UPMC SORBONNE UNIVERSITÉS

Arithmétique Flottante et Analyse d'Erreurs

Examen final du 31 janvier 2017 (2 heures)
Documents autorisés. Calculatrices autorisées.

Version du 30 janvier 2017

Le barème sur 30 est donné à titre indicatif. **Il n'est pas nécessaire de traiter tous les exercices pour avoir une bonne note.**

Exercice 1 – Arithmétique flottante (3 points)

1. (**1 point**) Soit x un nombre flottant double précision IEEE 754 vérifiant $1 \leq x < 2$. Montrer que, en arrondi au plus près, $\mathrm{fl}(x \times \mathrm{fl}(1/x))$ vaut soit 1 soit $1 - \mathbf{u}$ avec $\mathbf{u} = 2^{-53}$.
2. (**2 points**) Soit M un nombre flottant suffisamment grand pour que $\mathrm{fl}(10 + M) = M$. Quelles sont alors les valeurs possibles pour $\mathrm{fl}(\sum_{i=1}^{6} x_i)$ où $\{x_i\}_{i=1}^{6} = \{1, 2, 3, 4, M, -M\}$ sachant que la somme est évaluée par l'algorithme de sommation récursive classique ?

Exercice 2 – FastTwoSum (4 points)

On suppose que l'on travaille en double précision IEEE 754 et que l'on a deux nombres flottants double précision a, b **vérifiant** $|a| \geq |b|$. On rappelle ici l'algorithme `FastTwoSum`.

Algorithm 1 FastTwoSum

function $[s, t]$ = `FastTwoSum`(a, b)

1: $s \leftarrow \mathrm{fl}(a + b)$
2: $z \leftarrow \mathrm{fl}(s - a)$
3: $t \leftarrow \mathrm{fl}(b - z)$

Si l'on est en arrondi au plus près, on a $s + t = a + b$ où t est l'erreur d'arrondi qui est représentable par un nombre flottant double précision.

1. (**1 point**) Montrer que l'erreur d'arrondi n'est plus nécessairement représentable si l'on travaille en arrondi vers $+\infty$. Pour cela donner un contre-exemple.
2. (**2 points**) On suppose maintenant que l'on travaille en arrondi vers $+\infty$. On a donc $s + e = a + b$ où e est un nombre réel qui n'est pas nécessairement un nombre flottant. Montrer que dans l'algorithme `FastTwoSum`, on a bien $z = s - a$. Pour cela vous utiliserez le lemme de Sterbenz en distinguant les cas $a, b \geq 0$ et $a \geq 0, b \leq 0$ (dans ce cas vous pourrez distinguer $-b \geq a/2$ et $-b < a/2$). Vous ne traiterez pas les cas $a, b \leq 0$ et $a \leq 0, b \geq 0$.
3. (**1 point**) En déduire que l'on a $|t - e| \leq 2\mathbf{u}|e|$ où $\mathbf{u}$ est l'unité d'arrondi ($\mathbf{u} = 2^{-53}$ en double précision).

Exercice 3 – Fonctions élémentaires (6 points)

Pour une `libm`, on veut donner une implémentation de la fonction sinus hyperbolique $\sinh(x) = \frac{e^x - e^{-x}}{2}$.

On rappelle que cette fonction se développe autour de 0 de la façon suivante :

$$\sinh(x) = x + \frac{1}{6}x^3 + \frac{1}{120}x^5 + \dots$$

1. (**1 point**) Même si on a déjà une implémentation de la fonction e^x de disponible, pourquoi est-ce une mauvaise idée d'implémenter $\sinh$ comme suit ?

```
1 double sinh(double x) {
2   return 0.5 * (exp(x) - exp(-x));
3 }
```

2. (**3 points**) Supposons maintenant qu'on ait une implémentation de la fonction $\mathrm{expm1}(x) = e^x - 1$ à notre disposition. Que peut-on faire alors pour la fonction $\sinh$? Quel paramètre de l'implémentation s'améliore-t-il ? Peut-on faire encore mieux ? Discutez.
3. (**2 points**) Dans la `libm` de la GNU libc, la fonction $\sinh(x)$ est implémentée en exploitant d'abord sa symétrie et ensuite en découpant le domaine d'entrée en trois intervalles, pour les arguments petits, les moyens et les grands. Esquissez la structure du code pour une telle implémentation, en donnant et en justifiant des bornes pour les trois intervalles.

