Un résultat faux tout de suite ou correct trop tard ?

Vincent Lefèvre et Jean-Michel Muller

CNRS, Laboratoire LIP

École normale Supérieure de Lyon

46 Allée d'Italie, 69364 Lyon Cedex 07

FRANCE

Mars 1999

Corollaire de la loi de Murphy

- arithmétiques « virgule flottante » : rapides, mais résultats souvent imprécis ;
- arithmétiques « multi-précision », « exactes », calcul symbolique : le bon résultat... mais bien trop tard !

Quelques exemples

$$\begin{cases} u_0 & = & 2 \\ u_1 & = & -4 \\ u_{n+1} & = & 111 - \frac{1130}{u_n} + \frac{3000}{u_n u_{n-1}} \end{cases}$$

- u_n converge vers 6 $\left(u_n = \frac{4\times 5^{n+1} - 3\times 6^{n+1}}{4\times 5^n - 3\times 6^n}\right)$
- sur n'importe quelle machine u_n semble converger très vite vers 100

```
diverge := proc(a)
 x := a; for i to 100 do
     x := (3*x^4-20*x^3+35*x^2-24)/(4*x^3-30*x^2+70*x-50)
 od end
a:=1.510005072136258: Digits:=10: diverge(a);
                               4.000000033
Digits:=12: diverge(a);        1.00000000000
Digits:=14: diverge(a);        3.0000000000000
Digits:=16: diverge(a);        4.000000000000033
Digits:=18: diverge(a);        1.00000000000000000
```

Essai sur 10, puis 16 chiffres $\longrightarrow$ confiance totale en le résultat !

Dans le même genre (S. Rump)

$$f(a,b) = 333.75b^6 + a^2\left(11a^2b^2 - b^6 - 121b^4 - 2\right) + 5.5b^8 + \frac{a}{2b}$$

avec $a = 77617.0$ et $b = 33096.0$.

Sur un IBM 370 :

simple précision :	1.172603
double précision :	1.1726039400531
précision étendue :	1.172603940053178

