Progetto
Lauree Scientifiche
Università degli Studi di Trieste

http://www.laureescientifiche.units.it/

CIRD
Centro Interdipartimentale
per la Ricerca Didattica

Quello che tutti dovrebbero sapere sull' aritmetica dei computer

… ma anche delle calcolatrici tascabili o dei calcoli manuali

Giorgio Pastore, Maria Peressi
Dip. Fisica - Università di Trieste

(aggiornamento e formazione insegnanti;
corso CIRD - CP e M IDIFO3: Lab_A **I15**)

Schema

- Introduzione
- Implementazione dell' aritmetica su dispositivi elettronici e limiti conseguenti.
- Particolarità dell' aritmetica intera sui computer
- Particolarità dell' aritmetica non intera
    - errore di arrotondamento
    - effetto della base
    - comportamento dell' errore di arrotondamento nelle diverse operazioni
    - controllare l' errore: precisione macchina
    - gli standard esistenti
- Gestire l' aritmetica su una data piattaforma HW/SW

Introduzione

Anche se siamo abituati a pensare che calcolatrici tascabili e computer "non sbagliano mai", potremmo scoprire che in alcuni casi questi strumenti di calcolo automatico possono dare risultati "strani", imbarazzanti o apparentemente incomprensibili.

Introduzione

Qualche esempio:

$$(1.0 / n) * n$$

Ci aspetteremmo che sia uguale a 1.

A seconda **dello strumento di calcolo** che utilizziamo e **del valore di n** troveremo che il confronto con 1.0 può fallire.

Perché ?

Introduzione

I fattoriali. Output di un programma che calcola

n! = 1*2*3*….*n

```
n =  1  n! =  1
n =  2  n! =  2
n =  3  n! =  6
n =  4  n! =  24
n =  5  n! =  120
n =  6  n! =  720
n =  7  n! =  5040
n =  8  n! =  40320
n =  9  n! =  362880
n =  10  n! =  3628800
n =  11  n! =  39916800
n =  12  n! =  479001600
n =  13  n! =  1932053504
n =  14  n! =  1278945280
n =  15  n! =  2004310016
n =  16  n! =  2004189184
n =  17  n! =  -288522240
n =  18  n! =  -898433024
n =  19  n! =  109641728
n =  20  n! =  -2102132736
```

17! è chiaramente assurdo. Ma già 13! è decisamente sbagliato. Perché ?

Introduzione

Le regole dell' aritmetica impongono che

$$1 + x \neq 1 \qquad \text{per ogni } x \neq 0.$$

Procedendo per tentativi, possiamo scoprire che su un computer o su una calcolatrice esistono moltissimi numeri diversi da zero che sommati a 1 danno come risultato 1!

Le conseguenze di questo risultato per l' aritmetica dei computer appaiono a prima vista devastanti:

per esempio, non vale più la proprietà associativa della somma!

La situazione non è però disperata: è sufficiente avere presenti le limitazioni, per tenerne conto e poter quindi realizzare un' approssimazione controllata e migliorabile dell' aritmetica usuale.

Implementazione elettronica del calcolo numerico e sue conseguenze

Per ragioni di velocità, affidabilità ed economicità, il calcolo numerico viene attualmente implementato a partire da una rappresentazione dei dati come livelli di tensione all' interno di dispositivi elettronici.

Questo normalmente comporta vincoli sulla base della rappresentazione posizionale dei numeri e sul numero di cifre significative che possono essere gestite direttamente dall' hardware.

Rappresentazione dei dati numerici sui computer

Sui computer attuali i dati sono rappresentati in codifica binaria.
I singoli dati binari elementari (bit) sono organizzati in gruppi (byte, parole, ...), un' unità comune è il byte, di norma pari a 8 bit, anche se in realtà i trasferimenti di dati all' interno del computer avvengono a gruppi di più byte alla volta (p.es. 4 in architetture a 32 bit).

Nulla vieta di rappresentare dati numerici mediante un numero arbitrario di byte. Tuttavia, solo per alcuni multipli del byte, l' hardware sottostante implementa direttamente le operazioni aritmetiche di base. Per lunghezze dei dati differenti è sempre possibile implementare soluzioni software, ma a prezzo di un notevole degrado delle prestazioni.

