Calcolo Numerico

Luca De Paulis

3 marzo 2021

INDICE

1 ARITMETICA DI MACCHINA

1.1 RAPPRESENTAZIONE DEI NUMERI DI MACCHINA

Il primo problema da risolvere quando si vuole fare analisi numerica è scegliere un metodo per rappresentare i numeri reali su una macchina. Infatti un generico numero reale ha potenzialmente una scrittura decimale infinita, dunque essendo le risorse disponibili in una macchina *finite* dobbiamo trovare un modo di approssimarlo.

1.1.1 Virgola fissa

Il primo metodo è il metodo **a virgola fissa**: in questo metodo si rappresentano tutti i numeri nella loro forma decimale normale e si considerano esattamente k cifre dopo la virgola.

Questo metodo è molto semplice e ci consente di fare operazioni elementari (come le somme o i prodotti) immediatamente ("in colonna"), tuttavia ha anche degli svantaggi evidenti, come

- il range dei numeri rappresentabili su n cifre/bit è piccolo;
- siccome il numero di bit dedicato ai numeri dopo la virgola è basso, la precisione è molto bassa e assolutamente non adeguata ad applicazioni di analisi numerica.

Per questo è stata inventata la rappresentazione in virgola mobile.

1.1.2 Virgola mobile

Innanzitutto dobbiamo trovare un modo standard per rappresentare un qualsiasi numero reale. Per far ciò ci viene in aiuto il seguente teorema.

Teorema 1.1.1 **Teorema di rappresentazione in base.**
Sia $x \in \mathbb{R}$, $x \neq 0$ e sia $\beta \geqslant 2$ una base di rappresentazione.

*Allora esistono e sono univocamente determinati un **esponente** $p \in \mathbb{Z}$ e una successione di **cifre** $(d_i)_{i\in\mathbb{N}}$ (con $d_i \in \mathbb{Z}$ per ogni i) per cui vale che*

(i) $d_1 \neq 0$;

(ii) $0 \leqslant d_i \leqslant \beta - 1$ *per ogni* i;

(iii) *la successione* d *non è definitivamente uguale a* $\beta - 1$, *ovvero per ogni* $k > 0$ *esiste un* $j \geqslant k$ *tale che* $d_j \neq \beta - 1$;

(iv) *si ha che*

$$x = \operatorname{sgn}(x) \cdot \beta^p \cdot \left(\sum_{i=1}^{\infty} d_i \beta^{-i} \right).$$

La rappresentazione di x nella forma data dal Teorema 1.1.1 si dice **rappresentazione normalizzata in virgola mobile**.

Esempio 1.1.2. Consideriamo ad esempio il numero reale 123. Per il Teorema 1.1.1 possiamo esprimere 123 come

$$123 = +10^3 \cdot (0.123) = +10^3 \cdot (1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 3 \cdot 10^{-3}).$$

Questa rappresentazione è l'unica possibile se imponiamo che d_1 sia diverso da 0 e che le cifre della rappresentazione siano tutte considerate come *cifre decimali* moltiplicate per una certa potenza della base (in questo caso 3).

La rappresentazione data dal Teorema 1.1.1 è essenzialmente il concetto di *notazione scientifica*: in notazione scientifica portiamo il numero reale in modo che abbia una singola cifra diversa da zero prima della virgola, mentre nella notazione data dal Teorema quella cifra diventa la prima cifra dopo la virgola. Inoltre, il Teorema ci garantisce che questa rappresentazione non è solamente valida in base 10, ma lo è in qualsiasi base (noi lo useremo in particolare in base 2).

Il fattore β^p viene detto **esponente**, mentre la parte decimale (corrispondente alla sommatoria) viene detta **mantissa**.

Osservazione 1.1.1. Le ipotesi (i) e (iii) del Teorema sono fondamentali per l'unicità della rappresentazione. In effetti

- se venisse a mancare la prima condizione avremmo che

$$123 = +10^3 \cdot (0.123) = +10^4 \cdot (0.0123) = +10^5 \cdot (0.00123) = \dots;$$

- se venisse a mancare la terza condizione (che ci dice che la successione d non può terminare con una sequenza infinita di $(\beta - 1)$, ovvero il numero non può finire con $\beta - 1$ periodico) avremmo dei problemi più subdoli, che derivano dalla non esistenza del "9 *periodico*" (dimostrata in appendice, Proposizione A.1.1).

  Infatti in qualsiasi base β il numero $0.\overline{(\beta - 1)}$ è uguale a 1, quindi se ammettessimo entrambe le rappresentazioni verrebbe meno l'unicità.

Notazione. Useremo la notazione $(n)_\beta$ per riferirci al numero n espresso in base β.

Ad esempio il numero $(1011)_2$ si riferisce al numero

$$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 2 + 1 = 11 \text{ (in base 10)}.$$

Per rappresentare i numeri reali in macchina dobbiamo quindi risolvere ancora due problemi:

- la macchina usa (nella maggior parte dei casi) la base 2, quindi dobbiamo poter trasformare da base 10 in base 2;
- dobbiamo rappresentare i numeri infiniti in modo approssimato.

Per quanto riguarda il primo, l'algoritmo per trasformare un numero decimale (in base 10) in un numero in base 2 è il seguente:

1. si trasforma la parte intera (le cifre prima della virgola) in base 2 tramite divisioni successive per 2;
2. si trasforma la parte decimale (le cifre dopo la virgola) in base 2 tramite *moltiplicazioni* successive per 2.

Esempio 1.1.3. Trasformiamo 3.15 in base 2: ovviamente $(3)_{10} = (11)_2$ e quindi rimane solo da trasformare la parte decimale.

L'algoritmo consiste nel prendere il numero decimale 0.15 e moltiplicarlo per 2 ripetutamente: se il risultato ha come cifra prima della virgola uno 0 aggiungiamo uno 0 alla rappresentazione in base 2, altrimenti un 1. Nella pratica:

$$\begin{aligned}
0.15 \cdot 2 &= \boxed{0}.30 \\
0.30 \cdot 2 &= \boxed{0}.60 \\
0.60 \cdot 2 &= \boxed{1}.20 \\
0.20 \cdot 2 &= \boxed{0}.40 \\
0.40 \cdot 2 &= \boxed{0}.80 \\
0.80 \cdot 2 &= \boxed{1}.60 \\
0.60 \cdot 2 &= \boxed{1}.20 \\
&\vdots
\end{aligned}$$

Osserviamo che ripetendo il procedimento la sequenza $(0.60, 0.20, 0.40, 0.80)$ si ripete all'infinito, dunque il numero in base 2 termina con le cifre 1001 ripetute periodicamente. La parte decimale in base 2 corrisponde quindi a $(0.00\overline{1001})_2$.

Il numero completo è quindi

$$(11.00\overline{1001})_2 = 2^2 \cdot (0.1100\overline{1001})_2.$$

Per rappresentare i numeri infiniti dobbiamo approssimare la parte decimale, considerando solo un numero finito t di cifre dopo la virgola. Per far ciò esistono due metodi (che esamineremo più nel dettaglio nel seguito):

- nel caso del **troncamento** si considerano le prime t cifre dopo la virgola e si scartano tutte le successive;
- nel caso dell'**arrotondamento** analizziamo la cifra decimale di posto $t+1$:
  - se d_{t+1} appartiene all'intervallo $\left[0, \frac{\beta}{2}\right)$ (dove β è la base) consideriamo semplicemente le prime t cifre;
  - altrimenti (se $d_{t+1} \in \left[\frac{\beta}{2}, \beta\right]$ aggiungiamo 1 all'ultima cifra decimale (come riporto, quindi lo propaghiamo in caso sia necessario).

