F} :$$

$$x \oplus 0 = x$$

$$x \otimes 1 = x$$

Proprietà commutativa

$\forall x_1, x_2 \in \mathbb{F}$:

$$x_1 \oplus x_2 = x_2 \oplus x_1$$

$$x_1 \otimes x_2 = x_2 \otimes x_1$$

Legge di annullamento del prodotto

$\forall x_1, x_2 \in \mathbb{F}$, se $x_1 \otimes x_2$ non genera underflow oppure overflow :

$$x_1 \otimes x_2 = 0 \implies x_1 = 0 \vee x_2 = 0$$

Le seguenti proprietà non sono più valide
in aritmetica di meccanica

Proprietà associativa

In generale, non è vero che $\forall x_1, x_2, x_3 \in \mathbb{F}$:

$$(x_1 \oplus x_2) \oplus x_3 = x_1 \oplus (x_2 \oplus x_3)$$

$$(x_1 \otimes x_2) \otimes x_3 = x_1 \otimes (x_2 \otimes x_3)$$

Controesempio 1

Consideriamo $\beta = 10$, $t = 1$ (due cifre significative)

con arrotondamento, e i tre numeri di realtà:

$$x_1 = 1.2 \cdot 10^0; \quad x_2 = 1.3 \cdot 10^{-1}; \quad x_3 = 1.4 \cdot 10^{-1}$$

Risulta:

$$x_1 \oplus x_2 :$$

$$\text{fl} \left(\frac{1.2 \cdot 10^0 + 0.13 \cdot 10^0}{1.33 \cdot 10^0} \right) = 1.3 \cdot 10^0$$

$$(x_1 \oplus x_2) \oplus x_3 :$$

$$\text{fl} \left(\frac{1.3 \cdot 10^0 + 0.14 \cdot 10^0}{1.44 \cdot 10^0} \right) = 1.4 \cdot 10^0$$

Ora calcoliamo

$x_2 \oplus x_3 :$

$$\begin{array}{r} 1.3 \cdot 10^{-1} \\ + 1.6 \cdot 10^{-1} \\ \hline fl(2.7 \cdot 10^{-1}) = 2.7 \cdot 10^{-1} \end{array}$$

$x_1 \oplus (x_2 \oplus x_3) :$

$$\begin{array}{r} 1.2 \cdot 10^0 \\ + 0.27 \cdot 10^0 \\ \hline fl(1.47 \cdot 10^0) = 1.5 \cdot 10^0 \end{array}$$

Controesempio 2

Mn doppia precisione (con anotandamento)

Consideriamo i numeri di macchina:

$$x_1 = 1 \cdot 2^0 ; \quad x_2 = 1.0 \dots 0 \cdot 2^{-53} ; \quad x_3 = x_2$$

$$x_1 \oplus x_2 = \text{am} (1 + 2^{-53})$$

= am (1.0 $\underbrace{\dots}_{52}$ 01) = 1 · 2⁰

$$(x_1 \oplus x_2) \oplus x_3 = 1 \cdot 2^0 \oplus 1 \cdot 2^{-53} = 1 \cdot 2^0$$

D'altra parte

$$x_2 \oplus x_3 = 1 \cdot 2^{-53} \oplus 1 \cdot 2^{-53} = 1 \cdot 2^{-52}$$

$$\begin{aligned} x_1 \oplus (x_2 \oplus x_3) &= \text{avr } (\underbrace{1.0 \dots 01}_{52}) \\ &= 1.0 \dots 01 \cdot 2^0 (= 1 + 2^{-52}) \end{aligned}$$

Proprietà distributiva delle moltiplicazione
rispetto all'addizione

In generale, non è vero che, $\forall x_1, x_2, x_3 \in \mathbb{F}$:

$$x_1 \otimes (x_2 \oplus x_3) = (x_1 \otimes x_2) \oplus (x_1 \otimes x_3)$$

Semplificazione

In generale, non è vero che, $\forall x_1, x_2, x_3 \in \mathbb{F}$

con $x_2 \neq 0$:

$$(x_1 \odot x_2) \odot x_2 = x_1$$

$$x_1 \otimes x_2 = x_3 \otimes x_2 \implies x_1 = x_3$$

Commenti

In ogni caso, tutte le proprietà dell'aritmetica reale continuano a valere in aritmetica di macchina a meno di errori dell'ordine delle precisione di macchina.

