Numeri floating point

Corso di Calcolo Numerico, a.a. 2008/2009

Francesca Mazzia

Dipartimento di Matematica
Università di Bari

Rappresentazione dei numeri.

- L'utilizzo in modo corretto del calcolatore per fare calcoli di tipo scientifico, richiede la conoscenza di come sono rappresentati i numeri e degli errori che derivano da questa rappresentazione.

Rappresentazione dei numeri.

- L'utilizzo in modo corretto del calcolatore per fare calcoli di tipo scientifico, richiede la conoscenza di come sono rappresentati i numeri e degli errori che derivano da questa rappresentazione.
- L'uso dei numeri reali richiede una attenzione particolare, essendo questi infiniti, mentre il calcolatore ci da la possibilità di rappresentarne solo un numero finito.

Rappresentazione dei numeri.

- L'utilizzo in modo corretto del calcolatore per fare calcoli di tipo scientifico, richiede la conoscenza di come sono rappresentati i numeri e degli errori che derivano da questa rappresentazione.
- L'uso dei numeri reali richiede una attenzione particolare, essendo questi infiniti, mentre il calcolatore ci da la possibilità di rappresentarne solo un numero finito.
- La nostra notazione per rappresentare i numeri è una notazione posizionale a base 10. Ciò significa che se scriviamo 123 intendiamo esprimere il numero:

$$1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0.$$

Rappresentazione dei numeri.

Notazione scientifica:

$$\pm\gamma_0.\gamma_1\gamma_2\ldots\gamma_t\cdot\cdot\cdot\cdot N^q, \quad 0 \leq \gamma_i \leq N-1$$

mantissa: $m = \gamma_0.\gamma_1\gamma_2\ldots\gamma_t\ldots$
esponente: q
base : N
normalizzata se $\gamma_0 \neq 0$
I numeri di macchina o floating point sono del tipo:

$$\pm\gamma_0.\gamma_1\gamma_2\ldots\gamma_t \cdot N^q$$

con $\gamma_0 \neq 0$ e $M_1 \leq q \leq M_2$, più lo zero.

Numeri di Macchina

Il più grande numero rapparesentabile in questo sistema se utilizziamo la base 10 è

$$realmax = 9.999..9 \cdot 10^{M_2}$$

mentre il più piccolo numero vicino allo zero è:

$$realmin = 1.00...0 \cdot 10^{M_1}$$

Troncamento e Arrotondamento

Consideriamo il numero reale:

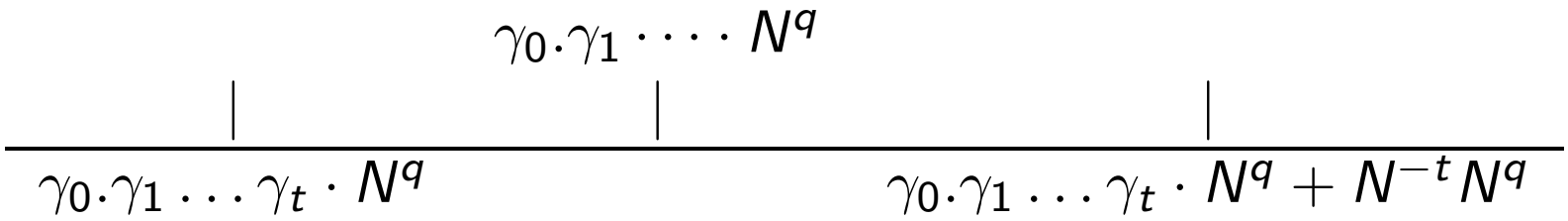
$$x = \pm\gamma_0 . \gamma_1\gamma_2 \ldots \gamma_t \cdots \cdot N^q$$

compreso fra il massimo e il minimo numero rappresentabile (Se si cerca di rappresentare un numero fuori da questo range si ha il problema dell'*underflow* o dell'*overflow*)
Vi sono due modi diversi per approssimare questo numero mediante un numero floating point:

1) Troncamento:
   si trascurano le cifre successive a γ_t ;
2) Arrotondamento:
   se $\gamma_{t+1} < N/2$ trascurare tutte le cifre dopo γ_t ;
   se $\gamma_{t+1} \geq N/2$, aggiungere 1 a γ_t e trascurare le rimanenti cifre.

