Chapitre 6

Etude des erreurs informatiques d'affectation et d'opération sur un ordinateur en arithmétique flottante

1 Introduction :

On considère le modèle d'arithmétique sur ordinateur suivant :
Soit b la base de l'arithmétique de l'ordinateur, en général b = 2 ou 16.
Tout nombre réel x compris entre les limites acceptées par la machine peut être représenté en virgule flottante normalisée par le nombre X tel que :

$$X = \pm M b^E \; avec \; \frac{1}{b} \leq M < 1$$

La mantisse M est codée sur n digits (chiffres associés à b) comprenant généralement eux-même un certain nombre de bits (par exemple en base 16, 4 bits) ; par normalisation, le premier digit est non nul, noté M_1.
M est égal à : $\sum_{i=1}^{n} M_i b^{-i}$, $0 \leq M_i < b$
C'est la partie fractionnaire de X.

Par exemple, un mot-mémoire a la structure binaire de la représentation en mode flottant normalisé suivante :
sur 64 bits (double précision), on a 1 bit de signe, 11 bits d'exposant, et 52 bits de mantisse.
Il y a un ensemble fini, noté $\mathbb{F}$ de nombres X.
Pour les exposants négatifs, on peut utiliser la représentation suivante :
$\mathrm{E}_{code'} = \mathrm{E}_{vrai}$ + biais où :

biais = $\frac{1}{2}2^k$ avec k bits de codage pour l'exposant avec son signe.

E_{vrai} est la valeur algébrique de l'exposant.

$\mathrm{E}_{code'}$ est la valeur décimale codée de l'exposant.

Par exemple sur 8 bits, le biais vaut 128, $\mathrm{E}_{code'}$ varie de 0 à 255, E_{vrai} varie de -128 à 127.
Dans la base b, un réel x s'écrit $\pm m b^E$ où m la mantisse généralement illimité, $\frac{1}{b} \leq m < 1$

$$m = \sum_{i=1}^{\infty} m_i \times b^{-i} \quad m_i \in \{0, 1, \cdots, b-1\} \quad m_1 \neq 0$$

Exemple en base 2 :

$$0.1 = \sum_{i=1}^{\infty} \lambda_i \times 2^{-i} \quad \lambda_i \in \{0, 1\}$$

En multipliant par 2, 0.1 et en prenant la partie entière on obtient λ_1 et on continue pour obtenir λ_2. Quand on a 1 en partie entière, on l'affecte à λ_i, puis on enlève 1 au nombre courant et on continue en multipliant par 2, le nouveau nombre (< 1) :

$0.1\times2 = \underline{0}.2 \quad (\lambda_1 = 0); 0.2\times2 = \underline{0}.4 \quad (\lambda_2 = 0); 0.4\times2 = \underline{0}.8 \quad (\lambda_3 = 0); 0.8\times2 = \underline{1}.6 \quad (\lambda_4 = 1); 1.6 - 1 = 0.6; 0.6 \times 2 = \underline{1}.2 \quad (\lambda_5 = 1); 1.2 - 1 = 0.2$; 0.2 a déjà été obtenu.
D'où le développement binaire infini périodique (de longueur de période 4) de 0.1 égal à $(0.00\underline{0110}0110\cdots)_2$
Pour obtenir l'écriture normalisé de la mantisse m (m_1 toujours égal à 1) on décale à gauche jusqu'au premier 1 et l'exposant devient -3 :

$$m = 0.11001100\cdots \text{ et } 0.1 = m2^{-3} = 0.8 \times 2^{-3}$$

Que se passe-t-il pour le codage de 0.1 en flottant ?

Il faut arrondir les nombres, c'est-à-dire renvoyer un des nombres flottants voisins.

La norme IEEE 754 a quatre modes d'arrondi :

L'arrondi vers $+\infty$ (ou par excès), noté $\triangle\,(x)$: retourne le plus petit nombre flottant supérieur ou égal à x.

L'arrondi vers $-\infty$ (ou par défaut), noté $\triangledown(x)$: retourne le plus grand nombre flottant inférieur ou égal à x.

L'arrondi vers 0, noté Z(x) : retourne $\triangle\,(x)$ pour les nombres x négatifs et $\triangledown(x)$ pour les nombres x positifs.

L'arrondi au plus près, noté $\circ(x)$: retourne le nombre flottant le plus proche de x (pour un réel au milieu de deux flottants voisins on choisit celui dont la mantisse se termine par un 0)

Les trois premiers modes d'arrondis sont dits dirigés.
L'arrondi correct est l'arrondi du calcul exact avec les quatre opérations arithmétiques réelles et les opérandes dans $\mathbb{F}$.

