UPMC SORBONNE UNIVERSITÉS

Arithmétique Flottante et Analyse d'Erreurs

Examen final du 26 janvier 2016 (2 heures)
Documents autorisés. Calculatrices autorisées.

Version du 25 janvier 2016

Le barème sur 42 est donné à titre indicatif. **Il n'est pas nécessaire de traiter tous les exercices pour avoir une bonne note.**

Exercice 1 – Représentation mémoire IEEE754-2008 (3 points)

1. (**3 points**) Soit $a = 4097 = 2^{12} + 1$ et $b = 8449 = 2^{13} + 2^8 + 1$. Soit $c = a \otimes b$ le nombre en virgule flottante obtenu par multiplication de a et de b en simple précision (binary32) avec arrondi au plus près. Donnez la représentation mémoire de c en simple précision (binary32).

Exercice 2 – Arithmétique en virgule flottante (2 points)

On suppose que l'on travaille en double précision (binary64) avec MATLAB.

1. (**1 point**) Quel est le résultat du calcul de $(-1+1)+2^{-53}$? Quel est le résultat du calcul de $-1+(1+2^{-53})$? Qu'en déduisez-vous sur les propriété mathématiques de l'addition en arithmétique à virgule flottante ?
2. (**1 point**) Quel est le résultat exact du calcul de $1 + \underbrace{2^{-53} + 2^{-53} + \cdots + 2^{-53}}_{2^{53} \text{ termes}}$? Quel est le résultat du calcul en double précision si l'on additionne les éléments de gauche à droite ? Comment s'appelle ce phénomène ?

Exercice 3 – Fused-Multiply-And-Add (5 points)

On travaille en arithmétique flottante binaire IEEE 754 avec arrondi au plus près. On note $\circ$ cet arrondi. On rappelle que $\texttt{FMA}(a, b, c)$ calcule l'arrondi au plus près de $a \cdot b + c$ en une seule opération et un seul arrondi. On propose l'algorithme de calcul d'un produit scalaire de deux vecteurs x et y de taille n suivant :

function $\texttt{res} = \texttt{Dot}(x, y)$
  $s = 0$;
  $p = 0$;
  for $i = 1 : n$
    $p = \circ(x_i \times y_i)$
    $s = \circ(s + p)$
  $\texttt{res} = s$

On peut montrer que l'on a la borne d'erreur suivante : $|\texttt{res} - x^T y| \leq \gamma_n |x|^T |y|$ où $\gamma_n = n\mathbf{u}/(1 - n\mathbf{u})$ et $\mathbf{u}$ l'unité d'arrondi.

1. (**1 point**) Sachant que l'on dispose de l'opérateur FMA, modifier l'algorithme précédent afin d'effectuer moins d'opérations. Est-ce que cela permet d'améliorer la borne d'erreur ? Justifier votre réponse.

2. (**1 point**) On souhaite calculer $1/a$ par la méthode de Newton en résolvant l'équation $f(x) = a - 1/x$. Proposer un algorithme sans `FMA`.
3. (**2 points**) On suppose maintenant que l'on dispose d'un `FMA`. Proposer un algorithme plus efficace permettant de calculer $1/a$.
4. (**1 point**) On sait que l'erreur d'arrondi pour la somme de deux flottants ou le produit de deux flottants est un nombre flottant. L'erreur d'arrondi lors du calcul de $\texttt{FMA}(a, b, c)$ est-elle un flottant ? Justifier votre réponse.

Exercice 4 – Fonctions élémentaires - $\sin(x)$ (14 points)

On veut implanter la fonction définie par $f(x) = \sin(x)$ pour des arguments raisonablement petits, disons pour $x \in \mathbb{F}_{53}$ tels que $0 \leq |x| \leq 2^{10} = 1024$.

On se place dans le cadre de l'arithmétique flottante IEEE754-2008 binaire. On utilise la précision double (binary64). On vérifie que $2^{52} + 2^{51} = 6755399441055744$.

1. (**2 points**) Que calcule le code suivant ? Affirmez une propriété liant k à x, puis démontrez-la.

```
1 double red_step_1(double x) {
2     double t, k;
3
4     t = x + 6755399441055744.0;
5     k = t - 6755399441055744.0;
6
7     return k;
8 }
```

2. (**1 point**) On note maintenant $\circ_{42}$ l'arrondi au plus près à une précision de 42 bits. On prend trois flottants définis comme suit :

$$\begin{aligned} c_h &= \circ_{42}(2\pi) \\ c_m &= \circ_{42}(2\pi - c_h) \\ c_l &= \circ_{42}(2\pi - c_h - c_m). \end{aligned}$$

Montrez que les flottants c_h, c_m et c_l sont représentables en precision double (binary64), c.-à-d. que $c_h, c_m, c_l \in \mathbb{F}_{53}$.