Esempio

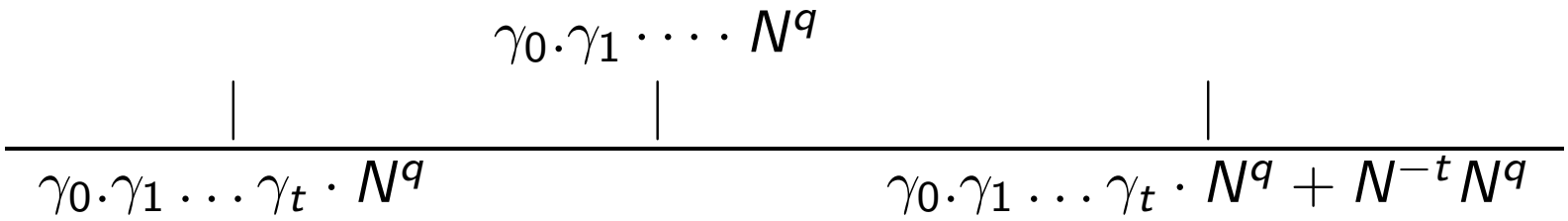
x	troncato t=2	arrotondato t=2
5.672	5.67	5.67
-5.672	-5.67	-5.67
5.677	5.67	5.68
-5.677	-5.67	-5.68



Sia $fl(x)$ l'approssimazione di x.

Troncamento:

$$|fl(x) - x| = 0.\gamma_{t+1} \cdots \cdot N^{q-t} \leq N^{q-t}.$$



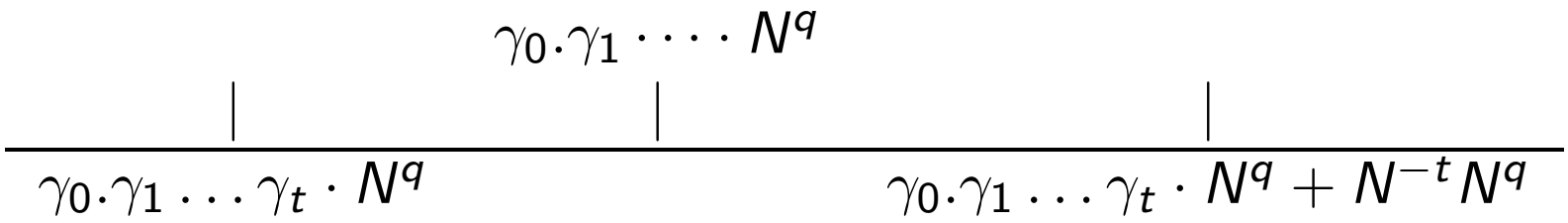
Sia $fl(x)$ l'approssimazione di x.

Troncamento:

$$|fl(x) - x| = 0.\gamma_{t+1} \cdots \cdot N^{q-t} \leq N^{q-t}.$$

Arrotondamento

$$|fl(x) - x| \leq \frac{1}{2} N^{q-t}$$



Sia $fl(x)$ l'approssimazione di x.

Troncamento:

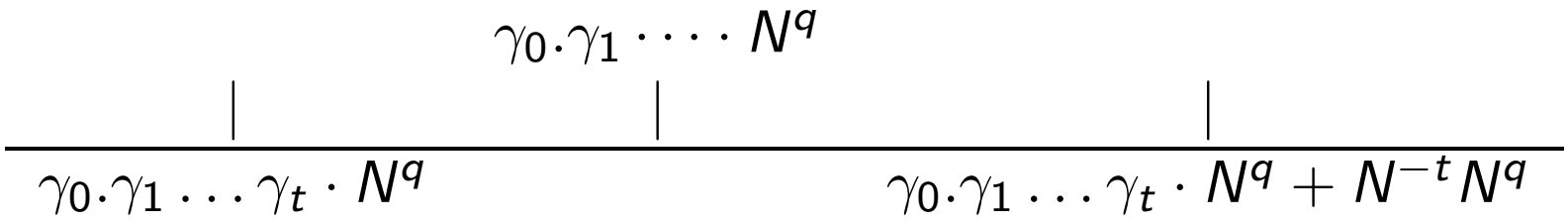
$$|fl(x) - x| = 0.\gamma_{t+1} \cdots \cdot N^{q-t} \leq N^{q-t}.$$

Arrotondamento

$$|fl(x) - x| \leq \frac{1}{2} N^{q-t}$$

Errore Assoluto

$\Delta x = |fl(x) - x|$



Sia $fl(x)$ l'approssimazione di x.

