TELECOM Nancy *(ESIAL)*
Maths Numériques

feuille 1 : nombres flottants

Exercice 1 *Un premier ensemble de flottants*

On se place dans l'ensemble de flottants $\mathcal{F} = \mathbb{F}(10, 4, -4, 4)$.

1. Calculer les nombres caractéristiques M, m et $\mathbf{u}$ de cet ensemble.
2. Donner la valeur de $fl(1,0015)$, $fl(1,234\ .10^{-3})$, $fl(1,234\ .10^{-5})$, $fl(1,234\ .10^{5})$, $fl(9,9995\ .10^{4})$ et $fl(9,99949\ .10^{4})$

Exercice 2 *Non associativité de l'addition flottante*

On se place dans $\mathbb{F}(10, 8, -\infty, +\infty)$ avec les règles IEEE (tout se passe comme si chaque calcul était effectué exactement puis on applique la fonction d'arrondi fl qui alors donne le nombre flottant le plus proche).

1. Calculer le epsilon machine $\mathbf{u}$ de cet ensemble de nombres flottants.
2. Montrer que :
$$(11111113 \oplus -11111111) \oplus 7,5111111 = 9,5111111$$
(le résultat obtenu étant exact) puis que :
$$11111113 \oplus (-11111111 \oplus 7,5111111) = 10$$
ce qui donne une erreur relative beaucoup plus importante que $\mathbf{u}$ (malgré le fait que les deux calculs consécutifs sont effectués chacun avec une précision relative inférieure à $\mathbf{u}$).

Exercice 3 *Une façon de noter l'erreur relative*

Soit $x \in \mathbb{R}$, $x \neq 0$ et y une approximation de x telle que l'erreur relative entre x et y soit inférieure à E :
$$\left|\frac{y-x}{x}\right| \leq E$$
Montrer qu'il est équivalent de dire qu'il existe un réel e (que l'on peut appeler erreur relative signée) avec $|e| \leq E$ pour lequel $y = x(1+e)$.

Exercice 4 *Changements de base*

1. Calculer $0,2$ en base 2 (aide : partir de $0,2 = \sum_{i>=1} d_i 2^{-i}$ avec $d_i = 0$ ou 1)
2. Même question pour $0,1$, $0,4$, $0,8$, etc. . . !
3. Il s'ensuit qu'en introduisant ces nombres dans un ordinateur fonctionnant en base 2 on commet nécessairement une (petite) erreur. Montrer que :
$$fl(0.2) = (\underbrace{1,100}_{1}\ \underbrace{1100}_{2}\ldots\underbrace{1100}_{12}\ \underbrace{1101}_{13}\ 0)_2\ 2^{-3}$$
dans $\mathbb{F}(2, 53, -1022, 1023)$ (cad en double précision), aide : $53 = 13 \times 4 + 1$. En base 10 ce nombre s'écrit exactement 0.2000000000000000011102230246251565404236316680908203125. C'est donc ce nombre que vous devez obtenir lorsque vous demandez une sortie avec suffisamment de chiffres (le format adéquat sera 56.54f).

Exercice 5 *Analyses d'erreur*

Dans cet exercice, on suppose que les calculs effectués par la machine ne provoquent ni overflow ni underflow et que sa F.P.U. respecte la norme IEEE, et donc si x et y sont deux nombres flottants alors :

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

D'autre part on pourra utiliser le fait que la multiplication ou la division par une puissance de 2 est exacte (sauf overflow ou underflow bien sûr).