3. (**2 points**) Donnez une borne pour l'erreur relative ε_c entre $c_h + c_m + c_l$ et 2π, définie par

$$\varepsilon_c = \frac{c_h + c_m + c_l}{2\pi} - 1.$$

Justifiez.

4. (**1 point**) On note $\circ_{53}$ l'arrondi au plus près en précision double (binary64). On vérifie que

$$\begin{aligned} c_h &= \circ_{53}(6.283185307180247) \\ c_m &= \circ_{53}(-6.605598148971295 \cdot 10^{-13}) \\ c_l &= \circ_{53}(6.578502774529327 \cdot 10^{-26}). \end{aligned}$$

On suppose toujours que x est un flottant double précision (binary64) tel que $|x| \leq 2^{10}$. On suppose aussi que k est le flottant double précision (binary64) représentant l'entier défini par

$$k = \left\lfloor \circ_{53}\left(\circ_{53}\left(\frac{1}{2\pi}\right) x \right) \right\rceil$$

c.-à-d. l'entier le plus proche à l'arrondi au plus proche de $\frac{1}{2\pi}$, représenté en double précision, multiplié par x.

Montrez que le code suivant calcule trois flottants t_h, t_m et t_l tels que $t_h + t_m + t_l = k \cdot (c_h + c_m + c_l)$ *exactement*, c.-à-d. sans qu'il y ait d'arrondi. Justifiez.

```
1 void red_step_2(double *th, double *tm, double *tl, double k) {
2   *th = k * 6.283185307180247;
3   *tm = k * -6.605598148971295e-13;
4   *tl = k * 6.578502774529327e-26;
5 }
```

5. (**3 points**) Montrez que la quantité r définie par

$$r = x - 2\pi \cdot \left\lfloor \frac{x}{2\pi} \right\rceil$$

est bornée par $-\pi \leq r \leq \pi$ et que $\sin(x) = \sin(r)$.

6. (**5 points**) Toujours en supposant que $x \in \mathbb{F}_{53}$ tel que $|x| \leq 2^{10}$ et en se servant d'un algorithme dû à William Kahan, il est possible de montrer la propriété suivante pour la quantité r :
— si x est tel que $|x| < 1$, alors $r = x$,
— si x est tel que $|x| \geq 1$, alors r vérifie $|r| > 2^{-58.5}$.
On admettra cette propriété sans chercher à la démonter (ce qui dépasserait de loin le cadre de cet examen).
Montrez que le code suivant calcule une approximation $\hat{r}$ à la quantité r définie ci-dessus et que l'erreur relative ε_r entre $\hat{r}$ et r, donnée par

$$\varepsilon_r = \frac{\hat{r}}{r} - 1,$$

est bornée par

$$|\varepsilon_r| \leq 2^{-53} + 2^{-55}.$$

Dans votre raisonnement, établissez d'abord le fait que les opérations flottantes faites aux lignes 10 et 11 ne provoquent pas d'erreur, c.-à-d. qu'elles sont exactes.
On vérifie que $\circ_{53}\left(\frac{1}{2\pi}\right) = \circ_{53}(0.15915494309189535)$.

```
1  double sin_red(double x) {
2    double q, k, th, tm, tl;
3    double u, v, r;
4
5    q = x * 0.15915494309189535;
6    k = red_step_1(q);
7
8    red_step_2(&th, &tm, &tl, k);
9
10   u = x - th;
11   v = u - tm;
12   r = z - tl;
13
14   return r;
15 }
```

Exercice 5 – Évaluation d'une suite (8 points)

Soit $(c_n)_{n\in\mathbb{N}}$ la suite de réels définie par

$$(3n+4)(3n+5)c_{n+2} - 10(n+1)(n+2)c_{n+1} + c_n = 0,$$
$$c_0 = \frac{1}{9\Gamma(2/3)^3}, \quad c_1 = \frac{1}{18\Gamma(2/3)^3} - \frac{1}{3\Gamma(1/3)^3},$$

où Γ est la fonction Gamma d'Euler. On admet qu'il existe une suite θ_n telle que

$$c_n = \frac{1+\theta_n}{4\sqrt{3}\pi 9^n} \quad \text{et} \quad |\theta_n| \leqslant \frac{3}{n^{1/4}}.$$

On suppose donné un algorithme permettant de calculer $\Gamma(n/3)$ à précision 2^{-p} pour n et p quelconques.

