1 Aritmetica computazionale

1.1 Rappresentazione dei numeri reali

I **numeri finiti** sono utilizzati dai calcolatori per rappresentare i numeri reali poiché questi ultimi possono avere un numero infinito di cifre, che i calcolatori, avendo una memoria limitata, non sono in grado di rappresentare.

Teorema (Rappresentazione in base)**.** *Sia α un numero reale non nullo. Possiamo rappresentare tale numero con una base $\beta \geq 2$, un numero intero scelto da noi, nel seguente modo:*

$$\begin{aligned} \alpha &= \pm(\alpha_1\beta^{-1} + \alpha_2\beta^{-2} + \ldots)\beta^p \\ \alpha &= \pm(\sum_{i=1}^{\infty} \alpha_i\beta^{-i})\beta^p \end{aligned} \tag{1}$$

I vari termini dell'uguaglianza vengono detti:

β	*base*
p	*esponente*
α_i	*cifre del numero*
$\sum_{i=1}^{\infty} \alpha_i\beta^{-i}$	*mantissa*

Ogni cifra α_i è un numero intero che varia tra 0 e $\beta - 1$. Ad esempio, se lavoriamo in base 10, le cifre saranno numeri interi compresi tra 0 e 9.
Per garantire l'unicità della rappresentazione, è necessario che $\alpha_1 \neq 0$. Se così non fosse, il numero 13 potrebbe essere rappresentato come 13, 013, 0013, eccetera, il che va contro l'unicità della rappresentazione.

Possiamo scrivere un numero $\alpha \in \mathbb{R}$ con $\alpha \neq 0$ in due modi:

1. **forma mista**.
$$\alpha = \begin{cases} \pm(0.000\alpha_1\alpha_2\ldots)_\beta & p \leq 0 \\ \pm(\alpha_1\alpha_2\ldots)_\beta & p > 0 \end{cases}$$
2. **forma scientifica**. L'idea è quella di spostare il punto decimale al primo numero $\neq 0$ e poi moltiplicare il tutto per β^p per riportare il numero al suo valore originale.
$$\alpha = \pm 0.\alpha_1\alpha_2\ldots \cdot \beta^p$$

Esempio:

$$\begin{aligned} \alpha = (12.37)_{10} \quad & \alpha = 0.12237 \cdot 10^2 \\ \alpha = (0.0045)_{10} \quad & \alpha = 0.45 \cdot 10^{-2} \\ & = (4 \cdot 10^{-1} + 5 \cdot 10^{-2}) \cdot 10^{-2} \end{aligned}$$

Definizione (Numeri finiti)**.** *L'insieme $\mathbb{F}$ dei numeri finiti è definito come l'insieme dei numeri espressi in base β (dove $\beta \geq 2$), utilizzando t cifre (con $t \geq 1$). Poiché anche l'esponente p potrebbe essere così grande da non poter essere rappresentato, è necessario limitare l'intervallo degli esponenti rappresentabili. Qui, λ indica il più piccolo esponente che può essere rappresentato e ω il più grande esponente rappresentabile.*

$$\begin{aligned} \mathbb{F}(\beta, t, \lambda, \omega) &= \{0\} \cup \{\alpha \in \mathbb{R} : \alpha = \pm 0.\alpha_1\alpha_2\ldots\alpha_t \cdot \beta^p, \\ &= \{0\} \cup \{\alpha \in \mathbb{R} : \alpha = \pm(\sum_{i=1}^{t} \alpha_i\beta^{-i})\beta^p, \\ &\text{con } 0 \geq \alpha_i < \beta, \text{ per } i = 1, 2, \ldots, t, \ \alpha_1 \neq 0, \lambda \leq p \leq \omega\} \end{aligned}$$

$\mathbb{F}$ è un sottoinsieme che rappresenta una discretizzazione di $\mathbb{R}$. In altre parole, $\mathbb{F}$ è un insieme discreto di numeri presi da $\mathbb{R}$, dove ciascun numero può essere espresso al più in t cifre. Questo significa che gli elementi di $\mathbb{F}$ sono una selezione discreta di numeri reali con una precisione limitata a t cifre decimali.

