Chapitre 1

Arithmétique flottante

1.1 Mini cours sur l'arithmétique flottante

1.1.1 Introduction

La représentation en virgule flottante est naturelle lorsque l'on veut écrire un nombre très grand ou très petit en valeur absolue, par exemple :

$A = 6,02252\ 10^{23}$ le nombre d'Avogadro
$h = 6,625\ 10^{-34}$ [*joule seconde*] la constante de Planck ($\hbar = h/2\pi$).

Cette écriture d'un nombre (appelée aussi "notation scientifique") comporte deux parties :
- la mantisse (ici $6,02252$ et $6,625$) ;
- la partie exposant (23 et -34), le 10 est ici obligatoire car c'est la base choisie pour écrire la mantisse et l'exposant).

On dit que la représentation est *normalisée* quand la mantisse est de la forme [1] :

$$c_0, c_1c_2c_2c_3 \ldots \text{ avec } c_0 \neq 0$$

Remarques :

1. Tout nombre réel (sauf zéro) peut s'écrire avec cette notation en virgule flottante normalisée avec en général un nombre de chiffres infini pour la mantisse (résultat provenant de la densité de $\mathbb{Q}$ dans $\mathbb{R}$) ; on peut aussi remarquer que certains nombres peuvent s'écrire de deux façons, par exemple $0,999\cdots = 1$.
2. Un changement de base peut introduire quelques bizarreries dans l'écriture d'un nombre alors que celle-ci est anodine dans la base initiale, par exemple (cf TD) :

$$(0,2)_{10} = (0,\ 0011\ 0011\ 0011\ \ldots)_2.$$

1.1.2 Définition des ensembles de flottants

Dans un ordinateur on est obligé de restreindre le nombre de chiffres pour les mantisses et de limiter l'étendue des exposants. En choisissant ces limites, on définit un ensemble de nombres flottants. Plus précisément un tel ensemble noté $\mathbb{F}(\beta, p, e_{min}, e_{max})$ est défini à partir de 4 entiers où :
- β est l'entier ($\beta \geq 2$) définissant la base ;

1. une autre définition souvent rencontrée est $0, c_1c_2c_2c_3 \ldots$ avec $c_1 \neq 0$

— p est le nombre de chiffres de la mantisse ;

— e_{min} est l'exposant minimum et e_{max} l'exposant maximum.

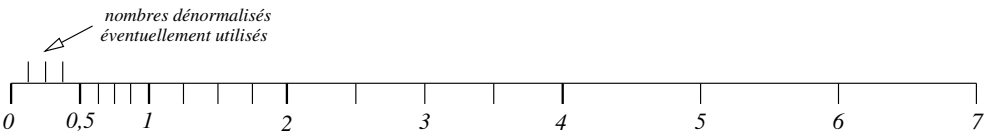
et il correspond à tous les nombres réels x s'écrivant :

$$x = s\left(\sum_{i=0}^{p-1} c_i \beta^{-i}\right)\beta^e, \text{ où } \begin{cases} s = \pm 1 \text{ le signe} \\ 0 \leq c_i \leq \beta - 1, \forall i \\ e_{min} \leq e \leq e_{max} \end{cases}$$

la représentation étant normalisée si $c_0 \neq 0$. Pour représenter 0 on utilise une écriture spéciale en ne mettant que des 0 dans la mantisse (ce qui est logique) et un exposant de $e_{min} - 1$ [2].

Exemple : le sous-ensemble des nombres normalisés positifs (plus zéro) de $\mathbb{F}(2, 3, -1, 2)$:

0		
$1,00\ 2^{-1}$	=	$(0,5)_{10}$
$1,01\ 2^{-1}$	=	$(0,625)_{10}$
$1,10\ 2^{-1}$	=	$(0,750)_{10}$
$1,11\ 2^{-1}$	=	$(0.875)_{10}$
$1,00\ 2^{0}$	=	1
$1,01\ 2^{0}$	=	$(1,25)_{10}$
$\vdots$	$\vdots$	$\vdots$
$1,11\ 2^{2}$	=	$(7)_{10}$



Remarques et notations :

— Dans les intervalles $[\beta^e, \beta^{e+1}]$ et $[-\beta^{e+1}, -\beta^e]$ l'incrément entre deux nombres flottants est constant et égal à β^{1-p+e}.

