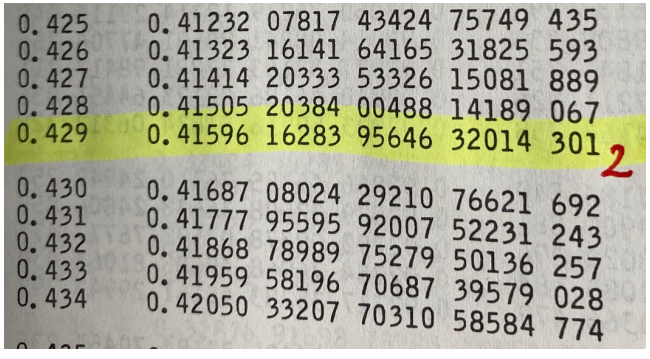
Le dilemme du fabricant de tables

Jean-Michel Muller

CNRS – Laboratoire LIP
Mai 2026

http://perso.ens-lyon.fr/jean-michel.muller/



0.425	0.	41232	07817	43424	75749	435
0.426	0.	41323	16141	64165	31825	593
0.427	0.	41414	20333	53326	15081	889
0.428	0.	41505	20384	00488	14189	067
0.429	0.	41596	16283	95646	32014	301
0.430	0.	41687	08024	29210	76621	692
0.431	0.	41777	95595	92007	52231	243
0.432	0.	41868	78989	75279	50136	257
0.433	0.	41959	58196	70687	39579	028
0.434	0.	42050	33207	70310	58584	774

$$2^{43}\cdot\log\left(3119607361246427\cdot 2^{627}\right) = 4136624566035640.5000000000000000000122\cdots$$

Ce qui m'intéresse…

Mettre en évidence et utiliser la structure qu'il y a dans l'arithmétique de nos ordinateurs… et en ajouter.

- le résultat de `c = a + b`, ou `c = a / b` est déterminé sans ambigüité, celui de `y = sin(x)` doit l'être ;
- permet de prouver des propriétés subtiles de programmes ;
- favorise la validation, le test, le debugging, la portabilité et la réutilisabilité des programmes.

… et tout ceci est lié à des «coïncidences» telles que

$$2^{53} \cdot \cos\left(\frac{5307249884967}{2^{41}}\right) = -6723126034308461.4999999999999999898\cdots$$

$$2^{43} \cdot \log\left(3119607361246427 \cdot 2^{627}\right) = 4136624566035640.5000000000000000000122\cdots$$

Arithmétique virgule flottante (VF) binaire

- de très loin la + utilisée pour du calcul numérique ;
- système VF de base 2 :

$$\begin{cases} \text{précision} & p \geq 1 \\ \text{exposants extrêmes} & e_{\min}, e_{\max} \end{cases}$$

Nombre VF x : mantisse M et exposant e, $(M, e) \in \mathbb{Z}$, $|M| \leq 2^p - 1$ et $e_{\min} \leq e \leq e_{\max}$, t.q.

$$x = \left(\frac{M}{2^{p-1}} \right) \times 2^e$$

Si plusieurs choix, on prend e minimal sous ces contraintes.
Taille : $w = \lceil \log_2 (e_{\max} - e_{\min} + 1) \rceil + p$.

nom officiel	nom d'usage	w	p	$e_{\min}$	$e_{\max}$
binary16		16	11	-14	15
binary32	simple précision	32	24	-126	127
binary64	double précision	64	53	-1022	1023
binary128	quad. précision	128	113	-16382	16383

Le monde du calcul virgule flottante jusqu'au milieu des années 1980…

- environnements très différents (précision, base, e_{min}, e_{max}, exceptions…) ;
- rarement bons, mais surtout aucune spécification :
  - médiocre portabilité du logiciel numérique ;
  - impossibilité de prouver/garantir quoi que ce soit : que prouver si on ne sait même pas ce que donne `c = a+b` ?
  - besoin de connaître des «trucs», comme pour x proche de 1, remplacer `x-1` par `(x-0.5)-0.5` ;
  - fonctions mathématiques : même l'ordre de grandeur était parfois incorrect.

