TEORIA degli ERRORI

Francesca Pelosi

Dipartimento di Matematica, Università di Roma "Tor Vergata"

CALCOLO NUMERICO e PROGRAMMAZIONE

http://www.mat.uniroma2.it/∼pelosi/

ANALISI NUMERICA

L'analisi Numerica è lo studio degli algoritmi per la soluzione di problemi della matematica del continuo
N. Trefethen

Notare l'uso (ed il leggero abuso) della nozione di algoritmo: in informatica la nozione di algoritmo è associata alla terminazione in un tempo finito, come vedremo questo non è immediatamente vero nei problemi che affronteremo.

ANALISI NUMERICA

OBBIETTIVO: dare una risposta numerica ad un problema matematico mediante un calcolatore automatico digitale

1. Problema reale
2. Costruire un modello
3. Formulare un problema matematico
4. Risolvere il problema matematico
5. Interpretare la soluzione

Esempio

1. PROBLEMA REALE: calcolare il periodo di oscillazione di un pendolo semplice, note le caratteristiche fisiche.
2. COSTRUZIONE MODELLO: dalla dinamica classica si fanno le seguenti ipotesi:
   1. Sfera puntiforme
   2. Filo flessibile ed inestensibile
   3. Attriti trascurabili
3. FORMULAZIONE MATEMATICA: radialmente la risultante delle forze è nulla, tangenzialmente (componente forza peso)

$$F = -mg \sin \theta$$

spostamento tangenziale: $x = \ell\theta$

Esempio

da $\mathbf{F} = m\mathbf{a} \Rightarrow m\ddot{x} + mg\sin\theta = 0$ con $x = \ell\theta \Rightarrow \ddot{\theta} + g\sin\theta/\ell = 0$

$$T = 2\pi\sqrt{\frac{\ell}{g}}\left(1 + \frac{1}{4}\left(\sin(\frac{\theta}{2}m)\right)^2 + \frac{9}{64}\left(\sin(\frac{\theta}{2}m)\right)^4 + \cdots\right) \qquad (1)$$

Formula troppo complessa $\Rightarrow$ ipotesi: piccole oscillazioni

$\theta \simeq 1°$ da cui $\sin\theta \simeq \theta = \frac{x}{\ell}$:

$$T = 2\pi\sqrt{\frac{\ell}{g}} \qquad (2)$$

4. RISOLVERE IL PROBLEMA MATEMATICO: Calcolando (1) occorre necessariamente troncare lo sviluppo in serie; l'espressione in (2), contenendo una radice non può essere calcolata esattamente, occorre una strategia per il suo calcolo approssimato.
   $\Rightarrow$ Si commettono errori di troncamento dovuti alla sostituzione dell'espressione teorica con una effettivamente calcolata.

Esempio

Inoltre i dati (ℓ, g) sono affetti da errore quindi

$$T(\ell, g) \rightarrow \hat{T}(\ell + \delta\ell, g + \delta g)$$

dove

$\delta\ell,\ \delta g$ sono gli errori sui dati

$\hat{T}$ è l'approssimazione calcolata di T

5. INTERPRETAZIONE DELLA SOLUZIONE: gli errori introdotti rendono la soluzione accettabile?

Errori

- In ognuno dei passi 2. 3. 4. si introducono degli errori:

  2-3 Ipotesi semplificative nella costruzione del modello

  4. 1 Errori di misura e errori di rappresentazione dei dati

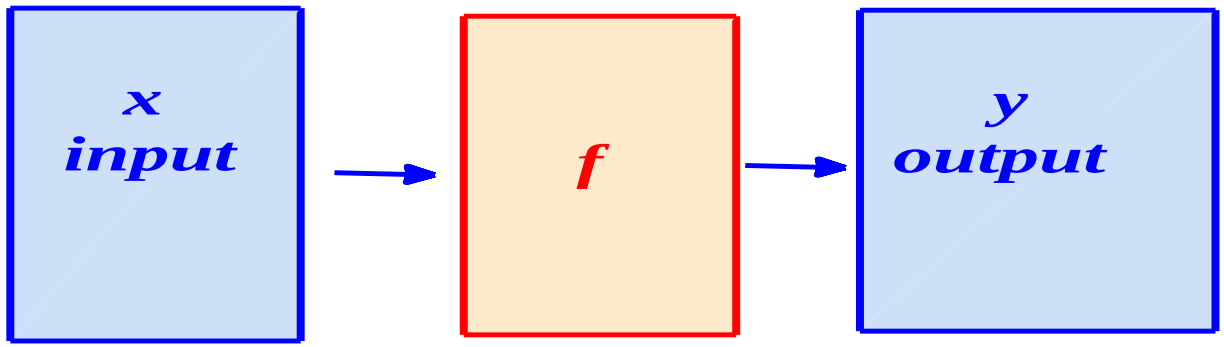
  4. 2 Errori di troncamento: approssimazione di un processo infinito con uno finito di calcolo effettivo