Esempio 1.1.4. Riprendiamo il numero considerato precedentemente, ovvero

$$(3.15)_{10} = 2^2 \cdot (0.1100\overline{1001})_2$$

e approssimiamolo a $t = 8$ cifre decimali.

Nel caso del troncamento basta considerare le prime 8 cifre decimali, quindi il numero troncato è

$$2^2 \cdot (0.11001001)_2.$$

Nel caso dell'arrotondamento invece dobbiamo considerare la nona cifra decimale: espandendo la rappresentazione decimale osserviamo che la nona cifra è

$$0.11001001\underline{1}00\ldots;$$

siccome $1 \in \left[\frac{2}{2}, 2\right] = [1, 2]$ dobbiamo *approssimare per eccesso*, ottenendo

$$2^2 \cdot (0.11001010)_2.$$

1.1.3 Insieme dei numeri di macchina

Siccome un singolo numero può occupare una quantità fissa in memoria (tipicamente 32 o 64 bit) dobbiamo fissare dei limiti per l'esponente e per il numero di cifre decimali che possiamo usare per la rappresentazione. Osserviamo inoltre che, se le nostre risorse sono limitate, aumentando il numero di cifre decimali disponibili dobbiamo necessariamente diminuire lo spazio dedicato a memorizzare l'esponente e viceversa.

Definizione 1.1.5

Insieme dei numeri di macchina. Sia $\beta \geqslant 2$ una base, t il numero di cifre decimali utilizzabili e siano $m,\ M \in \mathbb{Z}$ gli estremi per l'esponente (ovvero ogni esponente p rappresentabile deve appartenere a $[-m, M]$). Si dice allora **insieme dei numeri di macchina** l'insieme

$$\begin{aligned}\mathcal{F}(\beta, t, m, M) \coloneqq \{0\} \cup \Big\{\, x \in \mathbb{R} \;:\; & x = \operatorname{sgn}(x) \cdot \beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}, \\ & d_1 \neq 0, \\ & 0 \leqslant d_i \leqslant \beta - 1, \\ & -m \leqslant p \leqslant M \,\Big\}.\end{aligned}$$

Osservazione 1.1.2. Nelle condizioni in cui specifichiamo quali numeri sono rappresentabili compaiono le ipotesi (i) e (ii) del Teorema di rappresentazione in base, ma non l'ipotesi (iii). Quest deriva dal fatto che i numeri di macchina hanno una precisione finita (ovvero hanno al più t cifre decimali), dunque non è possibile avere un numero che termina in $\beta - 1$ periodico.

OMEGA GRANDE Chiamiamo Ω il più grande numero di macchina. Possiamo costruirlo in questo modo:

- scegliamo come segno $+$;
- scegliamo come esponente il massimo esponente possibile, ovvero $p = M$;
- poniamo tutte le t cifre decimali uguali al massimo possibile, che in base β è $\beta - 1$.

Segue quindi che

$$\Omega \coloneqq +\beta^M \sum_{i=1}^{t} (\beta - 1)\beta^{-i}.$$

Semplificando l'espressione di Ω tramite la formula per la serie geometrica si ottiene

$$\begin{aligned}\Omega &= +\beta^M \sum_{i=1}^{t} (\beta - 1)\beta^{-i} \\ &= \beta^M (\beta - 1) \sum_{i=1}^{t} (1/\beta)^i \\ &= \beta^M (\beta - 1) \left(\sum_{i=0}^{t} (1/\beta)^i - (1/\beta)^0 \right) \\ &= \beta^M (\beta - 1) \left(\frac{1 - (1/\beta)^{t+1}}{1 - 1/\beta} - 1 \right)\end{aligned}$$

Moltiplicando numeratore e denominatore della frazione per β:

$$\begin{aligned}
&= \beta^M(\beta-1)\left(\frac{\beta-(1/\beta)^t}{\beta-1}-1\right)\\
&= \beta^M(\beta-1)\cdot\frac{\beta-(1/\beta)^t-\beta+1}{\beta-1}\\
&= \beta^M\left(1-\beta^{-t}\right).
\end{aligned}$$

Esempio 1.1.6. Consideriamo l'insieme $\mathcal{F}(10,3,2,2)$: in questo sistema possiamo rappresentare numeri in base 10 con al massimo 3 cifre decimali e con un esponente compreso tra -2 e 2.

Il massimo numero rappresentabile in questo sistema è

$$\Omega = +10^2(0.999) = 10^2(1-10^{-3}).$$

OMEGA PICCOLO Chiamiamo ω il più piccolo numero *positivo* rappresentabile in un sistema. Possiamo costruirlo in questo modo:

- siccome è positivo, il segno è necessariamente $+$;
- come esponente scegliamo il più piccolo esponente possibile, ovvero $p = -m$;
- siccome le cifre decimali non possono essere tutte uguali a 0 poniamo $d_1 = 1$ e $d_i = 0$ per ogni $i > 1$.

Segue quindi che

$$\omega = +\beta^{-m}\left(1\cdot\beta^{-1}\right) = \beta^{-m}(0.1)_\beta.$$

CARDINALITÀ DELL'INSIEME DEI NUMERI DI MACCHINA Dati β, t, m, M calcoliamo $\#\mathcal{F}(\beta,t,m,M)$. Osserviamo che

1. dalla definizione di $\mathcal{F}(\beta,t,m,M)$, dobbiamo contare lo 0 separatamente;
2. per quanto riguarda i numeri non-nulli, per ogni possibile combinazione di esponente/mantissa abbiamo 2 scelte per il segno (segno positivo o negativo);
3. le scelte per i possibili esponenti corrispondono a tutti i numeri interi nell'intervallo $[-m, M]$, che sono $M+m+1$;
4. per quanto riguarda la mantissa dobbiamo scegliere t cifre comprese tra 0 e $\beta-1$ (e lo possiamo fare in β^t modi), ma dobbiamo escludere tutte quelle che iniziano per 0, che sono β^{t-1}: abbiamo quindi $\beta^t-\beta^{t-1}$ scelte per la mantissa di un numero.

In totale si ha quindi che

$$\#\mathcal{F}(\beta,t,m,M) = 1+2(M+m+1)\left(\beta^t-\beta^{t-1}\right).$$

Esempio 1.1.7. Ad esempio se $\beta=2$, $t=3$ e $m=M=1$ si ha che

$$\#\mathcal{F}(2,3,1,1) = 1+2(1+1+1)\left(2^3-2^2\right) = 25.$$

1.1.4 Standard IEEE (754)

Lo standard usato al giorno d'oggi per implementare i numeri a virgola mobile è lo standard IEEE (754). Esso definisce due rappresentazioni distinte, una *single precision*, dove ogni numero è rappresentato su 32 bit, ovvero su 4 byte, e una *double precision*, dove ogni numero è rappresentato su 64 bit (o equivalentemente 8 byte). In particolare noi analizzeremo solo il secondo.

