ESEMPIO: non associatività

Si assuma di avere a disposizione 4 cifre decimali per la mantissa e 2 cifre decimali per l'esponente. Per semplicità l'esempio è in base 10.

$a = 1258.47$

$b = 0.8941$

$c = 1.32932$

Si vuole eseguire la somma dei tre numeri, cercando l'ordine più conveniente per minimizzare gli effetti di approssimazione.

ESEMPIO: non associatività

- Si scrivono i numeri in notazione normalizzata e si esegue l'approssimazione delle mantisse:

  $a = 0.1258 \times 10^4$

  $b = 0.8941$

  $c = 0.1329 \times 10^1$

- Si capisce che se si sommasse prima a con b, quest'ultimo non darebbe alcun contributo (nel caso di troncamento).
- Conviene sommare prima i numeri più piccoli. Si esegue l'allineamento di b con c:

  $b = 0.0894 \times 10^1$

  $c = 0.1329 \times 10^1$

ESEMPIO: non associatività

- Ora i numeri hanno lo stesso esponente, si esegue $b + c$:

$$b + c = 0.2223 \times 10^1$$

- Questo risultato si somma ad a. Si esegue l'allineamento:

  $a = 0.1258 \times 10^4$

  $(b + c) = 0.0002 \times 10^4$

- Poi la somma $a + (b + c)$:

$$a + (b + c) = 0.1260 \times 10^4$$

"Overflow" e "Underflow"

Rappresentazione *floating point*:
$\pm 0.mmmmmmmmm \times N^{qqqq}$.
Problemi connessi con il numero finito di bit disponibili per l'**esponente**: gli esponenti rappresentabili vanno da q_{min} a q_{max}.

- *Overflow* per la rappresentazione floating-point si è superato q_{max}. In MATLAB $2^{1024} \simeq 1.7977e + 308$ è il primo numero in overflow.
- *Underflow* l'esponente del numero floating-point è inferiore a q_{min}. In MATLAB $2^{-1022} \simeq 2.2251e - 308$

In entrambi i casi il calcolatore invia un messaggio di errore, che il programmatore deve gestire.

Numero più piccolo rappresentabile

Nota sul numero più piccolo rappresentabile:
Non confondere la precisione macchina (o relativa) con il numero più piccolo rappresentabile.
Si preferisce parlare di **numero più piccolo normalizzato**.
Trattamento dell'underflow:

- Underflow brusco: se il risultato di una operazione è minore del numero più piccolo normalizzato, questo viene azzerato.
- Underflow graduale: se il risultato di una operazione è minore del numero più piccolo normalizzato, viene accettato come numero *denormalizzato* diverso da zero.

Estremi e precisione relativa

Formato	minimo	massimo	prec. macchina	bytes
singola	1.2e-38	3.4e38	5.96e-8	4
doppia	2.2e-308	1.8e308	1.11e-16	8
estesa	3.4e-4932	1.2e4932	5.42e-20	10
quadrupla	3.4e-4932	1.2e4932	9.63e-35	16

Nota: la precisione quadrupla è in emulazione.

Precisione doppia

```
s  qqq          mmm
-|-----------|--------------------..---------|
1   11 bit       52 bit
```

Il numero rappresentato è dato da: $(-1)^s \times 2^{qqq} \times mmm$

Guasti dovuti ad un cattivo calcolo numerico

`http://www.ima.umn.edu/~arnold/disasters`

- Fallimento del Missile Patriot in Dharan, Arabia Saudita, il 25-2-91: 28 morti. Attribuito ad un scarso trattamento degli errori di arrotondamento.
- Esplosione appena dopo la partenza del razzo Ariane 5, lanciato dall' Agenzia Spaziale Europea nella Guiana Francese, il 4-6-96. Conseguenza di un semplice overflow.
- Affondamento della piattaforma Sleipner A per la produzione di petrolio e gas nel Gandsfjorden vicino Stavanger, Norvegia, il 23-8-91. Causata da un'analisi non accurata mediante *elementi finiti*.