Exercice 4 – Calcul en virgule fixe (5 points)

1. (**2 points**) On considère le format virgule fixe signé $(6, -4)$ (position du bit de poids fort : 6, position du bit de poids faible : -4). Indiquez le nombre de bits de ce format, la valeur la plus petite ainsi que la plus grande. Quel est le pas de quantification ?
2. (**1.5 points**) Représenter les réels suivants en virgule fixe signée sur 8 bits, en indiquant le format virgule fixe (position du bit de poids fort et du bit de poids faible).

$$9.3452 \qquad 0.03434 \qquad -1012.125$$

3. (**1.5 points**) Indiquer pour chacun d'entre eux l'erreur relative et absolue commise. D'une manière générale, comment borner ces deux erreurs en fonctions de la largeur w utilisée pour représenter le nombre.

Exercice 5 – Arithmétique stochastique (6 points)

On calcule l'expression $f(x, y) = a + b + c$ avec $a = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2)$, $b = 5.5y^8$ et $c = x/(2y)$, pour $x = 77617$ et $y = 33096$.

En arithmétique à virgule flottante avec arrondi au plus près, on obtient les résultats suivants avec différentes précisions (pour lesquelles on indique le nombre de bits de la mantisse).

Précision	$f(77617, 33096)$
simple (24 bits)	$6.3382530\ 10^{29}$
double (53 bits)	1.1726039400531
étendue (64 bits)	1.172603940053178

1. (**3 points**) On utilise une implémentation de l'Arithmétique Stochastique Discrète (ASD) qui affiche pour chaque résultat les chiffres qu'elle estime corrects (@.0 s'il n'y en a aucun).

   Expliquer les résultats suivants, obtenus en double précision, c'est-à-dire avec une précision de 53 bits pour les mantisses :

   ```
   a=-0.791711134066896E+037
   b= 0.791711134066896E+037
   a+b= @.0
   c= 0.117260394005317E+001
   a+b+c=@.0
   ```

   Une instabilité numérique est générée. Ce type d'instabilité remet-il en cause la validité de l'estimation par l'ASD du nombre de chiffres corrects ? Pourquoi ?

2. (**3 points**) On obtient :
   — en arithmétique d'intervalles, en utilisant une précision de 53 bits pour les mantisses, un intervalle de longueur 10^{22} qui contient 0
   — en Arithmétique Stochastique Discrète, en utilisant une précision de 122 bits pour les mantisses, le résultat suivant :
   `a+b+c=-0.827396059946821368141165095479816292`
   et, dans ce cas, aucune instabilité numérique n'est détectée
   — en arithmétique d'intervalles, en utilisant une précision de 130 bits pour les mantisses, l'intervalle suivant :
   $[-0.82739605994682136814116509547981629200 5,$
   $-0.82739605994682136814116509547981629198 6].$

   Qu'en déduisez-vous ?

Exercice 6 – Series convergentes (6 points)

On note $\mathrm{M}(n)$ une borne sur la complexité de la multiplication d'entiers d'au plus n bits telle que la fonction $n \mapsto \mathrm{M}(n)/n$ soit croissante.

1. (**6 points**) Proposer un algorithme de complexité $O(\mathrm{M}(d)\log(d)^k)$ (pour un certain k que l'on explicitera) qui, pour tout d donné, calcule $\ln(2)$ avec une erreur relative bornée par 2^{-d}. Justifier la correction et la complexité de votre algorithme.

   On rappelle que $\ln(2) \approx 0.693\ldots$, que $\ln(x^{-1}) = -\ln(x)$ pour tout $x \in]0,\infty[$, et que la série entière

   $$\sum_{n=1}^{\infty}(-1)^{n+1}\frac{z^n}{n} = z - \frac{z^2}{2} + \frac{z^3}{3} - \cdots$$

   converge vers $\ln(1+z)$ pour tout $z \in]-1;1[$.