3. Struttura del sistema floating-poing

In questo capitolo studieremo la struttura del sistema floating-point, ovvero dell'insieme dei *reali-macchina* e dei valori rappresentabili per arrotondamento tramite i reali-macchina.
Tratteremo un sistema astratto in base b generica (facendo esempi in base 10), per poi discutere un modello semplificato dello standard a 64 bit in base 2, utilizzato in *Matlab* e in tutti i principali linguaggi di calcolo.

Ricordiamo che i reali-macchina in base b, composti da t cifre di mantissa e con range di esponenti $[L, U] \subset \mathbb{Z}$ sono definiti da:

$$\mathbb{F}(b, t, L, U) =$$

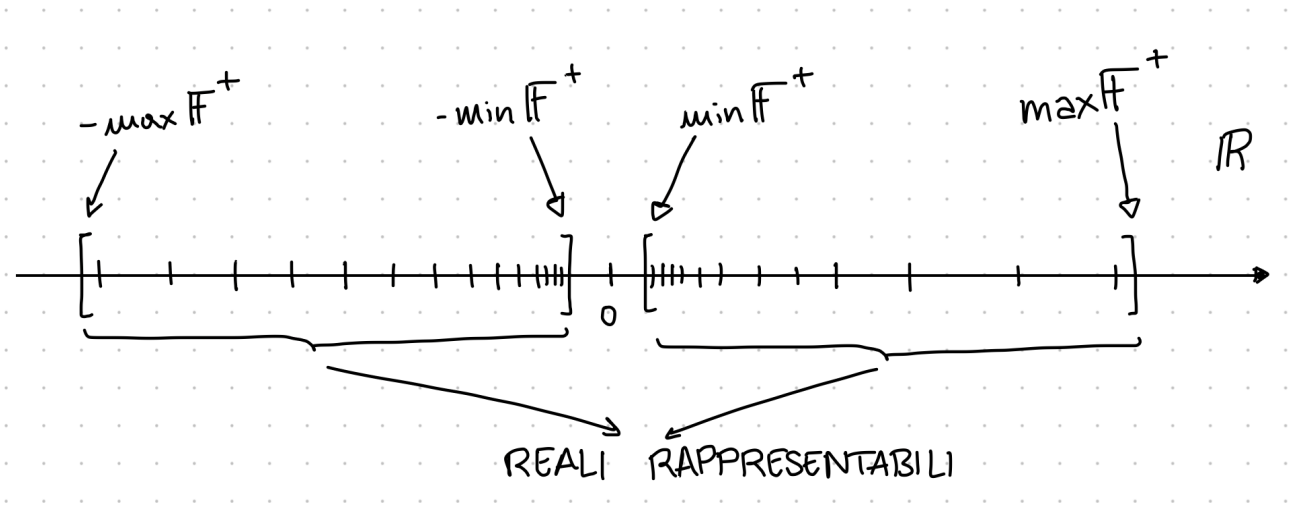
$$\{\mu \in \mathbb{Q}, \mu = \text{sgn}(\mu)(0.\mu_1\mu_2 \dots \mu_t)b^p : \mu \in \{0, 1, \dots, b-1\}, \mu_1 \neq 0, p \in [L, U] \subset \mathbb{Z}\}$$

dove $\mu_j, j = 1, \dots, t$ sono le cifre della mantissa e p l'esponente intero della potenza della base che sposta la virgola, incluso nell'intervallo $[L, U]$, dove $L < 0$ e $U > 0$.

Andremo a rispondere alle seguenti domande:

- quanti sono i reali-macchina?
- quali reali sono approssimabili per arrotondamento con i reali-macchina?
- come sono distribuiti sull'asse reale i reali-macchina?