  4. 3 Errori di arrotondamento nei calcoli: esecuzione effettiva dell'algoritmo.

- È importante che gli errori introdotti nei vari stadi siano dello stesso ordine.
- È inutile calcolare accuratamente la soluzione di un problema matematico derivante da una modellizzazione non accurata.
- L'Analisi numerica si occupa essenzialmente della fase 4.

Errori

- Ricapitolando la fase 4 si può schematizzare come segue



DEF: *Data una grandezza (scalare)* z *e una sua approssimazione* $\widetilde{z}$ *definiamo*

$\delta z = \widetilde{z} - z$ *Errore assoluto* ($\widetilde{z} = z + \delta z$)

$\varepsilon_z = \frac{\widetilde{z}-z}{z}$ *Errore relativo* ($\widetilde{z} = z + \varepsilon z$ *per* $z \neq 0$).

Errori

- I dati x sono affetti da errore (di misura o di rappresentazione dovuta al calcolatore) δx (errore inerente A)
$$\frac{f(x+\delta x)-f(x)}{f(x)}$$
- Anziché calcolare f si calcola una sua approssimazione g (numero finito di passi) (errore analitico o di troncamento B)
$$\frac{g(x)-f(x)}{f(x)}$$
- Il calcolo di g viene effettuato con l'esecuzione di uno specifico algoritmo su un calcolatore (precisione finita) ottenendo una approssimazione $\widetilde{g}$ (errore algoritmico C)
$$\frac{\widetilde{g}(x)-g(x)}{g(x)}$$
- Alla fine invece di $y=f(x)$ si ottiene $\tilde{y}=\widetilde{g}(x+\delta x)$

Condizionamento e stabilità

A: Dipende dal problema e dal dato e non dall'algoritmo utilizzato

DEF: *Un problema si dice bencondizionato se a piccole variazioni sui dati corrispondono piccole variazioni sui risultati*

$$\left| \frac{f(x+\delta x) - f(x)}{f(x)} \right| \simeq \left| \frac{\delta x}{x} \right| = \varepsilon_x$$

C: Dipende dalle caratteristiche dell'algoritmo utilizzato (operazioni e precisione) ed è dovuto all'uso di aritmetica finita

DEF: *Un algoritmo si dice stabile se amplifica poco (relativamente alla caratteristiche del calcolatore) gli errori di arrotondamento introdotti nelle singole operazioni*

Esempi

- Malcondizionamento :

$$\begin{aligned} (x-1)^4 &= 0 \quad \Rightarrow x_i = 1 \\ (x-1)^4 &= 10^{-8} \quad \Rightarrow x_i = 1 \pm 10^{-2} \end{aligned}$$

- Stabilità : calcolo di e^x per $x = -9$

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^k}{k!} + \cdots \\ e^{-9} &= 1 - 9 + \frac{81}{2} - \frac{729}{3!} + \cdots = \left(1 + 9 + \frac{81}{2} + \frac{729}{3!} + \cdots\right)^{-1} \\ &= (e^9)^{-1} \end{aligned}$$

- sommando fino al 23-esimo termine, con la seconda espressione O.K., mentre con la prima si ottiene un risultato negativo!

Osservazioni

- Un problema può essere malcondizionato per certi dati e per altri no.
- Anche un problema bencondizionato può dare risultati non sufficientemente corretti se risolto con un algoritmo instabile.
- La stabilità di un algoritmo è data dal bencondizionamento della successione delle trasformazioni elementari che lo compongono.