bonne valeur : $-0.8273960599\ldots$

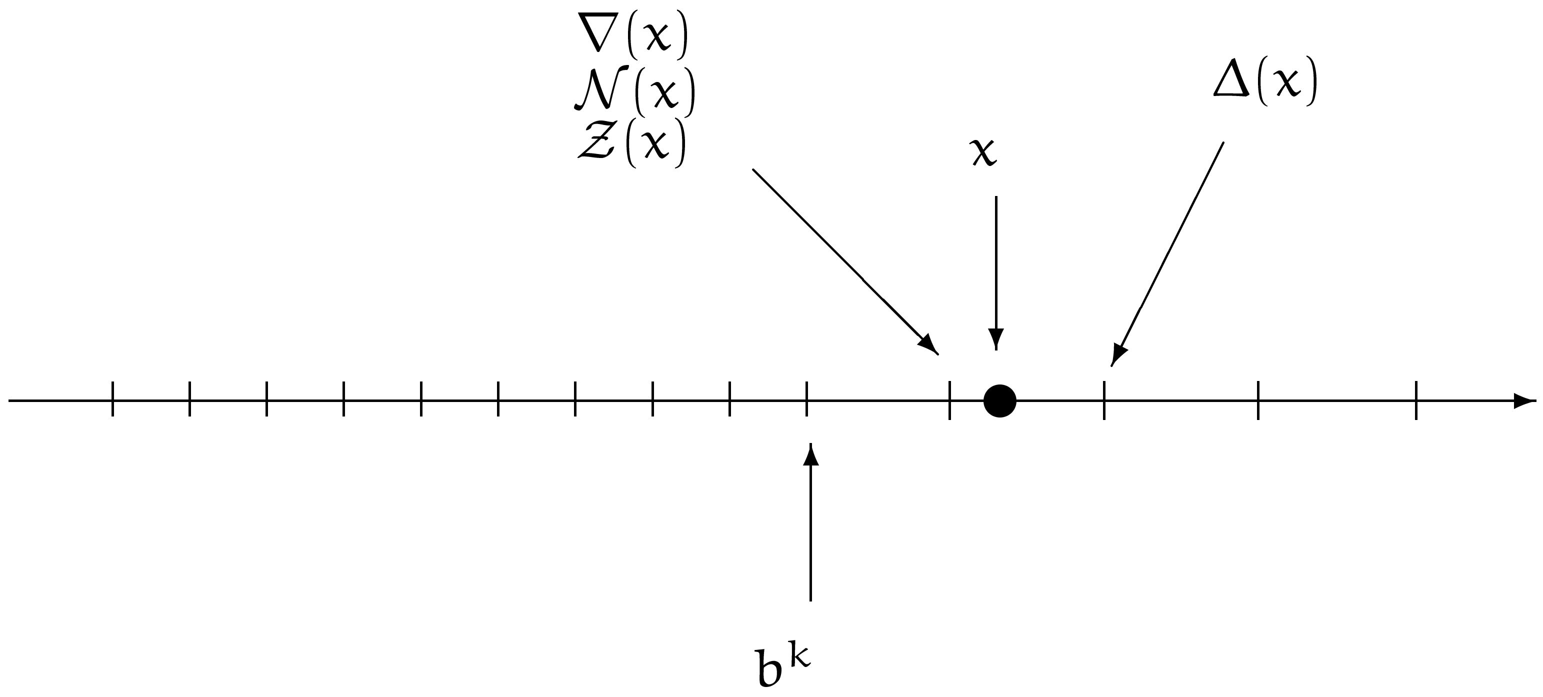
Autre chose que de la cuisine

- Virgule flottante presque toujours utilisée pour représenter des réels ;
- principales caractéristiques : base b, taille n de la mantisse, plage des exposants $E_{\text{min}} \ldots E_{\text{max}}$ et le mode d'arrondi actif

$$x = x_0.x_1x_2 \cdots x_{n-1} \times b^{e_x}$$

Modes d'arrondi

- nombre machine exactement représentable dans le système utilisé ;
- en général, la somme, le produit, le quotient de 2 nombres machines *ne sont pas* des nombres machine : il faut les arrondir. Modes d'arrondi usuels :
  - vers $-\infty$: $\nabla(x)$ est le plus grand nombre machine $\leq x$;
  - vers $+\infty$: $\Delta(x)$ est le plus petit nombre machine $\geq x$;
  - vers 0 : $\mathcal{Z}(x)$ vaut $\nabla(x)$ si $x \geq 0$, et $\Delta(x)$ si $x < 0$;
  - au plus près : $\mathcal{N}(x) =$ nombre machine le plus proche de x.


∇(x)
𝒩(x)
𝒵(x)
x
Δ(x)
b^k

Exemples

	base	n	e_{min}	e_{max}	dynamique
Cray-1	2	48	-8192	8191	$\approx 10^{\pm 2465}$
	2	96	-8192	8191	$\approx 10^{\pm 2465}$
DEC VAX	2	24	-127	127	$\approx 10^{\pm 38}$
	2	56	-127	127	$\approx 10^{\pm 38}$
IBM 370	16	6 (24 bits)	-64	63	$\approx 10^{\pm 75}$
	16	14 (56 bits)	-64	63	$\approx 10^{\pm 75}$
IEEE-754	2	23+1	-126	+127	$\approx 10^{\pm 38}$
	2	52+1	-1022	+1023	$\approx 10^{\pm 308}$

norme IEEE-754 (1985)

- caractéristiques les plus connues : *formats* des représentations
- caractéristique la plus intéressante : « arrondi exact »
- mode d'arrondi *actif* : $\diamond$
- x et y : nombres machine

Quand on calcule $x \star y$ ($\star$ étant $+$, $-$, $\times$ ou $\div$), le résultat obtenu doit *toujours* être $\diamond(x \star y)$, i.e. *l'arrondi du résultat exact*.