Per comprendere il comportamento dell' aritmetica dei computer è necessario entrare nel dettaglio delle diverse codifiche.

Interi - 1

I linguaggi di programmazione mettono a disposizione uno o più tipi dati per rappresentare gli interi. Le differenze riguardano gli intervalli possibili ed eventualmente la presenza o meno di interi con segno.

Se un particolare tipo intero viene rappresentato mediante n bit solo 2^n valori diversi saranno possibili.

Questo ha come prima conseguenza che usando n bit per rappresentare interi con segno, ci sarà un massimo ed un minimo intero (per convenzione $2^{(n-1)}-1$ e $-2^{(n-1)}$).

Inoltre, per vari motivi di convenienza, sui computer moderni, il superamento, nel corso delle operazioni aritmetiche, dei limiti non comporta un arresto dell' esecuzione. Invece il calcolo procede oltre assumento un' aritmetica modulo 2^n.

Interi - 2

Pertanto, il successivo di $2^{(n-1)} - 1$ sarà il più piccolo (negativo) intero: $-2^{(n-1)}$. Più che di retta numerica è opportuno parlare di circonferenza.

Da tener presente che

$2^{31}-1 = 2\ 147\ 483\ 647 \simeq 10^9$

$2^{63}-1 = 9\ 223\ 372\ 036\ 854\ 775\ 807 \simeq 10^{19}$

Una caratteristica di molti linguaggi, a cui occorre fare attenzione, è la possibile stretta osservanza delle regole della divisione, per cui il risutato della divisione è sempre un intero (ed eventualmente c' è un resto (intero). In tal caso un' espressione come 1/2 NON va intesa come frazione ma come divisione intera tra 1 e 2 (quante volte il 2 sta nell' 1 ?) il qui quoziente è 0.

Numeri con virgola - 1

Per i numeri con virgola, si utilizza la cosiddetta rappresentazione normalizzata in virgola mobile:
Ogni numero con virgola viene moltiplicato e diviso per un' opportuna potenza della base in modo da metterlo nella forma di un numero con virgola (la mantissa), con un' unica cifra non nulla a sinistra della virgola, che moltiplica una potenza della base.

Es. (base 10):

$123.45 = 1.2345\ 10^{2}$
$0.006543 = 6.543\ 10^{-3}$

La stessa cosa può esser fatta in qualsiasi base. In base 2 la convenzione di normalizzazione della mantissa implica che la prima cifra (quella a sinistra della virgola) sarà sempre 1.

Numeri con virgola - 2

Sui computer i numeri in virgola mobile vengono rappresentati per lo più secondo una convenzione standardizzata
(standard IEEE 754/IEC 60559).

Lo standard prevede che ci siano almeno due rappresentazioni, corrispondenti a diversa precisione (cifre della mantissa) e diversa estensione (intervallo di valori dell' esponente).

I due livelli di precisione di base corrispondono a: una più bassa implementata mdiante 32 bit (precisione singola), ed una più alta (precisione doppia) a 64 bit, di cui:

- un bit per il segno della mantissa;
- 8 bit (risp. 11) per l' esponente intero (segno incluso)
- 23 bit (risp. 52) per la mantissa.

Sono possibili ulteriori rappresentazioni, p.es. sui processori Intel è disponibile una precisione estesa a 80 bit.

Numeri con virgola - 3

A ciascun livello di precisione corrisponde una coppia di intervalli simmetrici rispetto allo zero di numeri "normalizzati" la cui precisione relativa è costante (circa 10^{-7} per la singola precisione e circa 10^{-16} per la doppia precisione) ed una coppia di intervalli di numeri "denormalizzati", compresi in valore assoluto tra 0 ed il minimo numero normalizzato positivo, la cui precisione relativa diminuisce col valore assoluto.

Numeri in valore assoluto compresi tra 0 ed il minimo dei dei denormalizzati positivi, sono considerati equivalenti a zero (errore di underflow).