Codage de 0.1 en flottant en norme IEEE 754 ($1 \leq M < 2$) :
Dans le cas de l'arrondi (au plus près) on rajoute 1 au 24^{eme} bit en SP et au 53^{eme} bit en DP de la mantisse m en partant de m_1 puis on ne conserve que les 23 premiers bits en SP et les 52 en DP.

Simple précision
Codage binaire
Sur 32 bits, on a :

```
(gdb) x/tw &zeroUn1
0x7fffffffde34: 00111101110011001100110011001101
```

FIGURE 1 – Codage binaire de 0.1 en simple précision

Le nombre flottant, en SP sur 32 bits lus de gauche à droite :
— est positif (bit de signe à 0) ;

— est normalisé (l'exposant n'est pas codé par 00000000) ;
— a un exposant codé par 01111011.
— a une mantisse codée par 10011001100110011001101.

Valeur décimale

— le bit de signe $s = 0$;

A partir de l'expression binaire de l'exposant codé $(e)_2 = (01111011)_2 = (e_7 e_6 \dots e_0)_2$, sa valeur est égale à :

$$\begin{aligned} e &= \sum_{i=0}^{7} e_i 2^i \\ &= 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^6 \\ &= 123 \end{aligned}$$

La valeur de l'exposant vrai $e - d = 123 - 127 = -4$
A partir de l'expression binaire de la mantisse codée $(m)_2 = (0, 10011001100110011001101)_2 = (0, m_1 m_2 \dots m_{23})_2$, sa valeur est égale à :

$$\begin{aligned} m &= \sum_{i=1}^{23} m_i 2^{-i} \\ &= 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + 2^{-23} \\ &\approx 0.60000002384 \end{aligned}$$

$M = 1 + m \approx 1.60000002384$. D'où la valeur est $\approx 1.60000002384 \times 2^{-4} \approx 0.10000000149$.
La valeur réelle est $(1+(m)_2)\times 2^{-4} = (1, 1001\ 1001\ 1001\dots)_2 \times 2^{-4} = 1.6 \times 2^{-4} = 0.1$.

Double précision

Codage binaire
Sur 64 bits, on a :

```
(gdb) x/tg &zeroUn2
0x7fffffffde38: 0011111110111001100110011001100110011001100110011001100110011010
```

FIGURE 2 – Codage binaire de 0.1 en double précision

Le nombre flottant, en DP sur 64 bits lus de gauche à droite :
— est positif (bit de signe à 0) ;
— est normalisé (l'exposant n'est pas codé par 00000000) ;
— a un exposant codé par 01111111011.
— a une mantisse codée par 1001100110011001100110011001100110011001100110011001100110011010.

Valeur exacte

— le bit de signe $s = 0$;

A partir de l'expression binaire de l'exposant codé $(e)_2 = (01111111011)_2 =$

$(e_{10}e_9 \dots e_0)_2$, sa valeur est égale à :

$$\begin{aligned} e &= \sum_{i=0}^{10} e_i 2^i \\ &= 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 + 2^9 \\ &= 1019 \end{aligned}$$

La valeur de l'exposant vrai $e - d = 1019 - 1023 = -4$

A partir de l'expression binaire de la mantisse codée
$(m)_2 = (0, 10011001100110011001100110011001100110011001100110011010)_2 = (0, m_1 m_2 \dots m_{52})_2$,
sa valeur est égale à :

$$\begin{aligned} m &= \sum_{i=1}^{52} m_i 2^{-i} \\ &= 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} + \dots + 2^{-45} + 2^{-48} + 2^{-49} + 2^{-51} \\ &\approx 0.60000000000000000888178419700125232338905334472656 25 \end{aligned}$$

$M = 1+m \approx 1.6$. D'où la valeur est $1.60000000000000000888178419700125232338905334472656 25 \times 2^{-4} \approx 0.10000000000000000055511151231257827021181583404541015625$.
La valeur réelle est $(1+(m)_2)\times 2^{-4} = (1, 1001\ 1001\ 1001 \dots)_2 \times 2^{-4} = 1.6 \times 2^{-4} = 0.1$.