Problema numerico ed algoritmo

Il problema numerico è una relazione funzionale (implicita od esplicita) tra due vettori di numeri x ed y che rappresentano rispettivamente i dati ed i risultati del problema.

Relazione esplicita: $y = f(x)$

Relazione implicita: $F(x, y) = 0$

Un algoritmo (del problema numerico) è una descrizione completa e ben definita delle operazioni che consentono di trasformare x in y secondo la relazione funzionale. Ad ogni problema numerico possono corrispondere più algoritmi.

Condizionamento

"**condizionamento**" di un problema: sensibilità ai dati del problema stesso; in un problema "ben-condizionato" le **variazioni relative** sui risultati sono dello stesso ordine di grandezza di quelli dei dati.
Dato il problema $y = f(x)$, con $x \in R^n$ e $y \in R^m$, si cercano maggiorazioni del tipo

$$\frac{\|f(x) - f(\bar{x})\|}{\|f(x)\|} \leq k \frac{\|x - \bar{x}\|}{\|x\|}$$

dove k è il numero di condizionamento. Buon condizionamento se k "non troppo grande". Il simbolo $\|\cdot\|$ indica una misura sui vettori. In seguito vedremo alcune definizioni di misure.

Stima del condizionamento

Un modo di stimare il condizionamento è di utilizzare le derivate parziali sugli errori relativi (in modo analogo alla propagazione degli errori nelle misure di grandezze fisiche):

$$\left|\frac{\Delta y}{y}\right| \leq \left(\left|\frac{\partial f}{\partial x_1}\right|\left|\frac{x_1}{y}\right|\right)\left|\frac{\Delta x_1}{x_1}\right| + \left(\left|\frac{\partial f}{\partial x_2}\right|\left|\frac{x_2}{y}\right|\right)\left|\frac{\Delta x_2}{x_2}\right| + \dots$$

Stabilità

Un algoritmo è stabile se la "sensibilità" dell'algoritmo ai dati ed alle operazioni è dello stesso ordine di grandezza di quello del problema numerico.
In altre parole, errori di arrotondamento e di calcolo numerico non producono amplificazione di errore sui risultati.
La stabilità di un algoritmo deve essere comprovata rispetto a problemi ben-condizionati, altrimenti non ha senso.

Analisi in avanti e all'indietro

Questo tipo di analisi "in avanti" (*forward*) può portare ad errori nel valutare la stabilità di un algoritmo (se per certi valori dei dati il problema è mal-condizionato).

Esiste l'analisi "all'indietro" (*backward*), che è più corretta: il risultato di un algoritmo viene "pensato" come risultato di un problema esatto applicato a dati perturbati. Se la perturbazione è dell'ordine della precisione macchina allora l'algoritmo è stabile.

Complessità di tempo degli algoritmi

Valutazione del numero di operazioni o del tempo di esecuzione di un algoritmo, per produrre il risultato.

- Addizioni, sottrazioni e moltiplicazioni, impiegano lo stesso tempo di CPU.
- le divisioni impiegano un tempo maggiore e non costante.

Contare insieme le addizioni, sottrazioni e moltiplicazioni. A parte le divisioni.

Algoritmo di Horner

Modo efficiente per valutare i polinomi.

- Dato un polinomio di grado n:

$$p_n(x) \equiv a_0 + a_1 x + a_2 x^2 \cdots + a_n x^n.$$

  Per valutare $p_n(\bar{x})$, dato $\bar{x}$, sono necessarie n addizioni e $2n-1$ moltiplicazioni: complessità $3n-1$.

- Moltiplicazioni annidate:

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

  sono necessarie n addizioni ed n moltiplicazioni: la complessità è $2n$.

Algoritmo di Horner 2

Procedimento iterativo:

$$b_n = a_n \,,$$
$$b_k = a_k + b_{k+1}\,\bar{x}\,, \quad k = n-1, n-2, \ldots 0$$

l'ultimo termine b_0 corrisponde al valore $p_n(\bar{x})$ cercato.
Esercizio: dati i coefficienti $a_0 \ldots a_n$ di un polinomio di grado n, implementare l'algoritmo di Horner per la valutazione dello stesso.