Ad esempio :

$$(x_1 \oslash x_2) \otimes x_2 = x_1 (1 + \varepsilon)$$

$$\text{con } |\varepsilon| < u$$

Conseguentemente, per verificare se, in aritmetica di macchine, due numeri x e $y \in F$ sono uguali, bisogna evitare di utilizzare l'uguaglianza logica, ma è bene confrontare i due numeri in termini della loro distanza,

ad esempio :

$$|x - y| < tol$$

dove tol è un'accuracy prefissata, ad esempio $tol = 10 \cdot u$

Analisi degli errori

Abbiamo visto che gli errori relativi introdotti in fase di rappresentazione dei numeri reali mediante i numeri di macchina e in fase di elaborazione dei dati mediante le operazioni elementari sono dell'ordine delle precisioni di macchina.

Tuttavia, ciò non garantisce che combinando le due fasi, l'errore nel risultato di un'operazione sia anch'esso dell'ordine

delle precisione di macchine. Il seguente esempio chiarisce la questione.

Esempio

Consideriamo $B = 10$, $t = 4$ (5 cifre significative) e andamento. Risolvere il seguente problema:

determinare, in \mathbb{F} , la differenza dei seguenti due numeri reali:

$$x_1 = 1.23456 \cdot 10^0; \quad x_2 = 1.23454 \cdot 10^0$$

e calcolare l'errore relativo nel risultato.

Osserviamo che $x_1, x_2 \notin \mathbb{F}$. Prima di tutto dovremo rappresentarli in \mathbb{F} :

$$\text{fl}(x_1) = 1.2346 \cdot 10^0;$$

$$\text{fl}(x_2) = 1.2345 \cdot 10^0.$$

Effettuiamo la sottrazione in FF:

$$\begin{aligned} f(x_1) \ominus f(x_2) : & \quad 1.2346 \cdot 10^0 \ominus \\ & \quad \underline{1.2345 \cdot 10^0} \\ & \text{an } (0.0001 \cdot 10^0) \\ = \text{an } (1.0000 \cdot 10^{-4}) \\ = 1.0000 \cdot 10^{-4} \end{aligned}$$

Al fine di valutare l'errore, determiniamo il risultato teorico:

$$x_1 - x_2 = 2 \cdot 10^{-5}$$

Risulta quindi

$$E_n = \frac{|(f(x_1) \ominus f(x_2)) - (x_1 - x_2)|}{x_1 - x_2}$$

$$= \frac{1 \cdot 10^{-4} - 2 \cdot 10^{-5}}{2 \cdot 10^{-5}} = \frac{10 \cdot 10^{-5} - 2 \cdot 10^{-5}}{2 \cdot 10^{-5}}$$

$$= 4 \cdot 10^0$$

Il risultato in \bar{H} contiene zero cifre

significative corrette

Osservazione

In questo esempio, l'unica fonte di errore è quella in fase di rappresentazione: La sottrazione è esatta! La precisione di macchina è $u = \frac{1}{2} \cdot 10^{-4}$ e, in effetti, $fl(x_1) - fl(x_2)$ contengono 4 cifre significative corrette. Questo piccolo errore si amplifica a seguito della sottrazione portando alle perdite di tutte le cifre significative nel risultato.