Conséquences

– compatibilité : même programme $\longrightarrow$ mêmes résultats sur différentes machines ;

– des algorithmes & preuves peuvent être construits en utilisant cette propriété (arithmétique à précision arbitraire, somme de plusieurs nombres machine, prise de décisions en géométrie. . .) ;

– arithmétique d'intervalles, minorations & majoration de résultats, . . .

recettes de cuisine $\longrightarrow$ structure mathématique

hélas : rarement accessible dans un langage de haut niveau, et pas encore de spécification pour les fonctions élémentaires.

Exemple de propriété de la norme

Sur toute machine compatible avec la norme IEEE le calcul de

$$X' = X/\sqrt{X^2 + Y^2}$$

où X et Y sont des nombres machine, donnera toujours un résultat compris entre -1 et $+1$.

Une telle propriété peut être cruciale, car le concepteur d'un logiciel peut utiliser ce résultat qui lui semble intuitif. Ce résultat est faux sur de nombreuses machines Cray.

Algorithme de Gentleman

A et B déclarés comme *nombres réels* (Virgule Flottante).

$A \leftarrow 1$

$B \leftarrow 1$

tant que $((A+1)-A)-1 = 0$ *faire* $A \leftarrow 2 \times A$

tant que $((A+B)-A)-B \neq 0$ *faire* $B \leftarrow B+1$

imprimer (B)

$\longrightarrow$ base du système de numération *interne* de votre machine

(calculettes : 10, autres machines : 2 ou 16)

Somme et erreur (Priest)

```
proc sum_and_error(a,b)
   begin
      c := a+b;   e := c-a;
      f := b-e;   g := c-e;
      h := g-a;   d := f-h;
      return (c,d)
   end;
```

donnera $c + d = a + b$, et soit $c = d = 0$, soit $|d| <$ dernier_bit (c) sur une arithmétique avec « arrondi exact » (marche sur une machine compatible IEEE, pas par exemple sur un Cray).

Une dernière propriété

– Sur une machine conforme à la norme IEEE, quand le résultat exact « tombe juste » (par exemple si c'est un entier), on l'obtient sans erreur.

– Sur une machine non conforme (ex. certains Cray), parfois faux. Sur certaines de ces machines, l'instruction FORTRAN `I = 14.0/7.0` donnera comme résultat 1. Sur certains Cray, il y a des nombres représentables x tels que le calcul $1 \times x$ provoque un dépassement.

Et les fonctions élémentaires ?

Environnement	Valeur calculée de $\sin 10^{22}$
Résultat exact	$-0.8522008497671888017727\ldots$
HP 28S, 48 SX	$-0.852200849\ldots$
HP 700	0.0
HP 375, 425t (4.3 BSD)	$-0.65365288\ldots$
Sun3, NeXt	$-0.65365288\ldots$
SPARC	$-0.852200849\ldots$
IBM RS/6000 AIX 3005	$-0.852200849\ldots$
IBM 3090/600S-VF AIX 370	0.0
PC : Borland TurboC 2.0	$4.67734e-240$
Sharp EL5806	-0.090748172
Casio fx-8100, fx180p, fx 6910 G	Error

Pourquoi ?

– pas d'exigence d'« arrondi exact » pour ces fonctions (sin, cos, exp, log, arctan)

– on a longtemps crû que c'était une tâche impossible

Pourquoi est-ce difficile ?

- fonction f (sin, cos, log, exp, arctg), système virgule flottante de base 2, n bits de mantisse ;
- en partant de x et m ($m > n$), on peut calculer une approximation de $f(x)$ dont l'erreur sur la mantisse y est $\leq 2^{-m}$

Problème obtient on l'arrondi de $f(x)$ en arrondissant y ?

Impossible si y est de la forme :

- En arrondi au plus près

$$\underbrace{\overbrace{1.xx\ldots xx}^{}}_{n\text{ bits}}\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\overbrace{1000\ldots00}^{m\text{ bits}}xx\ldots \quad \text{ou} \quad \overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}0111\ldots11}^{m\text{ bits}}xx\ldots$$

- avec les autres modes d'arrondi

$$\overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}000\ldots00}^{m\text{ bits}}xx\ldots \quad \text{ou} \quad \overbrace{\underbrace{1.xx\ldots xx}_{n\text{ bits}}111\ldots11}^{m\text{ bits}}xx\ldots$$

Ce problème est appelé le

Dilemme du Constructeur de Tables (TMD) .

