

Faire compter les ordinateurs...



quels algorithmes ?

Jean-Michel Muller
CNRS - Laboratoire LIP

(CNRS-INRIA-ENS Lyon-Université de Lyon)

[http://perso.ens-lyon.fr/
jean-michel.muller/](http://perso.ens-lyon.fr/jean-michel.muller/)

Grenoble — 24 octobre 2011



Mon métier ? J'apprends à compter aux ordinateurs !

Arithmétique des ordinateurs :

- systèmes de représentation des nombres et algorithmes de calculs arithmétiques (faire une addition est encore du domaine de la recherche) ;
- réalisations : algorithmes, preuves, programmes et circuits (processeurs) ;
- **critères** : vitesse, précision, fiabilité, consommation d'énergie, surface des circuits, etc.

⇒ solutions de compromis, qui différeront suivant les applications.

Arithmétique « virgule flottante »

$$60221415000000000000000000 \rightarrow 6.0221415 \times 10^{23}$$

- utilisée partout en calcul scientifique ;
- « notation scientifique » de nos calculettes ;
- $x = m_x \times \beta^{e_x}$;
- base β : 2, 4, 8, 16, 10... et même 3 et 256.
- approche « floue » pour analyser les algorithmes : valeur calculée de $x + y = (x + y)(1 + \epsilon)$.

Arithmétique « virgule flottante »

$$60221415000000000000000000 \rightarrow 6.0221415 \times 10^{23}$$

- utilisée partout en calcul scientifique ;
- « notation scientifique » de nos calculettes ;
- $x = m_x \times \beta^{e_x}$;
- base β : 2, 4, 8, 16, 10... et même 3 et 256.
- approche « floue » pour analyser les algorithmes : valeur calculée de $x + y = (x + y)(1 + \epsilon)$.

Meilleure approche ?

Le banquier infernal

Voulant sécuriser ma retraite, j'ai

$e - 1 = 1.718281828459045235360287471352662497757247093 \dots$

euros à placer...



je me rends à la **Société chaotique de banque**, qui fait de la pub pour de nouveaux placements...

Le banquier infernal

À la Société chaotique de banque, le banquier m'explique :

Au bout de 25 ans, je peux retirer mon argent. . . est-ce intéressant ?

Le banquier infernal

À la Société chaotique de banque, le banquier m'explique :

- la première année, mon capital est multiplié par 1, et on me retire 1 euro pour frais de gestion ;

Au bout de 25 ans, je peux retirer mon argent. . . est-ce intéressant ?

Le banquier infernal

À la Société chaotique de banque, le banquier m'explique :

- la première année, mon capital est multiplié par 1, et on me retire 1 euro pour frais de gestion ;
- la deuxième année, mon capital est multiplié par 2, et on me retire 1 euro pour frais de gestion ;

Au bout de 25 ans, je peux retirer mon argent. . . est-ce intéressant ?

Le banquier infernal

À la Société chaotique de banque, le banquier m'explique :

- la première année, mon capital est multiplié par 1, et on me retire 1 euro pour frais de gestion ;
- la deuxième année, mon capital est multiplié par 2, et on me retire 1 euro pour frais de gestion ;
- la troisième année, mon capital est multiplié par 3, et on me retire 1 euro pour frais de gestion ;
- ...

Au bout de 25 ans, je peux retirer mon argent. . . est-ce intéressant ?

Le banquier infernal

À la Société chaotique de banque, le banquier m'explique :

- la première année, mon capital est multiplié par 1, et on me retire 1 euro pour frais de gestion ;
- la deuxième année, mon capital est multiplié par 2, et on me retire 1 euro pour frais de gestion ;
- la troisième année, mon capital est multiplié par 3, et on me retire 1 euro pour frais de gestion ;
- ...
- la 25ème année, mon capital est multiplié par 25, et on me retire 1 euro pour frais de gestion ;

Au bout de 25 ans, je peux retirer mon argent. . . est-ce intéressant ?

Je n'ai pas signé tout de suite...

J'ai cherché à calculer ce que serait mon capital au bout de 25 ans. . .

Je n'ai pas signé tout de suite...

J'ai cherché à calculer ce que serait mon capital au bout de 25 ans. . .

- ma calculatrice (Casio) : **−747895876335** euros (1/16e de la dette des USA à moi tout seul);

Je n'ai pas signé tout de suite...

J'ai cherché à calculer ce que serait mon capital au bout de 25 ans. . .

- ma calculette (Casio) : **-747895876335** euros (1/16e de la dette des USA à moi tout seul);
- mon ordinateur (Proc. Intel Xeon, compilateur gcc, sous Linux) : **+1201807247** euros;

Je n'ai pas signé tout de suite...

J'ai cherché à calculer ce que serait mon capital au bout de 25 ans...

- ma calculette (Casio) : **-747895876335** euros (1/16e de la dette des USA à moi tout seul);
- mon ordinateur (Proc. Intel Xeon, compilateur gcc, sous Linux) : **+1201807247** euros;
- en fait, la « vraie » valeur est d'environ **0.0399** euros...



Conclusion de ce fâcheux épisode

Conclusion de ce fâcheux épisode

- ne faites pas aveuglément confiance à votre ordinateur ;



Conclusion de ce fâcheux épisode

- ne faites pas aveuglément confiance à votre ordinateur ;



- ne faites pas aveuglément confiance à votre banquier.

Virgule flottante... 6.0221415×10^{23}

- système babylonien de base 60 ; règle à calcul : Gunter (1681-1626), Oughtred (1575-1660) ;



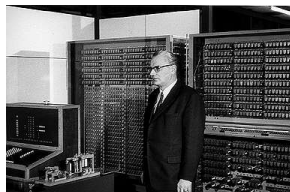
- Leonardo Torres y Quevedo (1914) : implantation électromécanique de la machine de Babbage, avec virgule flottante ;
- Konrad Zuse : Z3 (1941) : base 2, mantisses de 14 bits exposants de 7 bits. Mémoire de 16 nombres. Voir <http://www.epemag.com/zuse/>



Jean-Michel Muller



Faire compter les ordinateurs...



Besoins ? Quelques chiffres...

■ dynamique :

$$\frac{\text{Diamètre estimé Univers observable}}{\text{Longueur de Planck}} \approx 10^{62}$$

Besoins ? Quelques chiffres. . .

- **dynamique** :

$$\frac{\text{Diamètre estimé Univers observable}}{\text{Longueur de Planck}} \approx 10^{62}$$

- **précision** : certaines prédictions de la mécanique quantique et de la relativité générale vérifiées avec erreur relative $\approx 10^{-14}$

Besoins ? Quelques chiffres...