Operazioni che cercano di generare numeri maggiori in valore assoluto del massimo numero normalizzato danno luogo ai due valori speciali +Inf e -Inf (overflow).

Operazioni non definite (0/0, acos(-1.2), etc.) danno luogo ai valori speciali NaN (Not a Number).

Particolarità dell' aritmetica - 1

La rappresentazione finita e binaria dei numeri con virgola introduce per quasi tutti i numeri reali un errore di rappresentazione che dipende da

- la precisione utiizzata
- la particolare modalità di arrotondamento usato dal sistema
- il numero

E' pertanto utile definire una grandezza, la precisione macchina, o epsilon macchina (normalmente indicata con ϵ_M) che rappresenta il limite superiore all' errore di arrotondamento possibile con quella precisione. L' ordine di grandezza di ϵ_M per la singola e doppia precisione coincide con la precisione relativa.

Particolarità dell' aritmetica - 2

Un valore numerico sul computer (X_C) può essere pensato come

$$X_C = X(1+ \epsilon_X)$$

Dove X è il valore vero e ϵ_X l' errore relativo su X.

Varrà

$$|\epsilon_X| < \epsilon_M$$

Inoltre occorre tener conto di come l' errore relativo sui dati viene propagato nelle operazioni elementari

Particolarità dell' aritmetica - 3

Propagazione degli errori:

1. Prodotti e divisioni:

$$|\epsilon_{XY}| = |\epsilon_X + \epsilon_Y| \leq 2\,\epsilon_M$$
$$|\epsilon_{X/Y}| = |\epsilon_X - \epsilon_Y| \leq 2\,\epsilon_M$$

2. Somme o sottrazioni:

$$|\epsilon_{X+Y}| = |X\epsilon_X + Y\epsilon_Y|/|X+Y|$$

Se $X>>Y$ o $X<<Y$ o X e Y hanno segno concorde

$$|\epsilon_{X+Y}| \leq 2\epsilon_M$$

Ma se $X \simeq Y$ e hanno segno discorde

$$|\epsilon_{X+Y}| \leq 2|X|\epsilon_M/(|X\}-|Y|)$$

e il denominatore può essere molto piccolo, facendo crecere senza limiti l' errore relativo (<u>cancellazione sottrattiva</u>).

Particolarità dell' aritmetica - 4

Propagazione degli errori:

1. Nel caso di un gran numero (N) di operazioni, può risultare utile un' analisi statistica, nell' ipotesi di errori scorrelati, dell' errore relativo proveniente dalla somma di molti errori relativi:

$$|\epsilon| \leq \sqrt{N}\, \epsilon_M$$

In generale, nell' analisi di qualsiasi algoritmo numerico, l' analisi delle approssimazioni intrinseche all' algoritmo (approssimazioni analitiche) va completata con quella relativa agli errori di arrotondamento (round-off).

Esempio: derivata numerica

Diverse forme analiticamente equivalenti per la derivata.
Es:

$$Df(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \qquad [1]$$

$$Df(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} \qquad [2]$$

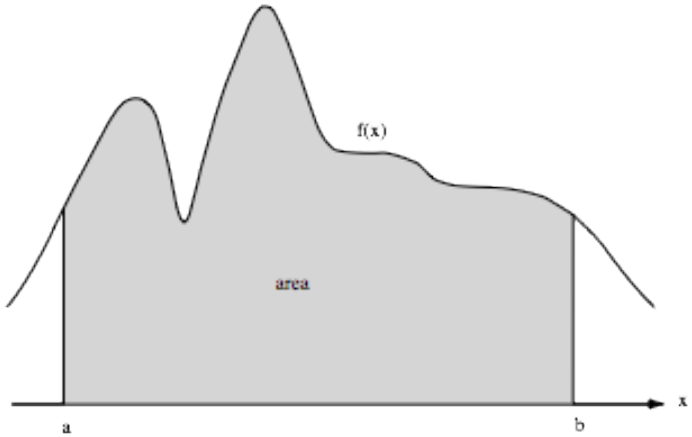
Numericamente le formule [1] e [2] vengono implementate sostituendo al processo di limite (non corrispondente ad un numero finito di passi) un valore di *h* che ottimizzi l' errore complessivo (analitico + arrotondamento)