Preliminari

Prodotto scalare (o interno) tra due vettori

Consideriamo due vettori di \mathbb{R}^m

(insieme dei vettori ad m componenti)

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Per convenzione i vettori saranno matrici a una colonna; per riguardarli come vettori riga se ne considererà il trasposto:

$$\underline{x}^\top = [x_1, x_2, \dots, x_m]$$

Si definisce prodotto scalare tra \underline{x} e \underline{y} il seguente numero reale:

$$\underline{x}^T \cdot \underline{y} = [x_1, x_2, \dots, x_n] \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$1 \times m$

$m \times 1$

$$= x_1 \cdot y_1 + x_2 y_2 + \dots + x_m \cdot y_m$$

$$= \sum_{i=1}^m x_i \cdot y_i$$

In parole: il prodotto scalare è la somma dei prodotti delle componenti omologhe dei due vettori. Geometricamente coincide con:

$$\|\underline{x}\|_2 \cdot \|\underline{y}\|_2 \cdot \cos \alpha$$

dove $\|\underline{x}\|_2$ e $\|\underline{y}\|_2$ (normaz.)

rappresentano le lunghezze euclidi del due vettori, mentre α è l'angolo che essi formano.

Derivate parziali

Sono l'estensione delle derivate classiche
al caso di funzioni a più variabili.

Sia $f: \mathbb{R}^m \rightarrow \mathbb{R}$, funzione scolare
ad m variabili, ad esempio

$$f(x_1, x_2) = x_1 + x_2$$

$$f(x_1, x_2) = x_1 \cdot x_2$$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2^2 - x_2 \cdot \log x_3$$

Si definisce derivate parziale di f rispetto
alla variabile x_i , il seguente limite
se esiste finito:

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_i + h, \dots, x_m) - f(x_1, \dots, x_i, \dots, x_m)}{h}$$

esso è il limite del rapporto incrementale
 definito sulla variabile x_i e lasciando
 fisse tutte le altre variabili.

Operativamente, si effettua la derivata
 rispetto alla variabile x_i , assumendo che
 tutte le altre variabili siano delle costanti.

Esempi

- $f(x_1, x_2) = \frac{x_1}{x_2}$

$$\frac{\partial f}{\partial x_1} = \frac{1}{x_2}; \quad \frac{\partial f}{\partial x_2} = -\frac{x_1}{x_2^2}$$

- $f(x_1, x_2, x_3) = x_1 \cdot x_2^2 - x_2 \cdot \log x_3$

$$\frac{\partial f}{\partial x_1} = x_2^2; \quad \frac{\partial f}{\partial x_2} = 2x_1 x_2 - \log x_3$$

$$\frac{\partial f}{\partial x_3} = -\frac{x_2}{x_3}$$

Il settore contenente le derivate parziali di una funzione si chiama gradiente.
 Esso individua la direzione di massima crescita della funzione. Si denota con il simbolo :

$$\nabla f(\underline{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\underline{x}) \\ \frac{\partial f}{\partial x_2}(\underline{x}) \\ \vdots \\ \frac{\partial f}{\partial x_m}(\underline{x}) \end{bmatrix}$$

Si osservi che, con f , anche

$$\frac{\partial f}{\partial x_i} : \mathbb{R}^m \longrightarrow \mathbb{R}$$

è una funzione scalare.

Sviluppo di Taylor arrestato al primo
ordine per funzioni scalari

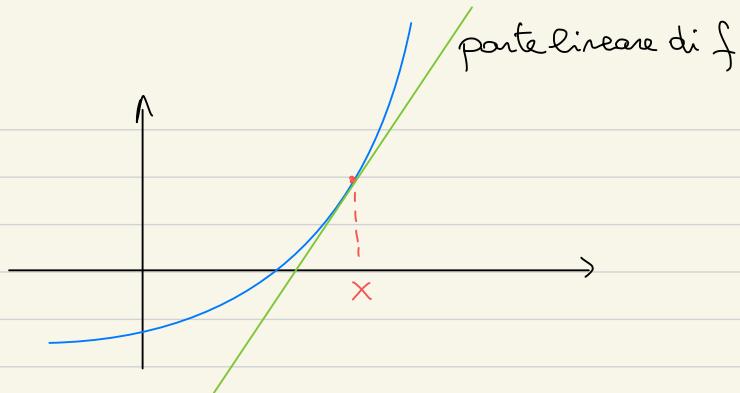
Ricordiamo che se $f: \mathbb{R} \rightarrow \mathbb{R}$