Troncamento:

$$|fl(x) - x| = 0.\gamma_{t+1} \cdots \cdot N^{q-t} \leq N^{q-t}.$$

Arrotondamento

$$|fl(x) - x| \leq \frac{1}{2} N^{q-t}$$

Errore Assoluto

$$\Delta x = |fl(x) - x|$$

Errore relativo

$$\left| \frac{fl(x) - x}{x} \right|$$

Teorema

Se $x \neq 0$ e usiamo t cifre per rappresentare la mantissa, allora:

$$\left| \frac{fl(x) - x}{x} \right| \leq u \tag{1}$$

dove:

$$u = \begin{cases} N^{-t} & \text{troncamento} \\ \frac{1}{2} N^{-t} & \text{arrotondamento} \end{cases}$$

Dimostrazione

Sia $x = \gamma_0.\gamma_1\gamma_2 \ldots \gamma_t \cdots \cdot N^q$, allora $|x| \geq N^q$ e, nel caso dell'arrotondamento, si ha:

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\frac{1}{2} N^{q-t}}{N^q} \leq \frac{1}{2} N^{-t},$$

Il numero u è detto unità di arrotondamento o precisione di macchina.

Secondo lo standard IEEE (lo standard di rappresentazione dei numeri floating point) utilizzando 8 byte per rappresentare i numeri la precisione di macchina risulta essere $u = 2^{-52} \approx 2.22 \cdot 10^{-16}$.
L'errore relativo, piuttosto che l'errore assoluto, è legato alle cifre esatte di un numero.

Poniamo:

$$\epsilon = \frac{fl(x) - x}{x}$$

Sappiamo che $|\epsilon| \leq u$ *e*

$$fl(x) = x(1 + \epsilon),$$

Operazioni con i numeri di macchina

Sia $o \in \{+, -, \times, /\}$ e siano x e y due numeri floating point. È improbabile che l'esatto valore di xoy sia un numero floating point.
Esempio t=3, N=10

$$1.111 \cdot 10^3 \times 1.111 \cdot 10^2 = 1.234321 \cdot 10^3$$

che richiede più di tre cifre decimali.
Il computer dovrebbe eseguire le operazioni aritmetiche di base in modo che il risultato finale sia il risultato esatto arrotondato al più vicino numero floating point. Cioè

Modello delle operazioni aritmetiche:

$$fl(xoy) = (xoy)(1 + \epsilon), \qquad |\epsilon| \leq u$$

o

$$fl(xoy) = (xoy)/(1 - \epsilon), \qquad |\epsilon| \leq u$$

L'aritmetica floating point IEEE richiede questo.

Esercizio

Consideriamo i seguenti numeri di macchina $\pm\gamma_0.\gamma_1\gamma_2 10^{\pm e_0 e_1}$:

- qual è il valore di realmin? qual è il valore di realmax?
- se usiamo l'arrotondamento qual è il valore della precisione di macchina?

Utilizzando i numeri di macchina appena definiti calcolare:

$$\frac{(x+2)^2 - 4}{x}$$

per $x = 6.00 \cdot 10^{-3}$ e $x = 2.00 \cdot 10^{-3}$.
Calcolare l'errore relativo in entrambi i casi e spiegare i risultati ottenuti.

Soluzione

Lavorando in base 10 le cifre della mantissa possono assumere i seguenti valori $0, 1, 2, 3, 4, 5, 6, 7, 8, 9,$ poichè i numeri di macchina sono rappresentati utilizzando la notazione esponenziale normalizzata $\gamma_0 \neq 0$. L'esponente può assumere i valori $\pm 0, 1, \cdots, 99$.

- Realmin è il più piccolo numero positivo rappresentabile. Realmin $= 1.00 \cdot 10^{-99}$
- Realmax è il più grande numero positivo rappresentabile.Realmax $= 9.99 \cdot 10^{99}$.
- Il valore della precisione di macchina è: $10^{-2}/2$.