Complessità di calcolo

- Un algoritmo è una successione finita di istruzioni, assegnate in modo non ambiguo, la cui esecuzione consenta di passare da una situazione iniziale (dati) ad una situazione finale (risultati), in un tempo finito.
- I requisiti fondamentali di un algoritmo sono la generalità: adatto a risolvere una classe di problemi e l'ottimalità, rispetto al tempo, al numero di operazioni, alla stabilità, ...
- Due algoritmi si possono confrontare anche rispetto al numero di operazioni. Maggior numero di operazioni corrisponde a maggior tempo di esecuzione, ma non sempre a peggior accuratezza
- Aumentando $\sharp$ cifre per rappresentare un numero aumenta il tempo di esecuzione delle operazioni (linearmente per somme, quadraticamente per moltiplicazioni)

Esempio: algoritmo di Horner

- Calcolo del valore di un polinomio in un punto

$$P(x) = = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_n x^n$$

$$= a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + x a_n)) \cdots)$$

Algoritmo 1

1. `Dati` $a_0, \ldots, a_n, x$
2. `Poni` $s := a_0, xp := 1;$
3. `Per` $i = 1, \ldots, n$
   1. $xp := xp * x$
   2. $s := s + a_i * xp$
4. `Stampa` s
5. `Stop`

$2n$ Molt. + n Add.

Algoritmo 2

1. `Dati` $a_0, \ldots, a_n, x$
2. `Poni` $s =: a_n$
3. `Per` $i = n-1, n-2, \ldots, 0$
   1. $s := s * x + a_i$
4. `Stampa` s
5. `Stop`

n Molt. + n Add.
OTTIMALE

Rappresentazione dei numeri

- Il sistema di numerazione decimale è basato sull'alfabeto decimale $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ed ogni numero è rappresentato come una sequenza di simboli di tale alfabeto. Ad ogni simbolo è associato un peso a seconda della posizione (notazione posizionale).
- Esempio: $2863 = 2 * 10^3 + 8 * 10^2 + 6 * 10^1 + 3 * 10^0$.
  In generale i sistemi numerici posizionali in base $\beta \geq 2$ rappresentano ogni numero con m cifre in base β:

$$N = d_{m-1} \dots d_0$$

- Dove i d_i denotano elementi di un insieme di β simboli che corrispondono ai primi β numeri naturali $0, \dots, \beta - 1$.
- Esempio: $(1100)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 12$

Rappresentazione dei numeri

- TEOREMA: *Sia data una base $\beta \geq 2$. Sia $x \in \mathbb{R}$ con $x \neq 0$, esiste uno e un solo intero e ed una successione $\{d_i\}$, $i = 1, 2, \ldots,$ con*

  1) $0 \leq d_i \leq \beta - 1, \quad d_1 \neq 0$

  2) d_i *non definitivamente uguali a* $\beta - 1$

  t.c.

$$x = \mathrm{segn}(x)\beta^e \sum_{i \geq 1} d_i \beta^{-i}$$

(Rappresentazione normalizzata in base β)

- DEF:
  - e: esponente o caratteristica
  - d_i: cifre della rappresentazione $\Rightarrow (0.d_1 d_2 \ldots)\beta^e \mathrm{segn}(x)$
  - $\sum_{i \geq 1} d_i \beta^{-i}$: mantissa
- $423.25 = 0.42325\ 10^3 = 0.0042325\ 10^5$

Numeri di macchina

- Poiché si hanno a disposizione un numero finito di bits, solo un sottoinsieme dei numeri reali è rappresentabile in un computer. Tale sottoinsieme è detto insieme dei numeri di macchina. Si suppone di rappresentare i numeri reali con il sistema posizionale a virgola mobile normalizzata

$$\pm(0.d_1 d_2 \ldots)\beta^e$$

- dove la prima cifra d_1 DEVE essere sempre diversa da zero. In generale l'insieme dei numeri rappresentabili in un calcolatore è dato da