Bref… de la cuisine ! (et rarement de la bonne)

Le standard IEEE-754 (première mouture : 1985)

Spécifie des formats, la gestion des exceptions, et requiert l'arrondi correct pour certaines opérations.

Definition 1 (Arrondi correct)

On se donne une *fonction d'arrondi* $\circ$ de $\mathbb{R}$ vers les nombres VF parmi :

- $\circ_n(x)$: au plus près (défaut), avec règle de départage si x est le milieu de 2 nombres VF consécutifs ;
- $\circ_u(x)$: vers $+\infty$;
- $\circ_d(x)$: vers $-\infty$;
- $\circ_z(x)$: vers zéro.

Une opération dont les entrées sont des nombres VF est correctement arrondie si le résultat retourné est ce qu'on obtiendrait en appliquant la fonction d'arrondi au résultat exact.

La norme IEEE-754 de 1985

Arrondi correct pour $+$, $-$, $\times$, $\div$, $\sqrt{}$ et certaines conversions.

En supposant que la fonction d'arrondie choisie est $\circ_n$:

$$\text{programme } \texttt{c = a+b} \rightarrow \text{calcul } c = \circ_n(a+b)$$

Avantages :

- si on n'utilise que $+$, $-$, $\times$, $\div$ et $\sqrt{}$, et si l'ordre des opérations ne change pas, arithmétique déterministe $\rightarrow$ algorithmes et preuves ;
- précision et portabilité améliorées ;
- $\forall t,\ |\circ_n(t) - t| \leq \frac{1}{2^{p+1}} \cdot |t| \rightarrow$ majorant de l'erreur relative de $+$, $-$, $\times$, $\div$ et $\sqrt{}$.

Mise à jour de 2008 : arrondi correct recommandé mais pas exigé pour un petit nombre de fonctions mathématiques : cos, exp, log, ...

Restent quelques ambigüités. Par exemple format de `b-c` dans `a*(b-c)`

Quelques petits exemples

Programme	Ce qui est réellement calculé	Propriété démontrable

`s = a+b;`
`z = s-a;`
`t = b-z;`
	$s = \circ_n(a+b)$
	$z = \circ_n(s-a)$
	$t = \circ_n(b-z)$
		si $\lvert a\rvert \geq \lvert b\rvert$ alors
		$t = (a+b) - s$
		(t est l'erreur de l'addition virgule flottante de a et b)

`z = x / sqrt(x*x+y*y)`	$z = \circ_n\left(\frac{x}{\circ_n\left(\sqrt{\circ_n\left(\circ_n(x^2) + \circ_n(y^2)\right)}\right)}\right)$	$\lvert z\rvert \leq 1$

`w = b*c;`
`e = fma(-b,c,w);`
`f = fma(a,d,-w);`
`x = f+e`
	$w = \circ_n(bc)$
	$e = \circ_n(w - bc)$
	$f = \circ_n(ad - w)$
	$x = \circ_n(f+e)$
		$e = w - bc$ (exact)
		et
		$\left\lvert \frac{x-(ad-bc)}{ad-bc} \right\rvert \leq 2^{-p+1}$

Note : `fma(a,b,c)` vaut $\circ_n(ab+c)$.

Proposer un ensemble de fonctions élémentaires correctement arrondies ?

Buts : portabilité, précision, reproductibilité, même majorant $\frac{1}{2^p+1}$ de l'erreur relative que pour les opérations.

Plusieurs problèmes à résoudre :

- quelles fonctions ?
- est-il possible de les calculer efficacement *et* avec arrondi correct ?
- dans les grands formats (64 ou 128 bits), comment garantir que nos programmes fournissent l'arrondi correct ?