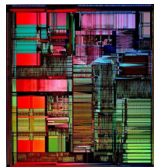
- **dynamique** :

$$\frac{\text{Diamètre estimé Univers observable}}{\text{Longueur de Planck}} \approx 10^{62}$$

- **précision** : certaines prédictions de la mécanique quantique et de la relativité générale vérifiées avec erreur relative $\approx 10^{-14}$
- **calculs intermédiaires** : « quad précision » (113 bits) et algorithmes sophistiqués pour la stabilité à très très long terme du système solaire
(J. Laskar, Observatoire de Paris).

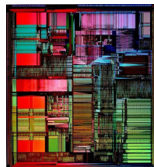
On peut faire du très mauvais travail...

- 1994 : bug de la division du Pentium,
8391667/12582905 donnait
0.666869... au lieu de 0.666910... ;



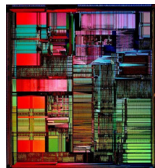
On peut faire du très mauvais travail...

- 1994 : bug de la division du Pentium, $8391667/12582905$ donnait $0.666869\dots$ au lieu de $0.666910\dots$;
- Sur certains ordinateurs Cray on pouvait déclencher un overflow en multipliant par 1 ;



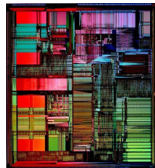
On peut faire du très mauvais travail...

- 1994 : bug de la division du Pentium, $8391667/12582905$ donnait $0.666869\dots$ au lieu de $0.666910\dots$;
- Sur certains ordinateurs Cray on pouvait déclencher un overflow en multipliant par 1 ;
- Maple, version 7.0, si l'on calcule $\frac{5001!}{5000!}$ on obtient 1 au lieu de 5001 ;



On peut faire du très mauvais travail...

- 1994 : bug de la division du Pentium, $8391667/12582905$ donnait $0.666869\dots$ au lieu de $0.666910\dots$;
- Sur certains ordinateurs Cray on pouvait déclencher un overflow en multipliant par 1 ;
- Maple, version 7.0, si l'on calcule $\frac{5001!}{5000!}$ on obtient 1 au lieu de 5001 ;
- Excel'2007 (premières versions), calculez $65535 - 2^{-37}$, vous obtiendrez 100000 ;



On peut faire du très mauvais travail...

- Novembre 1998, navire américain USS Yorktown, on a par erreur tapé un « zéro » sur un clavier → division par 0. Ce problème n'était pas prévu → cascade d'erreurs → arrêt du système de propulsion.



On peut faire du très mauvais travail...

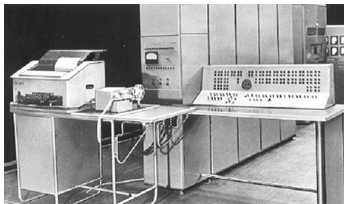
- Novembre 1998, navire américain USS Yorktown, on a par erreur tapé un « zéro » sur un clavier → division par 0. Ce problème n'était pas prévu → cascade d'erreurs → arrêt du système de propulsion.



- premier envol... et premier plongeon d'Ariane 5

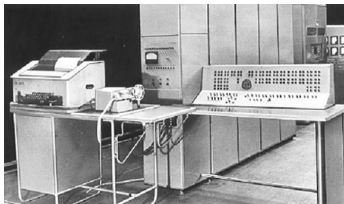


Beaucoup de bizzareries



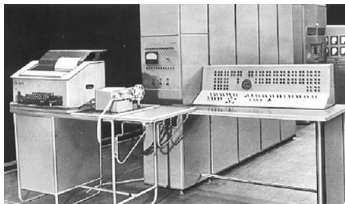
- Machine **Setun**, université de Moscou, 1958. 50 exemplaires ;

Beaucoup de bizzareries



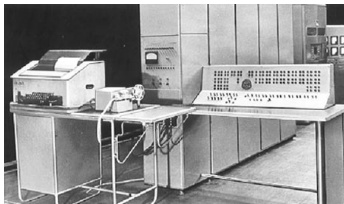
- Machine **Setun**, université de Moscou, 1958. 50 exemplaires ;
- base 3 et chiffres -1 , 0 et 1 .
Nombres sur 18 « trits » ;

Beaucoup de bizzareries



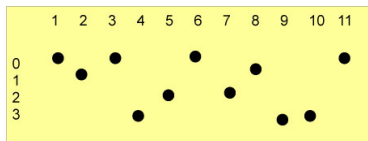
- Machine **Setun**, université de Moscou, 1958. 50 exemplaires ;
- base 3 et chiffres -1 , 0 et 1 . Nombres sur 18 « trits » ;
- idée : base β , nombre de chiffres n , + grand nombre représenté M . Mesure du « coût » : $\beta \times n$.

Beaucoup de bizzareries



Coût de la carte perforée ?

- Machine **Setun**, université de Moscou, 1958. 50 exemplaires ;
- base 3 et chiffres -1 , 0 et 1 .
Nombres sur 18 « trits » ;
- idée : base β , nombre de chiffres n , + grand nombre représenté M . Mesure du « coût » : $\beta \times n$.



Beaucoup de bizzareries

- minimiser $\beta \times n$ sachant que $\beta^n \approx M$;

Beaucoup de bizzareries

- minimiser $\beta \times n$ sachant que $\beta^n \approx M$;
- si variables réelles, optimum $\beta = e = 2.718\dots \approx 3$;

Beaucoup de bizzareries

- minimiser $\beta \times n$ sachant que $\beta^n \approx M$;
- si variables réelles, optimum $\beta = e = 2.718\dots \approx 3$;
- en variables entières, optimum $\beta = 2$ (rarement) ou 3 (plus souvent), et dès que

$$M \geq e^{\frac{5}{(2/\ln(2)) - (3/\ln(3))}} \approx 1.09 \times 10^{14}$$

l'optimum est toujours $\beta = 3$.

Système virgule flottante

$$\text{Paramètres : } \begin{cases} \text{base} & \beta \geq 2 \\ \text{précision} & p \geq 1 \\ \text{exposants extrêmes} & e_{\min}, e_{\max} \end{cases}$$

Un nombre VF fini x est représenté par 2 entiers :

■ **mantisse entière** : M , $|M| \leq \beta^p - 1$;

■ **exposant** e , $e_{\min} \leq e \leq e_{\max}$.

tels que

$$x = M \times \beta^{e+1-p}$$

On appelle **mantisse réelle**, ou **mantisse** de x le nombre

$$m = M \times \beta^{1-p},$$

ce qui donne $x = m \times \beta^e$, avec $|m| < \beta$.