2 Valeur moyenne et écart-type de l'erreur d'affectation :

On considère l'opération d'affectation : $X := x$, $X \in \mathbb{F}$, $x \in \mathbb{R}$
L'erreur relative d'affectation est donnée par : $\alpha = \dfrac{X - x}{X}$
en posant $r = m - M$, le résidu, la partie perdue de la mantisse, on a $\alpha = \dfrac{-r}{M}$
où :

- $0 \leq r < \dfrac{1}{b^n}$ avec une arithmétique d'arrondi vers 0 (de troncature) (on supprime dans la mantisse illimitée m, tous les digits de rang supérieur à n)
- $\dfrac{-1}{2b^n} \leq r < \dfrac{1}{2b^n}$ avec une arithmétique d'arrondi au plus près (d'arrondi) (on ajoute dans m, $b/2$ au digit de rang $n+1$, puis on tronque)

On suppose qu'un nombre flottant X représente de façon équiprobable tout nombre réel x. C'est-à-dire pour une mantisse M fixée, toutes les erreurs absolues d'affectation (résidus) r sont équiprobables.
On considère deux hyporhèses pour les mantisses :

Hypothèse 1 : La Porte et Vignes

Les mantisses sont équiréparties pour $\dfrac{1}{b} \leq M < 1$

Hypothèse 2 : Hamming et Knuth

Les mantisses sont logarithmiquement distribuées pour $\frac{1}{b} \leq M < 1$
L'hypothèse 1 s'interprète expérimentalement pour des nombres sans information antérieure, tandis que l'hypothèse 2 est valable pour des nombres provenant de calculs.
Dans le plan réel, on considère D_t (respectivement D_a) l'ensemble des points (M, r) vérifiant :

$$\frac{1}{b} \leq M < 1,\ 0 \leq r < \frac{1}{b^n}$$

$$(respectivement\ \ \frac{1}{b} \leq M < 1,\ \frac{-1}{2b^n} \leq r < \frac{1}{2b^n})$$

2.1 Ordinateur d'arithmétique de troncature

Soit p la densité de probabilité d'un point (M, r) dans le domaine D_t, on a dans l'hypothèse 1, p constant, égale à P :

$$\iint_{D_t} P dr dM = 1 \text{ d'où } P = \frac{b^{n+1}}{b-1}$$

et la valeur moyenne $\overline{\alpha}_t$ égale à :

$$-\iint_{D_t} p \frac{r}{M} dr dM = \frac{-b^{-n+1}}{2(b-1)} \log b$$

l'écart-type σ_t est donné par :

$$\sigma_t^2 = \iint_{D_t} p(\alpha - \overline{\alpha})^2 dr dM = b^{-2n} \left[\frac{b}{3} - \frac{1}{4} \frac{b^2 (\log b)^2}{(b-1)^2} \right]$$

Dans l'hypothèse 2, la densité de probabilité dans D_t est :

$$p(r, M) = \frac{K}{M \log b}$$

on a $\iint_{D_t} p(r, M) dr dM = 1$ d'où $K = b^n$

$$\overline{\alpha}_t = -\iint_{D_t} \frac{b^n r}{M^2 \log b} dr dM = -b^{-n} \frac{(b-1)}{2 \log b}$$

$$\sigma_t^2 = b^{-2n} \left[\frac{b^2 - 1}{6 \log b} - \frac{(b-1)^2}{4 (\log b)^2} \right]$$

2.2 Ordinateur d'arithmétique d'arrondi

Dans l'hypothèse 1, on a $\overline{\alpha_a} = 0$ et $\sigma_a^2 = \frac{1}{12} b^{1-2n}$

Dans l'hypothèse 2, on a $\overline{\alpha_a} = 0$ et $\sigma_a^2 = b^{-2n} \frac{b^2 - 1}{24 \log b}$

2.3 Comparaison des deux arithmétiques et hypothèses

Il s'agit de calculer les valeurs numériques des constantes précédentes et d'étudier les paramètres en fonction de la longueur de la mantisse de n' bits (avec n chiffres hexadécimaux de mantisse on a $n' = 4n$ et avec n chiffres binaires de mantisse on a $n' = n$).

	Hypothèse de Vignes		Hypothèse de Hamming	
$\overline{\alpha}_t$	-0,693*$2^{-n'}$	-1,48*$2^{-n'}$	-0,718*$2^{-n'}$	-2,70*$2^{-n'}$
σ_t	0,431*$2^{-n'}$	1,77*$2^{-n'}$	0,454*$2^{-n'}$	2,81*$2^{-n'}$
$\overline{\alpha}_a$	0	0	0	0
σ_a	0,408*$2^{-n'}$	1,33*$2^{-n'}$	0,425*$2^{-n'}$	1,95*$2^{-n'}$
	base 2	base 16	base 2	base 16

En base 2 ($n' = n$) les valeurs de α et σ sont peu différentes dans les deux hypothèses. La base 2 est plus précise que la base 16. L'arithmétique d'arrondi donne une meilleure précision que l'arithmétique de troncature.