— Pendant assez longtemps on a utilisé uniquement les nombres normalisés (plus zéro) de ces ensembles. Cependant, comme la figure précédente le montre, il en résulte un "vide" entre le plus petit nombre normalisé et le zéro. L'utilisation des nombres dénormalisés non redondants (cf figure précédente) permet d'aller vers zéro plus graduellement.

— On notera M le plus grand nombre positif de $\mathbb{F}(\beta, p, e_{min}, e_{max})$:

$$M = \left(\sum_{i=0}^{p-1}(\beta - 1)\beta^{-i}\right)\beta^{e_{max}} = (1 - \beta^{-p})\beta^{e_{max}+1},$$

m le plus petit nombre normalisé positif :

$$m = (1,00..0)\beta^{e_{min}} = \beta^{e_{min}},$$

et μ le plus petit nombre dénormalisé (> 0) :

$$\mu = (0,00..1)\beta^{e_{min}} = \beta^{1-p+e_{min}}.$$

— On notera $\oplus$, $\ominus$, $\otimes$, $\oslash$ les opérations d'addition, de soustraction, de multiplication et de division, effectuées par l'ordinateur.

— Dans les systèmes flottants actuels on rajoute des nombres spéciaux comme $+inf$, $-inf$ (inf comme infini) et NaN (pour Not a Number) qui ont une représentation spéciale (utilisant en particulier un exposant de $e_{max} + 1$). Enfin, on notera que, du fait du bit de signe, le zéro a deux représentations [3] qui conduisent à des résultats différents sur quelques calculs (par

2. ce qui semble naturel si on pense à l'algorithme de comparaison de deux nombres flottants normalisés

3. que l'on notera $+0$ et -0 : si mathématiquement c'est le même nombre, informatiquement non !

exemple $1 \oslash +0$ donnera $+inf$ alors que $1 \oslash -0$ donnera $-inf$).

1.1.3 Approximation d'un nombre réel par un nombre flottant

Étant donné $x \in \mathbb{R}$, en général $x \notin \mathbb{F}(\beta, p, e_{min}, e_{max})$ et on associe à x une approximation $fl(x) \in \mathbb{F}(\beta, p, e_{min}, e_{max})$ avec le critère suivant :

> $fl(x)$ est le flottant le plus proche de x et si x est à égale distance de deux flottants celui dont le dernier chiffre est pair est choisi (à partir de maintenant on suppose que β est pair).

Attention cependant cette règle ne fonctionne pas pour les nombres de magnitude supérieure à M : en toute rigueur il faut considérer d'abord $\widetilde{fl}(x)$ comme étant la même opération mais à valeur dans $\mathbb{F}(\beta, p, e_{min}, +\infty)$ et si :

- $|\widetilde{fl}(x)| > M$ alors $fl(x)$ est le nombre flottant spécial $sign(x)inf$ (on parle alors d' "overflow")
- et sinon $fl(x) = \widetilde{fl}(x)$.

Seuil d'overflow : avec cette règle M ne correspond pas exactement au seuil d'overflow. Dans $\mathbb{F}(\beta, p, e_{min}, +\infty)$ le nombre flottant juste après M est $\beta^{e_{max}+1}$, dont le dernier chiffre de la mantisse est pair. Ainsi le seuil d'overflow est exactement :

$$\tilde{M} := \frac{M + \beta^{e_{max}+1}}{2} = M + \frac{\beta}{2}\beta^{-p}\beta^{e_{max}}$$

mais du fait de la règle d'arrondi $\tilde{fl}(\tilde{M}) = \beta^{e_{max}+1}$ (et pas M) et donc $fl(\tilde{M}) = Inf$. Ainsi tout nombre réel x tel que $|x| < \tilde{M}$ doit être codé par un flottant "usuel" (cad qui n'est pas un nombre flottant spécial comme $\pm Inf$ ou Nan).