Quelques exemples

Système	β	p	$e_{\min}$	$e_{\max}$	+ grand représ.
DEC VAX	2	24	-128	126	$1.7 \dots \times 10^{38}$
(D format)	2	56	-128	126	$1.7 \dots \times 10^{38}$
HP 28, 48G	10	12	-500	498	$9.9 \dots \times 10^{498}$
IBM 370	16	6 (24 bits)	-65	62	$7.2 \dots \times 10^{75}$
et 3090	16	14 (56 bits)	-65	62	$7.2 \dots \times 10^{75}$
IEEE-754-1985 et	2	23+1	-126	127	$3.4 \dots \times 10^{38}$
IEEE-754-2008	2	52+1	-1022	1023	$1.8 \dots \times 10^{308}$
IEEE-754-2008 « binary 128 »	2	112+1	-16382	16383	$1.2 \dots \times 10^{4932}$
IEEE-754-2008 « decimal 64 »	10	16	-383	384	$9.9 \dots 9 \times 10^{384}$

Norme IEEE 754 (1985 et 2008)

- la norme IEEE 754-1985 a mis fin à une pagaille (très mauvaise portabilité des algorithmes numériques) ;
- leader : W. Kahan (père de l'arithmétique des HP35 et Intel 8087) ;
- formats ;
- spécification des opérations, des conversions, etc. ;
- gestion des *exceptions* ($\max+1$, $1/0$, $\sqrt{-2}$, $0/0$, etc.) ;
- en 1985 : aucune spécification des fonctions **élémentaires** : sin, cos, exp, log. . .
- nouvelle révision adoptée en 2008.

Definition 1 (Arrondi correct)

Mode d'arrondi actif parmi :

- **au plus près** (défaut) s'il y en a deux :
 - *round ties to even* : celui dont la mantisse entière est paire ;
 - *round ties to away* : (2008 – recomm. en base 10 seulement) celui de plus grande valeur absolue.
- **vers $+\infty$** .
- **vers $-\infty$** .
- **vers zéro**.

Une opération dont les entrées sont des nombres VF doit retourner ce qu'on obtiendrait en arrondissant le résultat exact suivant le mode d'arrondi actif.

Arrondi correct

IEEE-754-1985 : Arrondi correct pour $+$, $-$, $\times$, $\div$, $\sqrt{}$ et certaines conversions. Avantages :

- si le résultat d'une opération est exactement représentable, on l'obtient ;
- si on n'utilise que $+$, $-$, $\times$, $\div$ et $\sqrt{}$, l'arithmétique est *déterministe* : algorithmes et preuves qui utilisent ces spécifications ;
- précision et portabilité améliorées ;
- en jouant avec les arrondis vers $+\infty$ et $-\infty \rightarrow$ bornes certaines sur le résultat d'un calcul.

L'arithmétique VF devient une structure mathématique en elle-même, que l'on peut étudier.

IEEE-754-2008 : l'arrondi correct est recommandé (mais pas exigé) pour les principales fonctions mathématiques ($\cos$, $\exp$, $\log$, ...)

Premier exemple : lemme de Sterbenz

Lemme 1 (Sterbenz)

Base β . Soient a et b deux nombres VF positifs. Si

$$\frac{a}{2} \leq b \leq 2a$$

alors $a - b$ est un nombre VF ($\rightarrow$ il est calculé exactement dans n'importe lequel des 4 modes d'arrondi).

Preuve : élémentaire en utilisant la notation $x = M \times \beta^{e+1-p}$.

Erreur de l'addition VF (Møller, Knuth, Dekker)

Premier résultat : représentabilité. $RN(x) = x$ arrondi au plus près.

Lemme 2

Soient a et b deux nombres VF. Soient

$$s = RN(a + b)$$

et

$$r = (a + b) - s.$$

s'il n'y a pas de dépassement de capacité en calculant s , alors r est un nombre VF.

Erreur de l'addition VF (Møller, Knuth, Dekker)

Démonstration : Supposons $|a| \geq |b|$,

- 1** s est « le » nombre VF le plus proche de $a + b \rightarrow$ il est plus près de $a + b$ que a ne l'est. Donc

$|(a + b) - s| \leq |(a + b) - a|$, par conséquent

$$|r| \leq |b|.$$

Erreur de l'addition VF (Møller, Knuth, Dekker)

Démonstration : Supposons $|a| \geq |b|$,

- 1 s est « le » nombre VF le plus proche de $a + b \rightarrow$ il est plus près de $a + b$ que a ne l'est. Donc

$$|(a + b) - s| \leq |(a + b) - a|, \text{ par conséquent}$$

$$|r| \leq |b|.$$

- 2 posons $a = M_a \times \beta^{e_a - p + 1}$ et $b = M_b \times \beta^{e_b - p + 1}$, avec $|M_a|, |M_b| \leq \beta^p - 1$, et $e_a \geq e_b$.

$a + b$ est multiple de $\beta^{e_b - p + 1} \Rightarrow s$ et r sont multiples de $\beta^{e_b - p + 1}$ également $\Rightarrow \exists R \in \mathbb{Z}$ t.q.

$$r = R \times \beta^{e_b - p + 1}$$

mais $|r| \leq |b| \Rightarrow |R| \leq |M_b| \leq \beta^p - 1 \Rightarrow r$ est un nombre VF.

Obtenir r : l'algorithme fast2sum (Dekker)

Théorème 1 (Fast2Sum (Dekker))

$\beta \leq 3$. Soient a et b des nombres VF vérifiant $|a| \geq |b|$.

Algorithme suivant : s et r t.q.

- $s + r = a + b$ exactement ;
- s est « le » nombre VF le plus proche de $a + b$.

Algorithme 1 (FastTwoSum)

```
 $s \leftarrow RN(a + b)$   
 $z \leftarrow RN(s - a)$   
 $r \leftarrow RN(b - z)$ 
```

Programme C 1

```
s = a+b;  
z = s-a;  
r = b-z;
```

Se méfier des compilateurs « optimisants » .

Algorithme TwoSum (Møller-Knuth)

- pas besoin de comparer a et b ;

Algorithme TwoSum (Møller-Knuth)

- pas besoin de comparer a et b ;
- 6 opérations au lieu de 3 $\rightarrow$ moins cher qu'une mauvaise **prédiction de branchement** en comparant a et b .

