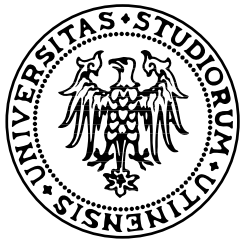
Analisi degli errori



Rossana Vermiglio

CD Lab CDLab
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine

Gruppo UMI *Modellistica Socio-Epidemiologica*

Outline

- Il calcolatore
  - Numeri di macchina
  - Precisione di macchina
  - Aritmetica di macchina
- Analisi dell'errore
  - Errore inerente
  - Errore analitico
  - Errore algoritmico
- Conclusioni

Numeri reali

Le proprietà dell'insieme dei numeri reali $\mathbb{R}$ sono note. In particolare sappiamo sono **infiniti** e che, in generale, sono necessarie **infinite** cifre d_i per la loro rappresentazione in base $B \geqslant 2, B \in \mathbb{N}$. Vale

$$\mathbb{R} = \{0\} \cup \left\{ \pm(d_1 B^{-1} + d_2 B^{-2} + \ldots) \times B^p \,\middle|\, \begin{array}{l} d_i \in \mathbb{N}, 0 \leqslant d_i \leqslant B-1, d_1 \neq 0 \\ d_i \text{ non definitivamente} = B-1 \\ p \in \mathbb{Z} \end{array} \right\}$$

Example

$x = 1/4 = 0.25\ 10^0$ $\qquad$ $x = 1/30 = 0.3333....\ 10^{-1}$

$-\sqrt{5} = -0.22360679774.....\ 10^1$ $\qquad$ $\pi = 0.314159265358\ldots\ 10^1$

$x = 0.1\ 10^0 = 0.\overline{1100}\ 2^{-3}$

Numeri di macchina o floating-point

Il sistema di **numeri di macchina** o **floating-point** (virgola mobile) è il sottoinsieme $\mathbb{F} := \mathbb{F}(B, t, p_{min}, p_{max})$ di $\mathbb{R}$ definito da

$$\mathbb{F} = \{0\} \cup \left\{ \pm(d_1 B^{-1} + d_2 B^{-2} + \ldots + d_t B^{-t}) \times B^p \,\middle|\, \begin{array}{l} 0 \leqslant d_i \leqslant B-1, \\ i = 1, .., t, \, d_1 \neq 0 \\ -p_{min} \leqslant p \leqslant p_{max} \} \end{array} \right\}$$

ed è caratterizzato dai quattro interi

- B: base di rappresentazione
- t: numero cifre della mantissa $\sum_{i=1}^{t} d_i B^{-i}$
- $p_{min}, p_{max} > 0$: forniscono limitazioni inferiore e superiore dell'esponente p, i.e. $-p_{min} \leqslant p \leqslant p_{max}$

Infine si definisce $c = p + p_{min}$ la caratteristica del numero, mentre $d_2 \ldots d_t$ è chiamata **frazione**.

Tale rappresentazione si dice **normalizzata**, perchè la cifra più significativa $d_1 \neq 0$, per $x \neq 0$. Nei sistemi binari, i.e. $B = 2$, risulta $d_1 = 1$.

Le proprietà dell'insieme $\mathbb{F}$

- l'insieme $\mathbb{F}$ è **finito** e la sua cardinalità è data da

$$1 + 2(B - 1)B^{t-1}(p_{max} + p_{min} + 1)$$

- il **più piccolo numero positivo** normalizzato di $\mathbb{F}$ è

$$realmin = B^{-p_{min}-1}$$

- il **più grande numero positivo** normalizzato di $\mathbb{F}$ è

$$realmax = B^{p_{max}}(1 - B^{-t})$$

- i numeri di $\mathbb{F}$ **non** sono uniformemente distribuiti
- sono **equispaziati** solo tra due potenze successive di B

I numeri denormalizzati

L'insieme dei numeri di macchina $\mathbb{F}$ può essere esteso includendo anche i **numeri denormalizzati**, che sono i numeri reali con cifra significativa principale $d_1 = 0$ e con esponente $p = -p_{min}$.

Tali numeri sono distribuiti tra $\pm realmin$ e lo zero e sono equispaziati con spaziatura pari a $B^{-p_{min}-t}$.

Exercise

Quanti sono i numeri denormalizzati? Che valore ha il più piccolo numero denormalizzato positivo?

Exercise

Descrivi l'insieme dei numeri di macchina $\mathbb{F}$ caratterizati da $B = 2$, $t = 3$, $p_{min} = 2$, $p_{max} = 1$, includendo anche i numeri denormalizzati.

Approssimazione dei numeri reali

Solitamente i calcolatori lavorano in una base $B \neq 10$, pertanto anche un numero che è rappresentabile con un numero finito di cifre decimali, può richiedere per la sua rappresentazione in una base diversa un numero infinito di cifre.

Example

$x = (0.3)_{10} = (0.0\overline{1001})_2 = (0.\overline{1001})_2 2^{-1}$.

Dato $x \in \mathbb{R}$ si cercherà un numero floating point $\mathrm{fl}(x) \in \mathbb{F}$ che lo approssimi, definendo così un'applicazione

$$\mathrm{fl} : \mathbb{R} \to \mathbb{F}$$

tale che

- fl segnala errore di **overflow** quando $|x| > realmax$.

Approssimazione dei numeri reali

Solitamente i calcolatori lavorano in una base $B \neq 10$, pertanto anche un numero che è rappresentabile con un numero finito di cifre decimali, può richiedere per la sua rappresentazione in una base diversa un numero infinito di cifre.

Example

$x = (0.3)_{10} = (0.0\overline{1001})_2 = (0.\overline{1001})_2 2^{-1}$.

Dato $x \in \mathbb{R}$ si cercherà un numero floating point $fl(x) \in \mathbb{F}$ che lo approssimi, definendo così un'applicazione

$$fl : \mathbb{R} \to \mathbb{F}$$

tale che

- fl segnala errore di **overflow** quando $|x| > realmax$.
- fl segnala errore di **underflow** quando $|x| < \min\{|y|, y \in \mathbb{F}, y \neq 0\}$. In alcuni sistemi viene posto $fl(x) = 0$.

Approssimazione dei numeri reali

Solitamente i calcolatori lavorano in una base $B \neq 10$, pertanto anche un numero che è rappresentabile con un numero finito di cifre decimali, può richiedere per la sua rappresentazione in una base diversa un numero infinito di cifre.

Example

$x = (0.3)_{10} = (0.0\overline{1001})_2 = (0.\overline{1001})_2 2^{-1}$.