$$F(\beta, t, L, U)$$

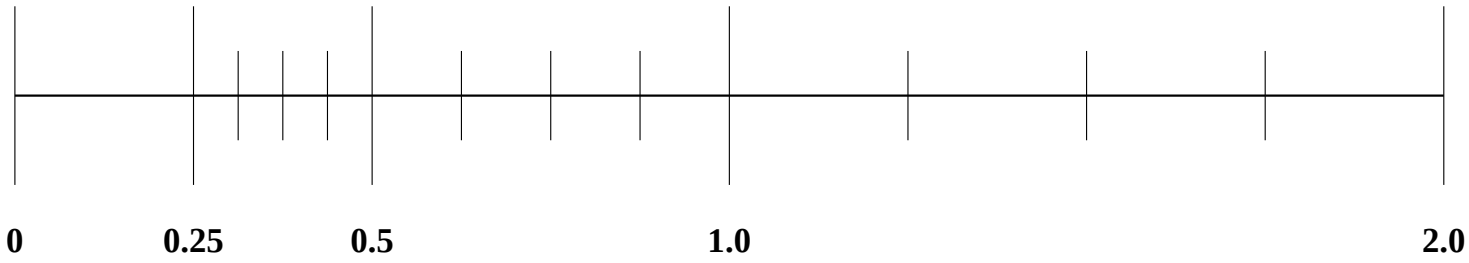
  - β è la base
  - t sono le cifre di mantissa
  - L e U sono i valori minimo e massimo dell'esponente

Numeri di macchina

Ad esempio con $\beta = 2$, $t = 3$, $L = -1$, $U = 3$ si hanno i valori:

$$0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875$$

$$1.0, 1.25, 1.50, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0$$



Numeri di macchina

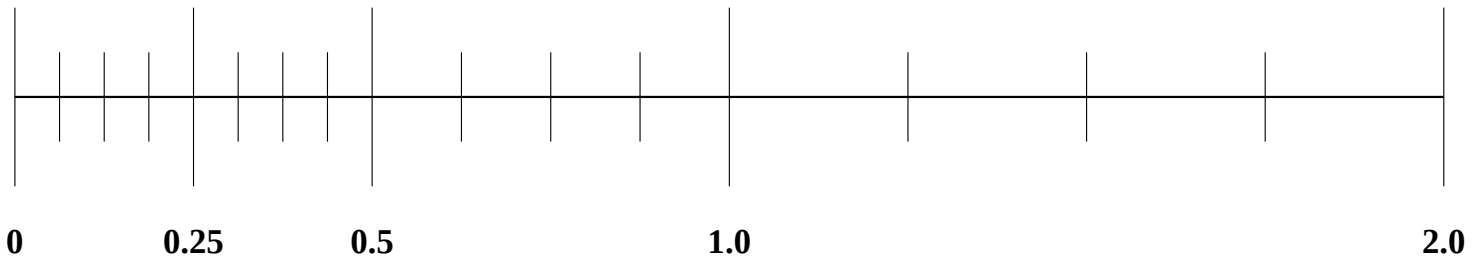
Se $\beta = 2$ allora

$$d_1 \neq 0 \Rightarrow d_1 = 1$$

pertanto il bit può essere sottinteso (“fantasma”).
Per avere spaziatura regolare, si introducono i numeri “denormalizzati” con $e = L, d_1 = 0$; nel nostro esempio

$$0.0625, 0.125, 0.1875$$

e vanno a riempire il “buco” nell’intorno dell’origine come segue:



Errori di rappresentazione

$$x = \pm(0.d_1 d_2 \ldots)\beta^e$$
$$fl(x) = \pm(0.d_1 d_2 \ldots d_t)\beta^e, \quad L \leq e \leq U$$

- Il numero di cifre che l'elaboratore riserva alla mantissa dei numeri reali (t) determina la precisione con la quale l'elaboratore lavora
  - Errori di underflow: l'esponente e è minore di L
  - Errori di overflow: l'esponente e è maggiore di U
  - Errori di arrotondamento: se e è compreso tra L e U ma il numero di cifre è maggiore di t

Precisione

- Sia $x \neq 0$ un numero reale tale che la sua rappresentazione in virgola mobile normalizzata richiede più di t cifre di mantissa $\Rightarrow$ dobbiamo fare un'approssimazione (errore di arrotondamento) e sostituire x con il numero reale di macchina più vicino (floating di x : $fl(x)$)
- La determinazione di $fl(x)$ può avvenire in due modi: troncamento o arrotondamento:
  - troncamento $fl(x) = \pm(0.d_1 d_2 \ldots d_t)\beta^e$
  - arrotondamento:

$$fl(x) = \begin{cases} \pm(0.d_1 d_2 \ldots d_t)\beta^e, & d_{t+1} < \beta/2 \\ \pm(0.d_1 d_2 \ldots (d_t+1))\beta^e, & d_{t+1} > \beta/2 \end{cases}$$

- *round to even*: se $d_{t+1} = \beta/2$ si sceglie di rendere d_t pari.