Algorithme 2 (TwoSum)

```
 $s \leftarrow RN(a + b)$   
 $a' \leftarrow RN(s - b)$   
 $b' \leftarrow RN(s - a')$   
 $\delta_a \leftarrow RN(a - a')$   
 $\delta_b \leftarrow RN(b - b')$   
 $r \leftarrow RN(\delta_a + \delta_b)$ 
```

Algorithme TwoSum (Møller-Knuth)

- pas besoin de comparer a et b ;
- 6 opérations au lieu de 3 $\rightarrow$ moins cher qu'une mauvaise **prédiction de branchement** en comparant a et b .

Algorithme 2 (TwoSum)

```
 $s \leftarrow RN(a + b)$   
 $a' \leftarrow RN(s - b)$   
 $b' \leftarrow RN(s - a')$   
 $\delta_a \leftarrow RN(a - a')$   
 $\delta_b \leftarrow RN(b - b')$   
 $r \leftarrow RN(\delta_a + \delta_b)$ 
```

Knuth : $\forall \beta$, en absence d'underflow et d'overflow $a + b = s + r$, et s est le nombre VF le plus proche de $a + b$.

Algorithme TwoSum (Møller-Knuth)

- pas besoin de comparer a et b ;
- 6 opérations au lieu de 3 $\rightarrow$ moins cher qu'une mauvaise **prédiction de branchement** en comparant a et b .

Algorithme 2 (TwoSum)

```
 $s \leftarrow RN(a + b)$   
 $a' \leftarrow RN(s - b)$   
 $b' \leftarrow RN(s - a')$   
 $\delta_a \leftarrow RN(a - a')$   
 $\delta_b \leftarrow RN(b - b')$   
 $r \leftarrow RN(\delta_a + \delta_b)$ 
```

Knuth : $\forall \beta$, en absence d'underflow et d'overflow $a + b = s + r$, et s est le nombre VF le plus proche de $a + b$.

Boldo et al : (preuve formelle) en base 2, marche même si underflow.

Definition 2

On appelle *Algorithme sans branchements à base de RN-additions* un algorithme

- sans comparaisons, expressions conditionnelles, ou instructions min/max ;
- n'utilisant que des additions/soustractions en arrondi au plus près : à l'étape i l'algorithme calcule $RN(a + b)$ ou $RN(a - b)$ où a et b sont soit des variables d'entrée, soit des valeurs calculées auparavant.

Optimalité de l'algorithme TwoSum

Théorème 2 (Kornerup, Lefèvre, Louvet, M.)

Un algorithme sans branchements à base de RN-additions qui calcule les mêmes résultats que 2Sum demande au moins 6 opérations arithmétiques (i.e., autant que 2Sum).

- démonstration : grand prix de l'inélégance mathématique ;

Optimalité de l'algorithme TwoSum

Théorème 2 (Kornerup, Lefèvre, Louvet, M.)

Un algorithme sans branchements à base de RN-additions qui calcule les mêmes résultats que 2Sum demande au moins 6 opérations arithmétiques (i.e., autant que 2Sum).

- démonstration : **grand prix de l'inélégance mathématique** ;
- 480756 algorithmes à 5 opérations (après suppression des symétries les plus immédiates) ;

Optimalité de l'algorithme TwoSum

Théorème 2 (Kornerup, Lefèvre, Louvet, M.)

Un algorithme sans branchements à base de RN-additions qui calcule les mêmes résultats que 2Sum demande au moins 6 opérations arithmétiques (i.e., autant que 2Sum).

- démonstration : **grand prix de l'inélégance mathématique** ;
- 480756 algorithmes à 5 opérations (après suppression des symétries les plus immédiates) ;
- chacun d'eux essayé avec 2 couples de valeurs d'entrée bien choisis.

Additionner n nombres : compensated summation (Kahan)

Arrondi : $\circ$. On veut calculer $x_1 + x_2 + x_3 + \cdots + x_n$.

Algorithme 3

```
 $s \leftarrow x_1$   
 $c \leftarrow 0$   
for  $i = 2$  to  $n$  do  
   $y \leftarrow \circ(x_i + c)$   
   $(s, c) \leftarrow \text{Fast2Sum}(s, y)$   
end for  
return  $s$ 
```

Table: Erreur des différentes méthodes pour $x_i = \text{RN}(1/i)$ – tous les x_i sont exactement représentables – dans le format Binary32, avec $n = 100,000$. Ici la méthode de Kahan donne le meilleur résultat possible (c.à.d., la somme exacte arrondie au nombre Binary32 le plus proche).

méthode	erreur en « ulps »
ordre croissant	6.86
ordre décroissant	738.9
compensated (Kahan)	0.137

Une autre propriété (Kahan)

$$z = \frac{x}{\sqrt{x^2 + y^2}}$$

- arrondi correct, arrondi au plus près, base 2 ; pas d'overflow/underflow.
- x et y sont des nombres VF,

Une autre propriété (Kahan)

$$z = \frac{x}{\sqrt{x^2 + y^2}}$$

- arrondi correct, arrondi au plus près, base 2 ; pas d'overflow/underflow.
- x et y sont des nombres VF,

La valeur calculée de z est comprise au sens large entre -1 et $+1$.

Une autre propriété (Kahan)

$$z = \frac{x}{\sqrt{x^2 + y^2}}$$

- arrondi correct, arrondi au plus près, base 2 ; pas d'overflow/underflow.
- x et y sont des nombres VF,

La valeur calculée de z est comprise au sens large entre -1 et $+1$.

- Propriété **très importante** car jugée (naïvement !) évidente par la plupart des programmeurs, qui pourront par exemple calculer une fonction de z définie seulement entre -1 et $+1$.
- Le programmeur expérimenté de 2010 rejoint le programmeur naïf de 1980 !

L'algorithme de Malcolm-Gentleman

- est l'un des 4 modes d'arrondi définis auparavant.