Dato $x \in \mathbb{R}$ si cercherà un numero floating point $fl(x) \in \mathbb{F}$ che lo approssimi, definendo così un'applicazione

$$fl : \mathbb{R} \to \mathbb{F}$$

tale che

- fl segnala errore di **overflow** quando $|x| > realmax$.
- fl segnala errore di **underflow** quando $|x| < \min\{|y|, y \in \mathbb{F}, y \neq 0\}$. In alcuni sistemi viene posto $fl(x) = 0$.
- $fl(x) = x$ quando $x \in \mathbb{F}$

Quando $x \notin \mathbb{F}$, ma

$$\min\{|y|, y \in \mathbb{F}, y \neq 0\} < |x| < realmax$$

allora $fl(x) \in \mathbb{F}$ viene scelto secondo una delle seguenti strategie:

- **Troncamento**: la mantissa di x viene troncata alla t-esima cifra.

  Esempio: $B = 10$, $t = 3$:

  $fl(0.1234567) = 0.123$; $fl(0.9876543) = 0.987$, $fl(0.1235) = 0.123$.

- **Arrotondamento ("rounding")**: il numero $fl(x)$ è il numero di macchina più vicino a x.

  Esempio: $B = 10$, $t = 3$:

  $fl(0.1234567) = 0.123$ e $fl(0.9876543) = 0.988$.

  Quando x è equidistante da due numeri di macchina, $fl(x)$ può essere scelto in modi diversi. In particolare ricordiamo

  - **"round away from zero"** dove $fl(x)$ è il numero più grande;
  - **"round to even"** dove $fl(x)$ è il numero di macchina con la cifra d_t pari. **Esempio**: $B = 10$, $t = 3$: $fl(0.1245) = 0.125$ (away from zero); $fl(0.1245) = 0.124$ (to even).

Precisione di macchina

L'accuratezza di un sistema di numeri di macchina è caratterizzata da una quantità definita **precisione di macchina** u

Ogni numero reale x nel range di $\mathbb{F}$ sarà approssimato da $fl(x)$ con un errore relativo (**errore di rappresentazione di** x) maggiorato dalla precisione di macchina u

$$\epsilon_x = \frac{|fl(x) - x|}{|x|} \leqslant u.$$

Vale:

$u := B^{1-t}$ nel caso di troncamento

$u := \frac{B^{1-t}}{2}$ nel caso dell'arrotondamento

La precisione di macchina u non va confusa con $realmin$! In tutti i sistemi vale $0 < realmin < u < realmax$

Sistemi di numeri di macchina $\mathbb{F}$

Modello	B	t	$-p_{min}$	p_{max}	u
IEEE - SP	2	24	-125	128	$6\ 10^{-8}$
IEEE - DP	**2**	**53**	**-1021**	**1024**	$\mathbf{1\ 10^{-16}}$
IEEE - EX	2	64	-16381	16384	$5\ 10^{-20}$
Cray- SP	2	48	-8192	8191	$4\ 10^{-15}$
Cray - DP	2	96	-8192	8191	$1\ 10^{-29}$
HP 28 and 48G calculator	10	12	-499	499	$5\ 10^{-12}$
IBM3090 - SP	16	6	-64	63	$5\ 10^{-7}$
IBM3090 - SP	16	14	-64	63	$1\ 10^{-16}$
IBM3090 - SP	16	28	-64	63	$2\ 10^{-23}$
DEC VAX G - SP	2	53	-1023	1023	$1\ 10^{-16}$
DEC VAX G - DP	2	56	-127	127	$1\ 10^{-17}$

- $realmin = 2^{-1022} \approx 2.2251e-308$
- $realmax = 2^{1024}(1 - 2^{-53}) \approx 1.7977e+308$
- $u = 2^{-53} \approx 1.11\ 10^{-16}$ (arrotondamento alla cifra pari)

L'esponente può assumere anche i valori $p = -p_{min} - 1 = -1022$ e $p = p_{max} + 1 = 1025$ per usi speciali:

- $p = -1022$, frazione $d_2....d_{53} = 0 \rightsquigarrow$ rappresentazione dello zero ± 0
- $p = -1022$, frazione $d_2....d_{53} \neq 0 \rightsquigarrow$ numeri denormalizzati
- $p = +1025$, frazione $d_2....d_{53} = 0 \rightsquigarrow$ Infinity (Inf) rappresenta una situazione di overflow, divisione per zero
- $p = +1025$, frazione $d_2....d_{53} \neq 0 \rightsquigarrow$Not a Number (NaN) rappresenta operazioni indefinite o indeterminate $\frac{0}{0}, 0 * Inf, \frac{Inf}{Inf}$.

La caratteristica c soddisfa $0 \leqslant c \leqslant 2047 = 2^{11} - 1$. Tenendo conto del bit necessario per il segno e delle 52 cifre della frazione risulta che per memorizzare i numeri di macchina (0, normalizzati e denormalizzati) ho bisogno di 64 cifre binarie.

MATLAB

Il MATLAB esegue tutti i suoi calcoli nello standard IEEE-DP. La funzione logica 'isieee' ritorna il valore 1 'vero' se sta usando l'aritmetica IEEE e 0 'falso' in caso contrario

```
>> isieee
ans = 1
```

La funzione 'eps' ci fornisce la distanza tra $x = 1.0$ e il successivo numero floating point ed è uguale al doppio della precisione di macchina

```
>>eps
eps = 2.2204e-16
```

Le funzioni MATLAB 'realmax' e 'realmin' forniscono rispettivamente i numeri (normalizzati) realmax e realmin:

```
>>realmax
ans =1.7977e+308
>>realmin
ans=2.2251e-308
```

MATLAB continua

Se il risultato di una computazione è più grande di realmax, si ha overflow segnalato dall' infinito

```
>>realmax*2
ans = Inf
```

Se il risultato di una computazione è più piccolo di realmin si ottiene in risposta un numero denormalizzato o il valore zero se è più piccolo di $eps * realmin$ (i.e. il più piccolo numero denormalizzato).

```
>>realmin*eps
ans =4.9407e-324
>>realmin*eps/2
ans = 0
```

MATLAB continua

I risultati di operazioni matematiche non valide producono un NaN :

```
>>0/0
Warning: Divide by zero ans = NaN
>>Inf/Inf
ans = NaN
>>Inf-Inf
ans = NaN
>>0*NaN
ans = NaN
```

Con il comando 'format hex ', i numeri binari floating point sono rappresentati nel formato esadecimale:

```
>>format hex
>>eps
eps = 3cb0000000000000
>>realmax
ans =7fefffffffffffff
>>realmin
ans = 0010000000000000
>>2*realmax
```

Le operazioni elementari (op $= \pm, /, \times$) che operano su numeri di macchina non è detto diano come risultato esatto un numero di macchina. Di qui la necessità di definire le **operazioni di macchina** fl(op).

Addizione o sottrazione Lo *shift* della mantissa necessario per rendere uguali gli esponenti dei numeri, può causare la perdita di cifre (anche tutte) nel numero più piccolo nella fase di arrotondamento.