I 64 bit vengono divisi nel seguente modo:

- un singolo bit di segno (0 per i numeri positivi, 1 per i negativi);
- 11 bit per rappresentare l'esponente;
- i restanti 52 bit vengono usati per rappresentare la mantissa.

Tuttavia possiamo ottimizzare un po' lo spazio usato facendo delle semplici osservazioni:

1. siccome siamo in base 2 la condizione $d_1 \neq 0$ ci dice che necessariamente $d_1 = 1$: ciò significa che rappresentare d_1 nella mantissa è inutile (ogni mantissa inizierebbe per 1) e quindi possiamo rappresentare le cifre da d_2 in poi. Questo significa che abbiamo in realtà $t = 53$ cifre decimali a disposizione;
2. per evitare di usare un bit per il segno dell'esponente lo rappresentiamo in *traslazione* (anche detto *bias* o *eccesso a*): invece di rappresentare il vero esponente p rappresentiamo $p + 1022$. Questo ci consente di indicare esponenti negativi senza dover usare strategie di complemento a 2.

L'insieme dei numeri di macchina definito dallo standard IEEE (754) è quindi

$$\mathcal{F}_{\text{IEEE}} \coloneqq \mathcal{F}(2, 53, 1021, 1024).$$

Osserviamo che questa scelta per i limiti degli esponenti non copre tutte le combinazioni possibili su 11 bit: in particolare possiamo rappresentare $1024 + 1021 + 1 = 2046$ esponenti diversi, mentre su 11 bit potenzialmente potremmo rappresentarne fino a $2^{11} = 2048$.

Questo ci permette di rappresentare dei numeri particolari.

ZERO Abbiamo visto che (per il Teorema 1.1.1) lo 0 non ammette una rappresentazione normalizzata. Nello standard IEEE (754) esso viene realizzato ponendo a 0 tutti i campi dell'esponente e della mantissa; tuttavia non viene specificato il segno, per cui esistono due rappresentazioni uguali per lo zero (-0 e $+0$).

Questa rappresentazione non va in conflitto con i numeri normalizzati, in quanto ponendo il campo dell'esponente uguale a undici 0 otterrei un esponente effettivo uguale a -1022, mentre il minimo rappresentabile è -1021.

NUMERI DENORMALIZZATI Se il campo dell'esponente contiene solo zeri ma la mantissa ha almeno un bit diverso da 0 stiamo rappresentando numeri *denormalizzati*. Un tale numero viene interpretato come se fosse $d_1 = 0$ e quindi ci consente di ottenere dei numeri in valore assoluto più piccoli di ω, risolvendo parzialmente i problemi di *underflow*.

Tuttavia questi numeri hanno una precisione minore dei numeri normalizzati, in quanto i primi k bit della mantissa possono essere tutti uguali a 0. Questo significa che invece di avere 53 cifre di precisione ne abbiamo $53 - k$ (dove k dipende dal numero denormalizzato scelto), dunque dobbiamo porre ancora più attenzione agli eventuali errori di precisione.

Il più grande numero denormalizzato è della forma

$$2^{-1021}(0.01\ldots1)_2 = 2^{-1022}(0.\overbrace{1\ldots1}^{52\text{ volte}})_2,$$

mentre il più piccolo numero denormalizzato è il numero

$$2^{-1021}(0.0\ldots01)_2.$$

INFINITI E NAN Ponendo il campo dell'esponente uguale a $(11111111111)_2$ (ovvero undici cifre 1) ottengo una rappresentazione che non appartiene all'insieme definito prima (in quanto il massimo esponente rappresentabile è 1024, che in eccesso a 1022 diventa $2046 = (11111111110)_2$). Sfruttando questo fatto possiamo rappresentare tre "numeri" diversi:

- se il campo della mantissa non è formato da tutti 0 il risultato è NaN (*Not a Number*): questa configurazione viene usata ad esempio per i risultati delle forme indeterminate, come $0/0$;
- se il campo della mantissa è formato da tutti 0 il numero viene interpretato come $\pm\infty$ (a seconda del bit di segno): ciò viene usato per rappresentare l'overflow, ovvero numeri che sono più grandi di Ω (o più piccoli di $-\Omega$).

1.2 ERRORI DI APPROSSIMAZIONE

Abbiamo iniziato ad introdurre il concetto di approssimazione nella sezione 1.1. Definiamolo ora formalmente.

Definizione 1.2.1 **Troncamento e arrotondamento.** Sia $x \in \mathbb{R}$, $x \neq 0$, tale che $\omega < |x| < \Omega$. Consideriamo inoltre la rappresentazione normalizzata di x:

$$x = \operatorname{sgn}(x)\beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}.$$

Allora definiamo

(1) il **troncamento** di x alla t-esima cifra, indicato con $\operatorname{trun}(x)$, dato da

$$\operatorname{trun}(x) := \operatorname{sgn}(x)\beta^p \sum_{i=1}^{t} d_i \beta^{-i};$$

(2) l'**arrotondamento** di x, indicato con $\operatorname{arr}(x)$, dato da

$$\operatorname{arr}(x) := \begin{cases} \operatorname{trun}(x), & \text{se } 0 \leqslant d_{t+1} < \beta/2 \\ \operatorname{trun}(x) + \beta^{p-t}, & \text{se } d_{t+1} \geqslant \beta/2. \end{cases}$$

Osservazione 1.2.1. Aggiungere β^{p-t} ad un numero significa aggiungere 1 alla sua t-esima cifra. Infatti (considerando lo stesso x della definizione) si ha che

$$\beta^p \sum_{i=1}^{\infty} d_i \beta^{-i} + \beta^{p-t} = \beta^p \left(\sum_{i=1}^{\infty} d_i \beta^{-i} + \beta^{-t} \right).$$

Esempio 1.2.2. Consideriamo il numero reale

$$\pi = 3.141592653\cdots = 10^1 \cdot (0.3141592653\ldots)_{10}.$$

Il suo troncamento alla quinta cifra sarà 3.1415, mentre il suo arrotondamento alla stessa cifrà (siccome la sesta cifra è $9 \geqslant 10/2$) sarà

$$3.1415 + 10^{1-5} = 3.1415 + 0.0001 = 3.1416.$$

D'ora in avanti quando parleremo dell'approssimazione di un numero reale x senza voler necessariamente specificare il metodo (troncamento o arrotondamento) usato scriveremo $\operatorname{fl}(x)$ oppure $\widetilde{x}$.

Approssimando numeri reali a numeri di macchina si genera un errore, detto **errore di rappresentazione**.

Definizione 1.2.3 **Errore di rappresentazione.** Sia $x \in \mathbb{R}$, $x \neq 0$ tale che $\omega \leqslant |x| \leqslant \Omega$. Chiameremo **errore assoluto** la quantità

$$\eta_x := \widetilde{x} - x,$$

mentre chiameremo **errore relativo** o **errore di rappresentazione** la quantità

$$\varepsilon_x := \frac{\widetilde{x} - x}{x}.$$

La qualità più importante dell'errore relativo è il fatto che è sempre **superiormente limitato**, come dimostreremo attraverso un lemma ed un teorema.

Lemma 1.2.4 **Limite superiore all'errore assoluto.** *Sia* $x \in \mathbb{R}$, $x \neq 0$, $\omega \leqslant x \leqslant \Omega$. *Sia inoltre* $x = \operatorname{sgn}(x)\beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}$ *la rappresentazione in base di* x. *Si ha che*

(i) $\left|\operatorname{trun}(x) - x\right| < \beta^{p-t}$

(ii) $\left|\operatorname{arr}(x) - x\right| \leqslant \frac{1}{2}\beta^{p-t}$.