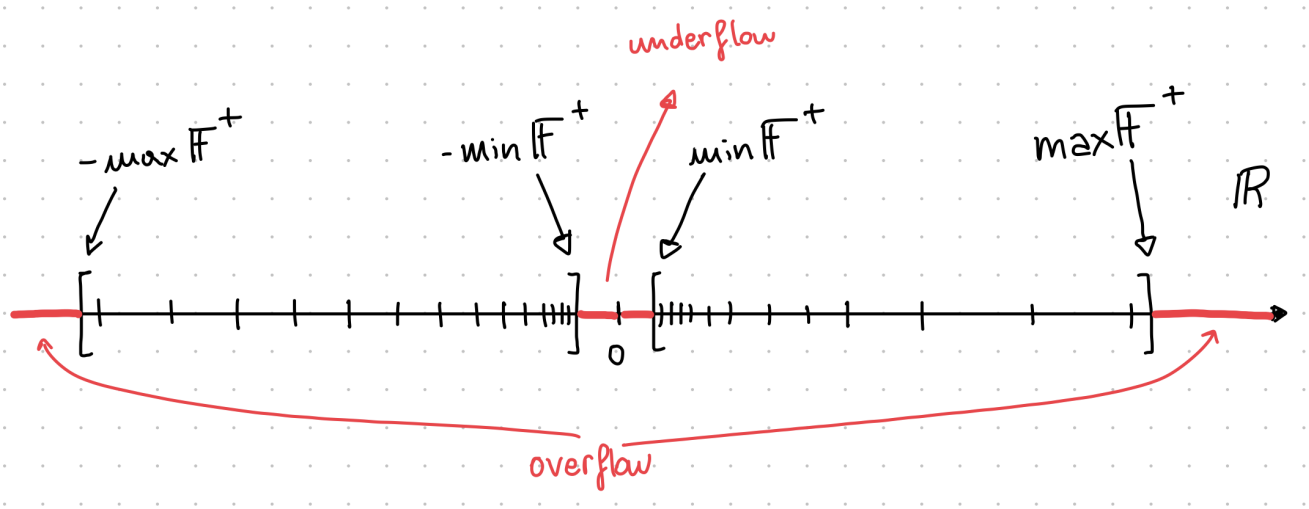
In modo molto schematico, sintetizziamo con un disegno:



In questo schemino grafico i reali-macchina corrispondono all'insieme *finito* di "tacche" sull'asse reale e, a parte lo 0, che ha una rappresentazione speciale visto che la mantissa nulla non è ammessa, *stanno nell'unione di due intervalli simmetrici*.

Infatti dato $\mathbb{F}^+$ che rappresenta i reali-macchina > 0 e $\mathbb{F}^-$ che rappresenta quelli < 0, abbiamo che $\mathbb{F}^- = -\mathbb{F}^+$.
I due intervalli $\mathbb{F}^+$ e $\mathbb{F}^-$ sono proprio, in $\mathbb{R}$, i *reali appossimabili* tramite i reali-macchina per arrotondamento a t cifre della mantissa, con un errore relativo $\leq \epsilon_M = \frac{b^{1-t}}{2}$, che equivale alla precisione di macchina.

Invece i reali che in modulo sono $> \max \mathbb{F}^+$ e quelli in modulo $\leq \min \mathbb{F}^-$, sono o troppo grandi o troppo piccoli per essere approssimati e costituiscono quello che in gergo si chiama, rispettivamente, *overflow* e *underflow* nel sistema floating-point:



Il problema è che l'esponente p di tali numeri è fuori dal range di esponenti ammissibili $[L, U] \subset \mathbb{Z}$ e la comparsa di numeri in overflow o underflow durante un processo di calcolo altera il processo stesso facendogli tendenzialmente perdere di significato.

In Matlab invece un numero in overflow, generato ad esempio dal prodotto di reali-macchina molto grandi in modulo, viene indicato col simbolo `Inf`, cioè infinito, ed entra nel processo di calcolo alterandolo, generando altri `Inf` o forme indeterminate come ad esempio $\frac{\text{Inf}}{\text{Inf}}$, che vengono indicate col simbolo `NaN`, che ha il significato di "Not a Number".
Per questo negli algoritmi numerici è importante cercare di prevenire overflow o underflow.