Example

Sia B $= 10$,t $= 3$. Dati $x_1 = 0.123 \times 10^1$,$x_2 = 0.123 \times 10^{-1} = 0.00123 \times 10^1$,vale

$$x_1 + x_2 = 0.12423\ 10^1 \notin \mathbb{F} \qquad \text{(aritmetica esatta)}$$

Il risultato deve essere approssimato: rimane così definita un'operazione approssimata fl(+), tale che

$$x_1 \text{fl}(+)\ x_2 = 0.124\ 10^1 \in \mathbb{F} \qquad \text{(aritmetica di macchina)}$$

Osserva che le ultime due cifre di x_2 non hanno effetto sul risultato.

Moltiplicazione Il prodotto di due mantisse di t cifre contiene fino a $2t$ cifre e quindi il risultato può non essere rappresentabile. Inotre si può verificare overflow.

Example

Sia $B = 10$,$t = 3$ e $p_{max} = 2$. Dati $x_1 = 0.123\ 10^1$, $x_2 = 0.123\ 10^2$, in aritmetica esatta si ottiene

$$x_1 \times x_2 = 0.015129\ 10^3 = 0.15129\ 10^2 \notin \mathbb{F}.$$

Il risultato deve essere pertanto approssimato: rimane così definita un'operazione approssimata, $fl(\times)$, tale che

$$x_1 fl(\times) x_2 = 0.151\ 10^2 \in \mathbb{F}.$$

Dati $x_1 = 0.123\ 10^2$, $x_2 = 0.124\ 10^2$, in aritmetica esatta si ottiene

$$x_1 \times x_2 = 0.015252\ 10^4 = 0.15252\ 10^3 \notin \mathbb{F}.$$

In questo caso si verifica overflow.

Divisione Il quoziente di due mantisse di t cifre può contenere piú di t cifre e quindi non essere rappresentabile.

Example

Sia $B = 10$,$t = 3$ e $p = 2$. Dati $x_1 = 0.123\ 10^1$, $x_2 = 0.456\ 10^1$, in aritmetica esatta si ottiene

$$x_1/x_2 = 0.269736842105263 2...10^0 \notin \mathbb{F}.$$

Il risultato deve essere pertanto approssimato: rimane così definita un'operazione approssimata, $\mathrm{fl}(/)$, tale che

$$x_1 \mathrm{fl}(/)\ x_2 = 0.270\ 10^0.$$

Modello di calcolo

Per analizzare gli errori di arrotondamento in un algoritmo, dobbiamo fare delle assunzioni sull'accuratezza nelle operazioni di base. Prendiamo come modello di **aritmetica di macchina** il seguente:

tutte le operazioni aritmetiche elementari ($\mathrm{op} = \pm, /, \times$) sono calcolate in modo da soddisfare per ogni $x_1, x_2 \in \mathbb{F}$

$$\mathrm{err}_{\mathrm{op}} = \frac{|(x_1 \,\mathrm{fl}(\mathrm{op}) x_2) - (x_1 \,\mathrm{op}\, x_2)|}{|x_1 \,\mathrm{op}\, x_2|} \leqslant u, \quad \mathrm{op} = \pm, /, \times.$$

Si considera tale maggiorazione valida anche per $\mathrm{op} = \sqrt{\cdot}$.

Il modello asserisce che il valore calcolato è "buono" quanto il valore ottenuto dall'arrotondamento del valore esatto, cioè idealmente si assume

$$x_1 \,\mathrm{fl}(\mathrm{op})\, x_2 = \mathrm{fl}(x_1 \,\mathrm{op}\, x_2).$$

Per lo standard IEEE tale modello è valido.

Limiti dell'aritmetica di macchina

L'aritmetica di macchina **NON** soddisfa tutte le proprietà dell'aritmetica esatta.

Si **conservano** le proprietà commutativa della somma e del prodotto.

$$x_1 \mathrm{fl}(+) x_2 = x_2 \mathrm{fl}(+) x_1, \quad x_1 \mathrm{fl}(\times)\, x_2 = x_1 \mathrm{fl}(\times) x_2$$

mentre **NON** sono valide, per esempio, la proprietà associativa della somma e del prodotto

$$\begin{array}{l} (x_1 \mathrm{fl}(+) x_2) \mathrm{fl}(+) x_3 \neq x_1 \mathrm{fl}(+) (x_2) \mathrm{fl}(+) x_3) \\ (x_1 \mathrm{fl}(\times) x_2) \mathrm{fl}(\times) x_3 \neq x_1 \mathrm{fl}(\times) (x_2) \mathrm{fl}(\times) x_3) \end{array} ,$$

la proprietà distributiva della somma rispetto al prodotto

$$(x_1 \mathrm{fl}(+) x_2) \mathrm{fl}(\times) x_3 \neq x_1 \mathrm{fl}(\times)\, x_3 \mathrm{fl}(+) x_2 \mathrm{fl}(\times) x_3 \ ,$$

e la legge di cancellazione

$$\begin{array}{l} (x_1 \mathrm{fl}(\times) x_2) \mathrm{fl}(/) x_2 \neq x_1 \\ (x_1 \mathrm{fl}(/) x_2) \mathrm{fl}(\times) x_2 \neq x_1 \end{array} .$$

Example

Sia $B = 10, t = 3$. Presi
$x_1 = 0.123\ 10^2, x_2 = 0.123\ 10^{-1} = 0.000123\ 10^2$, vale
$x_1 + x_2 = 0.123123 \times 10^2$, ma $x_1 fl(+)\ x_2 = 0.123\ 10^2$. Osserva che $x_2 > 0$ non ha effetto sul risultato in aritmetica approssimata.

Example

Sia $B = 10, t = 3$. Presi $x_1 = 0.559, x_2 = 0.555, x_3 = 0.4 \times 10^{-2}$, si ottiene
$(x_1 fl(+) x_2) fl(+) x_3 = 0.111 \times 10^1 fl(+)\ 0.0004 \times 10^1 = 0.111 \times 10^1$,
mentre $x_1 fl(+)(x_2 fl(+) x_3) = 0.559 fl(+) 0.559 = 0.112 \times 10^1$ che è il risultato esatto arrotondato.

Limiti dell'aritmetica di macchina: continua

Example

Vale $(1 + x) + x = 1$ ma $1 + (x + x) > 1$, dove x è un numero di macchina positivo leggermente più piccolo della precisione u.

L'uso dell'aritmetica di macchina comporta che **algoritmi matematicamente equivalenti non lo siano numericamente**!

Errore nel calcolo di una funzione $f : \mathbb{R} \to \mathbb{R}$

Le funzioni effettivamente calcolabili su un calcolatore sono le **funzioni razionali**, il cui valore è ottenuto con un numero finito di operazioni elementari. Le funzioni non razionali, come per esempio le funzioni trigonometriche o la funzione esponenziale, sono approssimate con opportune funzioni razionali.

Prendiamo in esame il problema del calcolo del valore

$$y = f(x)$$

con $x \in \mathbb{R}$ (dato di input) e $y \in \mathbb{R}$(dato di output).