Dimostrazione. Se fosse $x = \Omega$ oppure $x = \omega$ allora $\operatorname{trun}(x) = \operatorname{arr}(x) = x$, da cui l'errore assoluto $\widetilde{x} - x$ è nullo.

Supponiamo quindi $\omega < x < \Omega$ (il caso $-\Omega < x < -\omega$ è speculare). Siccome valgono le ipotesi del Teorema 1.1.1 possiamo scrivere

$$x = \beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}.$$

Esisteranno dunque due numeri di macchina $a, b \in \mathcal{F}$ tali che $a \leqslant x < b$ e b è il numero di macchina successivo rispetto ad a, ovvero a è il numero di macchina appena più piccolo di x e b è quello appena più grande.

Allora dovrà essere necessariamente

$$a = \beta^p \sum_{i=1}^{t} d_i \beta^{-i} = \operatorname{trun}(x),$$

mentre b dovrà essere

$$b = \beta^p \left(\sum_{i=1}^{t} d_i \beta^{-i} + \beta^{-t} \right) = a + \beta^{p-t},$$

da cui segue anche che $b - a = \beta^{p-t}$.

Si ha quindi

$$\left|\operatorname{trun}(x) - x\right| = |a - x| < |a - b| = \beta^{p-t},$$

dove il minore centrale deriva dal fatto che $x < b$.

Per quanto riguarda l'arrotondamento, osserviamo invece che

- se $d_{t+1} < \beta/2$ (ovvero se $x < 1/2(a + b)$) allora $\operatorname{arr}(x) = \operatorname{trun}(x) = a$;
- altrimenti $\operatorname{arr}(x) = \operatorname{trun}(x) + \beta^{p-t} = a + \beta^{p-t} = b$.

Nel primo caso si ha

$$\left|\operatorname{arr}(x) - x\right| = |x - a| \leqslant \left| \frac{a+b}{2} - a \right| = \frac{1}{2}|b - a| = \frac{1}{2}\beta^{p-t},$$

mentre nel secondo si ha

$$\left|\operatorname{arr}(x) - x\right| = |b - x| \leqslant \left| b - \frac{a+b}{2} \right| = \frac{1}{2}|b - a| = \frac{1}{2}\beta^{p-t},$$

da cui la tesi. □

Osservazione 1.2.2. Si ha l'uguaglianza $|\operatorname{arr} x - x| = \frac{1}{2}\beta^{p-t}$ se e solo se $x = \frac{1}{2}(a+b)$, ovvero se e solo se x è esattamente il punto medio dell'intervallo $[a, b]$.

Teorema 1.2.5 **Esistenza della precisione di macchina.** *Esiste un numero $u \in \mathbb{R}$, detto **precisione di macchina**, tale che per ogni $x \in \mathbb{R}$, $x \neq 0$, $\omega \leqslant |x| \leqslant \Omega$ vale che*

$$|\varepsilon_x| \leqslant u.$$

In particolare

(i) se $\mathrm{fl}(x) = \mathrm{trun}(x)$ si ha $u = \beta^{1-t}$;

(ii) se $\mathrm{fl}(x) = \mathrm{arr}(x)$ si ha $u = \frac{1}{2}\beta^{1-t}$.

Dimostrazione. Osserviamo che se $x = \pm\beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}$ allora varrà sicuramente che

$$|x| \geqslant \beta^p (0.10\ldots)_\beta = \beta^p \cdot (1\beta^{-1}) = \beta^{p-1}.$$

Infatti scegliendo il numero di macchina con lo stesso esponente di x ma la minima mantissa possibile otterremo necessariamente un numero più piccolo (o uguale) del valore assoluto di quello di partenza.

Consideriamo allora i due casi.

TRONCAMENTO Per il Lemma 1.2.4 sappiamo che $|\mathrm{trun}(x) - x| < \beta^{p-t}$. Si ha quindi che

$$|\varepsilon_x| = \frac{|\mathrm{trun}(x) - x|}{|x|} < \frac{\beta^{p-t}}{\beta^{p-1}} = \beta^{1-p}.$$

ARROTONDAMENTO Per il Lemma 1.2.4 sappiamo che $|\mathrm{arr}(x) - x| \leqslant \frac{1}{2}\beta^{p-t}$. Si ha quindi che

$$|\varepsilon_x| = \frac{|\mathrm{arr}(x) - x|}{|x|} \leqslant \frac{1/2\beta^{p-t}}{\beta^{p-1}} = \frac{1}{2}\beta^{1-p}.$$ □

Osserviamo che il Teorema 1.2.5 ci dice che u è un limite superiore *debole* (nel senso che $|\varepsilon_x| \leqslant u$ e non $|\varepsilon_x| < u$), tuttavia almeno nel caso del troncamento si vede dalla dimostrazione che vale anche la disuguaglianza stretta.

In realtà la disuguaglianza stretta vale anche nel caso dell'arrotondamento: la dimostrazione completa è lasciata all'appendice (in particolare nella sezione A.2).

1.3 ARITMETICA DI MACCHINA

Cerchiamo ora di estendere l'aritmetica tra numeri reali ad un'aritmetica per i numeri di macchina. Il primo tentativo è quello di usare semplicemente le solite operazioni definite tra numeri reali, tuttavia non è detto che il risultato di un'operazione tra numeri di macchina sia ancora un numero di macchina.

Esempio 1.3.1. Consideriamo il sistema $\mathcal{F}(10, 3, -5, 5)$ e i due numeri di macchina

$$a := 123 = 10^3(0.123)_{10}, \qquad b := 0.456 = 10^0(0.456).$$

Si ha che $a,\ b \in \mathcal{F}(10, 3, -5, 5)$, tuttavia $a + b = 123.456 = 10^3(0.123456)_{10}$ non è un numero di macchina poiché è rappresentato con $t = 6$ cifre di precisione.

Dobbiamo quindi definire delle nuove operazioni che siano *interne* all'insieme dei numeri di macchina.

Definizione 1.3.2 **Aritmetica di macchina.** Si definiscono quattro operazioni

$$\oplus,\ \otimes,\ \ominus,\ \oslash : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$$

tali che per ogni coppia di numeri di macchina $a, b \in \mathcal{F}$ ($b \neq 0$ nel caso di $\oslash$) si ha che

$$a \oplus b \coloneqq \mathrm{fl}(a+b) \qquad a \ominus b \coloneqq \mathrm{fl}(a-b)$$
$$a \otimes b \coloneqq \mathrm{fl}(ab) \qquad a \oslash b \coloneqq \mathrm{fl}(a/b)$$

Siamo ora interessati a studiare *quanto errore* commettiamo nel considerare i numeri di macchina invece dei numeri reali non approssimati.