1. Soient a, b, c, d et e des nombres flottants, on cherche à calculer $x = abc/(de)$. On suppose que ce calcul sera effectué selon le parenthésage : $x_c = ((a \otimes b) \otimes c) \oslash (d \otimes e)$. Donner une majoration de l'erreur relative entre x_c et le résultat exact x (utiliser le lemme de l'exercice suivant). On en déduira que, sauf problèmes d'overflow ou d'underflow, une séquence raisonnable de multiplications/divisions ne pose pas de problème de précision en arithmétique flottante.
2. Erreur pour une somme. On cherche à calculer $S = \sum_{k=1}^{n} x_k$, on suppose que le calcul se déroule selon le parenthésage :
$$S_c = ((((((x_1 \oplus x_2) \oplus x_3) \oplus x_4) \ldots \oplus x_n)$$
   (a) Montrer que s'il n'y a pas "d'overflow" ou "d'underflow" alors :
$$\begin{array}{rll} S_c = & (x_1+x_2)(1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3)\ldots(1+\epsilon_{n-1}) & \\ + & x_3(1+\epsilon_2)(1+\epsilon_3)\ldots(1+\epsilon_{n-1}) & \\ + & x_4(1+\epsilon_3)\ldots(1+\epsilon_{n-1}) & \text{avec } |\epsilon_k| \leq \mathbf{u}, \forall k \\ + & \ldots & \\ + & x_n(1+\epsilon_{n-1}) & \end{array}$$
   et en déduire, en utilisant le lemme de l'exercice suivant, que :
$$S_c - S = x_1\delta_{n-1} + x_2\delta_{n-1} + \sum_{k=3}^{n} x_k\delta_{n+1-k} \text{ avec } |\delta_k| \leq \frac{k\mathbf{u}}{1-k\mathbf{u}}, \forall k$$
   (b) En déduire que pour $S \neq 0$, on a :
$$\frac{|S_c - S|}{|S|} \leq \frac{\sum_k |x_k|}{|\sum_k x_k|} \left(\tfrac{(n-1)}{1-(n-1)\mathbf{u}}\right) \mathbf{u}$$
   Ainsi quand les x_k sont tous de même signe l'erreur relative est "quasi" bornée par $(n-1)\mathbf{u}$. Dans le cas contraire un coefficient d'amplification apparait dans la majoration et l'erreur relative peut donc être plus grande (voir question suivante) en particulier lorsque $|\sum_k x_k| << \sum_k |x_k|$.
3. Un problème important avec l'arithmétique flottante est la soustraction de deux nombres flottants de même signe et de magnitude voisine, ce qui semble contradictoire puisque ce calcul sera en général exact (si x et y sont 2 flottants de même signe tels que $|x|/2 \leq |y| \leq 2|x|$ on peut montrer que $x \ominus y = x - y$). Le problème est l'amplification des éventuelles erreurs contenues dans les 2 nombres que l'on soustrait. Soit deux nombres x et y résultats d'un calcul. Ces nombres sont entachés d'erreur et l'ordinateur a obtenu en fait $X = x+\delta x$ et $Y = y+\delta y$. On notera $r_x = x/(x-y)$, $r_y = y/(x-y)$ et e_x, e_y les erreurs relatives signées sur x et y. Montrer que l'erreur relative signée sur le résultat, c'est dire $\frac{(X \ominus Y)-(x-y)}{x-y}$ est égale à :
$$r_x e_x - r_y e_y$$
   (on supposera que X et Y sont tels que $X \ominus Y = X - Y$). Donner des exemples numériques concrets de perte de précision.

4. On cherche à calculer la fonction :

$$\phi(x) = \frac{1}{1+x} - \frac{1}{1-x} = \frac{-2x}{(1+x)(1-x)}$$

pour un nombre flottant x tel que $|x|$ est assez petit. Analyser l'erreur obtenue par les deux « algorithmes » :

(a) $y1 := (1 \oslash (1 \oplus x)) \ominus (1 \oslash (1 \ominus x))$

(b) $y2 := (-2 \otimes x) \oslash ((1 \oplus x) \otimes (1 \ominus x))$

5. Même question avec la fonction $f(x) = \sqrt{1+x} - \sqrt{1-x}$. On analysera d'abord l'erreur obtenue par "l'algorithme" immédiat : $y1 := sqrt(1 \oplus x) \ominus sqrt(1 \ominus x)$ où $sqrt(u)$ désigne la racine calculée en machine (pour cette opération la norme IEEE impose aussi que $sqrt(u) = fl(\sqrt{u})$, on a donc $sqrt(u) = \sqrt{u}(1+\epsilon)$ avec $|\epsilon| \leq \mathbf{u}$ pour tout flottant $u \geq 0$ qui n'est pas un nombre spécial (car avec la racine carrée $[\sqrt{\mu}, \sqrt{M}] \subset [m, M]$)). Puis on cherchera à réécrire f différemment pour trouver le bon algorithme (lorsque $|x|$ est petit). Rappel : $\sqrt{1+x} \approx 1 + x/2 + O(x^2)$.