Sia g la funzione razionale scelta per approssimare la funzione f. Tale funzione rappresenta l'algoritmo (**modello numerico**) in aritmetica esatta. Sia infine $\tilde{g}$ la funzione effettivamente calcolata in aritmetica di macchina.

A causa degli errori di rappresentazione del numero reale x sul calcolatore, sarà usato in input un valore approssimato $\tilde{x}$.

$$\begin{array}{ccc} x & \longrightarrow & y = f(x) \\ \updownarrow & & \updownarrow \quad \textit{errore totale} \\ \tilde{x} & \longrightarrow & \tilde{g}(\tilde{x}) \end{array}$$

Errore inerente, analitico e algoritmico

In prima approssimazione risulta

$$err_{totale} = \frac{f(x) - \tilde{g}(\tilde{x})}{f(x)} = err_{inerente} + err_{analitico} + err_{algoritmico}$$

dove

$$err_{inerente} = \frac{f(x) - f(\tilde{x})}{f(x)}$$

è l'errore generato dall'errore nel dato,

$$err_{analitico} = \frac{f(\tilde{x}) - g(\tilde{x})}{f(\tilde{x})}$$

è l'errore generato dalla approssimazione di una funzione non razionale con una funzione razionale,

$$err_{algoritmico} = \frac{g(\tilde{x}) - \tilde{g}(\tilde{x})}{g(\tilde{x})}$$

è l'errore generato dal fatto che le operazioni sono effettuate in aritmetica di macchina.

L'accuratezza del risultato finale dipende da tutte le componenti d'errore

Errore inerente e condizionamento

Esistono problemi tali che, qualunque algoritmo venga utilizzato per risolverli, l'errore generato nel risultato risulta elevato e talvolta tale da rendere privo di significato il risultato stesso. Questo fenomeno è una particolarità intrinseca del problema e non dipende dagli algoritmi utilizzati.

L'errore inerente è legato al problema assegnato e studia l'effetto degli errori nei dati di input sui dati di output. La scelta dell'algoritmo non ha effetto su tale errore.

Il problema si dice **ben condizionato** se non amplifica gli errori nei dati, ovvero se un errore relativo nei dati di input comporta un errore relativo dello stesso ordine di grandezza nella risposta.

Il problema si dice **mal condizionato** se un errore anche piccolo nei dati darà origine ad un grande errore nella risposta.

Come "misurare"il condizionamento nel calcolo della funzione f?

$$\begin{aligned} \mathrm{cond}_f(x) &:= \frac{|\mathrm{errore}_{\mathrm{risposta}}|}{|\mathrm{errore}_{\mathrm{dato}}|} \\ &= \frac{|f(x)-f(\tilde{x})||x|}{|f(x)||x-\tilde{x}|} \end{aligned}$$

Il problema è **mal condizionato** quando $\mathrm{cond}_f(x) \gg 1$.

Se la funzione f è differenziabile due volte in un intorno di x, otteniamo, in prima approssimazione,

$$\mathrm{cond}_f(x) = \frac{|x||f'(x)|}{|f(x)|}.$$

Osserviamo che il condizionamento dipende non solo da f ma anche dal dato x.

Example

Consideriamo il problema del calcolo della funzione $f(x) = \cos x$. Il condizionamento è misurato da

$$\text{cond}_f(x) = |x||\tan(x)|.$$

Il problema è dunque mal condizionato per $x \approx \frac{\pi}{2}$. Infatti

$$\cos(1.57079) = 6.32679489\ 10^{-6},$$
$$\cos(1.57078) = 1.632679489\ 10^{-5}$$

$$\epsilon_x = 6.36 \times 10^{-6} \rightarrow \epsilon_{f(x)} = 1.58$$

Errore inerente, continua

Vogliamo studiare il condizionamento del problema della valutazione di una funzione $F : \mathbb{R}^n \to \mathbb{R}$.

Usando anche in questo caso la formula di Taylor si trova, in prima approssimazione,

$$\frac{F(\tilde{x}) - F(x)}{F(x)} = \sum_{i=1}^{n} \left(\frac{x_i}{F(x)} \frac{\partial F(x)}{\partial x_i} \right) \frac{(\tilde{x}_i - x_i)}{x_i},$$

dove $x = (x_1, \ldots, x_n), \tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \in \mathbb{R}^n$.

$$|err_{inerente}| \leqslant \sum_{i=1}^{n} |c_i(x)| \epsilon_{x_i},$$

dove ϵ_{x_i} è l'errore relativo al dato i-esimo e $c_i(x) := \frac{\partial F(x)}{\partial x_i} \frac{x_i}{F(x)}$ è il **coefficiente di amplificazione** ad esso relativo. I coefficienti di amplificazione forniscono una misura del condizionamento del problema.

Esempio: errore inerente della somma di n numeri

$$F(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$$

I coefficienti di amplificazione sono

$$c_i(x_1, \ldots, x_n) = \frac{x_i}{x_1 + \ldots + x_n}, \quad i = 1, 2, \ldots, n,$$

e

$$|err_{inerente}| \leqslant \max_{i=1,\ldots,n} (\epsilon_{x_i}) \frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|}.$$

Il problema è fortemente mal condizionato se

$$\sum_{i=1}^{n} |x_i| >> |\sum_{i=1}^{n} x_i| \quad \textbf{(cancellazione)}$$

Nel caso di somma di numeri x_i di segno concorde, il problema è ben condizionato!

Esempio: errore inerente del prodotto di n numeri

$$F(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$$

I coefficienti di amplificazione sono

$$c_i(x_1, \ldots, x_n) = 1, \quad i = 1, 2, \ldots, n,$$

e pertanto il problema è **sempre** ben condizionato.

Exercise

Calcola il condizionamento nella divisione di due numeri e della radice quadrata di un numero

Esempio: errore inerente della soluzione di un sistema lineare $Ax = b$

Usando le norme vettoriali, le definizioni di errore possono essere estese al caso più generale di $y = F(x)$ con $F : \mathbb{R}^n \to \mathbb{R}^m, n, m, \geqslant 1$.

$$A = \begin{pmatrix} 0.7800 & 0.5630 \\ 0.4570 & 0.3300 \end{pmatrix}, b = \begin{pmatrix} 0.2170 \\ 0.1270 \end{pmatrix}$$

La soluzione esatta è $x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

Perturbiamo l'elemento $a_{1,1} = 0.7800$ della matrice A con $\tilde{a}_{1,1} = 0.7810$. L'errore relativo nel dato perturbato è $\epsilon_{a_{1,1}} = 1.2821 \times 10^{-3}$.

La soluzione del sistema perturbato risulta $\tilde{x} = \begin{pmatrix} 0.2483 \\ 0.0410 \end{pmatrix}$ con un errore misurato in norma infinito pari a $\|\tilde{x} - x\| = 1.0410$.

Esempio

Se il problema è mal condizionato, anche il miglior algoritmo potrà dare risultati non accurati. In questo caso si cerca una riformulazione del problema matematicamente equivalente, ma con un miglior indice di condizionamento.

Example

$$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3 = (\sqrt{2}-1)^6 = 99-70\sqrt{2} = \frac{1}{99+70\sqrt{2}}$$

- $f_1(x) = \left(\frac{x-1}{x+1}\right)^3 \rightsquigarrow \mathrm{cond}_{f_1}(\sqrt{2}) \approx 8.485$
- $f_2(x) = (x-1)^6 \rightsquigarrow \mathrm{cond}_{f_2}(\sqrt{2}) \approx 2.048\ 10^1$
- $f_3(x) = 99-70x \rightsquigarrow \mathrm{cond}_{f_3}(\sqrt{2}) \approx 1.960\ 10^4$ **mal condizionato !**
- $f_4(x) = \frac{1}{99+70x} \rightsquigarrow \mathrm{cond}_{f_4}(\sqrt{2}) \approx 0.5$

Errore analitico o di troncamento

L'errore analitico misura l'errore tra il risultato esatto e quello fornito dall'algoritmo che usa gli stessi dati di input e opera in aritmetica esatta.

Fornisce un'informazione sull'accuratezza della tecnica di approssimazione (modello numerico) usata.

Example

$$f(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \approx g_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$$

Per $x = 1$, $g_n(1)$ fornisce una stima del numero di Nepero e con un errore analitico (assoluto)

$$|e - g_n(1)| = \frac{e^a}{(n+1)!} < \frac{e}{(n+1)!}$$

dove $0 < a < 1$.

Errore analitico o di troncamento, continua

Example

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, h > 0$$

Per l'errore analitico (assoluto) vale per $x \in [a, b]$ e $x + h \in [a, b]$

$$|f'(x) - \frac{f(x+h) - f(x)}{h}| \leqslant \frac{|f''(\xi_x)|h}{2} \leqslant hC$$

dove $x < \xi_x < x + h$ e $C = \max_{t \in [a,b]} \frac{|f''(t)|}{2}$

Errore algoritmico

L'errore algoritmico è generato dal fatto che le operazioni sono effettuate in aritmetica di macchina: misura l'errore tra il risultato fornito da un algoritmo che opera in l'aritmetica esatta e quello prodotto dallo stesso algoritmo in aritmetica di macchina.

L'algoritmo scelto per la valutazione di $g(\tilde{x}), \tilde{x} \in \mathbb{F}$ può essere descritto da una sequenza finita di operazioni elementari (i.e. $\pm, \times, /, \sqrt{\cdot}$) $g_i : \mathbb{R}^{n_i} \to \mathbb{R}, n_i \geqslant 1, i = 1, 2, \ldots, N$.

La descrizione dello stesso algoritmo in aritmetica di macchina,$\tilde{g}$, si ottiene sostituendo ogni operazione in aritmetica esatta $g_i, i = 1, \ldots, N$, con la relativa operazione $\tilde{g}_i : \mathbb{F}^{n_i} \to \mathbb{F}, i = 1, \ldots, N$ in aritmetica di macchina

$$\begin{array}{ccccccccc} g & = & g_N & \circ & g_{N-1} & \circ & \ldots & \circ & g_1 \\ & & \downarrow & & \downarrow & & \ldots & & \downarrow \\ \tilde{g} & = & \tilde{g}_N & \circ & \tilde{g}_{N-1} & \circ & \ldots & \circ & \tilde{g}_1 \end{array}$$

Errore algoritmico e stabilità

Un algoritmo si dice **stabile in avanti** se $|err_{algoritmico}|$ è un **piccolo multiplo** della precisione di macchina u.

Oss: La stabilità in avanti dell'algoritmo **non** garantisce l'accuratezza del risultato finale.

L'accuratezza del risultato dipende dall'**errore totale**.

Un risultato non accurato può risultare sia dall'applicazione di un algoritmo instabile ad un problema ben condizionato sia dall'applicazione di un algoritmo stabile ad un problema mal condizionato.

Le ricette per migliorare sono diverse: scegliere un algoritmo più stabile nel primo caso, riformulare matematicamente il problema nel secondo caso.

Analisi dell'errore algoritmico

Nei casi semplici, a disposizione i **grafi computazionali**.

Un altro approccio consiste nell'**analisi dell'errore all'indietro** [J.H. Wilkinson (1919-1986)].

$$\begin{array}{ccc} \tilde{x} & \longrightarrow & g(\tilde{x}) \\ \text{errore} & \searrow & \text{errore} \\ \text{all'indietro} & & \text{in avanti} \\ \tilde{x}+\epsilon & \longrightarrow & \tilde{g}(\tilde{x}) = g(\tilde{x}+\epsilon) \end{array}$$

La soluzione $\tilde{g}(\tilde{x})$ fornita dall'algoritmo in aritmetica di macchina con dato $\tilde{x} \in \mathbb{F}$ viene vista come la soluzione ottenuta in aritmetica esatta su un dato perturbato $\tilde{g}(\tilde{x}) = g(\tilde{x}+\epsilon)$

Gli errori di arrotondamento sono interpretati come errori nei dati!

L'algoritmo per calcolare $g(\tilde{x})$ si dirà **stabile all'indietro** se, $\forall \tilde{x}$ il risultato $\tilde{g}(\tilde{x})$ sarà la soluzione ottenuta in aritmetica esatta con un dato "vicino" all'originale, i.e. $\frac{|\epsilon|}{|\tilde{x}|}$ è un piccolo multiplo della precisione di macchina u. La definizione di "vicino" può dipendere dal contesto.

Vale

$$|\mathrm{err}_{\mathrm{algoritmico}}| \approx \mathrm{cond}_g(\tilde{x}) \frac{|\epsilon|}{|\tilde{x}|},$$

dove $\mathrm{cond}_g(x)$ è il numero di condizionamento della funzione g.

Example

La somma di due numeri $x_1, x_2 \in \mathbb{F}$ con l'operazione di macchina $\mathrm{fl}(+)$ può essere riscritta come segue

$$x_1 \mathrm{fl}(+)\, x_2 = (x_1 + x_2)(1 + e) = x_1(1 + e) + x_2(1 + e) = \tilde{x}_1 + \tilde{x}_2$$

dove $|e| \leqslant u$.

La somma e la differenza sono operazioni "stabili all'indietro".

Analisi all'indietro, continua

Per l'analisi all'indietro applicata al problema $y = f(x), x \in \mathbb{F}$ si ha

$$\begin{array}{ccc} x & \longrightarrow & y = f(x) \\ \text{erroreall'indietro} & \searrow & \text{erroreinavanti} \\ x + \epsilon & \longrightarrow & \tilde{y} = \tilde{g}(x) = f(x + \epsilon) \end{array}$$

La soluzione $\tilde{y} = \tilde{g}(x)$ fornita dall'algoritmo in aritmetica di macchina con dato $x \in \mathbb{F}$ viene vista come la soluzione del problema su un dato perturbato $\tilde{g}(\tilde{x}) = f(x + \epsilon)$

Un metodo per calcolare $y = f(x)$ si dirà **stabile all'indietro** se, $\forall x \in \mathbb{F}$, il risultato $\tilde{y}$ sarà soluzione del problema con un dato "vicino" all'originale, i.e. $\frac{|\epsilon|}{|x|}$ è un piccolo multiplo della precisione di macchina u.