Per convenzione, utilizzeremo α per scrivere i numeri reali e $\tilde{\alpha}$ per scrivere i numeri finiti.

Esempio:

Determinare e posizionare sull'asse reale gli elementi di $\mathbb{F}(2, 3, -1, 2)$.
I numeri rappresentabili possono essere espressi come:

$$\tilde{\alpha} = \pm 0.\alpha_1\alpha_2\alpha_3 \cdot 2^p$$

$$\tilde{\alpha} = \pm(\sum_{i=1}^{3} \alpha_i \cdot 2^{-i}) \cdot 2^p$$

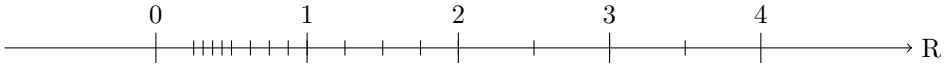
con $\tilde{\alpha} \in \mathbb{F},\ -1 \leq p < 3$ e $\alpha_1 \neq 0$
L'insieme delle possibili mantisse m_3 è dato da:

$$\begin{aligned} m_3 = \{&0.100, \\ &0.101, \\ &0.110, \\ &0.111\} \times \{2^{-1}, 2^0, 2^1, 2^2\} \end{aligned}$$

Pertanto, l'insieme degli elementi di $\mathbb{F}(2, 3, -1, 2)$ è composto da 33 elementi. Di questi, 16 sono positivi, 16 sono negativi e uno è lo zero.
Per capire come questi elementi sono posizionati sull'asse reale, li portiamo in base 10.

$$\begin{aligned} 0.100 &= 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} = \tfrac{1}{2} = \tfrac{4}{8} \\ 0.101 &= 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = \tfrac{5}{8} \\ 0.110 &= 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} = \tfrac{3}{4} = \tfrac{6}{8} \\ 0.111 &= 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = \tfrac{7}{8} \end{aligned}$$

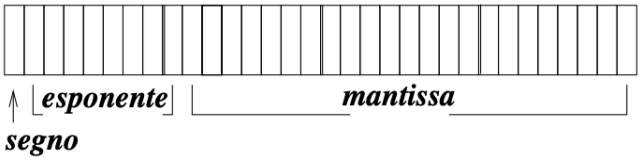
$\frac{4}{8} \cdot 2^{-1} = \frac{4}{16}$	$\frac{4}{8} \cdot 2^{0} = \frac{4}{8}$	$\frac{4}{8} \cdot 2^{1} = \frac{4}{4}$	$\frac{4}{8} \cdot 2^{2} = \frac{4}{2}$
$\frac{5}{8} \cdot 2^{-1} = \frac{5}{16}$	$\frac{5}{8} \cdot 2^{0} = \frac{5}{8}$	$\frac{5}{8} \cdot 2^{1} = \frac{5}{4}$	$\frac{5}{8} \cdot 2^{2} = \frac{5}{2}$
$\frac{6}{8} \cdot 2^{-1} = \frac{6}{16}$	$\frac{6}{8} \cdot 2^{0} = \frac{6}{8}$	$\frac{6}{8} \cdot 2^{1} = \frac{6}{4}$	$\frac{6}{8} \cdot 2^{2} = \frac{6}{2}$
$\frac{7}{8} \cdot 2^{-1} = \frac{7}{16}$	$\frac{7}{8} \cdot 2^{0} = \frac{7}{8}$	$\frac{7}{8} \cdot 2^{1} = \frac{7}{4}$	$\frac{7}{8} \cdot 2^{2} = \frac{7}{2}$



Notiamo come questi numeri sono equispaziati tra due potenze consecutive della base. Questo ci dà un'idea di come saranno fatti tutti gli insiemi di numeri finiti: tendono ad avere una densità maggiore vicino all'origine e si diradano man mano che ci si allontana da essa. La densità di questi numeri è direttamente influenzata dall'esponente negativo. Pertanto, è cruciale trovare un equilibrio tra numeri con esponenti sia positivi che negativi.