A questo punto conviene calcolare $\min \mathbb{F}^+$ e $\max \mathbb{F}^+$, in modo da quantificare gli estremi degli intervalli di rappresentazione visto che un reale-macchina ha la struttura:

$$\mu = \text{segno} \cdot \text{mantissa} \cdot b^p$$

Il minimo positivo si otterrà moltiplicando la minima mantissa per la potenza della base con il minimo esponente:

$$\text{mantissa minima} = (0.10\ldots0)_b = 1 \cdot b^{-1}$$

$$\text{esponente minimo} = L$$

$$\min \mathbb{F}^+ = b^{L-1}$$

Il massimo positivo invece si otterrà moltiplicando la mantissa massima per la potenza della base col massimo esponente:

$$\text{mantissa massima} = (0.[b-1][b-1]\dots[b-1])_b$$

cioè tutte le t cifre sono uguali alla cifra massima che è $b-1$.
Calcoliamo usando la somma geometrica di ragione b^{-1}:

$$\text{mantissa massima} = \sum_{j=1}^{t}(b-1)b^{-j} = 1 - b^{-t}$$

$$\text{esponente massimo} = U$$

$$\max \mathbb{F}^+ = (1-b^{-t})b^U$$

Ad esempio in base 10 con 16 cifre di mantissa ed esponenti minimo $L = -307$ e massimo $U = +308$:

$$\min \mathbb{F}^+ = 10^{-1-307} = 10^{-308}$$

$$\max \mathbb{F}^+ = (\underbrace{1 - 10^{-16}}_{\approx 1})10^{308} \approx 10^{-308}$$

I reali rappresentabili tramite reali-macchina in Matlab sono quindi intervalli estremamente ampi, con un estremo "grandissimo" e un estremo "piccolissimo"
Importante è ricordare ancora una volta che la rappresentazione avviene per arrotondamento della mantissa e che gli *unici* numeri effettivamente memorizzabili nel calcolatore sono i *reali-macchina*, un *insieme finito*.

Ma qual è la cardinalità di $\mathbb{F}$? Il calcolo è facile, osservando che basta contare gli elementi di $\mathbb{F}^+$ e raddoppiarli, tenendo poi conto dello 0, che ha una rappresentazione speciale:

$$\text{card}(\mathbb{F}^+) = \underbrace{(b-1)b^{t-1}}_{\text{numero mantisse}} \cdot \underbrace{(U-L+1)}_{\text{numero esponenti}}$$

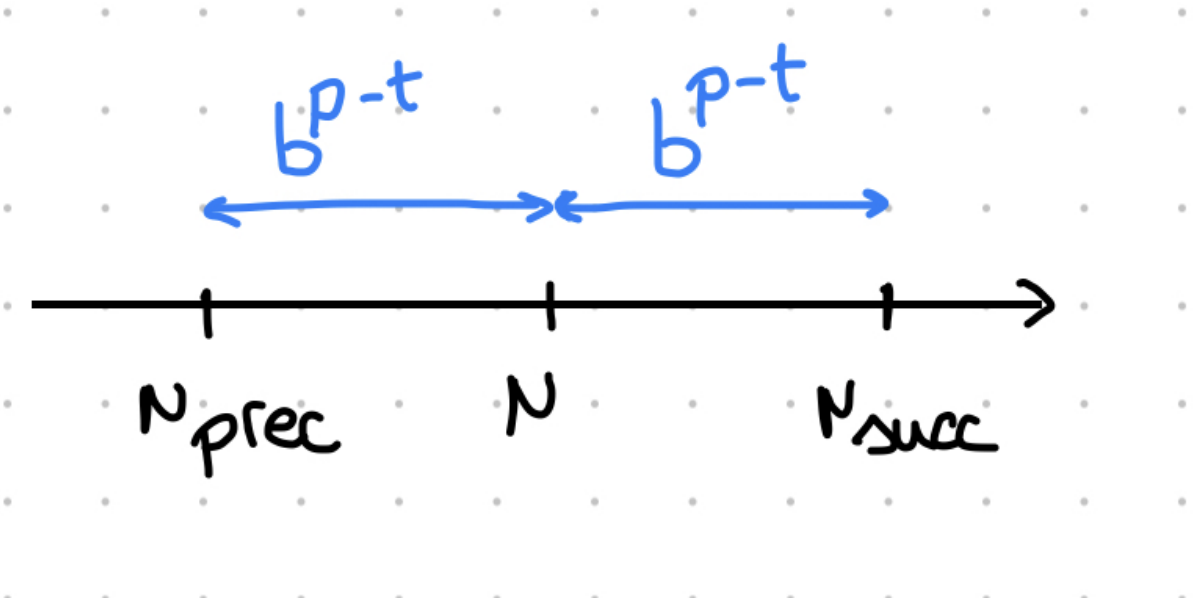
In questo conto si osservi che per la prima cifra di mantissa ci sono $b-1$ scelte, perché non è possibile iniziare con lo 0, mentre ci sono b scelte per tutte le altre cifre.
Alla fine si ottiene:

$$\text{card}(\mathbb{F}) = 1 + 2(b-1)b^{t-1} \cdot (U-L+1)$$

Ad esempio in Matlab abbiamo una $\text{card}(\mathbb{F}) = 1 + 2 \cdot 9 \cdot 10^{15} \cdot 616$ che è circa un numero dell'ordine di 10^{19}.

Ci resta ora da anailzzare come sono distribuiti i reali-macchina. Il concetto chiave è che *non sono distribuiti uniformemente*, ma bensì a *densità variabile*, i reali-macchina piccoli sono vicini, mentre quelli grandi sono distanti tra di loro: cioè la densità cresce andando verso $\pm \min \mathbb{F}^+$ e descresce andando verso $\pm \max \mathbb{F}^+$.

Questo si può capire bene calcolando la distanza tra reali-macchina consecutivi (con stesso esponente p), ad esempio:

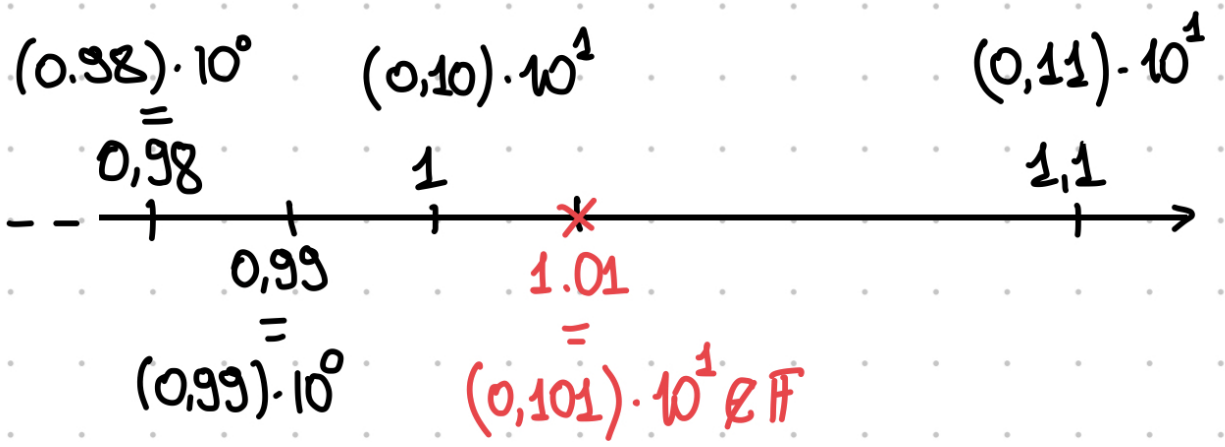


Infatti due reali-macchina consecutivi differiscono di 1 nella t-esima cifra di mantissa, ovvero:

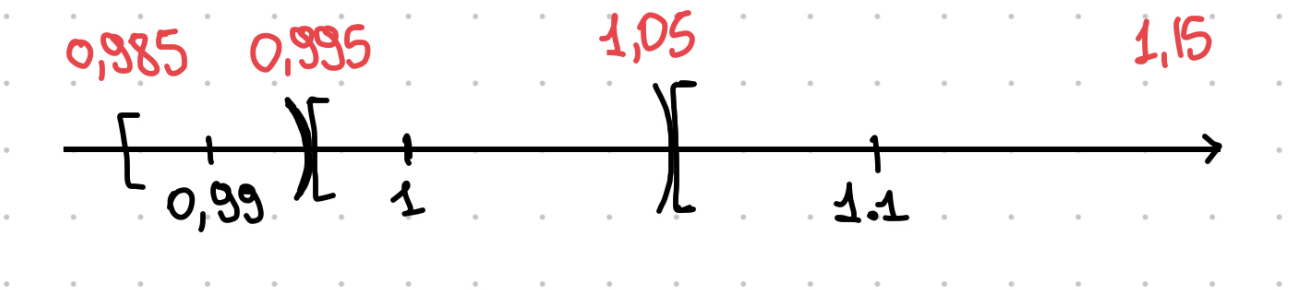
$$\text{mantissa}(\mu_{succ}) - \text{mantissa}(\mu) = 1 \cdot b^{-t}$$

$$\text{e} \quad \mu_{succ} - \mu = b^{-t} \cdot b^{p} = b^{p-t}$$

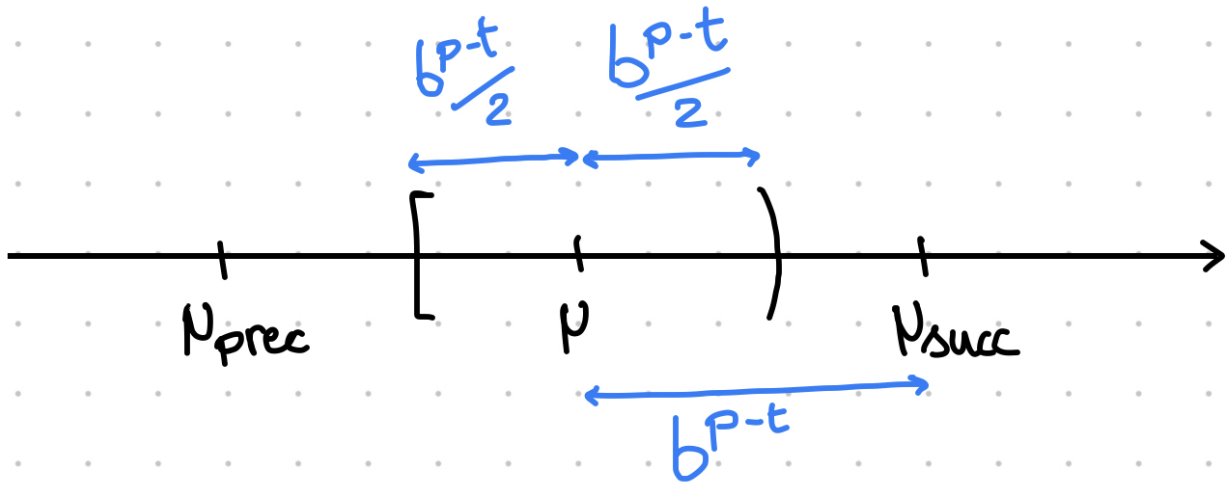
La distanza assoluta è quindi variabile con p, cioè con l'ordine di grandezza del numero in base b. Questa distanza varia quindi di un certo fattore dipendente da b quando si passa per una potenza della base.
Facciamo un esempio con $b = 10$ e $t = 2$: la domanda che possiamo farci ad esempio è: chi èi l reale-macchina successivo ad 1? Pensando a 2 cifre decimali, verrebbe da rispondere 1.01, ma non è così, perché $1.01 = (0.101) \cdot 10^1$ ha 3 cifre di mantissa e in questo esempio $t = 2$. Invece il reale-macchina successivo ad 1 è $(0.11) \cdot 10^1 = 1.1$.