3 Etude de l'erreur dans le calcul sur ordinateur d'une somme de deux flottants :

La somme de deux flottants se fait dans l'unité arithmétique et logique (UAL) à l'aide de registres. Le circuit associé est appelé additionneur.

Exemple sur 32 bits (SP) :
$X = 1.5 = 2^0(1 + 0.5) = \underline{|0|01111111|10000000000000000000000|}$
$Y = 0.5 = 2^{-1}(1 + 0.) = \underline{|0|01111110|00000000000000000000000|}$
Plus grand exposant 0 de X :
Mantisse de X avec à gauche le bit caché (poids de 2^0) et un bit supplémentaire à droite :
$\underline{|1|10000000000000000000000|0|}$
Mantisse de Y avec à gauche le bit caché (poids de 2^0) et un bit supplémentaire à droite :
$\underline{|1|00000000000000000000000|0|}$
Décalage d'un bit à droite (différence des deux exposants) :
$\underline{|0|10000000000000000000000|0|}$
Somme binaire de :

$$\begin{array}{r} |1| \\ |1|10000000000000000000000|0| \\ |0|10000000000000000000000|0| \end{array}$$

En arrondi (au plus près) on ajoute 1 au dernier bit (à droite).
On obtient le résultat (avec 1 de retenue) :

$$1|0|00000000000000000000000|1|$$

Normalisation (on rajoute 1 à l'exposant 0 et on décale d'un bit à droite) :

$$|1|00000000000000000000000|0|$$

On a le codage binaire du résultat flottant (le 1 du bit à gauche est le bit caché (poids de 2^0)) :
$\underline{|0|10000000|00000000000000000000000|}$

D'où la somme :
$X \oplus Y = 2^1(1 + 0.) = 2$
Remarque :
Soient X et $Y \in \mathbb{F}$
Soit Z le flottant de l'addition flottante $X \oplus Y$ et z le réel de l'addition réelle $X + Y$
On a $Z = z(1 + \alpha)$ avec $|\alpha| \leq u$ (u unité d'arrondi) et α l'erreur relative d'arrondi sur l'addition.
Si de plus $X = x(1 + \lambda)$ et $Y = y(1 + \mu)$, λ et μ erreurs relatives sur X et Y (par exemple de codage), alors :
$Z = (x(1 + \lambda) + y(1 + \mu))(1 + \alpha)$
$Z - z = x\lambda + y\mu + (x + y)\alpha + x\lambda\alpha + y\mu\alpha$

4 Etude de l'erreur dans le calcul sur ordinateur d'une somme de produits de nombres réels :

Soient les nombres réels x_i et y_i de $i = 1, \cdots, N$. Il s'agit de calculer sur ordinateur la somme r $= \sum_{i=1}^{N} x_i y_i$
Sur machine on obtient :

$$\mathrm{R} = X_1 * Y_1 \oplus X_2 * Y_2 \oplus \cdots \oplus X_N * Y_N$$

où les X_i, Y_i sont les nombres flottants associés aux x_i et y_i et $\oplus$ et $*$ sont les opérations d'addition et multiplication en virgule flottante de l'ordinateur.
La somme R est obtenue par l'algorithme des sommes partielles avec le programme suivant :

```
R <- 0
pour i = 1 jusqu'\'a N faire R <- R + X[i] * Y[i]
```

L'erreur absolue notée ρ est égale à $R - r$.

$$\text{On a} \begin{cases} X_i = x_i(1 + \lambda_i) \\ \qquad\qquad\qquad \text{i = 1, ..., N} \\ Y_i = y_i(1 + \mu_i) \end{cases}$$

Les erreurs relatives de codage λ_i et μ_i appartiennent à la population $\mathcal{P}(\alpha)$ des erreurs relatives d'affectation de valeur moyenne $\overline{\alpha}$ et d'écart-type σ donnés en fonction de l'arithmétique machine.
L'addition et la multiplication sur ordinateur donnent au premier ordre :

$$\begin{cases} X_i \oplus Y_i = x_i + y_i + \lambda_i x_i + \mu_i y_i + \alpha_i(x_i + y_i) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{i = 1, ..., N} \\ X_i * Y_i = x_i y_i(1 + \lambda_i + \mu_i + \beta_i) \end{cases}$$

avec α_i, λ_i, μ_i, $\beta_i \in \mathcal{P}(\alpha)$. α_i représente l'erreur relative d'arrondi liée à l'addition et β_i l'erreur relative d'arrondi liée à la multiplication.