I numeri coinvolti sono tutti rappresentabili esattamente nella nostra macchina. Supponiamo che ogni operazione aritmetica dia come risultato il valore esatto arrotondato.

Soluzione

Eseguiamo le operazioni con $x = 6.00 \cdot 10^{-3}$:

- $fl(x + 2) = fl(2.006 \cdot 10^0) = 2.01 \cdot 10^0$
- $fl((2.01 \cdot 10^0)^2 = fl(4.0401 \cdot 10^0) = 4.04 \cdot 10^0$
- $fl(4.04 \cdot 10^0 - 4.00 \cdot 10^0) = fl(0.04 \cdot 10^0) = 4.00 \cdot 10^{-2}$
- $fl(4.00 \cdot 10^{-2}/6.00 \cdot 10^{-3}) = fl(6.6\ldots6 \cdot 10^0) = 6.67 \cdot 10^0$

Il valore esatto è:

$$4.005999\ldots 10^0$$

L'errore relativo è:

$$\frac{|6.67 \cdot 10^0 - 4.00599\cdots \cdot 10^0|}{4.00599\cdots \cdot 10^0} \approx 1.47 \cdot 10^{-2}$$

Soluzione

Esaguiamo le operazioni con $x = 2.00 \cdot 10^{-3}$:

- $fl(x + 2) = fl(2.002 \cdot 10^0) = 2.00 \cdot 10^0$
- $fl((2.00 \cdot 10^0)^2 = fl(4.0000 \cdot 10^0) = 4.00 \cdot 10^0$
- $fl(4.00 \cdot 10^0 - 4.00 \cdot 10^0) = fl(0.00 \cdot 10^0) = 0.00$
- $fl(0.00/2.00 \cdot 10^{-3}) = 0.00$

Il valore esatto è:

$$4.001999999999 \cdots \cdot 10^0$$

L'errore relativo è:

$$\frac{|0.00 - 4.001999999999 \cdots \cdot 10^0|}{4.001999999999 \cdots \cdot 10^0} = 1.00 \cdot 10^0$$

Commenti

Se $x \approx y$ allora $|x - y|$ ha un errore relativo molto grande che si chiama

errore di cancellazione

Vedremo più avanti una spiegazione di questo errore.

Commenti

Se $x \approx y$ allora $|x - y|$ ha un errore relativo molto grande che si chiama

errore di cancellazione

Vedremo più avanti una spiegazione di questo errore.
Provare a ripetere l'esercizio usando $x = 6.00 \cdot 10^{-1}$ e $x = 6.00 \cdot 10^{-2}$.

Esempio cancellazione

Eseguiamo $\sqrt{x+1} - \sqrt{x}$, utilizzando il sistema floating point dell'esercizio precedente e $x = 1.00 \cdot 10^3$:
$fl(x+1) = fl(1.00 \cdot 10^3 + 0.001 \cdot 10^3) = fl(1.001 \cdot 10^3) = 1.00 \cdot 10^3$
quindi $\sqrt{1.00^3} - \sqrt{x}$ sarà uguale a zero.

Esempio cancellazione

Eseguiamo $\sqrt{x+1} - \sqrt{x}$, utilizzando il sistema floating point dell'esercizio precedente e $x = 1.00 \cdot 10^3$:
$fl(x+1) = fl(1.00 \cdot 10^3 + 0.001 \cdot 10^3) = fl(1.001 \cdot 10^3) = 1.00 \cdot 10^3$
quindi $\sqrt{1.00^3} - \sqrt{x}$ sarà uguale a zero.
Possiamo utilizzare delle formule equivalenti per calcolare la stessa quantità, il sistema floating point darà risultati diversi.

Esempio cancellazione

Eseguiamo $\sqrt{x+1} - \sqrt{x}$, utilizzando il sistema floating point dell'esercizio precedente e $x = 1.00 \cdot 10^3$:
$fl(x+1) = fl(1.00 \cdot 10^3 + 0.001 \cdot 10^3) = fl(1.001 \cdot 10^3) = 1.00 \cdot 10^3$
quindi $\sqrt{1.00^3} - \sqrt{x}$ sarà uguale a zero.
Possiamo utilizzare delle formule equivalenti per calcolare la stessa quantità, il sistema floating point darà risultati diversi.
Esempio:

$$(\sqrt{x+1} - \sqrt{x}) \frac{(\sqrt{x+1} + \sqrt{x})}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

L'errore relativo è $\approx 4.7 \cdot 10^{-4}$.