Passando per $1 = 10^0$ la distanza è cresciuta di un fattore 10, da 10^{-2} a 10^{-1} .
Pensando invece a quali reali sono approssimati da $0.99, 1$ e 1.1 abbiamo graficamente:

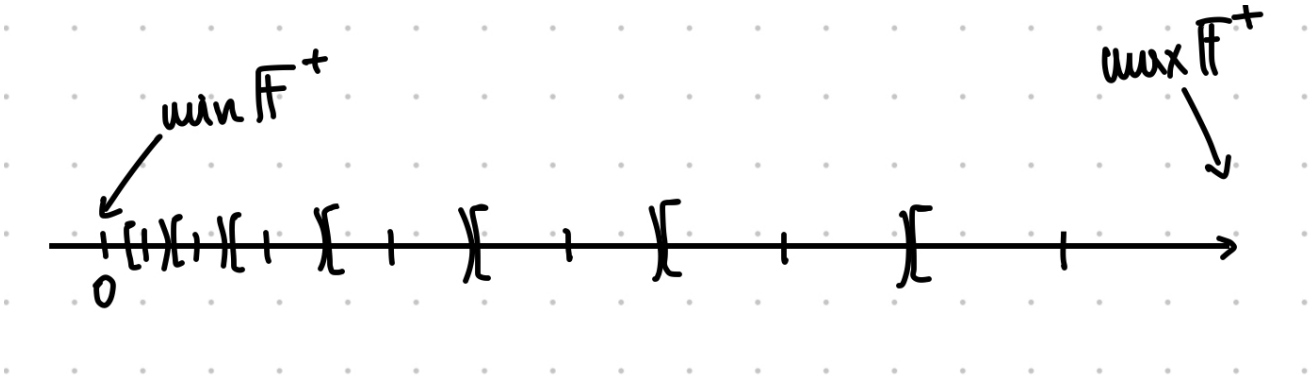


Vediamo che ogni reale-macchina è il centro di un intorno di approssimazione a t cifre di mantissa, nel nostro caso 2: questi intorni sono simmetrici e di raggio $\frac{b^{p-t}}{2}$, fino a quando non scatta il prossimo ordine di grandezza, nel qual caso l'intorno e' asimmetrico e l'intorno destro si allunga di un fattor b.

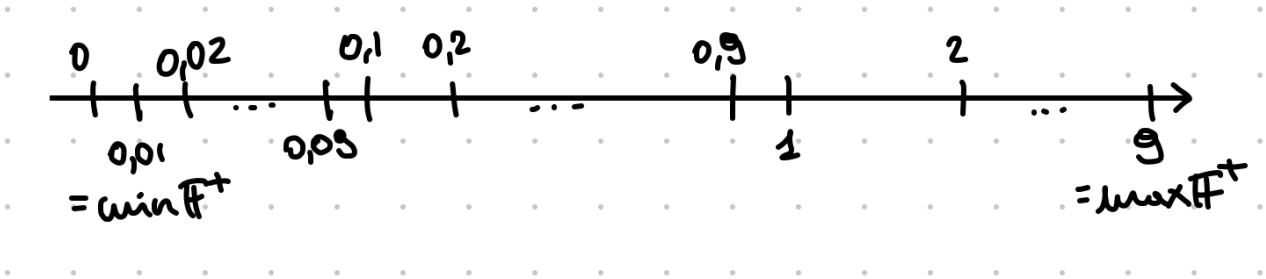


Tutti i reali di questi intorni vengono approssimati dal reale-macchina centro dell'intorno per arrotondamento a t cifre di mantissa, con un errore assoluto $\leq \frac{b^{p-t}}{2} = \text{raggio}$.
Ma se andiamo a stimare *l'errore relativo*, sappiamo che questo non può superare la precisione di macchina $\epsilon_M = \frac{b^{1-t}}{2}$, che è *indipendente da* p.