Algorithme 4

```
A ← 1.0
B ← 1.0
while ◦(◦(A + 1.0) - A) = 1.0 do
    A ← ◦(2 × A)
end while
while ◦(◦(A + B) - A) ≠ B do
    B ← ◦(B + 1.0)
end while
return B
```

Cet algorithme calcule β

- Base β , précision p ;
- Récurrence $\rightarrow$ si $2^i \leq \beta^p - 1$, alors $A_i = 2^i$ exactement.
Donne $A_i + 1 \leq \beta^p \rightarrow \circ(A_i + 1.0) = A_i + 1$.
- On en déduit $\circ(\circ(A_i + 1.0) - A_i) = \circ((A_i + 1) - A_i) = 1$.
Donc **tant que $2^i \leq \beta^p - 1$, on reste dans la 1ère boucle.**
- Considérons le plus petit j t.q. $2^j \geq \beta^p$. On a
 $A_j = \circ(2A_{j-1}) = \circ(2 \times 2^{j-1}) = \circ(2^j)$. Comme $\beta \geq 2$, on déduit

$$\beta^p \leq A_j < \beta^{p+1}.$$

- Donc le successeur VF de A_j est $A_j + \beta \rightarrow \circ(A_j + 1.0)$ vaudra soit A_j soit $A_j + \beta$ donc $\circ(\circ(A_j + 1.0) - A_j)$ vaudra 0 ou β : dans tous les cas, il sera $\neq 1.0 \rightarrow$ on quitte la boucle.

Donc à la fin de la 1ère boucle **while, A vérifie $\beta^p \leq A < \beta^{p+1}$.**

Cet algorithme calcule β

Considérons la 2ème boucle **while**.

- On a vu que le successeur VF de A est $A + \beta$.
- Donc tant que $B < \beta$, $\circ(A + B)$ vaut soit A soit $A + \beta \rightarrow \circ(\circ(A + B) - A)$ vaut 0 ou β : dans les 2 cas on reste dans la boucle.
- Dès que $B = \beta$, $\circ(A + B)$ vaut exactement $A + B$, donc $\circ(\circ(A + B) - A) = B$.

On quitte donc la 2ème boucle dès que $B = \beta$

On n'a pas besoin d'ordinateurs pour commettre des sottises

- la sonde Mars Climate Orbiter s'est écrasée sur Mars en 1999 ;



On n'a pas besoin d'ordinateurs pour commettre des sottises

- la sonde Mars Climate Orbiter s'est écrasée sur Mars en 1999 ;
- une partie des développeurs des logiciels supposait que l'unité de mesure était le mètre ;



On n'a pas besoin d'ordinateurs pour commettre des sottises

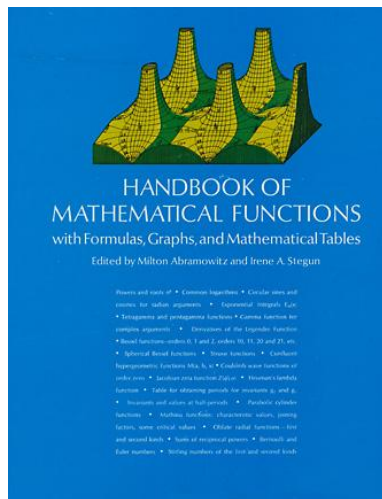
- la sonde Mars Climate Orbiter s'est écrasée sur Mars en 1999 ;
- une partie des développeurs des logiciels supposait que l'unité de mesure était le mètre ;
- l'autre partie croyait que c'était le pied.



Le dilemme du fabricant de tables



Le dilemme du fabricant de tables



Le dilemme du fabricant de tables

Considérons le nombre « decimal64 » ($\beta = 10, p = 16$)

$$x = 9.407822313572878 \times 10^{-2}.$$

On a

$$e^x = \underbrace{1.098645682066338}_{16 \text{ chiffres}} 5 \underbrace{0000000000000000}_{16 \text{ zéros}} 2780 \dots$$

Le dilemme du fabricant de tables

Considérons le nombre « decimal64 » ($\beta = 10, p = 16$)

$$x = 9.407822313572878 \times 10^{-2}.$$

On a

$$e^x = \underbrace{1.098645682066338}_{16 \text{ chiffres}} 5 \underbrace{0000000000000000}_{16 \text{ zéros}} 2780 \dots$$

Et alors ?

Le dilemme du fabricant de tables

Considérons le nombre « decimal64 » ($\beta = 10, p = 16$)

$$x = 9.407822313572878 \times 10^{-2}.$$

On a

$$e^x = \underbrace{1.098645682066338}_{16 \text{ chiffres}} 5 \underbrace{0000000000000000}_{16 \text{ zéros}} 2780 \dots$$

Et alors ?

Pire cas pour l'arrondi de la fonction e^x et des nombres decimal64.

Le tableau de la honte

Système	$\sin(10^{22})$
résultat exact	$-0.8522008497671888017727 \dots$
HP 48 GX	-0.852200849762
HP 700	0.0
HP 375, 425t (4.3 BSD)	$-0.65365288 \dots$
matlab V.4.2 c.1 pour Macintosh	0.8740
matlab V.4.2 c.1 pour SPARC	-0.8522
Silicon Graphics Indy	$0.87402806 \dots$
SPARC	-0.85220084976718879
IBM RS/6000 AIX 3005	$-0.852200849 \dots$
DECstation 3100	NaN
Casio fx-8100, fx180p, fx 6910 G	Error
TI 89	Trig. arg. too large

Jusqu'à août 2008, fonctions « élémentaires » non normalisées.

Le tableau de la honte

Système	$\sin(10^{22})$
résultat exact	$-0.8522008497671888017727 \dots$
HP 48 GX	-0.852200849762
HP 700	0.0
HP 375, 425t (4.3 BSD)	$-0.65365288 \dots$
matlab V.4.2 c.1 pour Macintosh	0.8740
matlab V.4.2 c.1 pour SPARC	-0.8522
Silicon Graphics Indy	0.87402806 $\dots$
SPARC	-0.85220084976718879
IBM RS/6000 AIX 3005	$-0.852200849 \dots$
DECstation 3100	NaN
Casio fx-8100, fx180p, fx 6910 G	Error
TI 89	Trig. arg. too large

Jusqu'à août 2008, fonctions « élémentaires » non normalisées.

Le tableau de la honte

Système	$\sin(10^{22})$
résultat exact	$-0.8522008497671888017727 \dots$
HP 48 GX	-0.852200849762
HP 700	0.0
HP 375, 425t (4.3 BSD)	$-0.65365288 \dots$
matlab V.4.2 c.1 pour Macintosh	0.8740
matlab V.4.2 c.1 pour SPARC	-0.8522
Silicon Graphics Indy	$0.87402806 \dots$
SPARC	-0.85220084976718879
IBM RS/6000 AIX 3005	$-0.852200849 \dots$
DECstation 3100	NaN
Casio fx-8100, fx180p, fx 6910 G	Error
TI 89	Trig. arg. too large

Jusqu'à août 2008, fonctions « élémentaires » non normalisées.

Le tableau de la honte

Système	$\sin(10^{22})$
résultat exact	$-0.8522008497671888017727 \dots$
HP 48 GX	-0.852200849762
HP 700	0.0
HP 375, 425t (4.3 BSD)	$-0.65365288 \dots$
matlab V.4.2 c.1 pour Macintosh	0.8740
matlab V.4.2 c.1 pour SPARC	-0.8522
Silicon Graphics Indy	0.87402806 $\dots$
SPARC	-0.85220084976718879
IBM RS/6000 AIX 3005	$-0.852200849 \dots$
DECstation 3100	NaN
Casio fx-8100, fx180p, fx 6910 G	Error
TI 89	Trig. arg. too large

Jusqu'à août 2008, fonctions « élémentaires » non normalisées.

Arrondi correct des fonctions élémentaires

- base 2, précision p ;
- nombre VF x et entier m (avec $m > p$) $\rightarrow$ approximation y de $f(x)$ dont l'erreur sur la mantisse est $\leq 2^{-m}$.
- peut être fait avec format intermédiaire plus grand, avec des algorithmes de type TwoSum, etc.
- obtenir un arrondi correct de $f(x)$ à partir de y : impossible si $f(x)$ est trop proche d'un point où l'arrondi change (en arrondi au plus près : milieu de 2 nombres VF consécutifs).

Arrondi correct des fonctions élémentaires

- Arrondi au + près,

$$\overbrace{1.\text{xxxxx} \dots \text{xxx}}^{m \text{ bits}} \overbrace{1000000 \dots 000000}^{p \text{ bits}} \text{xxx} \dots$$

ou

$$\overbrace{1.\text{xxxxx} \dots \text{xxx}}^{m \text{ bits}} \overbrace{0111111 \dots 111111}^{p \text{ bits}} \text{xxx} \dots ;$$

- autres modes,

$$\overbrace{1.\text{xxxxx} \dots \text{xxx}}^{m \text{ bits}} \overbrace{0000000 \dots 000000}^{p \text{ bits}} \text{xxx} \dots$$

ou

$$\overbrace{1.\text{xxxxx} \dots \text{xxx}}^{m \text{ bits}} \overbrace{1111111 \dots 111111}^{p \text{ bits}} \text{xxx} \dots .$$

Trouver m au delà duquel il n'y a plus de problème ?

- fonction f : sin, cos, arcsin, arccos, tan, arctan, exp, log, sinh, cosh,

Trouver m au delà duquel il n'y a plus de problème ?

- fonction f : sin, cos, arcsin, arccos, tan, arctan, exp, log, sinh, cosh,
- **Théorème de Lindemann** ($z \neq 0$ algébrique $\Rightarrow e^z$ transcendant) $\rightarrow$ à l'exception de quelques cas triviaux (e^0 , $\ln(1)$, $\sin(0)$, ...), si x est un nombre VF, il existe m_x , t.q. arrondir une approximation sur m_x bits $\Leftrightarrow$ arrondir $f(x)$;

Trouver m au delà duquel il n'y a plus de problème ?

- fonction f : sin, cos, arcsin, arccos, tan, arctan, exp, log, sinh, cosh,
- **Théorème de Lindemann** ($z \neq 0$ algébrique $\Rightarrow e^z$ transcendant) $\rightarrow$ à l'exception de quelques cas triviaux (e^0 , $\ln(1)$, $\sin(0)$, ...), si x est un nombre VF, il existe m_x , t.q. arrondir une approximation sur m_x bits $\Leftrightarrow$ arrondir $f(x)$;
- nombre **fini** de nombres VF $\rightarrow \exists m_{\max} = \max_x(m_x)$ t.q. $\forall x$, arrondir une approximation sur $m_{\max}$ bits de $f(x)$ est équivalent à arrondir $f(x)$;

Trouver m au delà duquel il n'y a plus de problème ?

- fonction f : sin, cos, arcsin, arccos, tan, arctan, exp, log, sinh, cosh,
- **Théorème de Lindemann** ($z \neq 0$ algébrique $\Rightarrow e^z$ transcendant) $\rightarrow$ à l'exception de quelques cas triviaux (e^0 , $\ln(1)$, $\sin(0)$, ...), si x est un nombre VF, il existe m_x , t.q. arrondir une approximation sur m_x bits $\Leftrightarrow$ arrondir $f(x)$;
- nombre **fini** de nombres VF $\rightarrow \exists m_{\max} = \max_x(m_x)$ t.q. $\forall x$, arrondir une approximation sur $m_{\max}$ bits de $f(x)$ est équivalent à arrondir $f(x)$;
- ce raisonnement ne donne aucune idée de l'ordre de grandeur de $m_{\max}$. Pourrait être trop grand pour être d'un quelconque intérêt pratique;

Trouver m au delà duquel il n'y a plus de problème ?

- fonction f : sin, cos, arcsin, arccos, tan, arctan, exp, log, sinh, cosh,
- **Théorème de Lindemann** ($z \neq 0$ algébrique $\Rightarrow e^z$ transcendant) $\rightarrow$ à l'exception de quelques cas triviaux (e^0 , $\ln(1)$, $\sin(0)$, ...), si x est un nombre VF, il existe m_x , t.q. arrondir une approximation sur m_x bits $\Leftrightarrow$ arrondir $f(x)$;
- nombre **fini** de nombres VF $\rightarrow \exists m_{\max} = \max_x(m_x)$ t.q. $\forall x$, arrondir une approximation sur $m_{\max}$ bits de $f(x)$ est équivalent à arrondir $f(x)$;
- ce raisonnement ne donne aucune idée de l'ordre de grandeur de $m_{\max}$. Pourrait être trop grand pour être d'un quelconque intérêt pratique;
- autres fonctions (erf, Γ) : on ne sait (presque) rien.

Un résultat de Baker (1975)

- $\alpha = i/j, \beta = r/s$, with $i, j, r, s < 2^p$;
- $C = 2^{800}$;

$$|\alpha - \log(\beta)| > (p2^p)^{-Cp \log p}$$

Application : Pour évaluer $\ln$ et $\exp$ en double précision ($p = 53$) avec arrondi correct, il suffit de calculer une approximation précise sur environ

Un résultat de Baker (1975)

- $\alpha = i/j, \beta = r/s$, with $i, j, r, s < 2^p$;
- $C = 2^{800}$;

$$|\alpha - \log(\beta)| > (p2^p)^{-Cp \log p}$$

Application : Pour évaluer $\ln$ et $\exp$ en double précision ($p = 53$) avec arrondi correct, il suffit de calculer une approximation précise sur environ

10^{244} bits

Un résultat de Baker (1975)

- $\alpha = i/j, \beta = r/s$, with $i, j, r, s < 2^p$;
- $C = 2^{800}$;

$$|\alpha - \log(\beta)| > (p2^p)^{-Cp \log p}$$

Application : Pour évaluer $\ln$ et $\exp$ en double précision ($p = 53$) avec arrondi correct, il suffit de calculer une approximation précise sur environ

10^{244} bits

Heureusement, en pratique, on constate que c'est beaucoup moins (≈ 100).

Un peu moins pessimiste...

Soit $i/j \in \mathbb{Q}$ avec $\gcd(i, j) = 1$. On définit $H(i/j) = \max\{|i|, |j|\}$.

Théorème 3 (Y. Nesterenko et M. Waldschmidt (1995))

Soit $(\alpha, \beta) \in \mathbb{Q}^2$. Soient A , B , et E des réels positifs, avec $E \geq e$, satisfaisant $A \geq \max(H(\alpha), e)$, $B \geq H(\beta)$. On a

$$\begin{aligned} |e^\beta - \alpha| \geq & \\ \exp\bigl(-211 \times (\ln B + \ln \ln A + 2 \ln(E|\beta|_+) + 10) & \\ \times (\ln A + 2E|\beta| + 6 \ln E) \times (3.3 \ln(2) + \ln E) \times (\ln E)^{-2}\bigr), & \end{aligned}$$

where $|\beta|_+ = \max(1, |\beta|)$.

Application : Pour calculer $\ln$ et $\exp$ en double précision ($p = 53$), il faut faire les calculs intermédiaires sur environ 10^6 bits.

Un peu moins pessimiste...

Soit $i/j \in \mathbb{Q}$ avec $\gcd(i, j) = 1$. On définit $H(i/j) = \max\{|i|, |j|\}$.

Théorème 3 (Y. Nesterenko et M. Waldschmidt (1995))

Soit $(\alpha, \beta) \in \mathbb{Q}^2$. Soient A, B , et E des réels positifs, avec $E \geq e$, satisfaisant $A \geq \max(H(\alpha), e)$, $B \geq H(\beta)$. On a

$$\begin{aligned} |e^\beta - \alpha| \geq & \\ & \exp\left(-211 \times (\ln B + \ln \ln A + 2 \ln(E|\beta|_+) + 10)\right) \\ & \times (\ln A + 2E|\beta| + 6 \ln E) \times (3.3 \ln(2) + \ln E) \times (\ln E)^{-2}, \end{aligned}$$

where $|\beta|_+ = \max(1, |\beta|)$.

Application : Pour calculer $\ln$ et $\exp$ en double précision ($p = 53$), il faut faire les calculs intermédiaires sur environ 10^6 bits.

impossible $\rightarrow$ très cher

Une explication qui n'est pas une preuve...

- la « mantisse infinie » y de $f(x)$ est de la forme :

$$y = y_0.y_1y_2 \cdots y_{p-1} \overbrace{01111111 \cdots 11}^{k\text{bits}} \text{xxxxx} \cdots$$

ou

$$y = y_0.y_1y_2 \cdots y_{p-1} \overbrace{10000000 \cdots 00}^{k\text{bits}} \text{xxxxx} \cdots$$

avec $k \geq 1$.

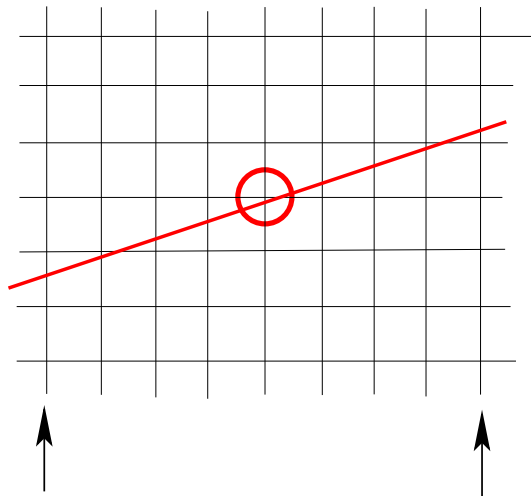
- En supposant qu'après le rang p les chiffres « 1 » et « 0 » ont la même probabilité d'apparaître, la « probabilité » d'avoir $k \geq k_0$ est 2^{1-k_0} ;
- si on considère N nombres VF en entrée, on aura $N \times 2^{1-k_0}$ valeurs pour lesquelles $k \geq k_0$;

→ ne se produira plus dès que k_0 est significativement supérieur à $\log_2(N)$ (pour une valeur donnée de l'exposant, dès que $k_0 \gg p$).

Pires cas en double précision : linéarisation

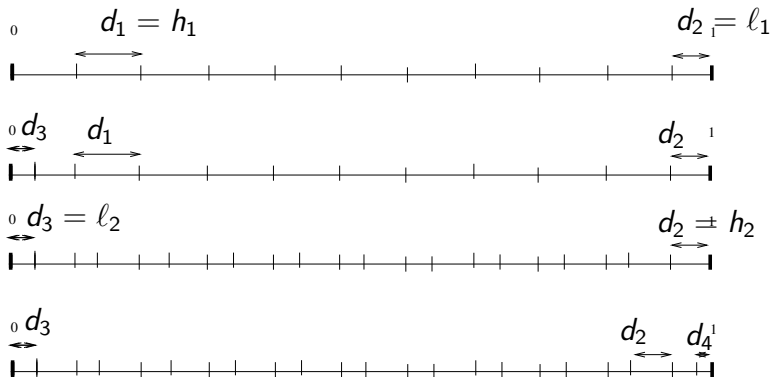
- couper le domaine $\rightarrow$ approximations linéaires des fonctions pour un **pré-filtrage**, pour qu'il ne reste que quelques cas à traiter en très grande précision ;
- pré-filtrage : variante de l'algorithme du PGCD d'Euclide ;
- double précision : pourquoi ?
 - de très loin la plus utilisée ;
 - calculer tous les sinus des 2^{32} nombres simple précision : quelques heures seulement ;
 - plus hautes précisions : semblent hors d'atteinte ;

Pires cas en double précision



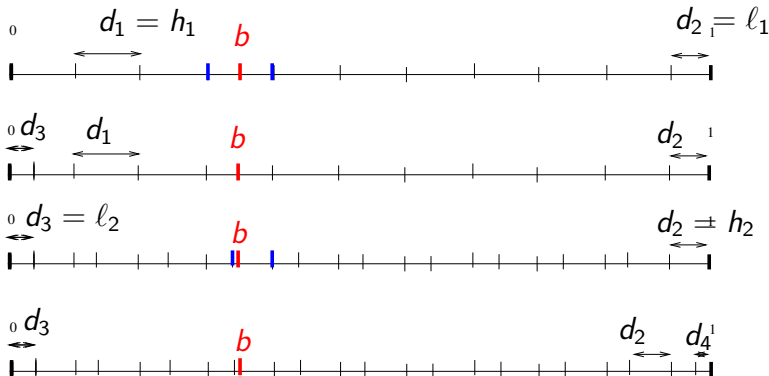
- grille : nombres VF et « milieux » ;
- mise à l'échelle $\rightarrow$ entiers ;
- est ce que la droite est très proche d'un point de la grille ?
- il est très probable que la réponse est « non » .

très petit $an - b \bmod 1$



- une infinité de fois : deux longueurs ;
- $h_{i+1} = \max(h_i - l_i, l_i)$, $l_{i+1} = \min(h_i - l_i, l_i) \rightarrow$ PGCD.

Plus petite distance entre $an \bmod 1$ et b , $n < T$?



- on ne construit pas tous les points : on les compte (pour s'arrêter dès qu'il y en a plus de T) ;
- on ne met à jour que les 2 points qui encadrent b , et on mémorise la distance à celui de gauche ;

Table: *Pires cas pour l'exponentielle de nombres double précision.*

Interval	worst case (binary)
$[-\infty, -2^{-30}]$	$\exp(-1.1110110100110001100011101111101101100010011111101010 \times 2^{-27})$ $= 1.1111111111111111111111111100 \dots 0111000100 \quad 1 \quad 1^{59}0001\dots \times 2^{-1}$
$[-2^{-30}, 0]$	$\exp(-1.0000000000000000000000000000000000000000000000000000000001 \times 2^{-51})$ $= 1.1111111111111111 \dots 11111111111111100 \quad 0 \quad 0^{100}1010\dots \times 2^{-1}$
$(0, +2^{-30}]$	$\exp(1.11111111111111111111111111111111111111111111111111111111111111 \times 2^{-53})$ $= 1.0000000000000000000000000000000000000000000000000000000000 \quad 1 \quad 1^{104}0101\dots$
$[2^{-30}, +\infty]$	$\exp(1.0111111111111111100111111111111111101110000000000100100 \times 2^{-32})$ $= 1.00000000000000000000000000000000000000101111111111111101000 \quad 0 \quad 0^{57}1101\dots$ $\exp(1.10000000000000001011111111111110110111111111111011100 \times 2^{-32})$ $= 1.0000000000000000000000000000000000000011000000000000010111 \quad 1 \quad 1^{57}0010\dots$ $\exp(1.100111101001111001111011111111111011011000001000000001011 \times 2^{-31})$ $= 1.000000000000000000000000000000000000000100111101001110010111 \quad 1 \quad 0^{57}1010\dots$ $\exp(110.00001111010100101111001101111010111011001111110100)$ $= 110101100.01010000101101000000100111001000101011101110 \quad 0 \quad 0^{57}1000\dots$

Table: *Pires cas pour les logarithmes de nombres double précision.*

Interval	worst case (binary)
$[2^{-1074}, 1)$	$\log(1.1110101001110001110110000101110011101110000000100000 \times 2^{-509})$ $= -101100000.00101001011010100110011010110100001011111111 \quad 1 \quad 1^{60}0000...$
	$\log(1.1001010001110110111000110000010011001101011111000111 \times 2^{-384})$ $= -100001001.10110110000011001010111101000111101100110101 \quad 1 \quad 0^{60}1010...$
	$\log(1.0010011011101001110001001101001100100111100101100000 \times 2^{-232})$ $= -10100000.101010110010110000100101111001101000010000100 \quad 0 \quad 0^{60}1001...$
	$\log(1.0110000100111001010101011101110010000000001011111000 \times 2^{-35})$ $= -10111.111100000010111110011011101011110110000000110101 \quad 0 \quad 1^{60}0011...$
$(1, 2^{1024}]$	$\log(1.0110001010101000100001100001001101100010100110110110 \times 2^{678})$ $= 111010110.01000111100111101011101001111100100101110001 \quad 0 \quad 0^{64}1110...$

Conclusion

- l'arrondi correct des fonctions les plus communes est faisable à coût raisonnable ;
- recommandé dans la norme IEEE 754-2008 (août 2008) ;
- **bibliothèque CRLIBM** disponible à <https://lipforge.ens-lyon.fr/projects/crlibm/>

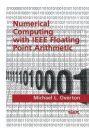
Quelques logiciels utiles

- CRLIBM : fonctions mathématiques avec arrondi correct.
<http://lipforge.ens-lyon.fr/projects/crlibm/>
La documentation qui explique les méthodes utilisées est à
<http://lipforge.ens-lyon.fr/frs/download.php/41/crlibm-0.10.pdf>
- GAPPA : outil de vérifications de propriétés VF (p.ex. bornes) et de génération de preuves formelles.
<http://lipforge.ens-lyon.fr/www/gappa/>
- MPFR : arithmétique multi-précision avec arrondi correct.
<http://www.mpfr.org/>
- PARI/GP : calculs rapides en arithmétique (factorisations, théorie algébrique des nombres, courbes elliptiques...).
<http://pari.math.u-bordeaux.fr/>

La virgule flottante sur le web

- le site de W. Kahan (père de la norme IEEE 754, de l'arithmétique du 8087 et de la HP35) :
- l'équipe Arénaire du LIP (ENS Lyon)
`http://www.ens-lyon.fr/LIP/Arenaire/`
- l'équipe CACAO du Loria (Nancy)
`http://www.loria.fr/equipes/cacao/`
- ma propre page
`http://perso.ens-lyon.fr/jean-michel.muller/`

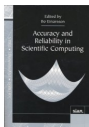
Livres sur l'arithmétique virgule flottante



Michael Overton

Numerical Computing with IEEE Floating Point Arithmetic

Siam, 2001



Bo Einarsson

Accuracy and Reliability in Scientific Computing

Siam, 2005



Jean-Michel Muller

Elementary Functions, algorithms and implementation, 2ème édition

Birkhauser Boston, 2006



Brisebarre, de Dinechin, Jeannerod, Lefèvre, Melquiond, Muller (coordinator), Revol, Stehlé and Torres

A Handbook of Floating-Point Arithmetic

Birkhauser Boston, 2010.