Esempio cancellazione

Eseguiamo $\sqrt{x+1} - \sqrt{x}$, utilizzando il sistema floating point dell'esercizio precedente e $x = 1.00 \cdot 10^3$:
$fl(x+1) = fl(1.00 \cdot 10^3 + 0.001 \cdot 10^3) = fl(1.001 \cdot 10^3) = 1.00 \cdot 10^3$
quindi $\sqrt{1.00^3} - \sqrt{x}$ sarà uguale a zero.
Possiamo utilizzare delle formule equivalenti per calcolare la stessa quantità, il sistema floating point darà risultati diversi.
Esempio:

$$(\sqrt{x+1} - \sqrt{x}) \frac{(\sqrt{x+1} + \sqrt{x})}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

L'errore relativo è $\approx 4.7 \cdot 10^{-4}$.
Provate ad eseguire i calcoli per esercizio.

Le operazioni con i numeri di macchina non godono di tutte le proprietà di cui godono le corrispondenti operazioni con i numeri reali. Per esempio non valgono più la proprietà associativa e quella distributiva.
Esempio $N = 10$ e $t = 2$:

$$5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$$

Le operazioni con i numeri di macchina non godono di tutte le proprietà di cui godono le corrispondenti operazioni con i numeri reali. Per esempio non valgono più la proprietà associativa e quella distributiva.
Esempio $N = 10$ e $t = 2$:

$$5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$$

$$5.24 \cdot 10^{-2} + (4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1})$$

- $= 5.24 \cdot 10^{-2} + (0.404 + 1.21)10^{-1}$
- $= 5.24 \cdot 10^{-2} + 1.61 \cdot 10^{-1}$

$$(5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2}) + 1.21 \cdot 10^{-1}$$

- $= 9.28 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$
- $= (0.928 + 1.21)10^{-1}$

Le operazioni con i numeri di macchina non godono di tutte le proprietà di cui godono le corrispondenti operazioni con i numeri reali. Per esempio non valgono più la proprietà associativa e quella distributiva.
Esempio $N = 10$ e $t = 2$:

$$5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$$

$5.24 \cdot 10^{-2} + (4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1})$

- $= 5.24 \cdot 10^{-2} + (0.404 + 1.21)10^{-1}$
- $= 5.24 \cdot 10^{-2} + 1.61 \cdot 10^{-1}$
- $= (0.524 + 1.61)10^{-1} = 2.13 \cdot 10^{-1}$;

$(5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2}) + 1.21 \cdot 10^{-1}$

- $= 9.28 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$
- $= (0.928 + 1.21)10^{-1}$
- $= 2.14 \cdot 10^{-1}$.

Addizione e Sottrazione

Siano x ed y numeri reali tali che $x + y$ è diverso da zero e calcoliamo la somma $fl(fl(x) + fl(y))$.
$fl(fl(x) + fl(y)) = (x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon)$
con $|\epsilon|, |\epsilon_x|, |\epsilon_y| \leq u$
Calcoliamo l'errore relativo trascurando i termini contenenti $\epsilon\epsilon_x$ ed $\epsilon\epsilon_y$,

$$\frac{|(x + y) - (fl(x) + fl(y))(1 + \epsilon)|}{|x + y|}$$
$$= \frac{|x + y - (x + x\epsilon_x + y + y\epsilon_y)(1 + \epsilon)|}{|x + y|} \approx$$
$$\approx |\epsilon| + |\epsilon_x| \frac{|x|}{|x + y|} + |\epsilon_y| \frac{|y|}{|x + y|}.$$

Se $x + y$ non è piccolo l'errore relativo è dello stesso ordine degli errori relativi $|\epsilon|$, $|\epsilon_x|$ ed $|\epsilon_y|$.

Se $x + y$ è molto piccolo l'errore relativo è grande. Esempio
$x = 0.147554326,\ y = 0.147251742,\ t = 5$ ed $N = 10$.
Usando l'arrotondamento
$fl(x) = 1.47554 \cdot 10^{-1}$
$fl(y) = 1.47252 \cdot 10^{-1}$
$fl(fl(x) - fl(y)) = 3.02000 \cdot 10^{-4}$,
$x - y = 3.02584 \cdot 10^{-4}$.
Le ultime tre cifre della mantissa risultano errate. L'errore relativo risultante è:

$$\frac{3.02584 - 3.02000}{3.02584} \approx 10^{-3}$$

che è piuttosto elevato.