Exemple 1 : base 10, n chiffres de mantisse

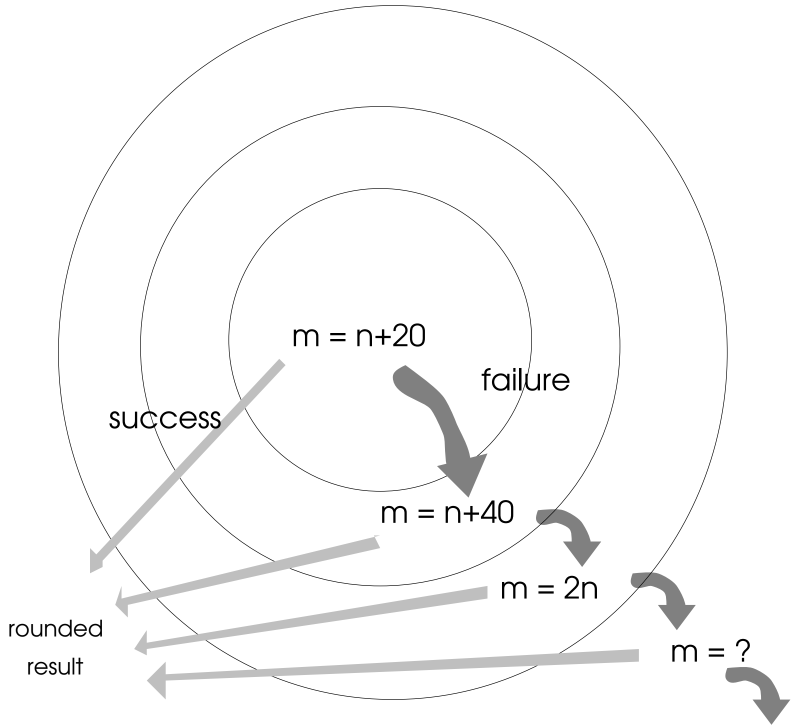
n	x	$\cos(x)$
4	1.149	0.409400000428...
5	1.6406	−0.69747000000219...
6	1.41162	0.15850500000180...
7	1.225910	0.33808970000005907...
8	1.9509449	−0.3710584400000000012...

Exemple 2 : base 2, 24 chiffres de mantisse

$$x = 1.00001100010011010100101_2 = \frac{8791717}{8388608},$$

$$\cos(x) = 0.0\underbrace{111111111001111010111000}_{24}0\underbrace{\color{red}{1111\ldots1111}}_{24}000011101\ldots$$

La stratégie “peau d’oignon” (Abraham Ziv)



Résoudre le TMD

- peut-on résoudre ce problème ? `oui`
- peut-on le résoudre en un temps raisonnable ?

- Lindemann, 1882 l'exponentielle d'un nombre algébrique $\neq 0$ n'est pas algébrique ;
- les nombres virgule flottante sont algébriques ;

$\Rightarrow \sin(x), \cos(x), \exp(x), \arctan(x)$ pour $x \neq 0$, et $\log_e x$, pour $x \neq 1$, ne peuvent pas avoir une infinité de 0 ou 1 consécutifs dans leur écriture binaire.

$\Rightarrow$ Pour tout x, il existe m tel que le TMD ne peut pas se produire.

Le nombre de nombres machine est fini $\Rightarrow$,il existe m tel que pour tout x le TMD ne peut se produire. Problème : trouver cet m.

Quelques expériences

24 bits de mantisse, nombres compris entre 1 et 2.

$f(x)$	m
$\sin x$	53
$\cos x$	50
e^x	49
$\ln x$	53
$\arctan x$	49
$\log_2 x$	51

Il semble que $m \approx 2n$

Un peu de cuisine

Attention approche non rigoureuse. On cherche à comprendre intuitivement d'où vient la relation $m \approx 2n$. On suppose

- arrondi au plus près ;
- quand x est un nombre machine, les bits de $f(x)$ après la n-ième position peuvent être vus comme des suites aléatoires de 0 et de 1 , le 0 et le 1 étant équiprobables ;
- ces suites peuvent être considérées « indépendantes » pour deux nombres machine différents.