Proposer un algorithme prenant en entrée n et p et renvoyant une approximation numérique de c_n à environ 2^{-p} près pour n et p assez grands. L'algorithme pourra faire appel à des opérations en virgule flottante à précision arbitraire et/ou à des opérations arithmétiques exactes sur des nombres rationnels. On veillera à préciser le type et la précision des opérations utilisées.

Estimer aussi précisément que possible la qualité de l'approximation renvoyée par l'algorithme. On pourra notamment proposer une estimation asymptotique pour n et p grands, ou, mieux, une borne.

Exercice 6 – Arithmétique en virgule fixe (4 points)

1. (**2 points**) Représenter les réels $\frac{\pi}{3}$, e^{10}, $-e^{-10}$ en virgule fixe avec $\beta = 8$ et $\beta = 16$.
2. (**2 points**) On cherche à calculer
$$\frac{\pi}{3}x + e^{10}y - e^{-10}z$$
avec x, y et z inconnus mais représentés sur 8 bits, avec comme LSB 4, -8 et 20, respectivement. On dispose d'un multiplieur accumulateur 16 bits (plus 5 bits de garde). Proposer une réalisation.

Exercice 7 – Arithmétique stochastique (6 points)

On souhaite calculer une racine du polynôme $f(x) = 1.47x^3 + 1.19x^2 - 1.83x + 0.45$ par la méthode de Newton. On calcule la suite : $x_0 = 0.5$, $x_{n+1} = x_n - f(x_n)/f'(x_n)$ avec le test d'arrêt $|x_n - x_{n-1}| \leq 10^{-12}$.

En arithmétique flottante IEEE avec arrondi au plus près, les derniers itérés obtenus sont :

```
x( 35) = +4.285714252078272e-01
x( 36) = +4.285714252078272e-01
```

Le calcul est ensuite effectué avec le programme ci-après qui utilise l'arithmétique stochastique discrète implantée dans la bibliothèque CADNA.

```
#include<cadna.h>
#include<stdio.h>
#include<math.h>
main()
{
  int i, nmax=100;
  double_st y, x;
  cadna_init(-1);   /*all instabilities detected */
  y = 0.5;
  for(i = 1;i<=nmax;i++){
    x = y;
    y = x-(1.47*pow(x,3)+1.19*pow(x,2)-1.83*x+0.45)/
          (4.41*pow(x,2)+2.38*x-1.83);
    if (fabs(x-y)<=1.e-12) break;
  }
  printf("x(%3d) = %s\n",i-1,strp(x));
  printf("x(%3d) = %s\n",i,strp(y));
  cadna_end();
}
```

Ce programmme fournit les résultats suivants.

```
x( 23) =  0.42857143E+000
x( 24) =  0.42857143E+000
-----------------------------------------------------------------
There are 50 numerical instabilities
1 UNSTABLE BRANCHING(S)
49 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)
```

1. (**2 points**) Commentez les résultats obtenus. Les instabilités détectées remettent-elles en cause l'estimation de la précision des résultats ?
2. (**2 points**) Dans le programme utilisant la bibliothèque CADNA, si on remplace le test d'arrêt par celui-ci :

```
14 if (fabs(x-y)<1.e-12) break;
```

   on obtient :

```
x( 82) =  0.42857142E+000
x( 83) =  0.42857142E+000
-----------------------------------------------------------------
CRITICAL WARNING: the self-validation detects major problem(s).
The results are NOT guaranteed.
There are 387 numerical instabilities
59 UNSTABLE DIVISION(S)
59 UNSTABLE BRANCHING(S)
45 UNSTABLE INTRINSIC FUNCTION(S)
224 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)
```

   Commentez ces résultats et expliquez la différence par rapport aux résultats obtenus précédemment. Les instabilités détectées remettent-elles en cause l'estimation de la précision des résultats ?
3. (**1 point**) Quel serait le test d'arrêt optimal dans ce programme ? Quel sera le nombre d'itérations effectuées avec ce nouveau test d'arrêt ?
4. (**1 point**) Ce programme fournit une approximation d'une racine double α. Il a été prouvé que, dans ce cas,

$$C_{x_n,x_{n+1}} \sim_\infty C_{x_{n+1},\alpha}$$

   où $C_{a,b}$ désigne le nombre de chiffres communs entre deux réels a et b et le symbole $\sim_\infty$ peut être considéré comme une égalité lorsque la phase de convergence est atteinte.

   Que peut-on en déduire concernant l'approximation fournie par ce programme, en particulier avec le test d'arrêt optimal ?