Definizione 1.3.3 **Errore locale.** Sia $*$ un'operazione tra numeri qualsiasi e sia $\circledast$ la corrispondente operazione su numeri di macchina. Dati $x, y \in \mathcal{F}$ si dice **errore locale** dell'operazione la quantità

$$\varepsilon \coloneqq \frac{(x \circledast y) - (x * y)}{x * y}.$$

Osserviamo che ogni operazione di macchina può essere scritta in termini della sua corrispondente operazione base e dell'errore locale: in effetti

$$x \circledast y = (x * y)(1+\varepsilon).$$

Esempio 1.3.4 (Errore nel calcolo della somma). Cerchiamo ora di calcolare l'errore totale commesso nell'approssimazione di un'operazione tra numeri reali (definiremo più avanti precisamente il concetto di errore totale). Prendiamo $a, b \in \mathbb{R}$ che approssimati non vadano in overflow (o underflow) e cerchiamo di calcolare $a+b$ in termini di numeri di macchina.

Siccome i numeri $a,\ b$ non sono necessariamente già numeri di macchina, l'unico modo per svolgere il calcolo è calcolare la quantità

$$\widetilde{a} \oplus \widetilde{b} = (\widetilde{a} + \widetilde{b})(1+\varepsilon).$$

Siccome sappiamo che, se $\varepsilon_a,\ \varepsilon_b$ sono gli errori di rappresentazione rispettivamente di a e b

$$\widetilde{a} = a(1+\varepsilon_a), \quad \widetilde{b} = b(1+\varepsilon_b)$$

otteniamo che

$$\begin{aligned}\widetilde{a} \oplus \widetilde{b} &= (a(1+\varepsilon_a) + b(1+\varepsilon_b))(1+\varepsilon) \\ &= a(1+\varepsilon_a)(1+\varepsilon) + b(1+\varepsilon_b)(1+\varepsilon).\end{aligned}$$

Nello svolgere il prossimo passaggio trascureremo gli addendi in cui compare il prodotto di due errori: effettueremo quindi un'**analisi al primo ordine**, indicata dal simbolo $\doteq$.

$$\doteq a + b + a\varepsilon_a + b\varepsilon_b + (a+b)\varepsilon.$$

L'errore complessivo che commettiamo è quindi dato dalla differenza tra il risultato ottenuto ($\widetilde{a} \oplus \widetilde{b}$) e il risultato non approssimato ($a+b$), dividendo per il risultato non approssimato in modo da ottenere un errore relativo:

$$\begin{aligned}\frac{(\widetilde{a} \oplus \widetilde{b}) - (a+b)}{a+b} &\doteq \frac{a+b+a\varepsilon_a + b\varepsilon_b + (a+b)\varepsilon - (a+b)}{a+b} \\ &= \frac{a}{a+b}\varepsilon_a + \frac{b}{a+b}\varepsilon_b + \varepsilon.\end{aligned}$$

Notiamo che questo errore può essere diviso in due parti diverse:

- la parte data da $\frac{a}{a+b}\varepsilon_a + \frac{b}{a+b}\varepsilon_b$ non dipende dal tipo di operazione che sto cercando di svolgere, ma dal fatto che devo approssimare gli argomenti dell'operazione: lo chiameremo **errore inerente**;

- la parte data da ε dipende dalla sequenza di passi scelta per calcolare la somma: lo chiameremo perciò **errore algoritmico**.

Esempio 1.3.5 (Errore di una funzione). Consideriamo la funzione $f : \mathbb{R} \to \mathbb{R}$ definita da

$$f(x) = x^2 - 1.$$

Calcoliamo l'errore totale commesso nel calcolare $f(x)$ tramite operazioni di macchina.

Osserviamo che potenzialmente abbiamo più *algoritmi* (intesi come sequenze di calcoli) per calcolare f: scegliamo ad esempio gli algoritmi di macchina g_1, g_2 definiti da

$$g_1(\widetilde{x}) \coloneqq (\widetilde{x} \otimes \widetilde{x}) \ominus 1, \qquad g_2(\widetilde{x}) \coloneqq (\widetilde{x} \ominus 1) \otimes (\widetilde{x} \oplus 1).$$

L'algoritmo g_1 prescrive di calcolare innanzitutto $z \coloneqq \widetilde{x} \otimes \widetilde{x}$ e dopo calcolare $z \ominus 1$. Se ε_1 è l'errore locale dell'operazione $\otimes$ si ha che

$$z = \widetilde{x}^2(1 + \varepsilon_1);$$

chiamando ε_2 l'errore locale dell'operazione $\ominus$ si ottiene

$$g_1(\widetilde{x}) = (z - 1)(1 + \varepsilon_2).$$

A questo punto, ricordando che $\widetilde{x} = x(1 + \varepsilon_x)$ dove ε_x è l'errore di rappresentazione, si può esprimere $g_1(\widetilde{x})$ in termini di x e dei vari errori. In effetti

$$\begin{aligned}
&(z - 1)(1 + \varepsilon_2) \\
&= \big(\widetilde{x}^2(1 + \varepsilon_1) - 1\big)(1 + \varepsilon_2) \\
&= \Big(\big(x(1 + \varepsilon_x)\big)^2(1 + \varepsilon_1) - 1\Big)(1 + \varepsilon_2) \\
&= \big(x^2(1 + \varepsilon_x)^2(1 + \varepsilon_1) - 1\big)(1 + \varepsilon_2) \\
&\doteq \big(x^2(1 + 2\varepsilon_x)(1 + \varepsilon_1) - 1\big)(1 + \varepsilon_2) \\
&\doteq \big(x^2(1 + 2\varepsilon_x + \varepsilon_1) - 1\big)(1 + \varepsilon_2) \\
&= x^2(1 + 2\varepsilon_x + \varepsilon_1)(1 + \varepsilon_2) - (1 + \varepsilon_2) \\
&\doteq x^2(1 + 2\varepsilon_x + \varepsilon_1 + \varepsilon_2) - (1 + \varepsilon_2) \\
&= x^2 - 1 + 2x^2\varepsilon_x + x^2\varepsilon_1 + (x^2 - 1)\varepsilon_2.
\end{aligned}$$

L'errore totale commesso nell'approssimare il risultato della funzione $f(x)$ con la funzione di macchina $g_1(\widetilde{x})$ è dunque

$$\begin{aligned}
\varepsilon_{g_1} &\coloneqq \frac{x^2 - 1 + 2x^2\varepsilon_x + x^2\varepsilon_1 + (x^2 - 1)\varepsilon_2 - (x^2 - 1)}{x^2 - 1} \\
&= \frac{2x^2}{x^2 - 1}\varepsilon_x + \frac{x^2}{x^2 - 1}\varepsilon_1 + \varepsilon_2.
\end{aligned}$$

Ripetiamo il procedimento per l'algoritmo g_2: siano δ_1, δ_2 e δ_3 i tre errori locali commessi; ovvero

$$\begin{aligned}
z_1 &\coloneqq \widetilde{x} \ominus 1 = (\widetilde{x} - 1)(1 + \delta_1) \\
z_2 &\coloneqq \widetilde{x} \oplus 1 = (\widetilde{x} + 1)(1 + \delta_2) \\
g_2(\widetilde{x}) &= z_1 \otimes z_2 = z_1 z_2 (1 + \delta_3).
\end{aligned}$$