I numeri finiti sui calcolatori vengono rappresentati seguendo uno standard, come l'**ANSI/IEEE 754-1985**, che definisce formati specifici per la rappresentazione dei numeri in base 2.
Questo standard definisce 4 formati di numeri finiti, ma solo due di essi sono rigorosamente specificati. Gli altri due formati sono lasciati alla discrezione dei produttori di processori.
Lo scopo di uno standard è garantire la portabilità del codice, così che sia possibile eseguire lo stesso programma su differenti architetture ottenendo gli stessi risultati.
Gli n bit consecutivi dedicati per la memorizzazione di un numero finito vengono suddivisi tra le t cifre della mantissa ed un certo numero di bit $(\omega - \lambda + 1)$ per l'esponente p, più un bit per il segno del numero. Alcune tipiche rappresentazioni sono:

Basic precisione single	$\mathbb{F}(2, 24, -127, 128)$	32 bit
Basic precisione double	$\mathbb{F}(2, 53, -1023, 1024)$	64 bit



In precisione singola vengono destinati 24 bit alla mantissa (in realtà solo 23[1]) e 8 all'esponente ($2^8 = 256 = \omega - \lambda + 1$, con $\lambda = -127$ e $\omega = 128$), mentre in precisione doppia le cifre della mantissa sono 53 (memorizzati 52 bit) e dell'esponente 11 ($2^{11} = 2048 = \omega - \lambda + 1$, con $\lambda = -1023$ e $\omega = 1024$).
Si osservi che l'esponente è memorizzato per traslazione (*exponent biased*) e che la costante di traslazione (*bias*) è $-\lambda$. Quindi, se p è l'esponente del numero e $\tilde{p}$ è l'esponente memorizzato, possiamo trovare l'esponente memorizzato a partire dall'esponente originale utilizzando la seguente relazione:

$$\tilde{p} = p - \lambda$$

Dato un numero reale non nullo, α, per associare un numero finito ad esso, procediamo come segue:

1. **Rappresentazione esatta**. Se α è scritto nella forma $\alpha = \pm(.\alpha_1\alpha_2\ldots) \times \beta^p$ tale che $\lambda \leq p \leq \omega$, $\alpha_i = 0$ per $i > t$, allora è rappresentabile esattamente come un numero finito t di cifre e $\alpha \in \mathbb{F}(\beta, t, \lambda, \omega)$.

2. **Rappresentazione approssimata**. Altrimenti $\alpha \notin \mathbb{F}(\beta, t, \lambda, \omega)$ e quindi bisogna associargli un numero approssimato $\tilde{\alpha}$ che indicheremo con $fl(\alpha)$. Si hanno i seguenti casi:

   - $p \notin [\lambda, \omega]$, viene segnalata una condizione d'errore:

     $$\begin{array}{ll} p < \lambda & \textit{underflow} \\ p > \lambda & \textit{overflow} \end{array}$$

   - $p \in [\lambda, \omega]$, ma le cifre a_i con $i > t$ non sono tutte nulle, allora viene assegnato un numero finito $fl(\alpha)$ seguendo due possibili criteri:
     - **Troncamento** di α alla t-esima cifra

       $$fl_T(\alpha) = \pm(\sum_{i=1}^{t} \alpha_i \beta^{-i})\beta^p$$

     - **Arrotondamento** di α alla t-esima cifra

       $$fl_A(\alpha) = \pm fl_T((\sum_{i=1}^{t+1} \alpha_i \beta^{-i} + \frac{\beta}{2}\beta^{-(t+1)})\beta^p)$$

Esempio:

Il numero $\alpha = (0.11011)_2$ ha una mantissa di lunghezza 5, che è più lunga delle 3 cifre consentite in $\mathbb{F}(2, 3, -1, 2)$. Quindi, procediamo con l'operazione di arrotondamento:

$$\begin{array}{rll} fl_A(\alpha) = & 0.11011 & + \\ & 0.00010 & = \\ \hline & 0.11100 & \end{array}$$

[1]Essendo sempre α_1=1 per la rappresentazione binaria, la prima cifra può essere sottintesa senza mai essere fisicamente memorizzata.