Epsilon machine : On montre que si $|x| \in [m, \tilde{M}[$ alors l'erreur relative est bornée par un nombre appelé epsilon machine :

$$\frac{|fl(x) - x|}{|x|} \leq \mathbf{u} := \frac{\beta^{1-p}}{2}.$$

Preuve : Soit donc $x \in \mathbb{R}$ tel que $|x| \in [m, \tilde{M}[$. Il existe $e \in [\![e_{min}, e_{max}]\!]$ tel que $|x| \in [\beta^e, \beta^{e+1}[$. Nous avons vu que l'incrément entre deux flottants dans la plage $[\beta^e, \beta^{e+1}]$ est égal à β^{1-p+e}, ainsi l'erreur absolue est bornée par la moitié de cette quantité (approximation par le flottant le plus proche) :

$$ea = |x - fl(x)| \leq \frac{1}{2}\beta^{1-p+e}$$

On peut alors borner l'ereur relative en divisant par le plus petit nombre de la plage, d'où :

$$er = \frac{|x - fl(x)|}{|x|} \leq \frac{ea}{\beta^e} = \frac{1}{2}\beta^{1-p} \quad \square$$

Une façon plus pratique de noter cette erreur relative est d'écrire que (cf TD) :

$$fl(x) = x(1 + \epsilon), \text{ avec } |\epsilon| \leq \mathbf{u}$$

Lorsque $|x| \in [0, m[$ avec $fl(x) \neq x$ on parle d'underflow. Dans ce cas l'erreur relative n'est plus maîtrisée. On passe d'une erreur relative quasi bornée par $\mathbf{u}$ au voisinage de $|m|$ à une erreur relative de 1 (pour les nombres non nuls très proches de 0 et qui seront codés par 0 (c'est à dire tous les nombres réels dans $[-\mu/2, \mu/2]$), excepté 0).

1.1.4 Règles des 4 opérations usuelles

Les 4 opérations usuelles sur les flottants doivent respecter le critère suivant : tout se passe comme si le calcul était exact puis arrondi au flottant le plus proche, c'est à dire que si $\cdot$ est l'une des 4 opérations et $\odot$ l'opération machine correspondante, et x et y sont deux nombres flottants alors :

$$x \odot y = fl(x \cdot y)$$

et donc en l'absence d'overflow et d'underflow on a :

$$x \odot y = (x \cdot y)(1+\epsilon), \text{ avec } |\epsilon| \leq \mathbf{u}$$

Remarques :

— La norme sur les flottants impose aussi cette précision pour la racine carrée.

— On peut montrer que la soustraction entre deux flottants x et y de même signe est exacte s'ils sont de magnitude voisine, plus exactement si $x \leq y \leq 2x$ ou si $y \leq x \leq 2y$ alors $x \ominus y = x - y$.
Si la soustraction de deux flottants voisins est exacte, elle est quand même à l'origine de certains problèmes de perte de précision car les nombres x et y résultent le plus souvent de calculs qui comporteront des erreurs et la soustraction amplifiera ces mêmes erreurs.

1.1.5 Flottants utilisés sur ordinateurs

Les deux ensembles de flottants les plus utilisés sur ordinateurs sont :

1. $\mathbb{F}(2, 24, -126, 127)$ appelés flottants "simple précision" dont le codage tient sur 4 octets,
2. $\mathbb{F}(2, 53, -1022, 1023)$ appelés flottants "double précision" tenant sur 8 octets.

et voici leurs nombres caractéristiques :

ensembles de flottants	$\mathbb{F}(2, 24, -126, 127)$	$\mathbb{F}(2, 53, -1022, 1023)$
$\mathbf{u}$	$5.9604645\ 10^{-8}$	$1.1102230246251565\ 10^{-16}$
m	$1.1754944\ 10^{-38}$	$2.2250738585072014\ 10^{-308}$
M	$3.4028235\ 10^{38}$	$1.7976931348623157\ 10^{308}$

Remarque : sauf problème d'overflow ou d'underflow toute multiplication/division par une puissance de 2 est exacte avec ces nombres flottants.