Errori

- Consideriamo l'errore assoluto e relativo

$$e = |x - fl(x)|, \quad e_r = \frac{|x - fl(x)|}{|x|}$$

- L'errore assoluto è influenzato dall'ordine di grandezza del dato, mentre l'errore relativo non è influenzato dall'ordine di grandezza del dato.
- L'errore relativo dà indicazioni sull'approssimazione operata sulla mantissa del dato ovvero sull'accuratezza (o precisione) con cui il dato è approssimato
- L'errore relativo non dipende dalla caratteristica ma dalla mantissa
- Se vogliamo una misura della precisione con cui l'elaboratore approssima un qualunque numero reale dobbiamo svincolarci da questa dipendenza COME?

Precisione di macchina

- Considerando una limitazione superiore degli errori di arrotondamento relativi:
- Limitazione superiore degli errori assoluti
  - con troncamento $|x - fl(x)| < \beta^{e-t}$
  - con arrotondamento $|x - fl(x)| \leq \frac{1}{2}\beta^{e-t}$
- Limitazione superiore degli errori relativi
  - con troncamento

$$\frac{|x - fl(x)|}{|x|} < \beta^{1-t} = \varepsilon_m, \quad \forall x \neq 0$$

  - con arrotondamento

$$\frac{|x - fl(x)|}{|x|} < \frac{1}{2}\beta^{1-t} = \frac{1}{2}\varepsilon_m, \quad \forall x \neq 0$$

Precisione di macchina

- La quantità ε_m è detta precisione di macchina: dipende da β, t e dalla tecnica utilizzata
- La conoscenza di ε_m è fondamentale per valutare i risultati e scegliere le tolleranze (nei programmi che traducono algoritmi)
- Da quanto visto:

$$fl(x) = x(1+\varepsilon) \quad |\varepsilon| < \varepsilon_m$$

Aritmetica IEEE-754

L'aritmetica di tutti i computer "normali" attualmente in uso.

Singola precisione $\beta = 2$, $t = 23 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -127$, $e_{max} = 127$, $\epsilon_m = 1.2 \times 10^{7}$, range $10^{\pm 38}$

Doppia precisione $\beta = 2$, $t = 52 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -1023$, $e_{max} = 1023$, $\epsilon_m = 2.2 \times 10^{-16}$, range $10^{\pm 308}$

Precisione estesa $\beta = 2$, $t = 63 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -16383$, $e_{max} = 16383$, $\epsilon_m = 1.1 \times 10^{-19}$

Precisione quadrupla $\beta = 2$, $t = 112 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -16383$, $e_{max} = 16383$, $\epsilon_m = 1.9 \times 10^{-34}$

Aritmetica IEEE-754

Se $e = 2048$, l'esponente effettivo vale $2048 - 1023 = 1024$; con $f = 0$ rappresenta l'infinito.
Se invece $f \neq 0$ si ottiene un NaN (Not a Number)

- $1/0 = \infty$
- $-1/0 = -\infty$
- $0/0 = \infty - \infty = \infty/\infty = 0 \times \infty = \sqrt{-1} = NaN$

Aritmetica finita

- Dati due numeri macchina x, y non è detto che $x\ (\mathrm{op})\ y$ sia ancora un numero macchina $\Rightarrow$ occorre definire un aritmetica approssimata o aritmetica finita:

$$x(\mathrm{op})y = fl(x\ \mathrm{op}\ y)$$

- In modo che valga: $x\ (\mathrm{op})\ y = (x\ \mathrm{op}\ y)(1+\varepsilon) \quad |\varepsilon| < \varepsilon_m$
- Un aritmetica basata sull'arrotondamento è più precisa ma necessita, per essere eseguita di registri più lunghi (per esaminare la $t+1$-esima cifra) e quindi di più tempo

$$\begin{array}{ll} x+y & \text{si scrive } x \oplus y = fl(fl(x)+fl(y)) \\ x-y & \text{si scrive } x \ominus y = fl(fl(x)-fl(y)) \\ x*y & \text{si scrive } x \otimes y = fl(fl(x)*fl(y)) \\ x/y & \text{si scrive } x \oslash y = fl(fl(x)/fl(y)) \end{array}$$