è sufficientemente regolare, possiamo considerarne lo sviluppo di Taylor arrestato al primo ordine in un intorno di x :

$$f(x+h) = \underbrace{f(x) + f'(x) \cdot h}_{\text{parte lineare di } f} + \underbrace{o(h)}_{\substack{\text{infinitesimo} \\ \text{di ordine} \\ \text{superiore} \\ \text{al primo}}}$$

Per h piccolo, possiamo trascurare $o(h)$ e approssimare f mediante la sua parte lineare:

$$f(x+h) \simeq f(x) + f'(x) \cdot h$$



Stiamo approssimando $f(x)$, in un intorno di x , mediante la retta tangente a f nel punto x .

Generalizziamo al caso di funzioni a più variabili:

$$f: \mathbb{R}^n \longrightarrow \mathbb{R}$$

f è una funzione a n variabili x_1, x_2, \dots, x_n

Denoteremo con

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Se f è sufficientemente regolare, possiamo considerarne lo sviluppo di Taylor arrestato al primo ordine in un intorno di \underline{x} :

$$f(\underline{x} + \underline{h}) = f(\underline{x}) + \nabla^T f(\underline{x}) \cdot \underline{h} + o(\underline{h})$$

parte lineare di f

infinitesimo
di ordine
superiore al primo

Per \underline{h} piccolo possiamo trascurare $o(\underline{h})$ e

scrivere:

$$\begin{aligned} f(\underline{x} + \underline{h}) &\simeq f(\underline{x}) + \nabla^T f(\underline{x}) \cdot \underline{h} \\ &= f(\underline{x}) + \sum_{i=1}^m \frac{\partial f}{\partial x_i}(\underline{x}) \cdot h_i \end{aligned}$$

Per $m=2$, stiamo approssimando la superficie
descritta da f mediante il piano tangente
ad f nel punto \underline{x} .

Ci proponiamo di studiare il seguente problema
e le generalizzazioni dell'esempio visto sopra.

Problema

Data una funzione razionale

$$g: \mathbb{R}^m \longrightarrow \mathbb{R}$$

valutare questa funzione in aritmetica di
macchina e stimare l'errore che ne risulta.

Osservazioni

- 1) Nel risolvere questo problema sul calcolatore
distinguiamo due fasi. Il dato di input,
un vettore \mathbf{x} di dati deve essere rappresentato
nella memoria del calcolatore; i dati devono
successivamente essere combinati mediante
le operazioni descritte dall'espressione di g .

Nelle prime fasi si generano gli enori di rappresentazione; nelle seconde fasi si generano gli enori di arrotondamento durante i calcoli.