Si ha quindi

$$\begin{aligned}
& z_1 z_2 (1+\delta_3) \\
&= (\widetilde{x}-1)(1+\delta_1)(\widetilde{x}+1)(1+\delta_2)(1+\delta_3) \\
&= (\widetilde{x}^2-1)(1+\delta_1)(1+\delta_2)(1+\delta_3) \\
&= \big(x^2(1+\varepsilon_x)^2-1\big)(1+\delta_1)(1+\delta_2)(1+\delta_3) \\
&\doteq \big(x^2(1+2\varepsilon_x)-1\big)(1+\delta_1)(1+\delta_2)(1+\delta_3) \\
&= \big(x^2(1+2\varepsilon_x)(1+\delta_1)-(1+\delta_1)\big)(1+\delta_2)(1+\delta_3) \\
&\doteq \big(x^2(1+2\varepsilon_x+\delta_1)-(1+\delta_1)\big)(1+\delta_2)(1+\delta_3) \\
&= \big(x^2(1+2\varepsilon_x+\delta_1)(1+\delta_2)-(1+\delta_1)(1+\delta_2)\big)(1+\delta_3) \\
&\doteq \big(x^2(1+2\varepsilon_x+\delta_1+\delta_2)-(1+\delta_1+\delta_2)\big)(1+\delta_3) \\
&= x^2(1+2\varepsilon_x+\delta_1+\delta_2)(1+\delta_3)-(1+\delta_1+\delta_2)(1+\delta_3) \\
&\doteq x^2(1+2\varepsilon_x+\delta_1+\delta_2+\delta_3)-(1+\delta_1+\delta_2+\delta_3) \\
&= x^2+2x^2\varepsilon_x+x^2(\delta_1+\delta_2+\delta_3)-1-(\delta_1+\delta_2+\delta_3) \\
&= x^2-1+2x^2\varepsilon_x+(x^2-1)(\delta_1+\delta_2+\delta_3).
\end{aligned}$$

L'errore totale generato dalla funzione g_2 è dunque

$$\begin{aligned}
\varepsilon_{g_2} &\coloneqq \frac{x^2-1+2x^2\varepsilon_x+(x^2-1)(\delta_1+\delta_2+\delta_3)-(x^2-1)}{x^2-1} \\
&= \frac{2x^2}{x^2-1}\varepsilon_x+\delta_1+\delta_2+\delta_3.
\end{aligned}$$

Osserviamo che entrambe le funzioni hanno lo stesso errore inerente (dato dal termine contenente ε_x): in effetti l'errore inerente deriva dall'approssimazione di x in $\widetilde{x}$, e ciò non dipende dall'algoritmo scelto.

Il resto del termine è l'errore algoritmico: siccome abbiamo due algoritmi diversi possiamo chiederci quale dei due sia migliore.

Osserviamo che il secondo non dipende da x, mentre il primo sì: in particolare l'errore algoritmico del primo tende ad infinito quando x tende a ± 1. Si dice quindi che l'algoritmo non è **stabile** per $x \to \pm 1$, mentre il secondo algoritmo è stabile per ogni $x \in \mathbb{R}$.

1.4 TEORIA DEGLI ERRORI

Diamo ora una definizione precisa dei vari tipi di errori.

Consideriamo nel resto della sezione

- una funzione razionale $f : A \to \mathbb{R}$ con $A \subseteq \mathbb{R}$, dove per *razionale* si intende che compaiono solo le operazioni di somma, prodotto, sottrazione e divisione
- un numero reale x qualunque e la sua approssimazione $\widetilde{x}$
- un algoritmo $g : \mathcal{F} \to \mathcal{F}$ per calcolare f, dove per *algoritmo* si intende una sequenza di operazioni tra numeri di macchina usata per calcolare il valore di f.

1.4.1 Errore inerente

Definizione 1.4.1

Errore inerente. Si dice **errore inerente** del problema la quantità

$$\varepsilon_{in} \coloneqq \frac{f(\widetilde{x})-f(x)}{f(x)}.$$

Osserviamo che l'errore inerente misura quanto è grande l'errore nell'approssimare x con il numero di macchina $\widetilde{x}$: se l'errore inerente è grande in un intorno di x nessun algoritmo ci consentirà di calcolare con una buona precisione il risultato della funzione f. In tal caso diremo che il problema è **mal condizionato**, mentre se l'errore inerente è sempre limitato il problema verrà definito **ben condizionato**.

Per studiare il condizionamento di un problema è quindi necessario studiare il suo errore inerente: vogliamo quindi una tecnica per calcolarlo più efficientemente.

Osserviamo che moltiplicando e dividendo per x e per $\widetilde{x} - x$ si ha

$$\varepsilon_{\text{in}} = \frac{f(\widetilde{x}) - f(x)}{f(x)} = \frac{f(\widetilde{x}) - f(x)}{\widetilde{x} - x} \cdot \frac{x}{f(x)} \cdot \underbrace{\frac{\widetilde{x} - x}{x}}_{\varepsilon_x}.$$

Il primo termine del prodotto ricorda un rapporto incrementale: se la funzione è derivabile in x possiamo semplificare l'espressione usando il Teorema di Taylor.

Supponiamo quindi che $f \in \mathcal{C}^2[a, b]$ (ovvero che f sia continua e derivabile due volte in un intervallo $[a, b]$): per il Teorema di Taylor con il resto di Lagrange si ha che esiste un numero ξ_x compreso tra x e $\widetilde{x}$ tale che

$$f(\widetilde{x}) = f(x) + f'(x)(\widetilde{x} - x) + f''(\xi_x)\frac{(\widetilde{x} - x)^2}{2!}$$

con $|\varepsilon_x - x| < |\widetilde{x} - x|$.

Sostituendo otteniamo

$$\begin{aligned}
\varepsilon_{\text{in}} &= \frac{f(\widetilde{x}) - f(x)}{\widetilde{x} - x} \cdot \frac{x}{f(x)} \cdot \varepsilon_x \\
&= \frac{f'(x)(\widetilde{x} - x) + f''(\xi_x)\dfrac{(\widetilde{x} - x)^2}{2!}}{\widetilde{x} - x} \cdot \frac{x}{f(x)} \cdot \varepsilon_x \\
&= \left(f'(x) + f''(\xi_x)\frac{\widetilde{x} - x}{2!} \right) \cdot \frac{x}{f(x)} \cdot \varepsilon_x \\
&= f'(x)\frac{x}{f(x)}\varepsilon_x + f''(\xi_x)\frac{\widetilde{x} - x}{2!}\frac{x}{f(x)} \cdot \varepsilon_x.
\end{aligned}$$

Osserviamo ora che il secondo addendo contiene (nascosto) un errore al quadrato, e pertanto in un analisi al prim'ordine viene approssimato a 0. In effetti moltiplicando e dividendo per x si ha

$$\begin{aligned}
& f''(\xi_x)\frac{\widetilde{x} - x}{2!}\frac{x}{f(x)} \cdot \varepsilon_x \\
&= f''(\xi_x)\frac{\widetilde{x} - x}{2!}\frac{x}{f(x)} \cdot \frac{x}{x} \cdot \varepsilon_x \\
&= f''(\xi_x)\frac{x^2}{2!f(x)} \cdot \frac{\widetilde{x} - x}{x} \cdot \varepsilon_x \\
&= f''(\xi_x)\frac{x^2}{2!f(x)} \cdot \varepsilon_x^2 \\
&\doteq 0.
\end{aligned}$$

Abbiamo quindi dimostrato il seguente Teorema.