On calcule la mantisse y de $f(x)$. Résultat de la forme :

$$y_0.y_1y_2\dots y_{n-1}\overbrace{01111\dots11}^{k\,\mathrm{bits}} \quad \text{ou} \quad y_0.y_1y_2\dots y_{n-1}\overbrace{10000\dots00}^{k\,\mathrm{bits}}$$

avec $k \geq 1$. Plus grande valeur de k ?

Nos hypothèses $\rightarrow$ la “probabilité” d’avoir $k \geq k_0$ est 2^{1-k_0}.

n bits de mantisse, et n_e exposants possibles (i.e. $N = n_e \times 2^{n-1}$ nombres machine) $\Rightarrow$ la probabilité d’avoir au moins un nombre conduisant à une valeur $k \geq k_0$ est :

$$\mathcal{P}_{k_0} = 1 - \left[1 - 2^{1-k_0}\right]^N$$

Donc le nombre k_0 tel que $k \geq k_0$ ait une probabilité $\leq 1/2$ vaut :

$$k_0 \geq n + \log_2(n_e) + 0.529$$

Si on ne considère qu'un petit nombre d'exposants possibles, la plus grande valeur de k sera en moyenne légèrement supérieure à n.

$\Rightarrow$ En pratique, m doit être légèrement supérieur à $2n$.

Nombre d'occurences de divers k pour $\sin(x)$

	k										
n	0	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	0	0	0	0	0	0
3	5	2	1	0	0	0	0	0	0	0	0
4	9	3	1	2	1	0	0	0	0	0	0
5	18	7	3	1	1	1	0	0	0	1	0
6	34	15	13	2	0	0	0	0	0	0	0
7	72	26	13	7	3	6	1	0	0	0	0
8	130	65	31	13	5	6	4	1	0	1	0
9	272	119	67	23	16	6	4	3	1	1	0
10	538	239	114	69	26	20	15	3	0	0	0
11	1037	527	215	121	71	50	14	6	2	4	0
12	2049	1030	521	255	127	49	27	21	7	6	3
13	4144	2017	1025	502	255	142	56	26	11	10	3
14	8247	4081	2044	1012	516	251	106	65	30	23	2

Passons à la théorie des nombres

Baker (1975) (cas particulier)

- $\alpha = \frac{p}{q}$, $\beta = \frac{p'}{q'}$, $p, q, p', q' \leq 2^n$
- $C = 16^{200}$

$$|\alpha - \log \beta| > (n2^n)^{-Cn \log n}$$

Application pour calculer $\log x$ et $\exp x$ en « double précision » ($n = 53$), faire les calculs intermédiaires sur $\approx 10^{244}$ bits suffit ! ! !

Une avancée significative. . .

Yu. Nesterenko et M. Waldschmidt (1995)

- α : *nombre algébrique* ;
- $h(\alpha)$ = *hauteur logarithmique absolue de Weil* de α : le polynôme minimal de α est $P(x) = a_0x^d + \cdots + a_d \in \mathbb{Z}[x]$, et ses racines complexes sont $\alpha_1, \cdots, \alpha_d$:

$$a_0x^d + \cdots + a_d = a_0(x - \alpha_1) \cdots (x - \alpha_d),$$

alors

$$h(\alpha) = \frac{1}{d}\left(\log|a_0| + \sum_{i=1}^{d} \log\max\{1, |\alpha_i|\}\right)$$

Théorème *Soient* $\theta \in \mathbb{C}$, $\theta \neq 0$, *et* α, β *des nombres algébriques ; soient* $\mathbb{K} = \mathbb{Q}(\alpha, \beta)$ *et* $D = [\mathbb{K} : \mathbb{Q}]$. *Soient* A, B *et* E *des réels* > 0, *avec* $E \geq e$ *satisfaisant*

$$\log A \geq \max\left(h(\alpha), D^{-1}\right), \quad \log B \geq h(\beta).$$

Alors

$$|e^{\theta} - \alpha| + |\theta - \beta| \geq$$
$$\exp\Big(-211D \times \left(\log B + \log\log A + 4\log D + 2\log(E|\theta|_+) + 10\right)$$
$$\times \left(D\log A + 2E|\theta| + 6\log E\right) \times \left(3.3D\log(D+2) + \log E\right) \times (\log E)^{-2}\Big)$$

où $|\theta|_+ = \max(1, |\theta|)$.