L'unione di tutti gli intorni di approssimazione per arrotondamento copre gli intervalli dei reali rappresentabili, ovvero $[-\max \mathbb{F}^+, -\min \mathbb{F}^+] \cup [\min \mathbb{F}^+, \max \mathbb{F}^+]$ come descritto in modo un po' ingenuo dal seguente disegno:

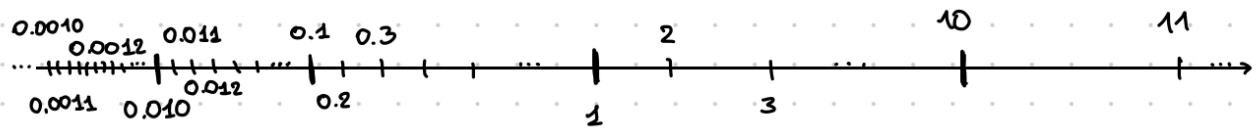


Concludiamo il seguente capitolo con due esempi semplici in base 10: disegnare $\mathbb{F}(10, 1, -1, 1)$ e $\mathbb{F}(10, 2, -2, 2)$.
Nel primo caso abbiamo una sola cifra di mantissa, che corrisponde al minimo possibile, ed esponenti $p = -1, 0, 1$:



Vediamo che si parte dall'ordine di grandezza dei centesimi con $\min \mathbb{F}^+ = 0.1 \cdot 10^-1 = 0.01$, notando anche che tra i centesimi non ci sono reali-macchina perché c'è una sola cifra di mantissa, passando poi ai decimi, poi alle unità sempre senza reali-macchina intermedi, fino a $\max \mathbb{F}^+ = (0.9) \cdot 10^1 = 9$.
La precisione di macchina è $\epsilon_M = \dfrac{10^{1-t}}{2} = \dfrac{10^{1-1}}{2} = \dfrac{1}{2} = 50\%$.
Si trata di un sistema floating-point molto povero, perché c'è una sola cifra di mantissa e un'estensione molto limitata dal piccolo range di esponenti.

Il secondo è un esempio di sistema più ricco del precedente infatti abbiamo $\min \mathbb{F}^+ = b^{L-1} = 10^{-3}$ e $\max \mathbb{F}^+ = (1 - b^{-t}) \cdot b^U = (1 - 10^{-2}) \cdot 10^2 = 99$ e infine $\epsilon_M = \dfrac{b^{1-t}}{2} = \dfrac{10^{-1}}{2} = 5\%$.
Partiamo quindi dai millesimi, ma fra questi ci sono i decimillesimi perché abbiamo 2 cifre di mantissa, poi ai centersimi coi millesimi, i decimi coi centesimi, le unità coi decimi e infine le decine con le unità.



Questo dovrebbe far capire quanto ricca è la struttura del sistema floating-point a 64 bit, detta anche a *doppia precisione*. Bisogna anche ricordare che esistono altri

standard come ad esempio quello a precisione *singola* a 32 bit con $\epsilon_M \approx 10^{-8}$ e la precisione *quadrupla* a 128 bit con $\epsilon_M \approx 10^{-32}$.
Quale sia la precisione di default dipende dal linguaggio, alcuni permettono di scegliere tra singola, doppia e quadrupla. Queste precisioni corrispondono ad operazioni aritmetiche implementate a livello hardware, di circuiti, nel processore, quindi estremamente veloci.
In alcuni linugaggi è possibile aumentare la precisione, cioè aumentare il numeri di cifre di mantissa, ma a quel punto le operazioni aritmetiche sono implementate a livello software, di conseguenza il costo delle operazioni cresce e cresce anche ovviamente l'occupazione di memoria, soprattutto elaborando una grossa mola di dati.

Questo è il motivo per cui si è fatto un compromesso e lo standard attuale è la precisione doppia con $\epsilon_M = 2^{-53} \approx 10^{-16}$, un massimo errore relativo di arrotondamento che è di solito molto più piccolo degli errori di misura sperimentale dei dati nelle applicazioni.
Ciononostante, vedremo che la propagazione degli errori in algoritmi instabili può far perdere anche tutta questa precisione nel processo di calcolo.
Questo conduce in modo naturale allo studio della *stabilità* degli algoritmi numerici.