Teorema 1.4.2 **Caratterizzazione dell'errore inerente.** *Sia* $f : A \subseteq \mathbb{R} \to \mathbb{R}$ *e sia* x *il punto di cui vogliamo calcolare l'immagine* $f(x)$*. Se* $f \in \mathcal{C}^2(A)$ *si ha che*

$$\varepsilon_{in} \doteq f'(x)\frac{x}{f(x)} \cdot \varepsilon_x,$$

dove ε_x *è l'errore di rappresentazione di* x*.*

La quantità $c_x \coloneqq f'(x)\frac{x}{f(x)}$ *viene detta* ***coefficiente di amplificazione****.*

Abbiamo quindi scoperto che l'errore inerente è sempre della forma $c_x\varepsilon_x$: possiamo quindi definire formalmente il concetto di condizionamento.

Definizione 1.4.3 **Condizionamento di un problema.** Sia $f : A \subseteq \mathbb{R} \to \mathbb{R}$ con $f \in \mathcal{C}^2(A)$. Se esiste $x_0 \in A$ tale che

$$\lim_{x \to x_0} |c_x| = +\infty$$

si dice che il problema è **mal condizionato** in un intorno di x_0, altrimenti si dice che il problema è **ben condizionato**.

Esempio 1.4.4. Riprendiamo l'Esempio 1.3.5: il coefficiente di amplificazione è

$$c_x = \frac{2x^2}{x^2 - 1}$$

il cui valore assoluto tende a $+\infty$ per $x \to \pm 1$. Si ha quindi che il problema è mal condizionato in un intorno di 1 e in un intorno di -1.

Il Teorema 1.4.2 può essere generalizzato per funzioni $\mathbb{R}^n \to \mathbb{R}$.

Corollario 1.4.5 **Errore inerente su funzioni multivariate.** *Sia* $f : \Omega \to \mathbb{R}$ *con* $\Omega \subseteq \mathbb{R}^n$ *aperto e* f *differenziabile due volte su* Ω *(ovvero esistono la derivata parziale prima e seconda rispetto ad ogni variabile).*

Sia inoltre $\mathbf{x} = (x_1, \ldots, x_n)$ *il punto di cui vogliamo calcolare* $f(\mathbf{x})$*. Si ha che*

$$\varepsilon_{in} = \sum_{i=1}^{n} c_{x_i}\varepsilon_{x_i}$$

dove $c_{x_i} \coloneqq \dfrac{x_i}{f(x)} \cdot \dfrac{\partial f}{\partial x_i}$ *è il* ***coefficiente di amplificazione****.*

Coefficienti di amplificazione delle operazioni elementari

Il Corollario 1.4.5 può essere molto comodo per calcolare i coefficienti di amplificazione delle operazioni elementari: ciò torna molto utile per il calcolo dell'errore totale.

SOMMA Consideriamo $f(x, y) = x + y$. Si ha che

$$c_x = \frac{x}{x+y} \cdot 1 = \frac{x}{x+y}, \qquad c_y = \frac{y}{x+y} \cdot 1 = \frac{y}{x+y}.$$

SOTTRAZIONE Consideriamo $f(x, y) = x - y$. Si ha che

$$c_x = \frac{x}{x-y} \cdot 1 = \frac{x}{x-y}, \qquad c_y = \frac{y}{x+y} \cdot (-1) = -\frac{y}{x-y}.$$

PRODOTTO Consideriamo $f(x, y) = xy$. Si ha che

$$c_x = \frac{x}{xy} \cdot y = 1, \qquad c_y = \frac{y}{xy} \cdot x = 1.$$

DIVISIONE Consideriamo $f(x,y) = \frac{x}{y}$. Si ha che

$$c_x = \frac{x}{x/y} \cdot \frac{1}{y} = 1, \qquad c_y = \frac{y}{x/y} \cdot \left(-\frac{x}{y^2}\right) = -1.$$

1.4.2 Errore algoritmico

Definizione 1.4.6

Errore algoritmico. Si dice **errore algoritmico** la quantità

$$\varepsilon_{alg} := \frac{g(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})}.$$

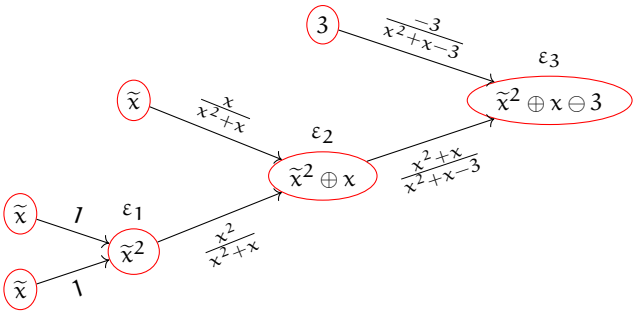
L'errore algoritmico misura quanto è grande calcolare il risultato attraverso l'algoritmo di macchina g invece della funzione non approssimata f *partendo da dati già approssimati*.

Per studiare l'errore algoritmico si usa la tecnica conosciuta come *errore in avanti* (*forward error* in inglese), che può essere resa più compatta attraverso l'uso di grafi.

Esempio 1.4.7. Calcoliamo l'errore algoritmico della funzione $f(x) = x^2 + x - 3$ attraverso l'algoritmo di macchina

$$g(\tilde{x}) = \left((\tilde{x} \otimes \tilde{x}) \oplus \tilde{x}\right) \ominus 3.$$

Per far ciò, costruiamo il seguente grafo.



Il grafo è costruito nel seguente modo:

- si considera la sequenza di operazioni da svolgere: in questo caso
  1. $z_1 := \tilde{x} \otimes \tilde{x}$
  2. $z_2 := z_1 \oplus \tilde{x}$
  3. $z_3 := z_2 \ominus 3 = g(\tilde{x})$
- si disegnano i nodi iniziali, ovvero quelli della prima operazione da svolgere, e sopra di essi si mettono gli errori commessi nel rappresentarli: siccome siamo interessati all'errore algoritmico partiamo già con numeri di macchina, quindi l'errore di rappresentazione è 0 e lo omettiamo;
- si crea un nodo corrispondente a z_1: l'errore commesso nella creazione di questo nodo è l'errore locale del prodotto (qui chiamato ε_1);
- sopra gli archi che portano i nodi $\tilde{x}$ in z_1 indichiamo il coefficiente di amplificazione c_x dei vari nodi: in questo caso sono entrambi uguali a 1 poiché l'operazione è un prodotto;
- ripetiamo il procedimento: creiamo un nuovo nodo con $\tilde{x}$ e "errore di creazione" uguale a 0 e lo usiamo per fare l'operazione di somma, creando un nuovo nodo (che corrisponde a z_2) con errore di creazione uguale all'errore locale della somma ε_2;
- per ricavare i coefficienti di amplificazione sfruttiamo le formule ricavate prima, che ci dicono che nella somma $\tilde{x}^2 \oplus \tilde{x}$ i coefficienti di amplificazione sono dati da

$$c_{x^2} = \frac{x^2}{x^2 + x}, \qquad c_x = \frac{x}{x^2 + x};$$

- ripetiamo un'ultima volta il ragionamento per quanto riguarda l'ultima operazione.

Possiamo ora calcolare l'errore totale a partire dal grafo: si parte dalla fine del grafo (nodo più a destra) e si calcola la somma tra l'errore del nodo (in questo caso ε_3) e gli errori dei sottoalberi (calcolati ricorsivamente), moltiplicati per il loro coefficiente di amplificazione.