D'où au premier ordre :

$$R = \sum_{i=1}^{N} x_i y_i + \sum_{i=1}^{N} (\lambda_i + \mu_i + \beta_i) x_i y_i + \sum_{k=2}^{N} \alpha_{k-1} r_k \text{ avec } r_k = \sum_{j=1}^{k} x_j y_j$$

On suppose que les erreurs α_i, λ_i, μ_i, $\beta_i \in \mathcal{P}(\alpha)$ sont aléatoires, indépendantes et représentatives de cette population.
Donc :

$\overline{\alpha_i} = \overline{\beta_i} = \overline{\lambda_i} = \overline{\mu_i} = \overline{\alpha}$

$\overline{\alpha_i^2} = \overline{\beta_i^2} = \overline{\lambda_i^2} = \overline{\mu_i^2} = \overline{\alpha}^2 + \sigma^2$

$\overline{\alpha_i \beta_k} = \overline{\alpha}^2$ avec $i \neq k$

Evaluation statistique de ρ

La valeur moyenne de ρ est la suivante :

$$\overline{\rho} = 3\overline{\alpha}r + \overline{\alpha}\left((N-1)x_1y_1 + (N-1)x_2y_2 + (N-2)x_3y_3 + (N-3)x_4y_4 + \cdots + 2x_{N-1}y_{N-1} + x_Ny_N\right)$$

Remarque :
Les premiers termes de la somme sont affectés des plus grands cœfficients dans l'expression $\overline{\rho}$ de la moyenne de l'erreur absolue sur la somme. Donc pour minimiser cette quantité sur une somme de termes positifs il faut sommer en premier les termes de plus petite valeur.

On peut simplifier cette expression avec l'hypothèse que la somme R est obtenue avec toutes les permutations possibles sur les opérandes et la valeur moyenne de ρ avec toutes ces sommes.

$$\overline{\rho} = \overline{\alpha}r\left(\frac{N^2 + 7N - 2}{2N}\right)$$

Evaluation statistique de ρ^2

$\rho^2 = S_1 + S_2 + S_3 + S_4 + S_5$
avec

$$S_1 = \sum_{i=1}^{N}(x_iy_i)^2(\lambda_i + \mu_i + \beta_i)^2$$
$$S_2 = \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} x_iy_ix_jy_j(\lambda_i + \mu_i + \beta_i)(\lambda_j + \mu_j + \beta_j)$$
$$S_3 = \sum_{k=2}^{N}\alpha_{k-1}^2 r_k^2$$
$$S_4 = \sum_{k=2}^{N}\sum_{\substack{l=2 \\ l\neq k}}^{N}\alpha_{k-1}\alpha_{l-1}r_kr_l$$
$$S_5 = \sum_{i=1}^{N}\sum_{k=2}^{N} x_iy_i(\lambda_i + \mu_i + \beta_i)\alpha_{k-1}r_k$$

Posons $S^2 = \sum_{i=1}^{N} (x_i y_i)^2$, $u = \sum_{k=2}^{N} r_k$, $v^2 = \sum_{k=2}^{N} r_k^2$

La valeur moyenne de ρ^2 est la suivante :

$\overline{\rho^2} = \sum_{i=1}^{5} \overline{S_i}$

en calculant chaque valeur moyenne de S_i et en remarquant que

$u^2 = v^2 + \sum_{k=2}^{N} \sum_{\substack{l=2 \\ l \neq k}}^{N} r_k r_l$, en fonction de $\overline{\alpha}$, σ, r, u, v^2, S^2

on obtient :

$\overline{\rho^2} = \overline{\alpha}^2(9r^2 + 6ru + u^2) + \sigma^2(3S^2 + v^2)$

On peut simplifier cette expression avec l'hypothèse que la somme R est obtenue avec toutes les permutations possibles sur les opérandes et la valeur moyenne de ρ^2 avec toutes ces sommes.
Si on conserve que les termes de plus haut degré, on a :

$$\overline{\rho^2} \approx \frac{\overline{\alpha}^2 N^2}{12}(3r^2 + S^2) + \frac{\sigma^2 N}{6}(2r^2 + S^2)$$

Avec une arithmétique de troncature :

$$\sqrt{\overline{\rho^2}} \approx \overline{\alpha} N \sqrt{\frac{r^2}{4} + \frac{S^2}{12}}$$

Avec une arithmétique d'arrondi :

$$\sqrt{\overline{\rho^2}} \approx \sigma \sqrt{N} \sqrt{\frac{r^2}{3} + \frac{S^2}{6}}$$

Remarque :
L'arithmétique d'arrondi donne une meilleure précision que l'arithmétique de troncature.

Remarque générale sur la formule algébrique de l'erreur absolue ρ :

Pour $N = 2$ cette formule est exacte à l'ordre 4.