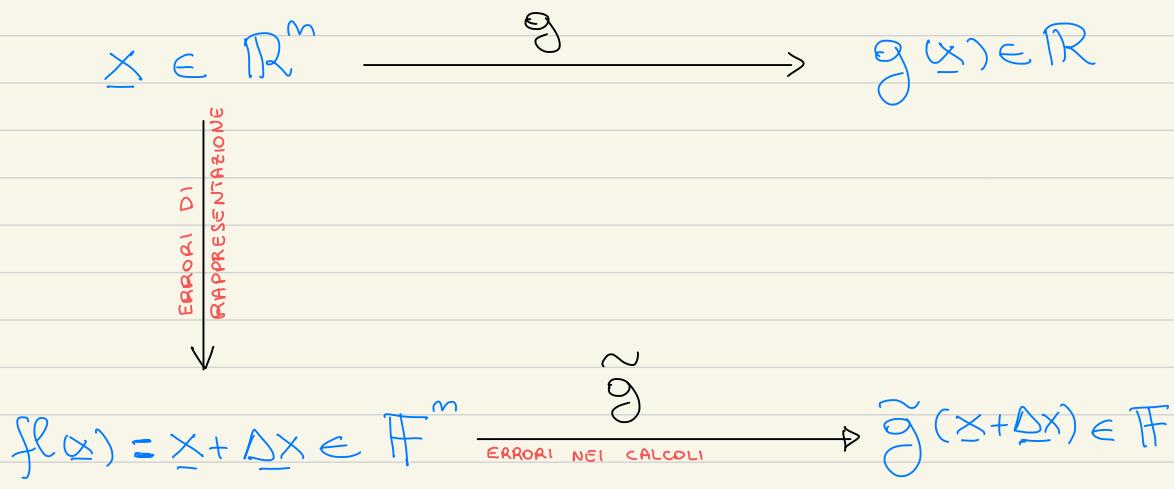
- 2) Abbiamo scelto y quale funzione razionale (cioè rapporto di due polinomi) poiché le funzioni razionali sono tutte e sole le funzioni che possono essere valutate mediante una sequenza finita di operazioni elementari. In effetti tutte le altre funzioni sono approssimate all'interno del calcolatore mediante opportune funzioni razionali.

Un esempio di funzione razionale che a noi interesserà è:

$$f(x_1, x_2) = x_1 \text{ op } x_2$$

con op: +, -, *, /

Il seguente diagramma riassume le due fasi descritte sopra



dove \tilde{g} ha la medesima espressione di g

con l'unica differenza che le operazioni in essa definite sono operazioni di macchine.

Ad esempio:

$$g(x_1, x_2) = \frac{x_1}{x_2} \implies \tilde{g}(x_1, x_2) = x_1 \oslash x_2$$

Osserviamo che

$$f(x) = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_m) \end{bmatrix} = \begin{bmatrix} x_1(1+\varepsilon_1) \\ x_2(1+\varepsilon_2) \\ \vdots \\ x_m(1+\varepsilon_m) \end{bmatrix}$$

con $| \varepsilon_i | < \mu$, $i = 1, \dots, m$

→ errori relativi in input

Liberando dalle parentesi e dividendo in due

settori, si ottiene:

$$f(x) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} x_1 \varepsilon_1 \\ x_2 \varepsilon_2 \\ \vdots \\ x_m \varepsilon_m \end{bmatrix}$$

x Δx

Introduciamo i seguenti errori

ERRORE TOTALE

$$E_{TOT} = \frac{\tilde{g}(\underline{x} + \Delta x) - g(\underline{x})}{g(\underline{x})}$$

nell'ipotesi che $g(\underline{x}) \neq 0$. È l'errore nel risultato che tiene conto di entrambe le fonti di errore prime descritte.

ERRORE INERENTE

$$E_{IN} = \frac{g(\underline{x} + \Delta x) - g(\underline{x})}{g(\underline{x})}$$

È l'errore nel risultato che tiene conto unicamente degli errori di rappresentazione dei dati. In altri termini vengono trascurati gli errori nei calcoli cioè si ritiene che i calcoli vengano eseguiti esattamente.

ERRORE ALGORITMICO

$$E_{ALG} = \frac{\tilde{g}(x + \Delta x) - g(x + \Delta x)}{g(x + \Delta x)}$$

È l'errore nel risultato che tiene conto unicamente degli errori di arrotondamento introdotti dalle operazioni floating-point. Si lavora direttamente su dati che sono numeri di macchine.

Sussiste la seguente relazione :

$$E_{TOT} = E_{IN} + E_{ALG} (1 + E_{IN})$$

La verifica è lasciata per esercizio.

