

CALCUL SCIENTIFIQUE POUR INGÉNIEURS
Notes de cours complémentaires

Représentation des nombres sur un ordinateur

L'étudiant doit comprendre qu'une machine calcule vite mais faux, très faux parfois. Il devra remettre en cause toute croyance maladroite à la qualité des résultats issus de sa calculatrice ou son ordinateur.

Dans le système décimal, nous avons l'habitude de représenter un nombre réel sous la forme:

$$x = \pm m \times 10^l,$$

où m est la mantisse, l est l'exposant (entier signé) et 10 est la base. De façon générale, selon une base quelconque, on peut écrire

$$x = \pm m \times b^l,$$

où b est la base utilisée. Par exemple $b = 2$ (binaire), $b = 10$ (décimale) et $b = 16$ (hexadécimale).

La forme générale de la mantisse est la suivante:

$$\begin{aligned} m &= d_0, d_1 d_2 d_3 d_4 \dots d_n \dots \\ m &= d_0 + d_1 \times b^{-1} + d_2 \times b^{-2} + d_3 \times b^{-3} + \dots + d_n \times b^{-n} + \dots \end{aligned}$$

Les d_i vérifient:

$$0 \leq d_i \leq b - 1 \text{ pour } i = 0, 1, 2, \dots$$

Notation flottante

Le système numérique d'un calculateur (ordinateur ou calculatrice de poche) est discret et dépend des contraintes comme l'espace mémoire pour représenter les nombres et la base utilisée. Lorsqu'un nombre ne peut pas être représenté exactement avec le nombre de chiffres (ou de bits) prévus dans la mantisse, on recourt à la troncature ou l'arrondi pour réduire la mantisse à n chiffres (bits si $b = 2$) après la virgule. La notation flottante ou la représentation en point flottant ou encore en virgule flottante est alors donnée par

$$\text{fl}(x) = \pm d_0^* d_1^* d_2^* d_3^* d_4^* \dots d_n^* \times b^{l^*},$$

avec $l^* \in [l_{\min}, l_{\max}]$ où n , b , l_{\min} et l_{\max} sont des caractéristiques du calculateur.

Exemples:

1. Les calculatrices de poche se distinguent des ordinateurs principalement par le fait qu'elles utilisent la base 10 ($b = 10$) et une mantisse d'environ 10 ($n = 10$). L'exposant l varie généralement entre $l_{min} = -100$ et $l_{max} = 100$. Les chiffres d_i de la mantisse vérifient:

$$\begin{cases} d_0 = 0; \\ 1 \leq d_1 \leq 9; \\ 0 \leq d_i \leq 9 \text{ pour } i = 2, 3, \dots, n. \end{cases}$$

La première inégalité signifie que la mantisse est normalisée, c'est-à-dire que le chiffre d_1 est toujours différent de 0. La normalisation permet d'assurer l'unicité de la représentation.

2. Le système binaire ($b = 2$) est à la base de la représentation des nombres sur la vaste majorité des ordinateurs. On utilise en général la norme IEEE-754. On y propose une représentation des nombres réels en simple précision avec $n = 23$ bits, $l_{min} = -126$, $l_{max} = 127$ et en double précision avec $n = 52$ bits, $l_{min} = -1022$ et $l_{max} = 1023$. Les bits de la mantisse vérifient:

$$\begin{cases} d_0 = 1; \\ 0 \leq d_i \leq 1 \text{ pour } i = 1, 2, 3, \dots, n. \end{cases}$$

Le premier bit d_0 est toujours 1 et il n'est pas nécessaire de le stocker en mémoire.

Erreurs dues à la représentation

Le fait d'utiliser un nombre limité de chiffres (ou de bits) pour représenter un nombre réel a des conséquences importantes.

- a) Quel que soit le nombre de chiffres (ou de bits) utilisés pour la mantisse, il existe un plus petit et un plus grand nombres positifs représentables (de même pour les nombres négatifs). En dehors de cette intervalle, on a un message de sous-dépassement (underflow) et de débordement (overflow). Par exemple, selon la norme IEEE double précision, on a $x_{min} = 2,2 \times 10^{-308}$ et $x_{max} = 1,8 \times 10^{308}$.
- b) À l'intérieur de cet intervalle fini, seulement un nombre fini de nombres sont représentables exactement et l'on doit recourir à la troncature ou à l'arrondi pour représenter les autres réels. Par exemple, selon la norme IEEE double précision, il y a 2^{52} nombres représentables exactement entre 1 et 2.

En général, l'erreur introduite par la notation flottante est dite erreur d'arrondi ou de représentation. Remarquons ici que le choix de la base b influence la précision de la

représentation. Ainsi le nombre $\frac{1}{10}$ sera représenté exactement sur une calculatrice mais pas sur un ordinateur qui travaille en binaire. La représentation en virgule flottante induit une erreur relative qui dépend du nombre de chiffres (ou de bits) de la mantisse, de l'utilisation de l'arrondi ou de la troncature ainsi que du nombre x que l'on veut représenter.

Définition: Précision machine

La précision machine ϵ_m est la plus grande erreur relative que l'on puisse commettre en représentant un nombre réel sur un ordinateur en utilisant **la troncature**.

$$\left| \frac{\text{fl}(x) - x}{x} \right| \leq \epsilon_m.$$

Remarque:

Si l'on utilise l'arrondi, la plus grande erreur relative que l'on puisse commettre est alors $\frac{\epsilon_m}{2}$ et on a dans ce cas

$$\left| \frac{\text{fl}(x) - x}{x} \right| \leq \frac{\epsilon_m}{2}.$$

Théorème: Borne pour la précision machine

La précision machine vérifie:

$$\epsilon_m \leq b^{1-n},$$

où b est la base utilisée et n le nombre de chiffres (ou de bits si $b = 2$) de la mantisse.

Exemple: Suivant la norme IEEE-754, la mantisse normalisée d'un réel contient 23 bits en simple précision (52 bits en double précision), mais avec une précision de 24 bits en simple précision (53 bits en double précision) puisque, après la normalisation, le premier bit $d_0 = 1$ n'est pas gardé en mémoire. La précision machine vaut alors

$$\epsilon_m = 2^{1-24} \simeq 0,119 \times 10^{-6}$$

en simple précision et

$$\epsilon_m = 2^{1-53} \simeq 0,222 \times 10^{-15}$$

en double précision.

Autres définitions:

1. La précision machine est la distance entre le nombre 1 et le prochain nombre représentable exactement suivant la norme IEEE-754.
2. La précision machine selon la norme IEEE-754 est le plus grand nombre positif x tel que

$$\text{fl}\left(1 + \frac{x}{2}\right) = 1.$$

On en déduit de ce résultat que

$$\text{fl}(1 + x) = 1 \text{ pour } x \leq \frac{\epsilon_m}{2}.$$

Exemple d'application sur Matlab:

```
x= 0;
for i = 1:10000
x= x+0.1;
end
y = 1.0e10*(x-1000);
```

On obtient $y = 1,58820$, ce résultat est erroné. Ce résultat est dû au fait que le nombre décimal $\frac{1}{10}$ possède un développement binaire illimité.

$$\left(\frac{1}{10}\right)_{10} = (0,1)_{10} = (0,11001100 \cdots \times 2^{-3})_2 = (1,1001100 \cdots \times 2^{-2})_2.$$