Fenomeno della cancellazione numerica.

Prodotto

Allo stesso modo si può analizzare il prodotto:

$$fl(fl(x) * fl(y)) = fl(x(1+e_x) * y(1+e_y)) = x(1+e_x)y(1+e_y)(1+e_*)$$

l'errore relativo è:

$$|x(1+e_x)y(1+e_y)(1+e_*) - xy|/|xy|$$

cioè

$$|(1+e_x)(1+e_y)(1+e_*) - 1|$$

semplificando ed eliminando i termini che contengono i prodotti di pi'**u**errori si ottiene:

$$\text{errore relativo nel prodotto } \approx |e_x| + |e_y| + |e_*|$$

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;
  - rendere l'aritmetica floating point il più accurata possibile;

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;
  - rendere l'aritmetica floating point il più accurata possibile;
  - produrre risultati sensati in situazioni eccezionali;

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;
  - rendere l'aritmetica floating point il più accurata possibile;
  - produrre risultati sensati in situazioni eccezionali;
  - standardizzare le operazioni floating point fra i computer;

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;
  - rendere l'aritmetica floating point il più accurata possibile;
  - produrre risultati sensati in situazioni eccezionali;
  - standardizzare le operazioni floating point fra i computer;
  - Dare al programmatore un controllo sulla manipolazione delle eccezioni;

Standard IEEE

- I due tipi di numeri rappresentati sono interi (fixed point) e reali (floating point).

Standard IEEE

- I due tipi di numeri rappresentati sono interi (fixed point) e reali (floating point).
- Un numero reale ha tipo float in c e real in Fortran, Matlab, Scilab, R.

Standard IEEE

- I due tipi di numeri rappresentati sono interi (fixed point) e reali (floating point).
- Un numero reale ha tipo float in c e real in Fortran, Matlab, Scilab, R.
- Nella maggior parte dei compilatori c e in Matlab, R, Scilab un float ha per default 8 byte invece di 4.

Standard IEEE

- I due tipi di numeri rappresentati sono interi (fixed point) e reali (floating point).
- Un numero reale ha tipo float in c e real in Fortran, Matlab, Scilab, R.
- Nella maggior parte dei compilatori c e in Matlab, R, Scilab un float ha per default 8 byte invece di 4.
- Il formato IEEE sostituisce la base 10 con la base 2 per rappresentare il numero.

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.
- Vi sono quindi $2^{32} \approx 4 \cdot 10^9$ interi a 32 bit che coprono l'intervallo da $-2 \cdot 10^9$ a $2 \cdot 10^9$.

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.
- Vi sono quindi $2^{32} \approx 4 \cdot 10^9$ interi a 32 bit che coprono l'intervallo da $-2 \cdot 10^9$ a $2 \cdot 10^9$.
- Addizione, sottrazione e moltiplicazione sono fatte esattamente se la risposta e compresa nell'intervallo.

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.
- Vi sono quindi $2^{32} \approx 4 \cdot 10^9$ interi a 32 bit che coprono l'intervallo da $-2 \cdot 10^9$ a $2 \cdot 10^9$.
- Addizione, sottrazione e moltiplicazione sono fatte esattamente se la risposta e compresa nell'intervallo.
- La maggior parte dei computer danno risultati imprevedibili se il risultato è fuori dal range (overflow).

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.
- Vi sono quindi $2^{32} \approx 4 \cdot 10^9$ interi a 32 bit che coprono l'intervallo da $-2 \cdot 10^9$ a $2 \cdot 10^9$.
- Addizione, sottrazione e moltiplicazione sono fatte esattamente se la risposta e compresa nell'intervallo.
- La maggior parte dei computer danno risultati imprevedibili se il risultato è fuori dal range (overflow).
- Lo svantaggio dell'aritmetica con gli interi è che non possono essere rappresentate le frazioni e l'intervallo dei numeri è piccolo.