Otteniamo quindi

$$\begin{aligned}
\varepsilon_{\text{alg}} &= \varepsilon_3 + \frac{x^2+x}{x^2+x-3}(\dots) + \frac{-3}{x^2+x-3}\cdot 0 \\
&= \varepsilon_3 + \frac{x^2+x}{x^2+x-3}\left(\varepsilon_2 + \frac{x^2}{x^2+x}(\dots) + \frac{x}{x^2+x}\cdot 0\right) \\
&= \varepsilon_3 + \frac{x^2+x}{x^2+x-3}\left(\varepsilon_2 + \frac{x^2}{x^2+x}(\varepsilon_1 + 1\cdot 0 + 1\cdot 0)\right) \\
&= \varepsilon_3 + \frac{x^2+x}{x^2+x-3}\left(\varepsilon_2 + \frac{x^2}{x^2+x}\cdot \varepsilon_1\right).
\end{aligned}$$

1.4.3 Errore totale

Definizione 1.4.8

Errore totale. Si dice **errore totale** del problema la quantità

$$\varepsilon_{\text{tot}} \coloneqq \frac{g(\widetilde{x}) - f(x)}{f(x)}.$$

Osserviamo chel'errore totale combina i concetti di errore inerente e algoritmico e ci dice quanto è grande l'errore nel calcolare g con il valore di $\widetilde{x}$ rispetto al calcolo esatto di $f(x)$.

Nell'Esempio 1.3.5 abbiamo notato come l'errore totale fosse dato dalla somma dell'errore algoritmico e dell'errore inerente: questa relazione vale sempre, come dimostrato dal prossimo teorema.

Teorema 1.4.9

$\varepsilon_{tot} \doteq \varepsilon_{in} + \varepsilon_{alg}$.

Dimostrazione. In effetti si ha

$$\begin{aligned}
\varepsilon_{\text{tot}} &= \frac{g(\widetilde{x}) - f(x)}{f(x)} \\
&= \frac{g(\widetilde{x}) - f(\widetilde{x}) + f(\widetilde{x}) - f(x)}{f(x)} \\
&= \frac{g(\widetilde{x}) - f(\widetilde{x})}{f(x)} + \frac{f(\widetilde{x}) - f(x)}{f(x)} \\
&= \frac{g(\widetilde{x}) - f(\widetilde{x})}{f(\widetilde{x})}\frac{f(\widetilde{x})}{f(x)} + \frac{f(\widetilde{x}) - f(x)}{f(x)} \\
&= \varepsilon_{\text{alg}}\cdot\frac{f(\widetilde{x})}{f(x)} + \varepsilon_{\text{in}} \\
&= \varepsilon_{\text{alg}}\cdot\frac{f(\widetilde{x}) - f(x) + f(x)}{f(x)} + \varepsilon_{\text{in}} \\
&= \varepsilon_{\text{alg}}\cdot\left(\frac{f(\widetilde{x}) - f(x)}{f(x)} + \frac{f(x)}{f(x)}\right) + \varepsilon_{\text{in}} \\
&= \varepsilon_{\text{alg}}\cdot(\varepsilon_{\text{in}} + 1) + \varepsilon_{\text{in}} \\
&\doteq \varepsilon_{\text{alg}} + \varepsilon_{\text{in}}.
\end{aligned}$$

□

A | APPENDICE A

A.1 "9-PERIODICO"

Proposizione A.1.1 **9-periodico.** *In base* 10 *i numeri* $0.\bar{9}$ *e* 1 *sono uguali.*

Forniamo due diverse dimostrazioni di questa proposizione.

Prima dimostrazione. Dalle formule per trasformare i numeri periodici in frazioni sappiamo che $0.\bar{9} = 9/9 = 1$. □

Seconda dimostrazione. Espandendo la definizione di numero periodico otteniamo che

$$0.\bar{9} = 0.999\cdots = 9\cdot 10^{-1} + 9\cdot 10^{-2} + 9\cdot 10^{-3} + \cdots = \sum_{i=1}^{\infty} 9\cdot 10^{-i}.$$

Sfruttando la formula della serie geometrica si ottiene che

$$\begin{aligned}
\sum_{i=1}^{\infty} 9\cdot 10^{-i} &= 9\cdot \sum_{i=1}^{\infty} \left(\frac{1}{10}\right)^{i} \\
&= 9\cdot \left(\left(\sum_{i=0}^{\infty} \left(\frac{1}{10}\right)^{i}\right) - \left(\frac{1}{10}\right)^{0}\right) \\
&= 9\cdot \left(\frac{1}{1-1/10} - 1\right) \\
&= 9\cdot \left(\frac{10}{9} - 1\right) \\
&= 9\cdot \frac{1}{9} \\
&= 1.
\end{aligned}$$

□

La proposizione vale in generale in una base β qualsiasi ($\beta \geqslant 2$).

Proposizione A.1.2 *In base* β *(*$\beta \geqslant 2$*) vale che* $0.\overline{(\beta-1)} = 1$.

Dimostrazione. La dimostrazione è uguale alla seconda dimostrazione della Proposizione A.1.1. □

A.2 LA PRECISIONE DI MACCHINA È UN LIMITE SUPERIORE STRETTO

In questa sezione mostriamo che la precisione di macchina $\mathbf{u}$ è sempre strettamente maggiore all'errore relativo ε_x per ogni $x \in \mathbb{R}$ tale che $x \neq 0$ e $\omega \leqslant |x| \leqslant \Omega$, anche usando l'arrotondamento come metodo di approssimazione.

Dalla dimostrazione del Teorema 1.2.5 si ha che

$$|\varepsilon_x| = \frac{1}{2}\beta^{1-p} = \mathbf{u}$$

se e solo se la disuguaglianza

$$\frac{\left|\mathrm{arr}(x) - x\right|}{|x|} \leqslant \frac{1/2\beta^{p-t}}{\beta^{p-1}}$$

è in realtà un'uguaglianza, ovvero se e solo se valgono le seguenti due condizioni:

- $\left|\mathrm{arr}(x) - x\right| = \frac{1}{2}\beta^{p-t}$
- $|x| = \beta^{p-1}$.

Tuttavia, come abbiamo notato nella Osservazione 1.2.2, la prima condizione vale se e solo se

$$|x| = \frac{1}{2}(a+b) = \frac{1}{2}(a+a+\beta^{p-t}) = a + \frac{1}{2}\beta^{p-t}.$$

È quindi necessario che

$$\begin{aligned}
\beta^{p-1} &= a + \frac{1}{2}\beta^{p-t} \\
&= \beta^{p}\left(\sum_{i=1}^{t} d_i\beta^{-i}\right) + \frac{1}{2}\beta^{p-t} \\
&= \beta^{p}\left(\sum_{i=1}^{t} d_i\beta^{-i} + \frac{1}{2}\beta^{-t}\right) \\
\iff \beta^{-1} &= \sum_{i=1}^{t} d_i\beta^{-i} + \frac{1}{2}\beta^{-t}.
\end{aligned}$$

Tuttavia ciò è assurdo: infatti il termine al primo membro ha una singola cifra decimale diversa da zero, ovvero quella di β^{-1}, mentre il termine del secondo membro ha anche delle cifre non nulle nelle posizioni β^{-t} e/o β^{-t-1}.

Segue quindi che $|\varepsilon_x| < u$ in qualsiasi caso.