Ne consegue che possiamo studiare indipendentemente i problemi più semplici di stimare

E_{IN} e E_{ALG} per poi combinare i risultati
e ottenere una stima di E_{TOT} . In particolare:

- Se $|E_{IN}| \ll 1$ e $|E_{ALG}| \ll 1$

lo sarà anche l'errore totale. In questo
caso:

$$E_{TOT} = E_{IN} + E_{ALG} + E_{ALG} \cdot E_{IN}$$
$$\simeq E_{IN} + E_{ALG}$$

In questo caso vale il principio di sovrapposizione

- Se $|E_{IN}| \gg 1$ oppure $|E_{ALG}| \gg 1$

potremo concludere che anche $|E_{TOT}| \gg 1$.

- L'errore inerente studia il condizionamento
di un problema.
- L'errore algoritmico studia la stabilità di
un algoritmo

Condizionamento di un problema

Studiare il condizionamento di un problema significa analizzare come minime piccole perturbazioni dei dati di input si riflettono nel risultato (dati di output). In particolare

- un problema si dice ben condizionato se a piccole perturbazioni dei dati di input corrispondono piccole perturbazioni dei dati di output.
- un problema si dice mal condizionato se a piccole perturbazioni dei dati di input possono corrispondere grandi perturbazioni dei dati di output. L'effetto del mal condizionamento si traduce in

una significativa perdita di cifre

Significative corrette

Stabilità di un algoritmo

Per uno stesso problema possono esserci più tecniche risolutive, cioè più algoritmi.

- Un algoritmo si dice stabile se le sequenze di operazioni di meccanica che lo definisce non amplifica notevolmente l'errore nel risultato a causa dell'accumularsi degli errori di arondamenti durante i calcoli
- Viceversa, un algoritmo si dà instabile.

Studiamo ora il condizionamento delle operazioni elementari.

Condizionamento delle operazioni elementari

Addizione

$$g(\underline{x}_1, \underline{x}_2) = \underline{x}_1 + \underline{x}_2 \quad \text{con } \underline{x}_1, \underline{x}_2 > 0$$

Per studiare il condizionamento dovremo valutare l'errore inherente. A tal fine riprendiamo lo sviluppo di Taylor arrestato al primo ordine, trascurando gli infinitesimi di ordine superiore:

$$g(\underline{x} + \Delta \underline{x}) \approx g(\underline{x}) + \nabla^T g(\underline{x}) \cdot \Delta \underline{x}$$

da cui

$$E_{IN} = \frac{g(\underline{x} + \Delta \underline{x}) - g(\underline{x})}{g(\underline{x})} \approx \frac{1}{g(\underline{x})} \cdot \nabla^T g(\underline{x}) \cdot \Delta \underline{x}$$

$$= \frac{1}{g(\underline{x})} \cdot \sum_{i=1}^m \frac{\partial g(\underline{x})}{\partial x_i} \cdot x_i \cdot \varepsilon_i$$

Portando $\frac{1}{g(x)}$ dentro la somma:

$$\underbrace{E_{IN}}_{\text{errore in output}} \simeq \sum_{i=1}^m \frac{x_i}{g(x)} \cdot \underbrace{\frac{\partial g}{\partial x_i}(x)}_{\text{erri im input}} \cdot \underbrace{\varepsilon_i}_{\text{errore}}$$

Poniamo $K(x_i, g) = \frac{x_i}{g(x)} \cdot \frac{\partial g}{\partial x_i}(x)$

Allora: $E_{IN} = \sum_{i=1}^m K(x_i, g) \cdot \varepsilon_i$

I coefficienti $K(x_i, g)$ prendono il nome di "numeri di condizionamento" o "coefficienti di amplificazione". Possiamo concludere che:

- se $|K(x_i, g)|$ non sono eccessivamente grandi, l'errore inerente sarà piccolo e il problema è ben condizionato

- viceversa, se $|K(x_i, g)| \gg 1$ per qualche i , $|E_{in}| \gg 1$ e il problema sarà mal condizionato.
-