Numeri Reali

Parola a 32 bit interpretata come un numero floating point

- il primo bit è il bit del segno, $s = 0$ equivale a $+$ $s = 1$ equivale a -.

Numeri Reali

Parola a 32 bit interpretata come un numero floating point

- il primo bit è il bit del segno, $s = 0$ equivale a $+$ $s = 1$ equivale a -.
- I successivi 8 bit formano l'esponente.

Numeri Reali

Parola a 32 bit interpretata come un numero floating point

- il primo bit è il bit del segno, $s = 0$ equivale a $+$ $s = 1$ equivale a -.
- I successivi 8 bit formano l'esponente.
- I rimanenti 23 bit determinano la mantissa.

Numeri Reali

Parola a 32 bit interpretata come un numero floating point

- il primo bit è il bit del segno, $s = 0$ equivale a $+$ $s = 1$ equivale a -.
- I successivi 8 bit formano l'esponente.
- I rimanenti 23 bit determinano la mantissa.
- Vi sono 2 possibili segni, 256 esponenti (che variano da 0 a 255) e $2^{23} \approx 8.4$ milioni di possibili mantisse.

Numeri Reali

- Esponenti negativi - convenzione: lo zero è nella posizione 127, dopo ci sono i numeri positivi e prima i numeri negativi.

Numeri Reali

- Esponenti negativi - convenzione: lo zero è nella posizione 127, dopo ci sono i numeri positivi e prima i numeri negativi.
- In memoria viene rappresentato $q^* = q + 127$.

Numeri Reali

- Esponenti negativi - convenzione: lo zero è nella posizione 127, dopo ci sono i numeri positivi e prima i numeri negativi.
- In memoria viene rappresentato $q^* = q + 127$.
- Il primo bit della mantissa, che rappresenta γ_0, è sempre uguale a 1, quindi non vi è necessità di memorizzarlo esplicitamente.

Numeri Reali

- Esponenti negativi - convenzione: lo zero è nella posizione 127, dopo ci sono i numeri positivi e prima i numeri negativi.
- In memoria viene rappresentato $q^* = q + 127$.
- Il primo bit della mantissa, che rappresenta γ_0, è sempre uguale a 1, quindi non vi è necessità di memorizzarlo esplicitamente.
- Nei bit assegnati alla mantissa viene memorizzato m^* e $m = 1.m^*$. Un numero floating point positivo ha quindi il valore $x = \pm(1.m^*)_2 2^{q^*-127}$ e la notazione $(1.m^*)_2$ indica che $1.m^*$ è interpretata in base 2.

Esempio

Il numero $2.752 \cdot 10^3 = 2752$ può essere scritto:

$$\begin{aligned} 2752 &= 2^{11} + 2^9 + 2^7 + 2^6 = \\ &= 2^{11}(1 + 2^{-2} + 2^{-4} + 2^{-5}) = \\ &= 2^{11}(1 + (0.01)_2 + (0.0001)_2 + (0.00001)_2)) = \\ &= 2^{11}(1.01011)_2 \end{aligned}$$

allora la rappresentazione di questo numero avrebbe segno $+$ esponente $q^* = q + 127 = 138 = (10001010)_2$ e $m^* = (010110\ldots0)_2$.

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.
- Se $q^* = 0$ il valore del numero memorizzato è $x = \pm(0.m^*)_2 2^{-126}$.

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.
- Se $q^* = 0$ il valore del numero memorizzato è $x = \pm(0.m^*)_2 2^{-126}$.
- Questo è chiamato underflow graduale (l'underflow è la situazione in cui il risultato di una operazione è diversa da zero ma è più vicina a zero di qualsiasi numero floating point).

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.
- Se $q^* = 0$ il valore del numero memorizzato è $x = \pm(0.m^*)_2 2^{-126}$.
- Questo è chiamato underflow graduale (l'underflow è la situazione in cui il risultato di una operazione è diversa da zero ma è più vicina a zero di qualsiasi numero floating point).
- I numeri corrispondenti vengono chiamati denormalizzati.