Application au calcul de e^x et $\ln(x)$

n	domaine pour l'exponentielle	m
24	$\ln 2$	494416
	10	3074888
53	$\ln 2$	1038560
	10	5234891
112	$\ln 2$	2527507
	10	10409113

– Encore beaucoup. . . mais il y a 2 transparents, c'était 10^{244}

– Impossible $\longrightarrow$ Très cher

Une estimation pessimiste

- $\sin x$, sur un Cyrix 83D87 : 121 cycles ;
- 200 MHz, ne calculant que des sinus, en permanence ;
- résultat double précision ; $m = 106$ bits ;
- une “mauvaise configuration” tous les 170 ans.

Le problème de mon arrière-arrière petit fils ?

Calculer sur un million de bits ?

$M(n)$ = nombre d'opérations élémentaires pour multiplier 2 nombres de n bits.

Théorème (Brent, 1976)
Si π *et* $\log 2$ *sont pré-calculés,* $f(x)$ *peut être calculé sur* n *chiffres en* $(K + o(1))\, M(n) \log_2 n$ *opérations élémentaires, où*

$$K = \begin{cases} 13 & \textit{si} \quad f(x) = \log(x) \textit{ ou } \exp(x) \\ 34 & \textit{si} \quad f(x) = \arctan(x) \textit{ ou } \sin(x) \textit{ ou } \cos x \end{cases}$$

Et la multiplication ?

- Dan Zuras, 1993, analyse de différents algorithmes de multiplication
- HP-9000/720, 50 MHz
- multiplication de nombres de 1 million de bits : $\simeq$ 4 secondes

Conséquence avec une machine à 50 MHz, calculer une exponentielle sur un million de bits demande moins de 18 minutes

Que tout cela est décevant !

- But le format « IEEE double précision » (base 2, $n = 53$), de très loin le plus utilisé ;
- on « sait » que la bonne valeur de m est certainement proche de 110 ;
- meilleure borne : environ 10^6

Or, faire des circuits de calcul avec $m = 110$ serait très facilement réalisable. . .

Seule solution : tester tous les cas possibles

Tests exhaustifs pour e^x

- ne pas calculer $2^{53} = 9007199254740992$ exponentielles (10^6 par seconde demanderait 285 ans) ;
- Filtre très vite éliminer la plupart des cas ;
- découpage en très petits sous-domaines : dans chaque domaine, l'exponentielle est approchée par une fonction affine.

Le théorème des trois distances

- cercle unité ;
- x_0 sur le cercle ;
- angle α donné.

le point x_{i+1} est obtenu en faisant tourner x_i d'un angle α.