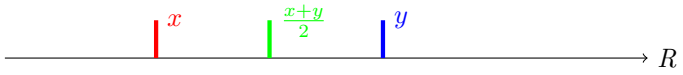
Esempio:

Consideriamo l'insieme dei numeri finiti $\mathbb{F}(10, 5, -50, 49)$. Per rappresentare un numero finito in questo insieme in memoria, dobbiamo definire il numero di posizioni necessarie. Nello specifico:

- **Segno**: una posizione è riservata per il segno. Se il numero è positivo si usa 0; se è negativo, si usa $\beta - 1$.
- **Esponente**: due posizioni sono destinate all'esponente. Usando la tecnica di memorizzazione per traslazione ($p - \lambda = \tilde{p}$), possiamo rappresentare gli esponenti da -50 a 49 attraverso valori memorizzati da 00 a 99.
- **Mantissa**: cinque posizioni sono dedicate alla mantissa.

$$\begin{aligned} \alpha = 0.0532 = 0.532 \cdot 10^{-1} \qquad & fl(\alpha) = 04953200 \\ \alpha = -237141 = -0.237141 \cdot 10^{6} \qquad & fl(\alpha) = 95623714 \end{aligned}$$

Osservazione. *Siano x ed y due numeri $\in \mathbb{F}$ consecutivi positivi. Sia $\alpha \in \mathbb{R}$ tale che $x \leq \alpha < y$.*



Allora possiamo affermare che α non appartiene all'insieme $\mathbb{F}$ perché, per ipotesi, x e y sono consecutivi e non ci può essere un altro numero tra loro. Tuttavia, la rappresentazione approssimata $fl(\alpha)$ risulta essere:

$$fl_T(\alpha) = x \quad fl_A(\alpha) = \begin{cases} x & se\ \alpha < \frac{x+y}{2} \\ y & se\ \alpha \geq \frac{x+y}{2} \end{cases}$$

L'errore commesso nel troncamento sarà sempre maggiore o uguale dell'errore commesso nell'arrotondamento. Questo è il motivo per cui, con una base numerica pari, si preferisce utilizzare l'arrotondamento, poiché fornirà una migliore approssimazione del numero reale rispetto al troncamento.

La modalità di arrotondamento dello standard ANSI/IEEE-754 coincide con quella precedentemente descritta, con la particolarità dell'**arrotondamento ai pari**. Questa particolarità si applica quando un numero reale α è esattamente equidistante dai numeri finiti consecutivi x ed y, in altre parole, quando $\alpha = \frac{x+y}{2}$. In questa situazione l'arrotondamento funziona nel seguente modo:

$$fl_{AP}(\alpha) = \begin{cases} x & \text{se x è pari} \\ \text{y} & \text{se y è pari} \end{cases}$$

Sempre parlando dello standard ANSI/IEEE-754, per gestire risultati non rappresentabili, vengono utilizzati due valori speciali:

- **NaN**
- **Inf**

Invece, di avere un buco vicino allo zero dove i numeri molto piccoli verrebbero immediatamente arrotondati a zero, vengono inseriti dei numeri ulteriori per riempire questo buco e permettere ai valori di avvicinarsi progressivamente a zero. Questo meccanismo è chiamato **gradual underflow**.
Per rappresentare questi numeri estremamente piccoli, si fa uso della rappresentazione **denormalizzata**. In questa rappresentazione, la mantissa non inizia con il solito bit implicito di 1, ma con una serie di 0.