Vale

$$|\text{errorre in avanti}| \approx \text{cond}_f(x)|\text{errorre all'indietro}|$$

dove $\text{cond}_f(x)$ è in numero di condizionamento relativo a f. La soluzione calcolata di una problema mal condizionato può avere un grande errore in avanti.

Esempio di analisi all'indietro

Example

Per la funzione esponenziale $f(x) = e^x$ otteniamo

$$\tilde{y} = e^{x+\epsilon} \Rightarrow x + \epsilon = \log(\tilde{y})$$

Per esempio $x = 1$ e $n = 3$ abbiamo

$$e^x = e = 2.7182818284590..., \tilde{y} = 2.666666666666667$$
$$x + \epsilon = \log(\tilde{y}) = 0.980829253011\,73$$
$$\text{errore in avanti} = \tilde{y} - e^x = -0.051615161792\,38$$
$$\text{errore all'indietro} = \epsilon = -0.019170746988\,27$$

Cancellazione

La sottrazione di due numeri approssimati (a causa di errori di arrotondamento o di altri errori attribuibili a computazioni precedenti) aventi lo stesso segno e modulo quasi uguale può essere causa di inaccuratezza nel risultato.

Tale fenomeno è indicato come **cancellazione**.

Example

$$0.192403\ 10^2 - 0.192275\ 10^2 = 0.128000\ 10^{-1}$$

Il risultato è corretto e esattamente rappresentabile, ma ha solo tre cifre significative, perchè le cifre principali sono state **cancellate**.

Anche se il risultato è esatto, la cancellazione implica una perdita di informazioni che può essere causa di risultati non accurati.

Cancellazione: calcolo delle radici di equazioni di secondo grado

Le soluzioni dell'equazione di secondo grado $ax^2+bx+c=0$ sono fornite dalla ben nota formula

$$x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}.$$

Ma, mentre il problema matematico è banale, dal punto di vista numerico tale formula richiede più attenzione in quanto può dare risposte inaccurate o essere non valutatabile in corrispondenza di particolari scelte dei dati a, b e c.

- $b^2 \gg |4ac| \rightsquigarrow \sqrt{b^2-4ac} \approx |b|$

  Il fenomeno della cancellazione si può presentare nel calcolo di una delle due soluzioni: $\sqrt{b^2-4ac}$ sarà affetto da errori di approssimazione e pertanto la sottrazione amplificherà tale errore. La cancellazione può essere evitata calcolando

$$x_1 = \frac{-(b+\text{sign}(b)\sqrt{b^2-4ac})}{2a}$$

$$x_2 = \frac{c}{}$$

Cancellazione: radici di equazioni di secondo grado, continua

- Una cancellazione ben più pericolosa numericamente si può presentare nella valutazione dell'espressione $b^2 - 4ac$.

  Nel caso di radici quasi coincidenti (i.e. $b^2 \approx 4ac$), ci può essere una significativa perdita di accuratezza nel risultato, che **non** può essere risolta con una trasformazione algebrica. L'unica possibilità consiste nel valutare tale espressione con una precisione maggiore.

- Un'altra difficoltà numerica può essere dovuta all'overflow, che, per esempio, si può incontrare nel calcolo di b^2 e $4ac$ con a, b e c con esponente $p = (p_{max} + 1)/2$. In alcuni casi l'overflow si può risovere con uno "scaling" dei dati.

Cancellazione: deviazione standard

La media m_n di n numeri $x_1, \ldots, x_n$, è data da

$$m_n = \frac{\sum_{i=1}^{n} x_i}{n}$$

mentre la deviazione standard σ_n è

$$\sigma_n = \sqrt{\frac{\sum_{i=1}^{n} (x_i - m_n)^2}{n}}.$$

Per calcolare la deviazione σ_n, spesso, onde evitare due passaggi di dati, viene proposta la seguente formula matematicamente equivalente

$$\sigma_n = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2}{n}}.$$

In presenza di errori di arrotondamento, questa seconda formulazione che calcola la deviazione come differenza di due numeri positivi, può fornire una risposta non accurata per il fenomeno della cancellazione.

Cancellazione: deviazione standard, continua

Un algoritmo numericamente stabile per calcolare la deviazione standard è il seguente: definite le quantità

$$m_k = \frac{\sum_{i=1}^{k} x_i}{k}, \quad k = 1, \ldots, n,$$

e

$$q_k = \sum_{i=1}^{k} (x_i - m_k)^2, \quad k = 1, \ldots, n,$$

valgono le relazioni ricorsive

$$m_1 = x_1, \quad m_k = m_{k-1} + \frac{x_k - m_{k-1}}{k}, \quad k = 2, \ldots, n,$$

$$q_1 = 0, \quad q_k = q_{k-1} + \frac{(x_k - m_{k-1})^2 (k-1)}{k}, \quad k = 2, \ldots, n,$$

che mi permettono di ricavare la deviazione standard da $\sigma_n^2 = q_n/n$.

Cancellazione: riformulazione dell'algoritmo di Archimede

L'algoritmo di Archimede può essere riscritto, per evitare la cancellazione, nel seguente modo

$$
\begin{aligned}
l_1 &= \sqrt{2}, \\
l_{i+1} &= \frac{l_i}{\sqrt{2+\sqrt{4-l_i^2}}}, \quad i = 1, 2, \ldots, \\
p_i &= l_i \times 2^i, \quad i = 1, 2, \ldots.
\end{aligned}
$$

Cancellazione: l'algoritmo di Archimede risultati

i	p_i	err_{ass}	err_{rel}
1	2.828427124746190	3.13e-01	9.97e-02
2	3.061467458920718	8.01e-02	2.55e-02
4	3.136548490545939	5.04e-03	1.60e-03
5	3.140331156954753	1.26e-03	4.02e-04
⋮	⋮	⋮	⋮
11	3.141592345570118	3.08e-07	9.80e-08
12	3.141592576584873	7.70e-08	2.45e-08
13	3.141592634338564	1.92e-08	6.13e-09
14	3.141592648776986	4.81e-09	1.53e-09
15	3.141592652386592	1.20e-09	3.83e-10
16	3.141592653288993	3.01e-10	9.57e-11
17	3.141592653514594	7.52e-11	2.39e-11
⋮	⋮	⋮	⋮
22	3.141592653589721	7.19e-14	2.29e-14
23	3.141592653589776	1.69e-14	5.37e-15
24	3.141592653589790	3.11e-15	9.89e-16
25	3.141592653589794	4.44e-16	1.41e-16