Exercice 6 *Un lemme de simplification*

Lorsque l'on conduit une étude de la précision d'une formule en arithmétique flottante on se retrouve souvent avec des quantités de la forme suivante :

$$Q = \frac{(1+\epsilon_1)(1+\epsilon_2)\dots(1+\epsilon_k)}{(1+\epsilon_{k+1})\dots(1+\epsilon_n)}$$

où les ϵ_i sont très petits devant 1. En développant en série les termes en $1/(1+\epsilon)$, on obtient :

$$Q = 1 + \delta, \text{ avec } \delta = \epsilon_1 + \ldots + \epsilon_k - \epsilon_{k+1} - \ldots - \epsilon_n + \text{ termes croisés}$$

Ainsi si on néglige les termes croisés, il vient $|\delta| \leq \sum_{i=1}^{n} |\epsilon_i|$ soit $|\delta| \leq n\mathbf{u}$ si $|\epsilon_i| \leq \mathbf{u}$ pour tout i. Cependant cette borne n'est pas exacte du fait que l'on a négligé les termes croisés.

Une manière élégante de traiter ce problème est d'utiliser le résultat suivant : si $|\epsilon_k| \leq \mathbf{u}$, $s_k = \pm 1$ pour $k = 1, \dots n$ et si $n\mathbf{u} < 1$ alors :

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

Démontrer ce résultat par récurrence sur n.

Exercice 7 *Problème d'overflow et d'underflow parasites*

Parfois au cours d'un calcul, la magnitude d'une quantité intermédiaire q peut ne pas tomber dans l'intervalle $[m, M]$ [1] alors que le résultat final pourrait être correctement approché dans le système flottant. On parle dans ce cas d'overflow ou d'underflow parasite, l'exemple le plus simple étant celui du calcul de la norme d'un vecteur (ici en dimension 2 mais c'est encore plus vrai en dimension supérieure) : $\sqrt{x^2+y^2}$.

1. donner des cas d'overflow et d'underflow parasites pour les flottants 4 octets ;
2. trouver un algorithme simple qui permet d'éviter ce problème puis obtenir une majoration de l'erreur relative de ce dernier.

Exercice 8 *La bonne façon de résoudre l'équation du second degré*

Soit l'équation du second degré $ax^2 + bx + c = 0$. On se place dans le cas où $a \neq 0$ et $\Delta = b^2 - 4ac > 0$.

1. Dans le cas où $|4ac| << b^2$ quel problème peut-on rencontrer dans le calcul des racines ?

1. Plus précisément si $|q| < m$ avec $fl(q) \neq q$ on obtient alors une perte de précision (erreur relative plus forcément bornée par $\mathbf{u}$) et dans le cas où $|q| \geq \tilde{M}$ on obtient un Inf.

2. Comment peut-on remédier à ce problème (aide : $x_1 x_2 = c/a$) ?

Exercice 9 *extrait du partiel de 2011*

Le but de l'exercice est d'étudier l'algorithme utilisé par la fonction `dnrm2` de la bibliothèque BLAS standard. Cette fonction permet de calculer la norme euclidienne d'un vecteur de $\mathbb{R}^n$:

$$\|x\| = \left(\sum_{k=1}^{n} x_k^2 \right)^{1/2} \tag{2}$$

en évitant les problèmes d'overflow parasite.

1. Rappeler ce qu'est un overflow parasite.
2. Une première version consiste à généraliser la méthode de l'exercice 5 : on recherche le maximum des $|x_k|$ qui sert alors de facteur d'échelle (on factorise l'expression (2) par $\max_k |x_k|$). Ecrire précisément cet algorithme.
3. L'algorithme précédent a le défaut de parcourir deux fois les composantes du vecteur x (une fois pour le max et une deuxième pour le calcul effectif) ce qui n'est pas très efficace du point de vue des accès mémoire. On aimerait faire juste une seule passe sur x ce qui implique l'utilisation d'un facteur d'échelle "dynamique" que l'on modifie au besoin au fur et à mesure des itérations. A la fin de l'itération k le facteur d'échelle doit être égal à :

$$s := \max_{j \in [\![1,k]\!]} |x_j|$$

et on a calculé :

$$t := \sum_{j=1}^{k} \left(\frac{x_j}{s} \right)^2$$

(a) montrer que si $k = n$ on obtient alors $\|x\| = s\sqrt{t}$.

(b) écrire la mise à jour des deux quantités s et t à l'itération $k+1$ (aide : il faut bien sûr tester si $|x_{k+1}| > s$) ;

(c) écrire alors l'algorithme complet.

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	$3.4028235\ 10^{38}$	$1.7976931348623157\ 10^{308}$

TABLE 1 – valeurs caractéristiques (approchées) des deux jeux de flottants usuels