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.
- Se $q^* = 0$ il valore del numero memorizzato è $x = \pm(0.m^*)_2 2^{-126}$.
- Questo è chiamato underflow graduale (l'underflow è la situazione in cui il risultato di una operazione è diversa da zero ma è più vicina a zero di qualsiasi numero floating point).
- I numeri corrispondenti vengono chiamati denormalizzati.
- L'underflow graduale ha la conseguenza che due numeri floating point sono uguali se e solo se sottraendone uno dall'altro si ha esattamente zero.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0 \cdots 01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0 \cdots 01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.
- L'altro caso è $q^* = 255$ che ha due sottocasi, Inf se $m^* = 0$ e NaN se $m^* \neq 0$.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0 \cdots 01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.
- L'altro caso è $q^* = 255$ che ha due sottocasi, *Inf* se $m^* = 0$ e *NaN* se $m^* \neq 0$.
- Molti linguaggi di programmazione stampano *Inf* o *NaN* quando si stampa una variabile floating point che contiene questo risultato.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0 \cdots 01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.
- L'altro caso è $q^* = 255$ che ha due sottocasi, *Inf* se $m^* = 0$ e *NaN* se $m^* \neq 0$.
- Molti linguaggi di programmazione stampano *Inf* o *NaN* quando si stampa una variabile floating point che contiene questo risultato.
- Il computer produce *Inf* se il risultato di una operazione è più grande del più grande numero rappresentabile.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0\cdots01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.
- L'altro caso è $q^* = 255$ che ha due sottocasi, Inf se $m^* = 0$ e NaN se $m^* \neq 0$.
- Molti linguaggi di programmazione stampano Inf o NaN quando si stampa una variabile floating point che contiene questo risultato.
- Il computer produce Inf se il risultato di una operazione è più grande del più grande numero rappresentabile.
- Operazioni invalide che producono come risultato NaN sono: Inf/Inf, $0/0$, $Inf - Inf$. Operazioni con Inf hanno il significato usuale: $Inf + finito = Inf$, $Inf/Inf = NaN$, $Finito/Inf = 0$, $Inf - Inf = NaN$.

Accuratezza

- L'accuratezza delle operazioni Floating Point è determinata dalla grandezza degli errori di arrotondamento.

Accuratezza

- L'accuratezza delle operazioni Floating Point è determinata dalla grandezza degli errori di arrotondamento.
- Questo errore di arrotondamento è determinato dalla distanza di un numero dal numero floating point più vicino.

Accuratezza

- L'accuratezza delle operazioni Floating Point è determinata dalla grandezza degli errori di arrotondamento.
- Questo errore di arrotondamento è determinato dalla distanza di un numero dal numero floating point più vicino.
- Eccetto che per i numeri denormalizzati, i numeri floating point differiscono di un bit, l'ultimo bit di m^*. Cioè $2^{-23} \approx 10^{-7}$ in singola precisione. Questo è l'errore relativo e non l'errore assoluto.

IEEE - singola precisione

Occupa 4 byte = 32 bits

	$\longleftarrow$ 8 $\longrightarrow$	$\longleftarrow$ 23 $\longrightarrow$ —
s	q	m
segno	esponente	mantissa

Esso rappresenta $(-1)^s \cdot 2^{q-127} \cdot (1.m)$ Si noti che $\gamma_0 = 1$ nella mantissa non deve essere esplicitamente memorizzato, quindi $m = \gamma_1\gamma_2 \ldots \gamma_t$.
Range di numeri positivi normalizzati da 2^{-126} a $2^{127} \times 1.11 \cdots 1 \approx 2^{128}$ (da 1.2×10^{-38} a 3.4×10^{38})

IEEE - doppia precisione

Occupa 8 byte = 64 bits

	$\longleftarrow$ 11 $\longrightarrow$	$\longleftarrow$ 52 $\longrightarrow$ —
s	q	m
segno	esponente	mantissa

Esso rappresenta $(-1)^s \cdot 2^{q-1023} \cdot (1.m)$

Range di numeri positivi normalizzati da 2^{-1022} a $2^{1023} \times 1.11\cdots1 \approx 2^{1024}$ (da 2.2×10^{-308} a 1.8×10^{308})

IEEE - Eccezioni aritmetiche e risultati di default

tipo di eccezione	esempio	risultato di default
operazione invalide	$0/0 \quad 0 \times \infty$ $\sqrt{-1}$	NaN (Not a number)
Overflow		$\pm\infty$
Divisione per zero	Numero finito non nullo/0	$\pm\infty$
Underflow		numeri denormali