Floating-point arithmetic and error analysis (AFAE)

Examen final du 4 février 2020

Documents autorisés. Calculatrices autorisées - Durée : 2h00

Le barème est donné à titre indicatif

Le sujet se décompose en 4 exercices indépendants. La qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'évaluation de la copie.

Exercice 1 (Multiplications flottantes (5 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 avec arrondi au plus près. Donnez la représentation en simple précision de c.

Exercice 2 (Nombres flottants (5 points))**.** On suppose que l'on travaille en double précision IEEE 754 avec arrondi au plus près et que l'on a deux nombres flottants double précision a, b **vérifiant** $|a| \geq |b|$. On note $\mathrm{fl}(\cdot)$ le résultat d'un calcul en double précision. On note $x = \mathrm{fl}(a+b)$ le résultat de l'addition flottante de a et b. On note e l'erreur d'arrondi, c'est-à-dire que l'on a $x + e = a + b$. On peut montrer (et on l'admet ici) que e est toujours un nombre flottant double précision.
On effectue dans l'ordre les calculs suivants en double précision :

$$x = \mathrm{fl}(a+b), \quad z = \mathrm{fl}(x-a), \quad y = \mathrm{fl}(b-z).$$

On rappelle le lemme de Sterbenz : si s et t sont deux flottants positifs vérifiant $t/2 \leq s \leq 2t$ alors $s - t = \mathrm{fl}(s-t)$.

1. Montrer que l'on a $z = x - a$.
   Indication : 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 $a \leq 0, b \geq 0$.
2. En déduire que $y = e$.

Exercice 3 (FMA (5 points))**.** On travaille en arithmétique flottante binaire IEEE 758 avec arrondi au plus près. On rappelle que $\mathtt{FMA}(a,b,c)$ 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 $\mathtt{res} = \mathtt{Dot}(x, y)$
  $s = 0$;
  $p = 0$;
  for $i = 1 : n$
    $p = \mathrm{fl}(x_i \times y_i)$
    $s = \mathrm{fl}(s + p)$
  $\mathtt{res} = s$

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

1. 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. On souhaite calculer $1/a$ par la méthode de Newton en résolvant l'équation $f(x) = a - 1/x$. Proposer un algorithme sans FMA. On suppose maintenant que l'on dispose d'un FMA. Proposer un algorithme plus efficace permettant de calculer $1/a$.
3. 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 FMA(a, b, c) est-elle un flottant ? Justifier votre réponse.

Exercice 4 (Arithmétique stochastique (5 points))**.** Soit f une fonction réelle C^∞ sur l'intervalle $[a,b]$ telle que $f'(a) \neq f'(b)$ et soit $I = \int_a^b f(x)dx$.
On considère I_n l'approximation de I par la méthode des trapèzes avec le pas $h = \frac{b-a}{2^n}$:

$$I_n = \frac{h}{2}\left(f(y_0) + 2f(y_1) + \ldots + 2f(y_{2^n-1}) + f(y_{2^n})\right)$$

avec $y_i = a + ih$, $i = 0, \ldots, 2^n$.

Le programme ci-après calcule les approximations I_n successives de $I = \int_0^1 \frac{\arctan(\sqrt{2+t^2})}{(1+t^2)\sqrt{2+t^2}}dt$.
Ce programme utilise une implementation de l'Arithmétique Stochastique Discrète (ASD) qui permet d'afficher pour chaque résultat les chiffres qu'elle estime corrects (@.0 s'il n'y en a aucun).

```
#include<cadna.h>
#include<stdio.h>
#include<math.h>

double_st f(double_st x)
{ double_st aux=sqrt(2.+x*x);
  return(atan(aux)/(aux*(1.+x*x)));
}

int main()
{
double_st x, somme, integ, h, integold, aux , a, b;
double eps=1.E-14;
int n=1, i=0, j;
cadna_init(-1);
somme=0.;
a=0.;
b=1.;
aux=f(a)+f(b);
h = b-a;
integold = 1.;
integ=0;
while( (fabs(integold - integ)>eps) && i<30 )
  {integold = integ;
   i = i+1;
   x = a + h/2.;
   for (j=1; j<=n ; j++)
     { somme = somme + f(x);
       x = x + h;
```

```
      }
    n = 2*n;
    h = h/2.;
    integ = h*(aux + 2*somme)/2.;
   }
  printf("%d %s %s\n", i, strp(integ),strp(fabs(integ-integold)));
  cadna_end();
}
```

Ce programmme fournit les résultats suivants.

```
CADNA_C software
Self-validation detection: ON
Mathematical instabilities detection: ON
Branching instabilities detection: ON
Intrinsic instabilities detection: ON
Cancellation instabilities detection: ON
----------------------------------------------------------------
21  0.51404189589003E+000  @.0
----------------------------------------------------------------
There are 20 numerical instabilities
1 UNSTABLE BRANCHING(S)
2 UNSTABLE INTRINSIC FUNCTION(S)
17 LOSS OF ACCURACY DUE TO CANCELLATION(S)
```

1. Commentez les résultats obtenus. Les instabilités détectées remettent-elles en cause l'estimation de la précision des résultats ?

2. Si dans le test d'arrêt, on remplace >**eps** par >**=eps**, 30 itérations sont effectuées. Comment l'expliquez-vous ?

3. Quel serait le test d'arrêt optimal dans ce programme ?
Quel sera le nombre d'itérations effectuées avec ce nouveau test d'arrêt ?

4. Il a été prouvé que

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{4}{3}\right) + \mathcal{O}\left(\frac{1}{4^n}\right)$$

où $C_{a,b}$ désigne le nombre de chiffres décimaux communs entre deux réels a et b.
Que peut-on en déduire concernant l'approximation fournie par le programme ?