Operazioni macchina

- Somma algebrica di 2 numeri reali:
  - Si trasforma il numero con caratteristica minore in modo che i 2 numeri abbiano la stessa caratteristica (uno dei due perde la forma v.m.n.)
  - Si sommano le mantisse (lasciando invariate le caratteristiche)
  - Si ricava il floating del risultato (si rinormalizza il numero troncando o arrotondando se necessario)
- Prodotto/divisione di 2 numeri reali
  - Si esegue il prodotto/divisione delle mantisse e si sommano/sottraggono le caratteristiche
  - Si ricava il floating del risultato (si rinormalizza il numero troncando o arrotondando se necessario)

Operazioni macchina

- ESEMPIO:

$F(10, 4, -4, 4)$,
$x = 10^{-1}$, $y = -0.00054$
$fl(x) = 0.1000\ 10^{0}$,
$fl(y) = -0.5400\ 10^{-3}$

a) $x \oplus y = fl(0.1000\ 10^{0} - 0.0005\ 10^{0}) = 0.0995\ 10^{0} = 0.9950\ 10^{-1}$

b) $x \otimes y = fl(0.1000\ 10^{0} * (-0.0005\ 10^{0})) = -0.5000\ 10^{-4}$

Osservazioni sulla precisione finita

- Per le operazioni di macchina non valgono più tutte le proprietà delle quattro operazioni:
  - Vale la proprietà commutativa
  - Non vale la proprietà associativa
  - Non vale la proprietà distributiva
- Formule o algoritmi matematicamente equivalenti (che portano allo stesso risultato se applicati in precisione infinita) possono produrre risultati diversi in aritmetica finita.
- Lo studio degli errori di arrotondamento e della loro propagazione attraverso gli algoritmi è di fondamentale importanza per poter interpretare e valutare i risultati di un qualunque algoritmo che operi con numeri reali

Esempio 1

- $\beta = 10,\ t = 4$, per troncamento, $\varepsilon = \beta^{1-t} = 10^{-3}$

  $a = 2000 \quad b = 2.5 \quad c = 7.8$

$$
\begin{aligned}
(a \oplus b) \oplus c &= (0.2000\ 10^4 \oplus 0.2500\ 10^1) \oplus 0.7800\ 10^1 \\
&= fl(0.2000\ 10^4 + 0.0002500\ 10^4) \oplus 0.7800\ 10^1 \\
&= 0.2002\ 10^4 \oplus 0.7800\ 10^1 \\
&= fl(0.2002\ 10^4 + 0.0007800\ 10^4) = 0.2009\ 10^4 \\
a \oplus (b \oplus c) &= 0.2000\ 10^4 \oplus (0.2500\ 10^1 \oplus 0.7800\ 10^1) \\
&= 0.2000\ 10^4 \oplus 0.1030\ 10^2 = \\
&= 0.2000\ 10^4 \oplus 0.0010(3)\ 10^4 = 0.2010\ 10^4
\end{aligned}
$$

$$(a + b) + c = a + (b + c) = 2010.3$$

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

- L'errore relativo nel primo caso: $1.3/2010.3 \simeq 10^{-3}$, mentre nel secondo $0.3/2010.3 \simeq 10^{-4}$, entrambi accettabili rispetto alla precisione usata

Esempio 2: punto medio

- $\beta = 10,\ t = 3$ per troncamento, $\varepsilon = \beta^{1-t} = 10^{-2}$
  $a = 0.651 \quad b = 0.653$

$$
\begin{aligned}
c &= \frac{a+b}{2} = fl((0.651\ 10^0 \oplus 0.653\ 10^0)/2) \\
&= fl(0.130\ 10^1/2) = 0.650\ 10^0 \notin [a,b] \\
c &= a + \frac{b-a}{2} = 0.651\ 10^0 \oplus fl((0.653\ 10^0 \oplus (-0.651\ 10^0))/2) \\
&= 0.651\ 10^0 \oplus 0.100\ 10^{-2} = 0.652\ 10^0
\end{aligned}
$$

- Formule matematicamente equivalenti non lo sono più quando si opera in precisione finita $(a+b)/2 = 0.652$