1.1.6 Exemple de calcul d'erreur

Pour faire une analyse de précision de calculs menés avec les flottants, on fait souvent l'hypothèse qu'il n'y a pas eu d'overflow ni d'underflow. Exemple : on considère a, b et c des flottants et on veut analyser la précision de $x_c := (a \otimes b) \oslash (c \otimes d)$. Si on fait cette hypothèse alors :

$$\begin{aligned} a \otimes b &= (a \times b)(1+\epsilon_1), \quad |\epsilon_1| \leq \mathbf{u} \\ c \otimes d &= (c \times d)(1+\epsilon_2), \quad |\epsilon_2| \leq \mathbf{u} \\ (a \otimes b) \oslash (c \otimes d) &= \frac{(a \otimes b)}{(c \otimes d)}(1+\epsilon_3), \quad |\epsilon_3| \leq \mathbf{u} \end{aligned}$$

soit finalement :

$$x_c = \underbrace{\frac{a \times b}{c \times d}}_{x} \underbrace{\frac{(1+\epsilon_1)(1+\epsilon_3)}{(1+\epsilon_2)}}_{(1+\delta)}$$

où comme $\mathbf{u} << 1$, l'erreur relative $|\delta|$ est "quasi-bornée" par $3\mathbf{u}$.

Une manière élégante de traiter ce problème est d'utiliser le résultat suivant :

lemme : (de *simplification*, c'est l'exercice 6 du TD). Si $|\epsilon_k| \leq \mathbf{u}$, $s_k = \pm 1$ pour $k = 1, \ldots n$ et si $n\mathbf{u} < 1$ alors :

$$\prod_{k=1}^{n}(1+\epsilon_k)^{s_k} = 1 + \delta_n \quad \text{avec} \quad |\delta_n| \leq \frac{n\mathbf{u}}{1-n\mathbf{u}}$$

Preuve : Montrons que $\mathcal{P}(1)$ est vraie. Si $s_1 = 1$ le résultat est évident. Examinons le cas où $s_1 = -1$. On obtient :

$$\frac{1}{1+\epsilon_1} = 1 - 1 + \frac{1}{1+\epsilon_1} = 1 + \underbrace{\frac{-\epsilon_1}{1+\epsilon_1}}_{\delta_1}$$

et donc :

$$|\delta_1| \leq \frac{\mathbf{u}}{1-\mathbf{u}}.$$

Supposons maintenant que $\mathcal{P}(j-1)$ soit vraie (rmq : si $j \leq n$ alors on a bien $j\mathbf{u} < 1$) et montrons que $\mathcal{P}(j)$ l'est également. En utilisant l'hypothèse de récurrence :

$$\prod_{k=1}^{j}(1+\epsilon_k)^{s_k} = (1+\delta_{j-1})(1+\epsilon_j)^{s_j} \text{ avec } |\delta_{j-1}| \leq \frac{(j-1)\mathbf{u}}{1-(j-1)\mathbf{u}}$$

Si $s_j = 1$ alors :

$$(1+\delta_{j-1})(1+\epsilon_j) = 1 + \underbrace{\delta_{j-1} + \epsilon_j + \delta_{j-1}\epsilon_j}_{\delta_j}$$

d'où :

$$|\delta_j| \leq |\delta_{j-1}| + |\epsilon_j| + |\delta_{j-1}\epsilon_j| \leq \frac{(j-1)\mathbf{u}}{1-(j-1)\mathbf{u}} + \mathbf{u} + \frac{(j-1)\mathbf{u}^2}{1-(j-1)\mathbf{u}} = \frac{j\mathbf{u}}{1-(j-1)\mathbf{u}} \leq \frac{j\mathbf{u}}{1-j\mathbf{u}}$$

Et si $s_j = -1$:

$$\frac{1+\delta_{j-1}}{1+\epsilon_j} = 1 - 1 + \frac{1+\delta_{j-1}}{1+\epsilon_j} = 1 + \underbrace{\frac{\delta_{j-1}-\epsilon_j}{1+\epsilon_j}}_{\delta_j}$$

d'où :

$$|\delta_j| \leq \frac{\frac{(j-1)\mathbf{u}}{1-(j-1)\mathbf{u}} + \mathbf{u}}{1-\mathbf{u}} = \frac{(j-1)\mathbf{u} + \mathbf{u} - (j-1)\mathbf{u}^2}{(1-(j-1)\mathbf{u})(1-\mathbf{u})} = \frac{j\mathbf{u} - (j-1)\mathbf{u}^2}{1-j\mathbf{u}+(j-1)\mathbf{u}^2} \leq \frac{j\mathbf{u}}{1-j\mathbf{u}}$$

□