Torniamo all'addizione $g(x_1, x_2) = x_1 + x_2$

$$E_{in} = K(x_1, g) \varepsilon_1 + K(x_2, g) \cdot \varepsilon_2$$

$$= \frac{x_1}{g(x_1, x_2)} \cdot \frac{\partial g(x_1, x_2)}{\partial x_1} \varepsilon_1 + \frac{x_2}{g(x_1, x_2)} \cdot \frac{\partial g(x_1, x_2)}{\partial x_2} \varepsilon_2$$

$$= \frac{x_1}{x_1 + x_2} \cdot 1 \cdot \varepsilon_1 + \frac{x_2}{x_1 + x_2} \cdot 1 \varepsilon_2$$

Ponendo ai valori assoluti e sfruttando la diseguagliante triangolare $|a+b| \leq |a| + |b|$:

$$|E_{in}| \leq \frac{x_1}{x_1 + x_2} \mu + \frac{x_2}{x_1 + x_2} \cdot \mu = \mu$$

Conclusione: l'addizione tra due numeri dello stesso segno è sempre ben condizionata.

Infatti l'errore inerente è maggiore della precisione di macchina.

Sottrazione

$$g(x_1, x_2) = x_1 - x_2, \quad x_1, x_2 > 0$$

$$E_{IN} = \frac{x_1}{x_1 - x_2} \cdot 1 \varepsilon_1 + \frac{x_2}{x_1 - x_2} (-1) \varepsilon_2$$

$\underbrace{}_{K(x_1, g)}$ $\underbrace{}_{K(x_2, g)}$

Non possiamo maggiorare $|K(x_i, g)|$ mediante una quantità indipendente da x_1 e x_2 .

In effetti

$$\lim_{x_2 \rightarrow x_1} |K(x_i, g)| = +\infty$$

Se x_1 e x_2 sono vicini, il problema diventa mal condizionato poiché l' $|E_{IN}|$ diventa grande.

Conclusione: la sottrazione tra due numeri dello stesso segno è mal condizionata se i due numeri sono molto vicini tra loro.

In questo caso si ha il cosiddetto fenomeno di "cancellazione numerica" cioè perdita di cifre significative nel risultato.

Moltiplicazione

$$g(x_1, x_2) = x_1 * x_2 \quad x_1, x_2 \neq 0$$

$$\begin{aligned} E_{IN} &= \frac{x_1}{x_1 * x_2} * x_2 \varepsilon_1 + \frac{x_2}{x_1 * x_2} * x_1 \varepsilon_2 \\ &= \varepsilon_1 + \varepsilon_2 \end{aligned}$$

e ponendo ai valori assoluti:

$$|E_{IN}| \leq |\varepsilon_1| + |\varepsilon_2| \leq 2\mu$$

Conclusione: la moltiplicazione è sempre ben condizionata

Divisione

$$g(x_1, x_2) = \frac{x_1}{x_2}, \quad x_1, x_2 \neq 0$$

Svolgere i calcoli per esercizio

Si vede che $|E_{IN}| < 2\mu$

Conclusione: la divisione tra due numeri non nulli è sempre ben condizionata.

Ora vediamo un esempio e lo studia la stabilità di un algoritmo.

Stabilità di un algoritmo che effettua la somma di n numeri di macchina positivi

Problema:

Dati $x_1, x_2, \dots, x_n \in \mathbb{F}$, determinarne la somma in aritmetica di macchina e studiarne l'errore algoritmico. Come algoritmo considereremo:

$$\begin{cases} S_1 = x_1 \\ S_i = S_{i-1} + x_i, \quad i = 2, \dots, n \end{cases}$$

È chiaro che

$$S_i = x_1 + x_2 + \dots + x_i$$

e quindi

$$S_n = \sum_{i=1}^n x_i$$

In aritmetica di macchina avremo:

$$\begin{cases} \tilde{S}_1 = x_1 \\ \tilde{S}_i = \tilde{S}_{i-1} \oplus x_i, i=2, \dots, m \end{cases}$$