Esempio 3: equazione di secondo grado

- Le radici di $ax^2 + bx + c$ sono date da $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Se $\beta = 10,\ t = 5,\ \varepsilon = \beta^{1-t} = 10^{-4}$
  $a = 1,\ b = -3.6778,\ c = 0.0020798$ si ottiene come risultato
  $x_2 = 0.00055000$ invece di $x_2 = 0.000565\ldots$
- Il modo corretto è invece:

$$\begin{aligned} x_1 &= -\frac{b + \text{sign}(b)\sqrt{b^2 - 4ac}}{2a} \\ x_1 x_2 &= \frac{c}{a} \end{aligned}$$

che produce $x_2 = 5.6558 \cdot 10^{-4}$

Importante

- Il concetto di uguaglianza va modificato quando si lavora sui numeri reali in precisione finita
- Risultati diversi possono essere considerati uguali nei limiti della precisione usata
- Due numeri sono uguali quando hanno uguale caratteristica ed uguale mantissa MA due numeri sono da considerarsi uguali quando la loro differenza è piccola rispetto alla precisione di macchina $|a - b| < \varepsilon$
- ε è la tolleranza ed il suo valore dipende dalla precisione di macchina
- Non ha senso chiedere errori relativi minori della precisione di macchina

Algoritmo per determinare ε_m

- La precisione di macchina può essere determinata operativamente come la più piccola quantità che sommata ad 1 dà un risultato diverso da 1:

Dim: supponiamo di operare per taglio: $\varepsilon_m = \beta^{1-t}$

$$\begin{aligned} 1 \oplus \varepsilon_m &= 0.10\ldots0\ \beta^1 + 0.10\ldots0\ \beta^{2-t} \\ &= 0.10\ldots0\ \beta^1 + 0.\underbrace{0\ldots01}0\ldots0\ \beta^1 \\ &= 0.10\ldots01\ \beta^1 > 1 \end{aligned}$$

Consideriamo un numero macchina minore di ε_m:
$x = 0.(\beta-1)(\beta-1)\cdots(\beta-1)\ \beta^{1-t}$:

$$\begin{aligned} 1 \oplus x &= 0.10\ldots0\ \beta^1 + 0.\underbrace{0\ldots00}(\beta-1)(\beta-1)\cdots(\beta-1)\ \beta^1 \\ &= 0.10\ldots0(\beta-1)(\beta-1)\cdots(\beta-1)\ \beta^1 \\ &= 0.10\ldots0\ \beta^1 = 1 \end{aligned}$$

Algoritmo per determinare ε_m

- La precisione di macchina può essere determinata operativamente come la più piccola quantità che sommata ad 1 dà un risultato diverso da 1:
- supponiamo di operare per taglio:

1. Poni $\varepsilon_m := 1$
2. Poni $\varepsilon_m := \beta^{-1} * \varepsilon_m$
3. Poni $a := 1 + \varepsilon_m$
4. Se $a > 1$ vai a 2.
   altrimenti vai a 5.
5. Poni $\varepsilon_m := \beta * \varepsilon_m$
6. Stampa ε_m

STABILITÀ:

Propagazione degli errori sui dati

- Conclusione:
  - si commettono errori nel rappresentare i numeri reali
  - si commettono errori nell'esecuzione delle operazioni aritmetiche
  - Idea tranquillizzante e sbagliata: poiché gli errori sono molto piccoli anche i risultati sono affetti da errori molto piccoli

$\Rightarrow$ Occorre studiare come si propagano gli errori delle operazioni dell'algoritmo.

DEF: *Un algoritmo si dice stabile se rimane limitato l'influsso degli errori*

STABILITÀ:

Propagazione degli errori sui dati

- Ad esempio nella SOMMA si ha una grande amplificazione degli errori sui dati sommando numeri quasi uguali in modulo ma opposti di segno (cancellazione numerica)
- L'analisi della propagazione degli errori in un algoritmo è estremamente complessa. Dato un problema P
  - Analisi in avanti: si valuta la differenza tra la soluzione esatta y e quella calcolata $\tilde{y}$
  - Analisi in indietro: si interpreta la soluzione calcolata $\tilde{y}$ come soluzione esatta di un problema perturbato Q e si valuta P-Q

STABILITÀ:

Esempio

Consideriamo la successione definita da:

$$\begin{aligned} x_{k+2} &= 2.25x_{k+1} - 0.5x_k \\ x_1 &= \frac{1}{3} \\ x_2 &= \frac{1}{12} \end{aligned}$$

Apparentemente si ha un andamento del tipo

$$x_k = \frac{1}{3}4^{1-k}$$

Esercizio: studiare cosa accade realizzando questa successione in Matlab.