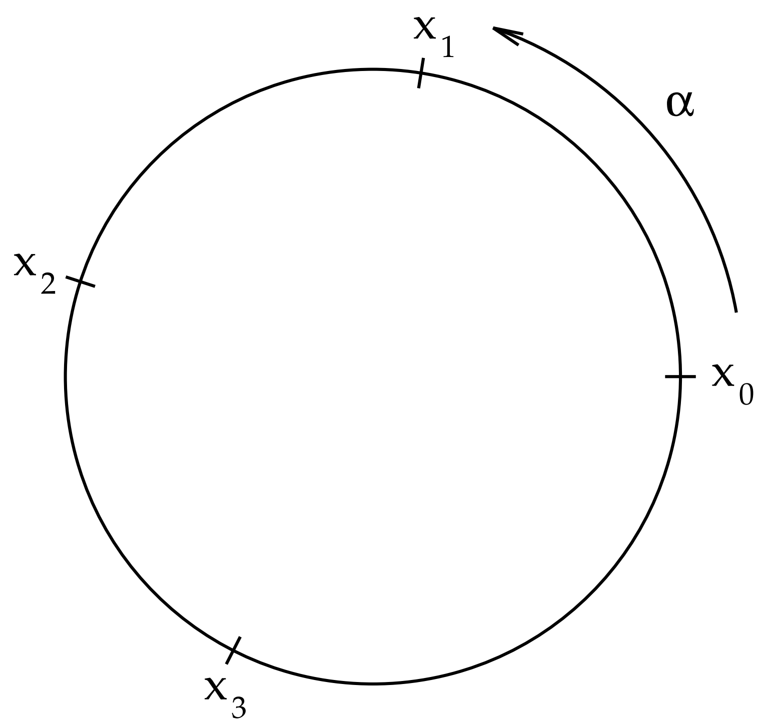


FIG. 1: Construction des premiers points.

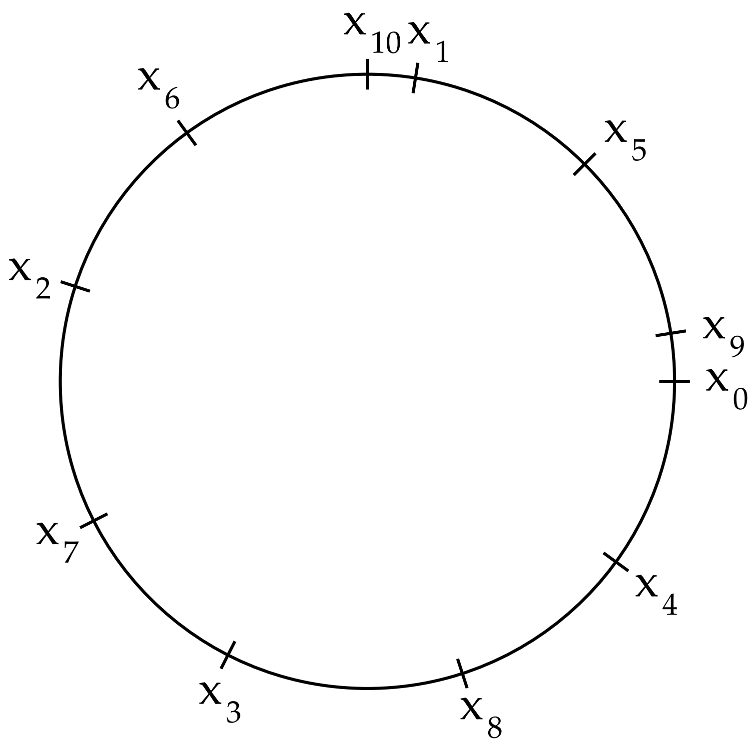


FIG. 2: Construction des points suivants.

Propriété À tout moment durant ce processus de construction (i.e. pour un n donné, quand les points $x_0, x_1, x_2, \ldots, x_n$ sont construits), la distance angulaire entre 2 points qui sont voisins sur le cercle ne peut prendre qu'au plus 3 valeurs possibles . Ces valeurs dépendent de n et α. De plus, il y a une infinité de valeurs de n pour lesquelles cette distance ne prend que 2 valeurs possibles

Quel est le rapport avec le TMD ?

- Sous domaines dans lesquels l'exponentielle est approchée par une fonction affine. Nombres de même exposant dans chaque sous-domaine ;
- changements de variable : on se ramène à des calculs sur des entiers (en multipliant par des puissances de 2) $\Rightarrow$ étant donnés une grille dont les coordonnées sont des entier, une droite et un (petit) réel positif ϵ, a-t-on des points de la grille à distance $\leq \epsilon$ de la droite ?
- translation : a-t-on des points de la grille au dessus de la droite, et à distance $\leq 2\epsilon$ de celle-ci ?

Calcul modulo 1

- Équation de la droite $y = ax - b$;
- on cherche s'il y a un entier x dans le domaine considéré tel que $b - ax$ modulo 1 est $\leq 2\epsilon$;
- on retrouve le théorème des 3 distances avec $x_0 = b$ et $\alpha = -2\pi a$,
- on se contente de construire la suite des distances atteintes pour les valeurs de n pour lesquelles il y a 2 distances.