Osserviamo:

$$\begin{cases} \tilde{S}_1 = x_1 (1 + \varepsilon_1) \\ \tilde{S}_i = (\tilde{S}_{i-1} + x_i)(1 + \varepsilon_i), i=2, \dots, m \end{cases}$$

con $\varepsilon_1 = 0$ e $|\varepsilon_i| < \mu$, $i=2, \dots, m$

Possiamo scrivere

$$\begin{aligned} \tilde{S}_m &= (\tilde{S}_{m-1} + x_m)(1 + \varepsilon_m) \\ &= \tilde{S}_{m-1} (1 + \varepsilon_m) + x_m (1 + \varepsilon_m) \\ &= (\tilde{S}_{m-2} + x_{m-1})(1 + \varepsilon_{m-1})(1 + \varepsilon_m) + x_m (1 + \varepsilon_m) \\ &= \tilde{S}_{m-2} (1 + \varepsilon_{m-1})(1 + \varepsilon_m) + x_{m-1} (1 + \varepsilon_{m-1})(1 + \varepsilon_m) \\ &\quad + x_m (1 + \varepsilon_m) \end{aligned}$$

..... andando avanti si ottiene

$$\begin{aligned}\tilde{S}_m &= x_1 (1 + \varepsilon_1) (1 + \varepsilon_2) \dots (1 + \varepsilon_m) \\ &\quad + x_2 (1 + \varepsilon_2) \dots (1 + \varepsilon_m) \\ &\quad \vdots \\ &\quad + x_m (1 + \varepsilon_m) \\ &= \sum_{i=1}^m \left(x_i \cdot \prod_{k=i}^m (1 + \varepsilon_k) \right)\end{aligned}$$

Studiamo il generico prodotto :

$$\prod_{k=i}^m (1 + \varepsilon_k) = (1 + \varepsilon_i) \cdot (1 + \varepsilon_{i+1}) \cdot \dots \cdot (1 + \varepsilon_m)$$

$$= 1 + \sum_{k=i}^m \varepsilon_k + o(u) \simeq 1 + \sum_{k=i}^m \varepsilon_k$$

$$= 1 + S_i, \text{ avendo posto}$$

$$S_i = \sum_{k=i}^m \varepsilon_k$$

Possiamo osservare che

$$|S_i| \leq \sum_{k=i}^m |\varepsilon_k| \leq n \cdot \sum_{k=i}^m 1$$

$$= (m-i+1) \cdot n \leq mn$$

Introducendo l'approssimazione

δ_i nell'espressione di \tilde{S}_m :

$$\tilde{S}_m = \sum_{i=1}^m x_i (1 + \delta_i) = \sum_{i=1}^m x_i + \underbrace{\sum_{i=1}^m x_i \delta_i}_{S_m}$$

da cui:

$$E_{\text{Alc}} = \frac{\tilde{S}_m - S_m}{S_m} = \frac{1}{S_m} \cdot \sum_{i=1}^m x_i \delta_i$$

Ponendo ai valori assoluti

$$|E_{\text{Alc}}| \leq \frac{1}{S_m} \sum_{i=1}^m x_i |S_i| \stackrel{\leq mn}{\leq} mn$$

Conclusione: l'errore eredita al più
linearmente rispetto ad n e quindi l'
algoritmo è stabile.

Commento.

Per un algoritmo generico, se l'errore
algoritmico è maggiorato da un'espressione
della forma

$$|E_{Alg}| \leq p(m) \cdot \epsilon$$

dove m indica la dimensione del problema
da risolvere e $p(m)$ denota un polinomio
in m , l'algoritmo può ritenersi stabile se
il grado di $p(m)$ è basso (ad esempio
 $1, 2, 3$).