Cancellazione: calcolo della funzione e^x per $x < 0$

L'instabilità dell'algoritmo visto in precedenza per il calcolo di e^x per $x < 0$ può essere risolta riscrivendo $e^x = 1/e^{-x}$ per $x < 0$.

x	n	$g_n(x)$	err_{rel}
-0.5	16	6.065306597126335e-01	1.83e-16
-1	20	3.678794411714423e-01	1.50e-16
-20	68	2.061153622438558e-09	2.01e-16
-40	103	4.248354255291590e-18	1.81e-16
-100	192	3.720075976020839e-44	8.03e-16

Derivata e rapporto incrementale

Assumendo che l' errore relativo nei dati di input $f(x), f(x+h)$ sia maggiorato dalla precisione di macchina u e che h sia un numero di macchina, si ha

$$|err_{algoritmo}| \leqslant 2u + u\frac{|f(x+h)| + |f(x)|}{|f(x+h) - f(x)|} \approx 2u + 2u\frac{|f(x)|}{h|f'(x)|}$$

L'andamento dell'errore di arrotondamento rispetto al parametro h è diverso dall'errore analitico: riducendo h tale errore diventa arbitrariamente grande causa la cancellazione.
Il passo h_{ott} che fornisce la maggiorazione ottimale si può stimare ponendo $\frac{Ch}{2|f'(x)|} = \cdot\frac{2|f(x)|u}{h|f'(x)|}$ da cui si ottiene

$$h_{ott} \simeq 2\sqrt{\frac{u|f(x)|}{C}}.$$

Somma di n numeri

$$s_n = \sum_{i=1}^{n} x_i, \quad x_i \in \mathbb{F}, i = 1, \ldots, n$$

Per $\mathrm{fl}(+)$ non vale la proprietà associativa e l'errore in un singolo passo dipende dagli addendi da sommare $\rightsquigarrow$ il valore finale calcolato in aritmetica di macchina dall'algoritmo della **somma ricorsiva**

```
s(1)=x(1)
for i=2:n
    s(i)=s(i-1)+x(i);
end
```

dipenderà dall'ordinamento. In prima approssimazione vale

$$err_{algoritmo} = \frac{1}{s_n} \sum_{i=2}^{n} s_i \delta_{i-1}$$

dove δ_i è l'errore nell'i-esima operazione di somma.

Da $|\delta_i| \leqslant u, i = 1, \ldots, n-1$, si ottiene

$$|err_{algoritmo}| \leqslant \frac{u}{|s_n|} \sum_{i=2}^{n} |s_i|,$$

e, dalla maggiorazione $|s_i| \leqslant \sum_{k=1}^{n} |x_k|, i = 2, \ldots, n$, segue

$$|err_{algoritmo}| \leqslant (n-1)u \frac{\sum_{i=1}^{n} |x_i|}{|s_n|}$$

In particolare, se i dati x_i, $i = 1, \ldots, n$, hanno lo **stesso segno**

$$|err_{algoritmo}| \leqslant (n-1)u.$$

Somma di n numeri: l'algoritmo PSUM

In generale possiamo dedurre che l'accuratezza migliore si otterrà scegliendo l'ordinamento che **minimizza** $|s_i|, i = 2, \ldots, n$. Questo problema di ottimizzazione combinatoria è troppo costoso da risolvere.

Se x_i, $i = 1, \ldots, n$ hanno stesso segno, la soluzione ottimale è data dall'ordinamento di modulo crescente che fornisce la maggiorazione

$$|err_{algoritmo}| \leqslant \frac{(n+1)u}{2}.$$

Nel caso generale, si possono proporre soluzioni di compromesso quali l'ordinamento per modulo crescente oppure l'ordinamento determinato sequenzialmente minimizzando passo dopo passo $|s_1|, |s_2|, \ldots$ (**algoritmo PSUM**)

```
s(0)=0;
for k=1:n-1
    j(k) tale che min{|x(j)+s(k-1)|, j=k,...,n}
    j(k) <---->k
end
```

L'ordine decrescente può fornire una risposta più accurata sia di PSUM che dell'ordinamento crescente in presenza di forte cancellazione.

Example

Sia $x \in \mathbb{F}$ tale che $\mathrm{fl}(1+x) = x$ e supponiamo di dover sommare 1, x, $2x$ e $-3x$. I risultati che si ottengono sono

- ordinamento crescente: $\mathrm{fl}(1 + x + 2x - 3x) = 0$;
- ordinamento PSUM: $\mathrm{fl}(1 + x - 3x + 2x) = 0$;
- ordinamento decrescente: $\mathrm{fl}(-3x + 2x + x + 1) = 1$.

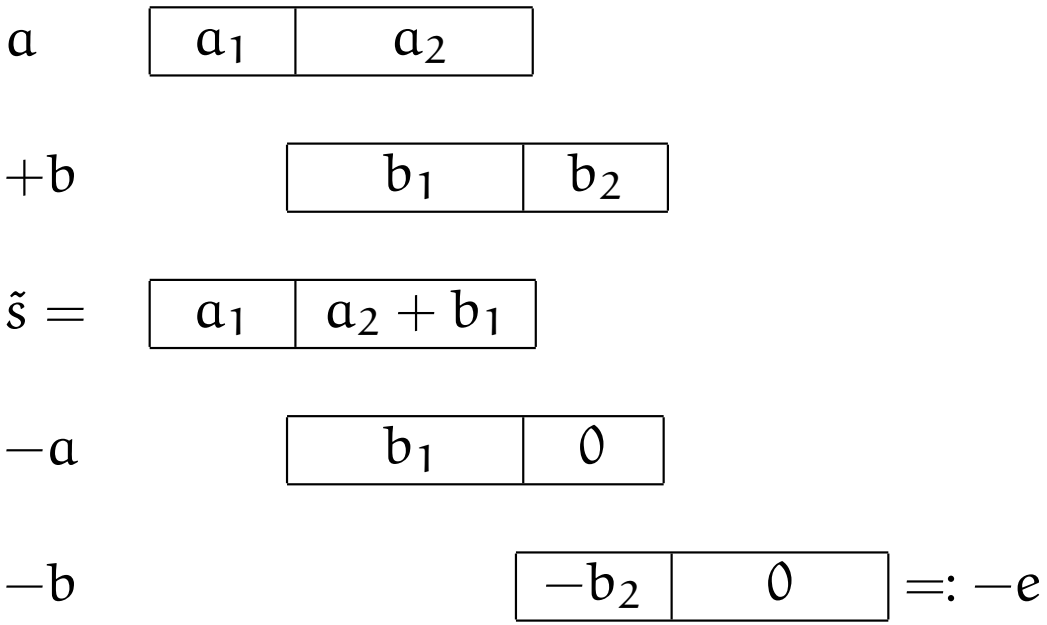
Somma di n numeri: l'algoritmo dell'inserzione e della somma binaria

Nell' **algoritmo dell'inserzione** i dati x_i vengono ordinati secondo il modulo crescente e ogni somma parziale s_i calcolata per $i = 2, \ldots, n$ viene inserita nella lista $x_{i+1}, \ldots, x_n$ mantenendo l'ordinamento crescente.