  (petits formats : énumérer toutes les entrées possibles)

Le programme calculant l'exponentielle, le sinus, … sera appelé des milliards de fois :

→ Sa conception peut prendre du temps ;

→ son exécution doit être rapide.

Construction d'un noyau de fonctions de précision «ultime»

Bien plus appelées (expériences de Piparo and Innocente, du CERN), et servent à construire les autres :

$$e^x, \log(x), x^y,$$
$$\sin(x), \cos(x), \tan(x), \arcsin(x), \arccos(x), \arctan(x).$$

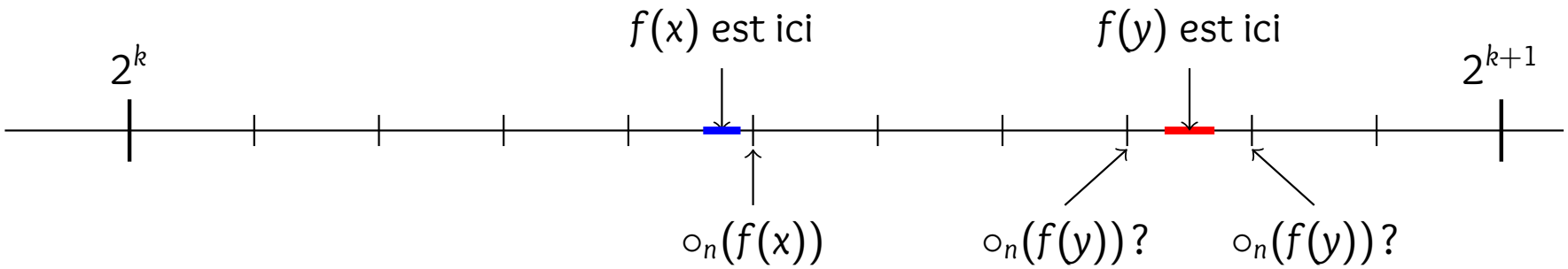
Programme de travail (étalé sur 20 ans) :

- «presque meilleures» approximations polynomiales dont les coefficients sont des nombres VF ;
- bornes fines et certaines des erreurs de ces approximations polynomiales ;
- bornes fines et certaines de l'erreur d'évaluation de ces polynômes ;
- avec quelle précision approcher la fonction pour être certain qu'arrondir l'approximation est équivalent à arrondir la valeur exacte ? Cette question : dilemme du fabricant de table (TMD).

Nicolas Brisebarre ; Sylvain Chevillard ; Florent de Dinechin ; Guillaume Hanrot ; Mioara Joldes ; Christoph Lauter ; Vincent Lefèvre ; Erik Martin-Dorel ; Guillaume Melquiond ; Arnaud Tisserand ; Paul Zimmermann.

Le dilemme du fabricant de tables

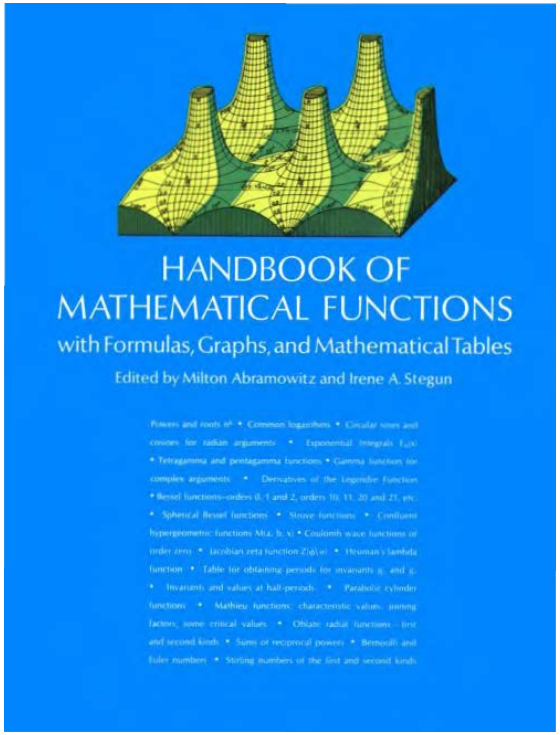
- Points de seuil : discontinuités de la fonction d'arrondi. Pour $\circ_n$: milieux de deux nombres VF consécutifs ;
- approximation de $f(x)$ à une certaine précision $\rightarrow$ $f(x)$ est dans un intervalle ;
- cet intervalle contient-il un point de seuil ?