L' errore relativo algoritmico di [1] è proporzionale a *h*.
L' errore relativo algoritmico di [2] è proporzionale a h^2.
L' errore relativo di arrotondamento è proporzionale a ϵ_M / h
Pertanto, il valore ottimale di h sarà proporzionale a $\epsilon_M^{1/2}$ con [1] e a $\epsilon_M^{1/3}$ con [2] con errori rispettivamente propozionali a $\epsilon_M^{1/2}$ e $\epsilon_M^{2/3}$

Integrazione numerica

Partiamo dall'interpretazione geometrica di un integrale definito:

$$F = \int_a^b f(x)dx$$



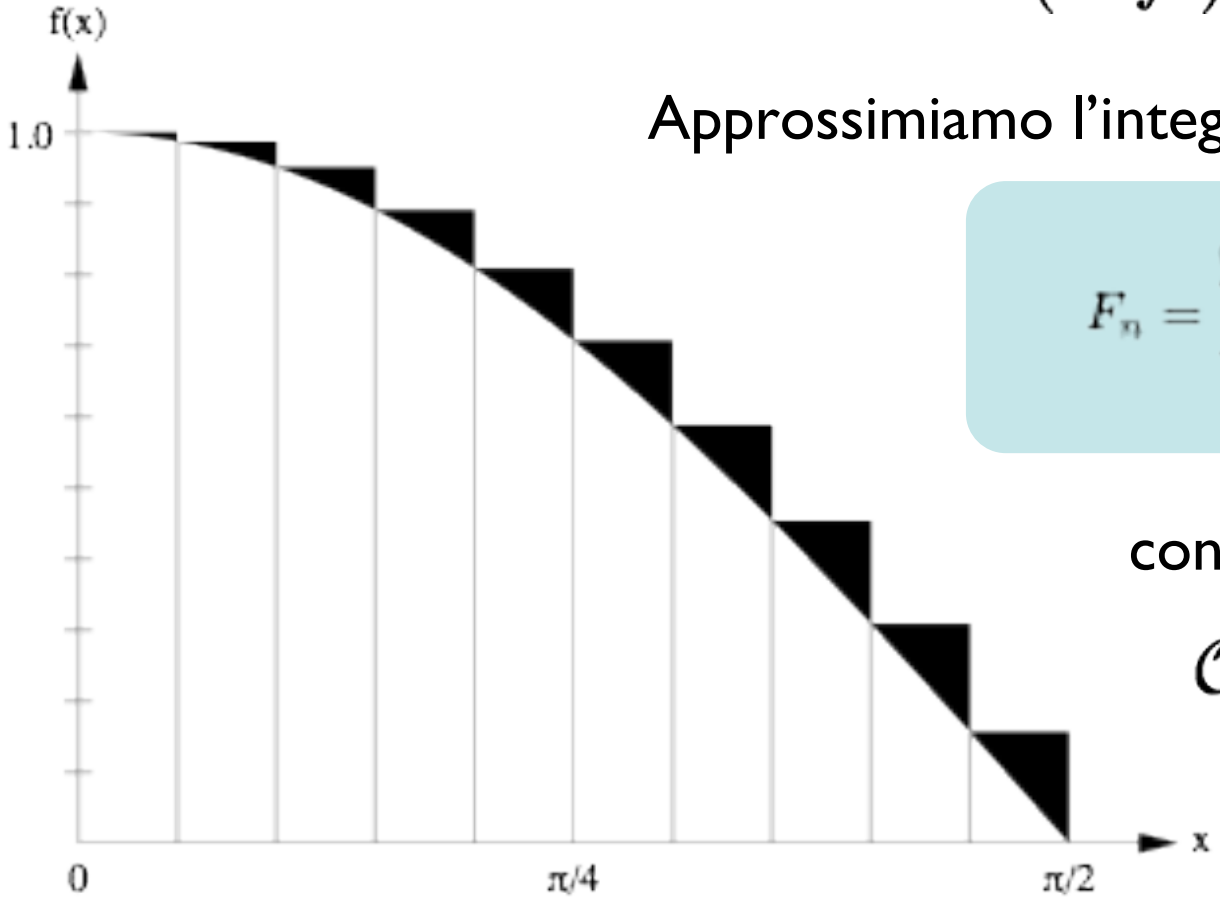
Dividiamo l'intervallo di integrazione in "piccoli" intervalli:

$$\Delta x = \frac{b-a}{n},$$

$$x_n = x_0 + n\,\Delta x.$$

Approssimiamo l'integrale in un intervallo: $$\int_{x_i}^{x_{i+1}} f(x)dx = hf_i$$

L' errore e': $\mathcal{O}(h^2 f'), \propto 1/n^2$



Approssimiamo l'integrale su *n* intervalli:

$$F_n = \sum_{i=0}^{n-1} f(x_i)\Delta x.$$

con un errore totale:

$$\mathcal{O}(hf'), \propto 1/n$$

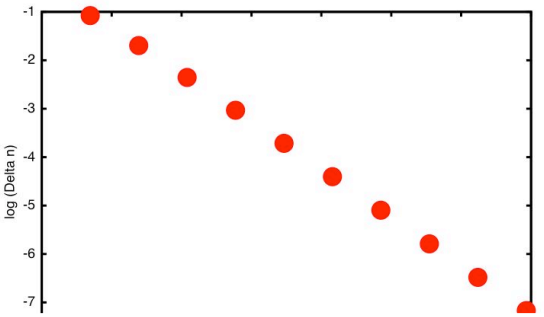
: The rectangular approximation for $f(x) =$ per x for $0 \leq x \leq \pi/2$.

Integrazione numerica regola dei rettangoli

$$I = \int_0^{\pi/2} \cos(x)dx = 1$$

$$F_n = \frac{\pi}{2n} \sum_0^{n-1} \cos x_i; \quad x_i = i\frac{\pi}{2n}$$

n	F_n	Δ_n
2	1.34076	0.34076
4	1.18347	0.18347
8	1.09496	0.09496
16	1.04828	0.04828
32	1.02434	0.02434
64	1.01222	0.01222
128	1.00612	0.00612
256	1.00306	0.00306



$$\Delta_n = F_n - I$$

n^{-1}, that is, if n is increased by a factor of 2, Δ_n decreases by a factor 2.

L'approssimazione con rettangoli dell'integrale di f(x)=cos(x) da x=0 a x=π/2 in funzione del numero *n* di intervalli. L'errore Δ_n e' la differenza tra l'approssimazione con i rettangoli e il risultato esatto, Notare come l'errore Δ_n decresce approssimativamente come *1/n,* cioe', se *n* aumenta di un fattore 2, Δ_n decresce di un fattore 2.

Integrazione numerica
regola dei trapezi

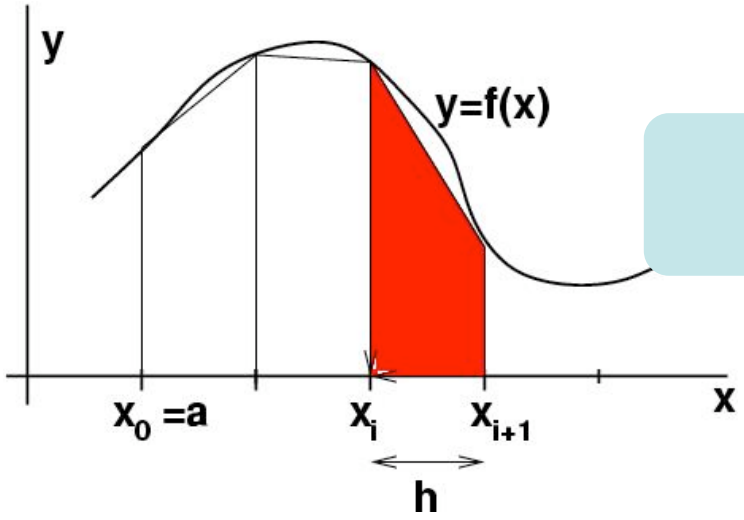
In un intervallo:

$$\int_{x_i}^{x_{i+1}} f(x)dx = h\left[\frac{1}{2}f_i + \frac{1}{2}f_{i+1}\right]$$

con errore:

$$\mathcal{O}(h^3 f'''), \propto 1/n^3$$

Applicando iterativamente su intervalli consecutivi:

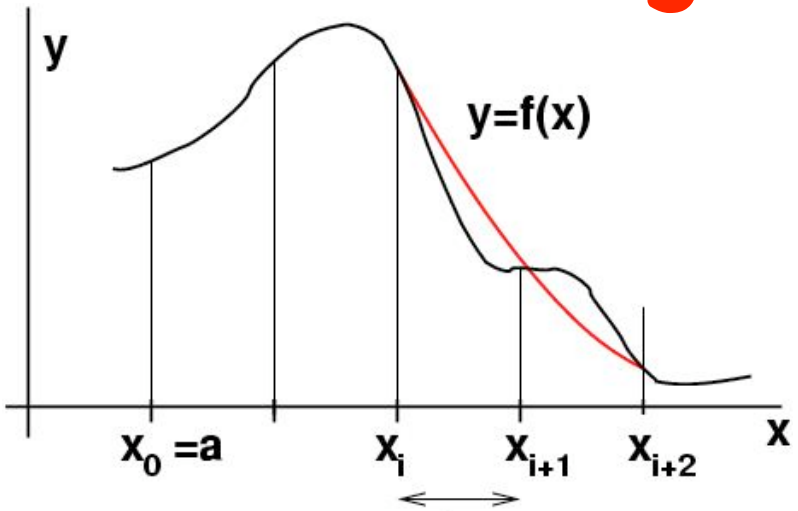


$$F_n = \left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right]\Delta x.$$

con un errore totale:

$$\mathcal{O}(h^2 f'''), \propto 1/n^2$$

Integrazione numerica
regola di Simpson



Si basa su un procedimento di interpolazione parabolica tra terne di punti adiacenti.

In un intervallo:

$$\int_{x_i}^{x_{i+2}} f(x)dx = h\left[\frac{1}{3}f_i + \frac{4}{3}f_{i+1} + \frac{1}{3}f_{i+2}\right] + \mathcal{O}(h^5 f^{IV}) \; (error \;\; \propto 1/n^5)$$

Iterativamente applicato su tutto l'intervallo di integrazione (numero di punti dispari!):

$$\int_{x_0}^{x_n} f(x)dx = h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \ldots + \frac{2}{3}f_{n-2} + \frac{4}{3}f_{n-1} + \frac{1}{3}f_n\right] + \mathcal{O}(h^4 f^{IV}) \; (error \;\; \propto 1/n^4)$$

Gestire una particolare piattaforma HW/SW

Tutti i linguaggi di programmazione mettono a disposizione diversi tipi dati per interi e per numeri con virgola. Alcuni di questi sono implementati in modo da sfruttare direttamente l' hardware e le possibilità di aritmetica offerte da questo. In alcuni casi è anche possibile controllare il tipo di arrotondamento utilizzato.

Tipi dati che non rientrino nelle possibilità dirette dell' HW possono essere implementati via software, sia in modo diretto, attraverso tipi dati implementati nei linguaggi, sia attraverso librerie di programmi esterne.

Per saperne di più

Note su rappresentazione posizionale, cambi di base e rappresentazione su computer:
http://www-dft.ts.infn.it/~pastore/DIDA/MNF/posizionale.html
http://www-dft.ts.infn.it/~pastore/DIDA/info/appunti/aritmetica.html
What every computer scientist should knw about floating-point arithmetic:
http://portal.acm.org/citation.cfm?id=103163

Metodi numerici:
G. Naldi, L. Pareschi, G. Russo "Introduzione al calcolo scientifico", McGraw-Hill, 2001 (Esempi con Matlab, utilizzabile anche con Octave)

Numerical recipes:
http://www.nr.com/oldverswitcher.html