- Infinité de valeurs de n pour lesquelles la distance entre 2 points sur le cercle peut prendre 2 valeurs ;
- γ_i et δ_i ces distances la $i$ème fois qu'il y a 2 distances.

Théorème (V. Lefèvre) L'algorithme suivant calcule γ_i et δ_i.
Notation : $\{x\} = x - \lfloor x \rfloor$. Résultat $\{b - ax\}$, où x est un entier dans $[0, N-1]$.

Initialization : $\gamma = \{a\};\ \delta = 1 - \{a\};\ d = \{b\};\ u = v = 1;$

Infinite loop :

```
if (d < γ)
        while (γ < δ)
                if (u + v ⩾ N) exit
                δ = δ − γ;  u = u + v;
        if (u + v ⩾ N) exit
        γ = γ − δ;  v = v + u;
else
        d = d − γ;
        while (δ < γ)
                if (u + v ⩾ N) exit
                γ = γ − δ;  v = v + u;
        if (u + v ⩾ N) exit
        δ = δ − γ;  u = u + v;
```

Returned value : d

Quelques résultats

Fonctions testées, en double précision : $\exp$ entre 2^{-1} et 2^{5}, $\log$ entre $1+2^{-13}$ et 2, $\sin$ entre 2^{-5} et 2^{1}, $\cos$ entre 2^{-2} et 2^{0}. Résultats complets pour $\log_2$ et 2^x.

Pires cas :

$$x = .011111111001110110011101110011100111010000111101101101_2$$

$$= \frac{8980155785351021}{18014398509481984}$$

avec

$$\begin{aligned} \sin x \quad = \quad & .011110100110010101000001110011 \\ & 000011000100011010010101\ 1\ \underbrace{1111\ldots1111}_{65}\ 00\ldots. \end{aligned}$$

$$x = .0111101001100101010000001110011000011000100011010010110_2$$

$$= \frac{4306410053968715}{9007199254740992}$$

avec

$$\arcsin x = .011111111001110110011101110011$$
$$1001110100001111101101101\ 0\ \underbrace{0000\ldots0000}_{64}\ 10\ldots.$$

Donc la valeur de m pour $\sin x$ dans $[2^{-5}, 2]$ est $120 = 53 + 66 + 1$

et la valeur de m pour $\arcsin x$ est $119 = 53 + 65 + 1$

Conclusion

– calculs en cours $\Rightarrow$ d'ici environ 10 ans, les systèmes disponibles garantiront l'« arrondi exact » des fonctions en « double précision » ;

– très grandes précisions (≥ 100 bits) peu d'espoirs de résultats certains, mais estimations probabilistes + détection ;

– Conséquence préservation de propriétés (p.ex. monotonie), système complètement spécifié $\Rightarrow$ axiomes sur lesquels on peut (et doit !) bâtir des preuves et des algorithmes

La perfection n'est pas de ce monde...

Format IEEE-754 simple précision ($n = 24$). Tangente et arctangente arrondies au + près. Le nombre machine le plus proche de $\arctan(2^{30})$ est

$$\frac{13176795}{8388608} = 1.57079637050628662109375 > \frac{\pi}{2}.$$

Sur notre système, on obtiendra donc un résultat + grand que $\pi/2$. On aura

$$\tan\left(\arctan\left(2^{30}\right)\right) = -2.2877\cdots \times 10^{7}.$$

Le pire n'est pas toujours certain !

Théorème élémentaire Si une fonction f vérifie

- $f(x)$ est défini $\forall x \in [a, b]$,
- $c = \min_{x \in [a,b]} f(x)$ est un nombre machine,
- $d = \max_{x \in [a,b]} f(x)$ est un nombre machine,

alors si $f(x)$ est calculé avec arrondi exact pour un nombre machine $x \in [a, b]$, le résultat obtenu sera dans $[c, d]$.
$\Rightarrow$ pas de problème avec sin, cos, th...