- si oui, l'approximation ne suffit pas pour connaître $\circ_n(f(x))$… on peut décider de continuer avec une approximation plus précise, mais…
- ce processus s'arrête-t-il toujours ? Et quand ?
- en 1971, W. Kahan forge le nom de Table Maker's Dilemma, pour un problème qu'il estime insoluble.

Le dilemme du fabricant de tables s'est posé… aux concepteurs de tables

Le dilemme du fabricant de tables s'est posé… aux concepteurs de tables



Abramowitz & Stegun, Handbook of Math. Functions (1964) :

sin(0.429) :

- valeur tabulée :
  `0. 41596 16283 95646 32014 301`
- valeur exacte :

  $0.4159616283956463201430 1\,500372652\cdots$

sin(1.108) :

- valeur tabulée :
  `0. 89480 75718 42671 89157 146`
- valeur exacte :

  $0.8948075718426718915714 6\,50062808\cdots$

Détection de plagiat !

En arrondi au plus près (le mode par défaut)

En supposant $f(x) \in [2^k, 2^{k+1}[$, et en approchant $f(x)$ avec une erreur pouvant aller jusqu'à $2^{1+k-p-m}$, on ne saura pas arrondir $f(x)$ lorsque

$$f(x) = 2^k \times \underbrace{1.xxxxx\cdots xxx}_{p \text{ bits}} \overbrace{1000000\cdots000000}^{m \text{ bits}} xxx\cdots$$

ou

$$f(x) = 2^k \times \underbrace{1.xxxxx\cdots xxx}_{p \text{ bits}} \overbrace{0111111\cdots111111}^{m \text{ bits}} xxx\cdots;$$

«Pires cas» en «simple précision» (32 bits, $p = 24$)

f	x (décimal)	mantisse de $f(x)$ (en binaire)
exp	11615567×2^{-33}	$1.00000000010110001010111\underbrace{0000000000000000000000000}_{29}10\cdots$
sin	1258291×2^{-7}	$-1.01100011111101001011101\underbrace{1111111111111111111111111}_{30}01\cdots$
cos	12860426×2^{28}	$1.10000100101111101100010\underbrace{1111111111111111111111111}_{31}01\cdots$
arccos	4272401×2^{-34}	$1.10010010000011110110100\underbrace{0000000000000000000000000}_{33}10\cdots$
log	14192851×2^{53}	$1.10101001101000111111000\underbrace{0000000000000000000000000}_{34}11\cdots$
Γ	14583465×2^{-71}	$1.00100110100000100110011\underbrace{1111111111111111111111111}_{32}01\cdots$

Plus grandes valeurs de m observées : toutes proches de 32… est-ce un hasard ?

Approximation avec une erreur $\leq 2^{-p-k+1}$ sur la mantisse de $f(x)$

$\frac{f(x)}{2^{e_{f(x)}}} = \boxed{1.xxxx \cdots xxxxxxx}$

p bits, suivis de :

$k=2$	$k=3$	$k=4$
00	000	0000
01	001	0001
10	010	0010
11	011	0011
	100	0100
	101	0101
	110	0110
	111	0111
		1000
		1001
		1010
		1011
		1100
		1101
		1110
		1111

Proportion des cas conduisant à un échec : $1/2$ $1/4$ $1/8$

k bits $\rightarrow$ proportion des cas d'échec 2^{1-k}

Heuristique probabiliste

- format sur w bits $\rightarrow$ 2^w entrées possibles ;
- on s'attend à ce que le nombre de nombres VF x pour lesquels on a une chaîne de m bits de la forme $01111\cdots111$ ou $1000\cdots000$ après le $p^{\text{ème}}$ bit dans l'écriture binaire de la mantisse de $f(x)$ soit de l'ordre de

$$2^{w+1-m}.$$

$\rightarrow$ on s'attend à ce que la plus grande valeur de m soit environ $w+1$.

Années 90 : Gal-Bachelis ; Dunham ; Lefèvre-M.-Tisserand

Justification rigoureuse pour $m < p/3$: Brisebarre, Hanrot & Robert, *Exponential sums and correctly-rounded functions*, IEEE Trans. Computers, Déc. 2017.

Quelques résultats tirés de la théorie des nombres

- Hermite-Lindemann (1882) : z algébrique $\neq 0 \Rightarrow e^z$ transcendant $\rightarrow$ pour $f = \exp, \log, \sin, \cos, \tan, \arctan, \arcsin, \arccos \ldots$, si x est un nombre VF, $f(x)$ n'est jamais un point de seuil.
- Nesterenko-Walschmidt (1995) :
  Soient $\alpha = p/q$ et $\alpha' = p'/q' \in \mathbb{Q}$. Soient $\theta, A, A', E \in \mathbb{R}^*$ tels que

$$E \geq e, \quad A \geq \max(p, q, e), \quad A' \geq \max(p', q').$$

  On a

$$\begin{aligned} &\left|e^\theta - \alpha\right| + |\theta - \alpha'| \geq \\ &\exp\Big\{-211 \cdot \Big(\ln A' + \ln\ln A + 2\ln(E \cdot \max\{1, |\theta|\}) + 10\Big) \\ &\cdot \Big(\ln A + 2E|\theta| + 6\ln E\Big) \cdot \Big(3.7 + \ln E\Big) \cdot \Big(\ln E\Big)^{-2}\Big\}. \end{aligned}$$

  Donne des résultats pour exp, log, cos, sin, tan, arctan, etc.
  $\rightarrow$ approximations intermédiaires sur quelques millions de bits ;
- Khémira-Voutier (2011) :
  approximations intermédiaires sur quelques dizaines de milliers de bits.

C’est quand même frustrant…

- approximations intermédiaires sur quelques dizaines de milliers de bits : faisable, mais cher…

- et en plus on «sait» que la plus grande valeur de m vaut environ w…

Trouver un moyen de calculer cette plus grande valeur de m ?

Tests exhaustifs : 2^w valeurs à tester. Trivial pour $w \leq 32$, accessible à l’humanité mais d’un coût déraisonnable pour $w = 64$, impensable pour les plus grandes valeurs.

Stratégie adoptée pour $w \geq 64$

- découper le domaine initial en N (petits) sous-domaines ;
- dans chaque sous-domaine, approximation polynomiale de f ;
- degré 1 : Lefèvre, M., Tisserand (2000), premiers «pires cas» pour la double précision (exponentielle et log dans tout le domaine) :
  - coût $\approx \mathcal{O}\left(2^{w-p/3}\right)$;
  - a permis de construire la bibliothèque «preuve de concept» CRLibm ;
  - reste très cher (mois de CPU en double précision, impensable en plus hautes précision).

- plus hauts degrés : algorithme de Stehlé-Lefèvre-Zimmermann (SLZ).

Trouver les pires cas… ou pas. L'algorithme SLZ

- on se ramène à $f : \left[\frac{1}{2}, 1\right] \to \left[\frac{1}{2}, 1\right]$;
- sous domaine $\left[\frac{X_0 - a}{2^p}, \frac{X_0 + a}{2^p}\right]$ de $\left[\frac{1}{2}, 1\right]$, f remplacée par polynôme F ;
- on cherche les éventuelles solutions de l'équation

$$2^p \cdot F\left(\frac{X}{2^p}\right) - \frac{1}{2} - z = 0 \mod 1;$$

  avec $X \in \mathbb{Z}$, $|X| \leq a$ et $|z| \leq \frac{1}{M}$, M grand entier (typiquement 2^p ou 2^w) ;
- on se débarrasse des dénominateurs

$$P(u, v) = 0 \mod R,$$

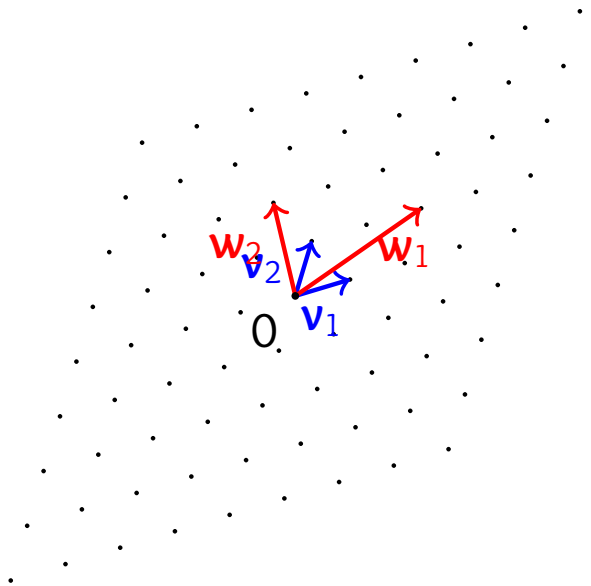
  $P \in \mathbb{Z}[X]$, $(u, v) \in \mathbb{Z}^2$, $|u| < U$, $|v| < V$.

Réseaux euclidiens

- $b_1, b_2, \ldots, b_d$ vecteurs linéairement indépendants de $\mathbb{R}^n \rightarrow$ réseau engendré

$$\mathcal{L} = \sum_{i \leq d} k_i \cdot b_i, \quad k_i \in \mathbb{Z};$$

- Objet très utile (p.ex. en cryptographie) ;



- Problème difficile : trouver un petit vecteur dans $\mathcal{L}$.
- Algorithme LLL (Lenstra, Lenstra and Lovász, 1982) $\rightarrow$ base $(c_1, c_2, \ldots, c_d)$ de relativement petits vecteurs en temps polynomial en la dimension d du réseau.
- En particulier, $||c_1|| \leq 2^{(d-1)/2}\lambda(L)$, où $\lambda(L)$ est la plus petite norme d'un vecteur non nul de $\mathcal{L}$.

Trouver les pires cas…ou pas. L'algorithme Stehlé-Lefèvre-Zimermann

$$P(u,v) = 0 \mod R, \tag{1}$$

$P \in \mathbb{Z}[X]$, $(u,v) \in \mathbb{Z}^2$, $|u| < U$, $|v| < V$.

- méthode de Coppersmith : pour un paramètre α donné, on construit une famille

  $$P_{i,j,k}(u,v) = R^{\alpha-i} P(u,v)^i u^j v^k$$

  $(i \leq \alpha, j \leq j_{max}, k \leq k_{max})$;
- (1) implique

  $$P_{i,j,k}(u,v) = 0 \mod R^\alpha.$$

  Si on trouve une combinaison linéaire (à coefficients entiers) $Q = \sum q_{m,n} u^m v^n$ des $P_{i,j,k}$ telle que

  $$\sum |q_{m,n}| \cdot U^m V^n < R^\alpha,$$

  alors (1) $\Rightarrow Q(u,v) = 0$ dans les entiers.
- trouver des petits vecteurs est précisément ce que LLL sait faire !
- deux tels polynômes Q_1 et Q_2 : on résout le système $Q_1(u,v) = 0, Q_2(u,v) = 0$.

D. Stehle, V. Lefevre & P. Zimmermann, *Searching worst cases of a one-variable function using lattice reduction,* IEEE Trans. on Computers, Mars 2005

Quelques remarques sur cet algorithme SLZ

- coût pour une valeur de l'exposant d'entrée : [1]

$$\text{Poly}\,(p + \log(M)) \cdot 2^{\frac{p^2}{p+\log(M)}};$$

- recherche de pires cas : $M \approx 2^p$ (heuristique probabiliste) $\rightarrow$ coût en $2^{p/2}$.

$\rightarrow$ pires cas maintenant connus [2] en double précision pour

exp, log, exp2, log2, exp10, log10, log1p, log2p1, log10p1, expm1, exp2m1, exp10m1, rsqrt, sin, cos, tan, sinpi, cospi, tanpi, acospi, asinpi, atanpi, cosh, sinh, tanh, acosh, asinh, atanh, acos, asin, atan, cbrt, erf, erfc.

$\rightarrow$ bibliothèque CORE-MATH (P. Zimmermann) : toutes les fonctions 32 et 64 bits du standard C. Très efficace.

- Reste compliqué pour la quadruple précision.

1. D. Stehlé, On the randomness of bits generated by sufficiently smooth functions, Algorithmic Number Theory. ANTS 2006.
2. `https://www.reliable-computing.org/correctrounding.html`

Des espoirs en quadruple précision ?

- coût pour une valeur de l'exposant d'entrée :

$$\mathrm{Poly}\,(p + \log(M)) \cdot 2^{\frac{p^2}{p+\log(M)}} ;$$

- Compliqué pour la quadruple précision… mais si on accepte $M \gg 2^p$: résultats du style *avec des résultats intermédiaires sur $6p$ (voire p^2) bits, alors on peut toujours arrondir correctement*, peut devenir jouable ;
- Problème :
  - version usuelle ($M \approx 2^p$) : «pires cas» que l'on peut tester ;
  - version «grand M» : algorithme compliqué, programme fortement optimisé, qui au bout de jours de calcul, répond juste «oui» ou «non»… quelle confiance ?

→ nécessité absolue de preuve formelle !

  - prouver formellement l'algorithme, le programme : peu d'espoir ;
  - approche par certificat basée sur Q_1 et Q_2.

Double précision : Plus grande erreur (en ulps), février 2023 et février 2026

Pour $x \in \mathbb{R}$, $\mathrm{ulp}(x)$ = distance entre les 2 nombres VF qui encadrent x.
f arrondie correctement : erreur $\leq \frac{1}{2}\,\mathrm{ulp}(f(t))$.
Il y a encore 20 ans, $5-6$ ulps était considéré comme une erreur honorable.

	GNU libc 2.37	GNU libc 2.43	IML 2023.0.0	IML 2025.0.0	LLVM 21.1.8
acos	0.523	0.523	0.531	0.531	0.500
acosh	2.25	0.500	0.509	0.509	3.22
asin	0.516	0.516	0.531	0.531	0.500
asinh	1.92	0.500	0.507	0.507	2.05
atan	0.523	0.523	0.528	0.528	1.00
atanh	1.81	0.500	0.507	0.507	2.50
cos	0.516	0.516	0.518	0.518	0.500
cosh	1.93	1.93	0.516	0.516	1.42
erfc	5.19	0.500	0.505	0.827	5.85
exp	0.511	0.511	0.530	0.530	0.500
log	0.520	0.520	0.518	0.518	0.500
sin	0.516	0.516	0.518	0.518	0.500
pow	0.523	0.523	1.73	1.73	Inf

Source : Gladman et al., *Accuracy of Mathematical Functions in Single, Double, Extended Double and Quadruple Precision*, `https://inria.hal.science/hal-03141101`

Scoop : π est rationnel !

Les horreurs n'ont pas toutes disparu…

Si vos enfants ont une calculette Casio FX-92 :

- calculez $11^6/13$, vous obtiendrez

$$\frac{156158413}{3600}\pi;$$

- calculez 97027288/89521, vous obtiendrez

$$345\pi.$$