L'**algoritmo della somma binaria** per $n = 2^k$, somma al primo passo gli addendi a coppie $y_i = x_{2*i-1} + x_{2*i}$, $i = 1, \ldots, n/2 = 2^{k-1}$ e ripete il procedimento ricorsivamente a partire da y_i. Il risultato si raggiunge in $\log_2 n$ passi.

Somma di n numeri: algoritmo di somma compensata

Particolarmente interessante è l'**algoritmo di somma compensata** [Kahan (1972)]. Dati due numeri $a, b \in \mathbb{F}, |a| > |b|$, sia $\tilde{s} = a\mathrm{fl}(+)b$



suggerisce che

$$e = -[((a\mathrm{fl}(+)b)\mathrm{fl}(-)\ a)\mathrm{fl}(-)\ b] = (a\mathrm{fl}(-)\ \tilde{s})\mathrm{fl}(+)\ b$$

è una stima dell'errore commesso $(a + b) - \tilde{s}$.

L'algoritmo di Kahan applica la correzione fornita da e ad ogni passo della somma ricorsiva, i.e.

```
s=0; e=0
for i=1:n
    t=s
    y=x(i)+e
    s=t+y
    e=(t-s)+y
end
s=s+e
```

E' stato dimostrato [Knuth (1981)] che vale la maggiorazione

$$|err_{algoritmo}| \leqslant \left(2u + O(nu^2)\right) \frac{\sum_{i=1}^{n} |x_i|}{|s_n|}.$$

Finchè $nu \leqslant 1$ la costante di questa limitazione è indipendente da n. In caso di forte cancellazione anche l'algoritmo di somma compensata non è in grado di garantire un errore algoritmico piccolo.

Cancellazione degli errori di arrotondamanto: esempio

La valutazione della funzione $f(x) = (e^x - 1)/x$ presenta il fenomeno della cancellazione per valori di $x \approx 0$. Si possono proporre due algoritmi seguenti

- **Algoritmo 1:**

```
if x=0 then
    f=1
else
    f=(exp(x)-1)/x
end
```

- **Algoritmo 2:**

```
y=exp(x)
if y=1 then
    g=1
else
    g=(y-1)/log(y)
end
```

Confronto tra gli algoritmi 1 e 2 per il calcolo di $f(x) = (e^x - 1)/x$.

x	Algoritmo1	Algoritmo2
1.e-05	1.000005000006965e+00	1.000005000016667e+00
1.e-06	1.000000499962183e+00	1.000000500000167e+00
1.e-07	1.000000049433680e+00	1.000000050000002e+00
1.e-10	1.000000082740371e+00	1.000000000050000e+00
1.e-11	1.000000082740371e+00	1.000000000005000e+00
1.e-12	1.000088900582341e+00	1.000000000000500e+00
1.e-13	9.992007221626408e-01	1.000000000000050e+00
1.e-14	9.992007221626408e-01	1.000000000000005e+00
1.e-15	1.110223024625157e+00	1.000000000000000e+00
1.e-16	0	1

Commenti sui risultati

L'algoritmo 1 risente della cancellazione per $|x| \ll 1$ e questo causa la risposta non accurata.

L'algoritmo 2 valuta in maniera approssimata sia $y = e^x$ che $\log(y)$, ma pur ottenendo valori poco accurati per il numeratore e denominatore il loro rapporto risulta molto accurato $\rightsquigarrow$

gli errori di arrotondamanto si cancellano nella divisione!!

Questo risultato si spiega con l'analisi del condizionamento di $g(y)$ in un intorno di $y = 1$. Posto $v = y - 1$ vale vale

$$g(1+v) = \frac{v}{\log(1+v)} = \frac{v}{v - v^2/2 + v^3/3 + \ldots)} = 1 + \frac{v}{2} + O(v^2)$$

da cui si ottiene per $y \approx 1$

$$g(\tilde{y}) - g(y) \approx \frac{\tilde{y} - y}{2} \approx \frac{y\epsilon_y}{2} \approx \frac{\epsilon_y}{2} \approx \frac{\epsilon_y g(y)}{2}$$

Instabilità senza cancellazione

Si vuole analizzare il risultato dell'algoritmo seguente per $x > 0$

$$
\begin{aligned}
y_1 &= x^2 \\
y_{i+1} &= \sqrt{y_i}, \quad i = 1, \ldots, m \\
z_1 &= y_{m+1} \\
z_{i+1} &= z_i^2, \quad i = 1, \ldots, m-1
\end{aligned}
$$

Implementando in MATLAB l'algoritmo si ottiene

```
>>x=.25:.25:1.5
x=
   0.2500    0.5000    0.7500    1.0000    1.2500    1.5000
>>[z,err]=mroot(x,50)
z=
   0.2375    0.4724    0.7316    1.0000    1.1331    1.4550
err=
   0.0499    0.0553    0.0245         0    0.0935    0.0300
```

Analizziamo l'errore algoritmico mediante l'uso dei grafi computazionali in maniera ricorsiva.

$\sqrt{\cdot}$ ha un coefficiente di amplificazione pari a 1/2 mentre per $(\cdot)^2$ è 2

$$\begin{aligned} |\epsilon_{alg}| &\leqslant u\left(\sum_{i=0}^{m-1} 2^i + 2^{m-1}\sum_{i=0}^{m} 2^{-i}\right) = \\ &= u\left(2^m + 2^{m-1} - \tfrac{3}{2}\right) = \tfrac{3u}{2}\left(2^m - 1\right). \end{aligned}$$

L'instabilità dell'algoritmo aumenta con m con coefficiente di amplificazione totale dell'ordine di 2^m. Per IEEE-DP $u = 2^{-53}$, con $m = 50$ l'errore algoritmico totale risulta dell'ordine di $3/16 = 0.1875$.

Osservazioni conclusive

L'analisi degli errori di rappresentazione e dell'aritmetica di macchina è stata un importante argomento dell'analisi numerica quando il calcolatore è diventato uno strumento essenziale nella risoluzione di problemi matematici complessi, perchè ha posto l'attenzione sul ruolo che esso gioca nell'accuratezza della risposta numerica e ha permesso di definire importanti concetti come la **stabilità di algoritmi** e il **condizionamento del problema**.

Qui

- http://www.ima.umn.edu/ arnold/455.f96/disasters.html
- http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html

sono presentati dei casi reali in cui computazioni numeriche non eseguite in maniera attenta hanno causato situazioni disastrose.

Ma il cuore della disciplina sta nello sviluppo e analisi di **algoritmi** accurati ed efficienti.