1.2 Errori di rappresentazione

Definizione. *Consideriamo un valore $\alpha \in \mathbb{R}$. Se $\alpha \notin \mathbb{F}(\beta, t, \lambda, \omega)$, allora la sua migliore approssimazione all'interno di questo insieme è data da $\tilde{\alpha} \in \mathbb{F}(\beta, t, \lambda, \omega)$. L'approssimazione di α con $\tilde{\alpha}$ introduce un **errore di rappresentazione**. Per quantificare tale errore, definiamo le seguenti metriche:*

$$\begin{aligned} E_{abs} &= |\alpha - fl(\alpha)| && \textit{errore assoluto} \\ E_{rel} &= |\tfrac{\alpha - fl(\alpha)}{\alpha}| \ \textit{se } \alpha \neq 0 && \textit{errore relativo} \end{aligned}$$

Nel calcolo scientifico, l'errore relativo è preferito poiché fornisce una misura dell'errore "normalizzata", che non dipende dalla grandezza dei numeri confrontati.

Esempio:

Si eseguano i passi necessari per rappresentare il numero reale $(-13.9)_{10}$ in un'area di memoria di 8 bit (1 per il segno, 3 per l'exponent biased e 4 per la mantissa), che permettono di memorizzare $\mathbb{F}(2, 5, -3, 4)$ per troncamento e arrotondamento.

1. **Conversione in binario**, prima la parte intera, quindi la parte decimale:

   - *Parte intera*:
     (a) Dividi il numero per 2.
     (b) Registra il resto della divisione (sarà 0 o 1).
     (c) Usa il quoziente ottenuto come nuovo numero e ripeti la divisione per 2.
     (d) Continua il processo fino a quando il quoziente diventa 0.
     (e) Leggi i resti della divisione in ordine inverso: questo sarà il numero in base 2 della parte intera.

   $$(13)_{10} = (1101)_2$$

   - *Parte decimale*:
     (a) Moltiplica la parte decimale per 2.
     (b) Registra la parte intera del risultato (sarà 0 o 1).
     (c) Usa la parte decimale del risultato come nuovo numero e ripeti la moltiplicazione per 2.
     (d) Continua questo processo finché non ottieni una parte decimale di 0 o si arriva al limite di precisione della mantissa.
     (e) Leggi i numeri interi in ordine di apparizione: questo sarà il numero in base 2 della parte decimale.

   $$0.9 \times 2 = \underline{1}.8$$
   $$0.8 \times 2 = \underline{1}.6$$
   $$0.6 \times 2 = \underline{1}.2$$
   $$0.2 \times 2 = \underline{0}.4$$
   $$0.4 \times 2 = \underline{0}.8$$
   $$(0.9)_{10} = (11100\ldots)_2$$

   da cui
   $$(-13.9)_{10} = (-1101.11100\ldots)_2$$

2. **Normalizzazione**: nello standard IEEE-754, la rappresentazione normalizzata dei numeri in virgola mobile prevede che la parte intera sia sempre 1.

   $$(-1101.11100\ldots)_2 = (-1.10111100\ldots)_2 \times 2^3$$

3. **Calcolo dell'esponente biased**:

   $$p - \lambda = \tilde{p} \to 3 - (-3) = 6$$

   $$(-1.10111100\ldots)_2 \times 2^3 = (-1.10111100\ldots)_2 \times 2^{(110)_2}$$

4. **Rappresentazione della mantissa**:

arrotondamento	troncamento
$1.10111 + 0.00001 = 1.1100$	1.1011

5. **Rappresentazione in memoria**: nello standard IEEE-754, con una mantissa di 5 bit, solo 4 bit vengono effettivamente memorizzati in memoria.

1	1	1	0	1	1	0	0

1	1	1	0	1	0	1	1

Esempio: