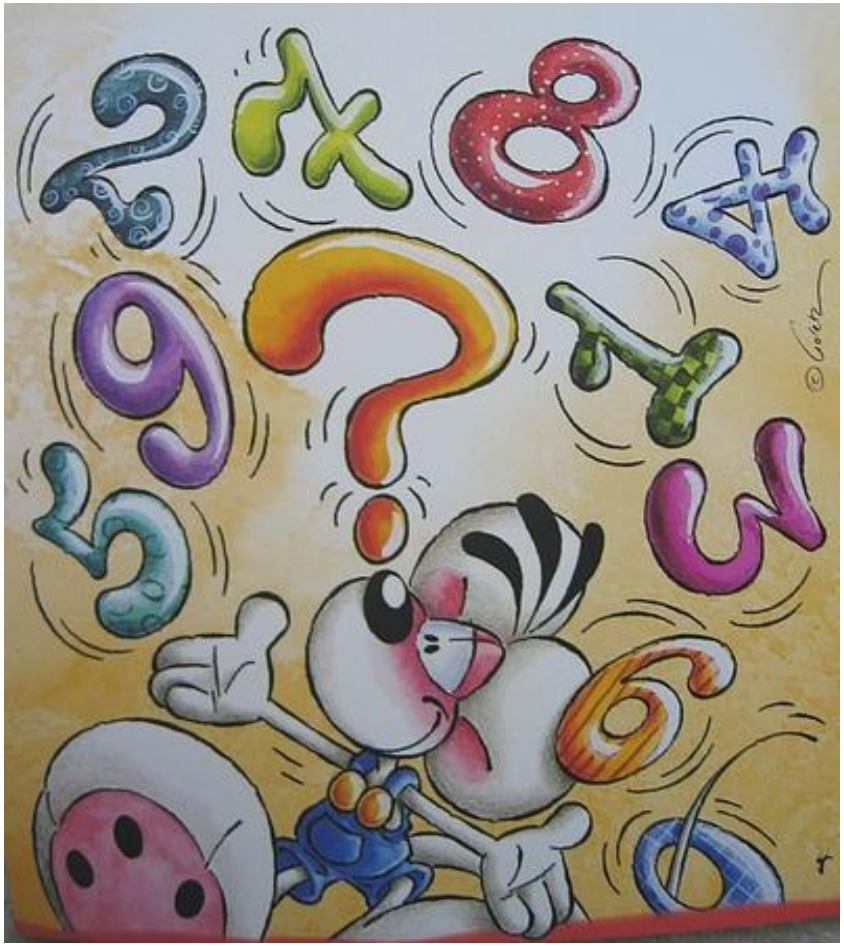


Diddl - Goletz

(cf. http://www.col-camus-soufflenheim.ac-strasbourg.fr/Page.php?IDP=137&IDD=0)

Arithmétique des ordinateurs
calculer de façon rapide, fiable, précise

Nathalie Revol

10 avril 2007

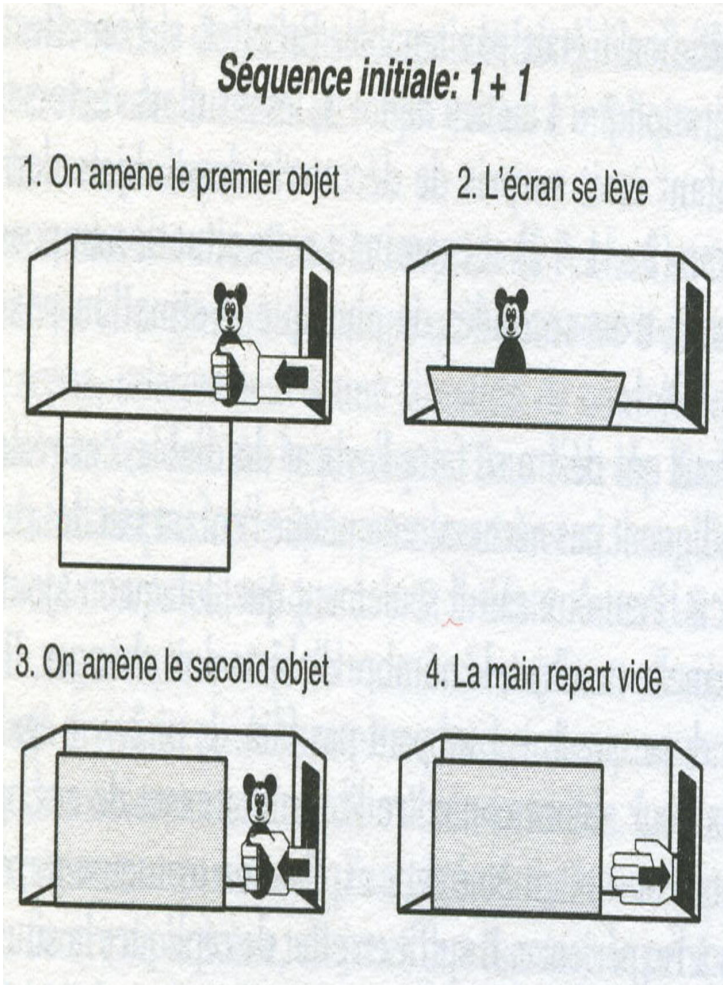
Compter
Même les animaux comptent !

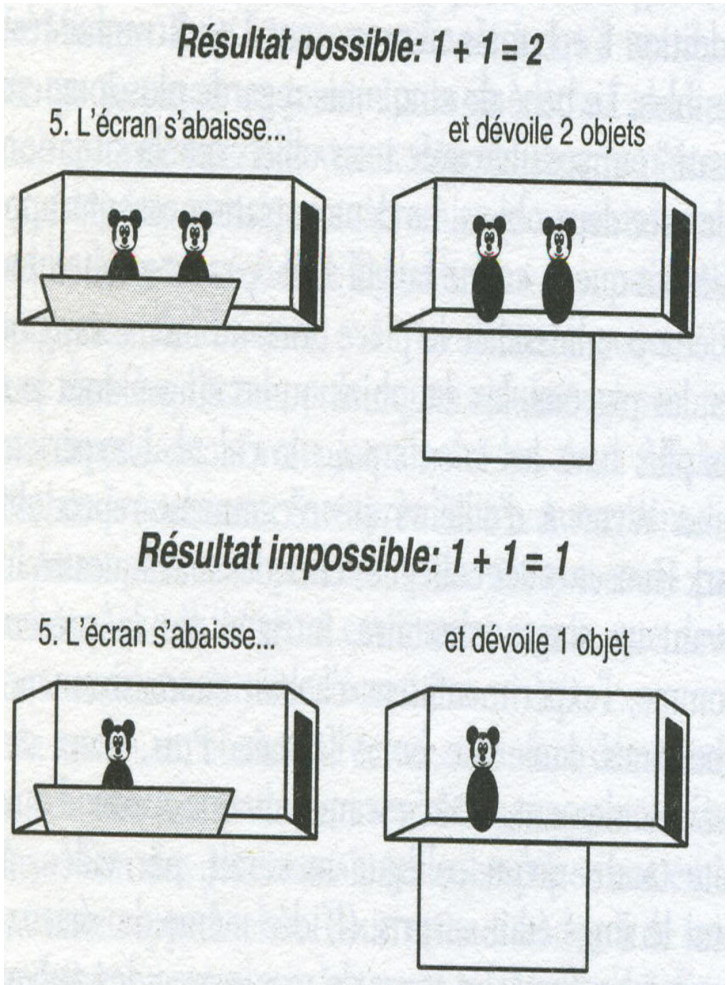
Un châtelain voulait abattre une corneille qui s'était nichée au sommet d'une tour. Mais dès qu'il s'approchait du nid, l'oiseau s'envolait hors de portée de fusil et guettait le départ du chasseur. Sitôt celui-ci parti, elle revenait de plus belle infester les parages de la tour. Notre homme eut l'idée de demander l'aide d'un voisin. Les deux hommes armés entrèrent ensemble dans la tour, puis l'un d'eux seul en ressortit. Mais la corneille ne se laissa pas piéger et attendit sagement que le deuxième homme s'éloigne avant de regagner son logis. Trois hommes, puis quatre, puis cinq, ne suffirent pas à la tromper. Chaque fois, la corneille attendait le départ de tous les chasseurs avant de revenir. Finalement, les chasseurs vinrent au nombre de six. Lorsque cinq d'entre eux furent sortis de la tour, l'oiseau revint au nid, trop confiant, et le sixième chasseur l'abattit.

Compter

Les bébés aussi savent compter

(tiré de *La bosse des maths*, Stanislas Dehaene, Odile Jacob 1997)





Expérience de Karen Wynn : les bébés de 4 à 5 mois sont surpris par le second événement.

Compter
Et votre ordinateur, il compte comment ?

C'est l'objet de ce cours :

- comment représenter les nombres
- les opérations apprises à l'école primaire
- le calcul en virgule flottante (ou la notation scientifique, vue au collège)
- le calcul par intervalles

expliqués à votre ordinateur.

Compter sur ordinateur
À quoi ça sert ?

Wikipedia : Scientific computing = constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific and engineering problems.

Different objectives :

- Reconstruct and understand known events (e.g., earthquake, tsunamis and other natural disasters).
- Optimise known scenarios (e.g., technical and manufacturing processes, front end engineering).
- Predict future or unobserved situations (e.g., weather, sub-atomic particle behaviour).

Compter sur ordinateur
Exemples de domaines d'application

- bio-informatique
- biologie
- imagerie médicale
- chimie (conformations moléculaires...)
- économie
- modélisation financière
- systèmes d'information géographique (GIS)
- mécanique
- physique
- électromagnétisme
- mécanique des fluides
- météorologie, climatologie
- environnement
- machine learning
- pattern recognition

Compter sur ordinateur
À quelle vitesse ?

http://www.top500.org/

28th TOP500 List, released in Tampa, FL during SC06.

Rank	Site	Computer
1	DOE/NNSA/LLNL, US	BlueGene/L - eServer Blue Gene Soluti
2	NNSA/Sandia NL, US	Red Storm - Sandia/ Cray Red Storm,
3	IBM Thomas J. Watson RC, US	BGW - eServer Blue Gene Solution IBM
4	DOE/NNSA/LLNL, US	ASC Purple - eServer pSeries p5 575 1
5	Barcelona Supercomp. C, Spain	MareNostrum - BladeCenter JS21 Clus
6	NNSA/Sandia NL, US	Thunderbird - PowerEdge 1850, 3.6 GH
7	CEA, France	Tera-10 - NovaScale 5160, Itanium2 1.
8	NASA/Ames RC/NAS, US	Columbia - SGI Altix 1.5 GHz, Voltaire
9	GSIC Center, Tokyo IT, Japan	TSUBAME Grid Cluster - Sun Fire x46
10	ORNL, US	Jaguar - Cray XT3, 2.6 GHz dual Core

Compter sur ordinateur
À quelle vitesse ?

Nov 2006 : the IBM BlueGene/L system [. . .] retains the No. 1 spot with a Linpack performance of 281.6 teraflops.
No. 2 (Sandia NL Cray Red Storm) : 101.4 Tflops.

Un Teraflops $= 10^{12}$ *floating-point operation per second*

Exemples d'utilisation de Blue Gene :

- simulation temps réel de repliements de protéines (cf. prion)
- data mining intensif sur des statistiques médicales
- analyse temps réel de signaux radioastronomiques
- Blue Brain Project - simulation d'une colonne corticale humaine.

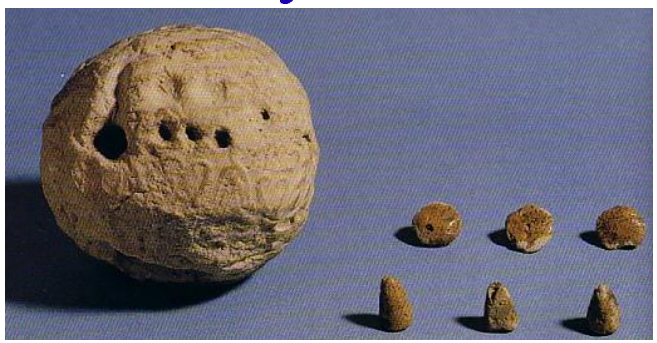
Arithmétique des ordinateurs
calculer de façon rapide, fiable, précise

- comment représenter les nombres
- les opérations apprises à l'école primaire
- le calcul en virgule flottante (ou la notation scientifique, vue au collège)
- le calcul par intervalles

Systèmes de numération

- **"1, 2, beaucoup"** est peut-être une exagération ?

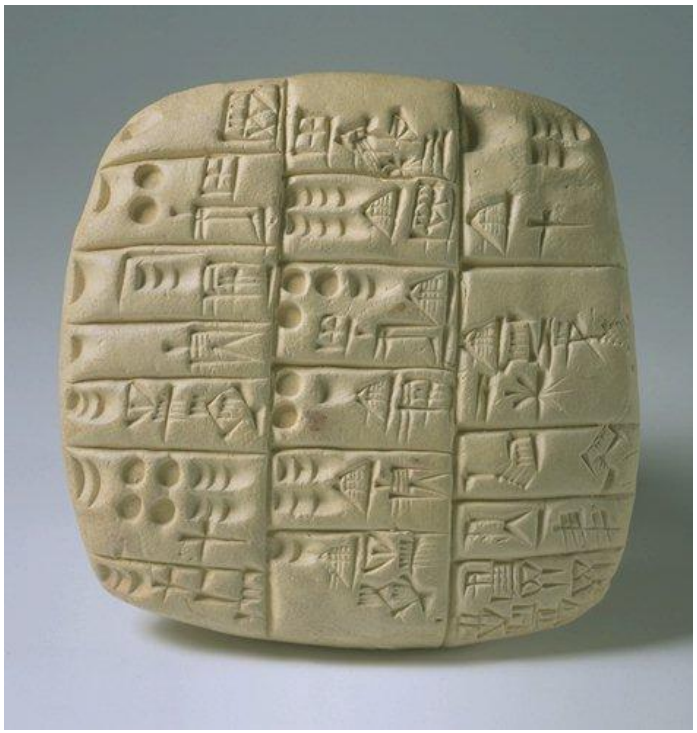
- **des encoches sur des os :**
- **calcul = caillou**
- **à Sumer il y a 6000 ans :**



Petit cône = 1
Petite bille = 10
Grand cône = 60
Grand cône percé = 600
Grosse bille = 3600
Grosse bille percée = 36000

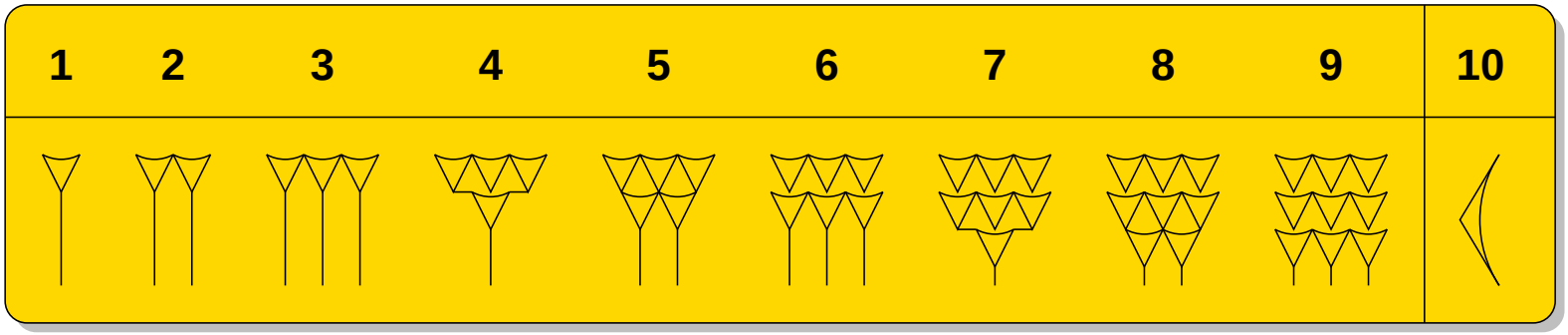
Systèmes de numération et calcul

Babylone il y a 4000 ans : systèmes de position en base 60

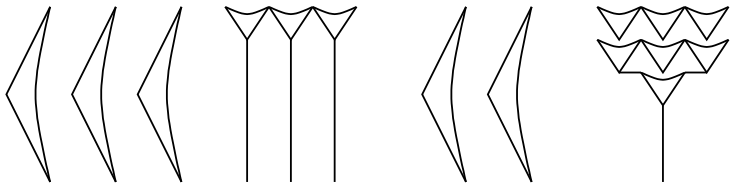


Systèmes de numération et calcul

Babylone il y a 4000 ans : systèmes de position en base 60



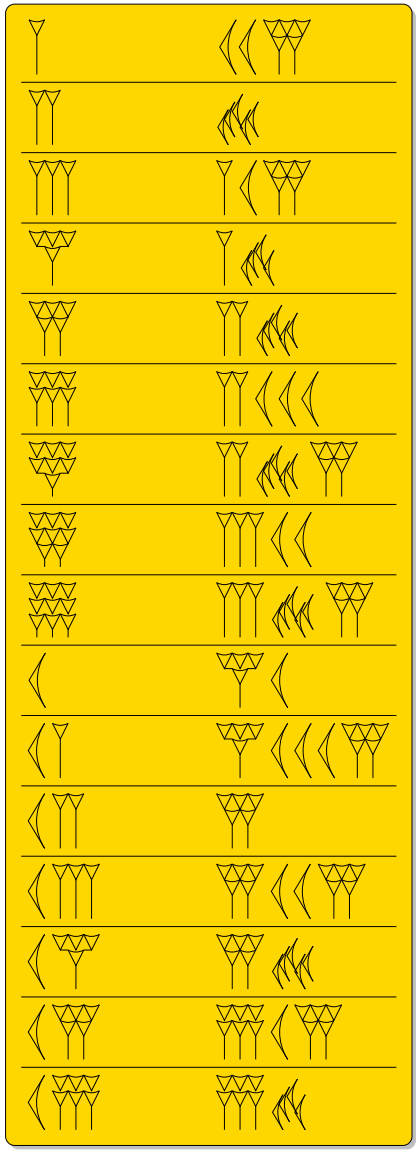
(source : A. Tisserand, cours 2005-2006)

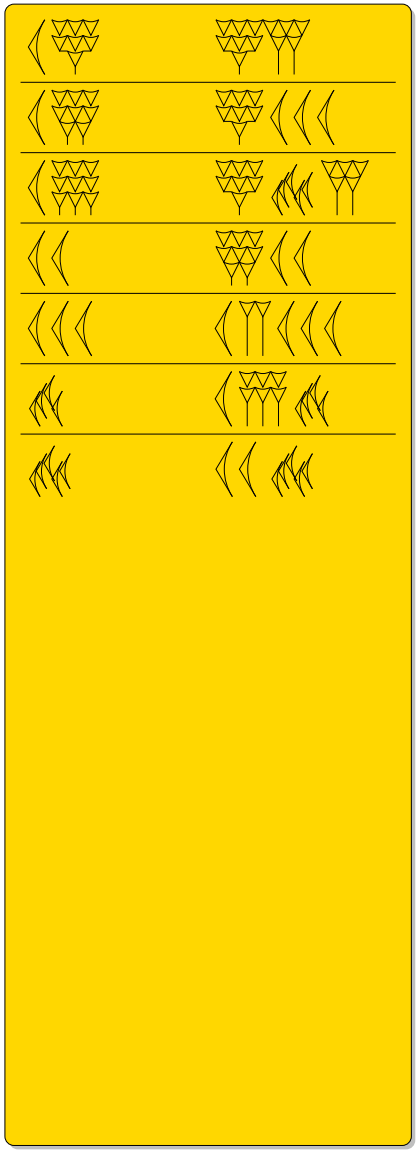


$2007 = 33*60 + 27 = 1980 + 27$

Systèmes de numération et calcul

Babylone il y a 4000 ans : table de multiplication par 25

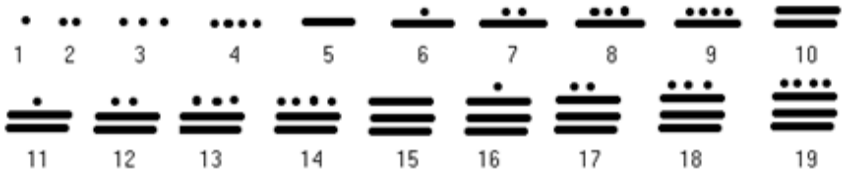




(source : A. Tisserand, cours 2005-2006)

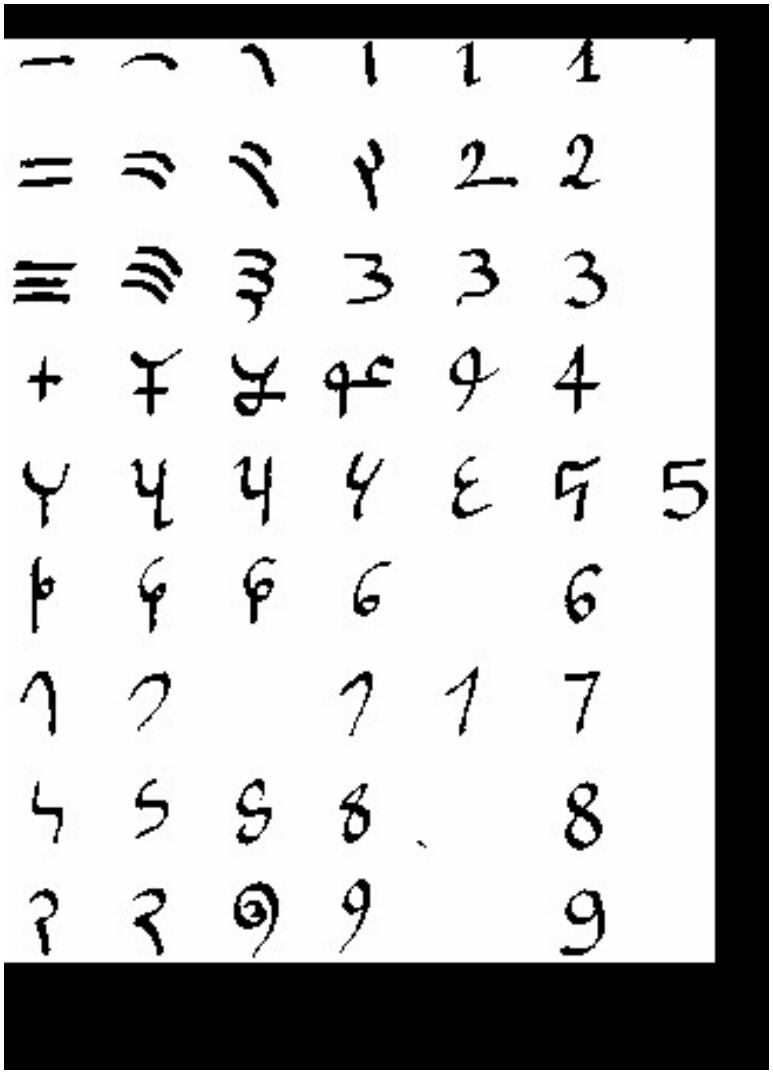
Systèmes de numération et calcul

chez les Mayas il y a 15 siècles : calcul en base 20



Systèmes de numération et calcul

nos chiffres "arabes" :
issus de l'Inde,
adoptés par le monde arabe au VIIIe ou IXe siècle,
puis par l'Europe grâce au pape Sylvestre II (Gerbert d'Aurillac) vers 990.



Systèmes de numération et calcul

Exercice : diviser MMCCCXCIII par CLXXXII.
Généralisation : proposer un algorithme de division.
Attention ! obligation de se mettre dans la peau de quelqu'un qui n'a aucune idée de ce que peut être un système "de position".

Compter
Et votre ordinateur, il compte comment ?

- comment représenter les nombres
- **les opérations apprises à l'école primaire**
- le calcul en virgule flottante (ou la notation scientifique, vue au collège)
- le calcul par intervalles

Algorithme

suite d'instructions à effectuer pour accomplir une tâche
Pas d'implicite ni d'imprécision.
Syntaxe rigide quand on s'adresse à un ordinateur.

Compter avec des entiers

Pour des humains ayant 10 doigts :

représentation des nombres entiers :

$$13 = 1 \times 10 + 3 \times 1 = 1 \times 10^1 + 3 \times 10^0$$

Pour des ordinateurs ayant 2 poings :

représentation des nombres entiers :

$$13_{10} = 1\times8+1\times4+0\times2+1\times1 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0 = 1101_2$$

Addition en base 10

$$\begin{array}{cccccc} & & & 1 & 1 & \\ & & 1 & 2 & 3 & 4 \\ + & & & 5 & 6 & 7 \\ \hline & & 1 & 8 & 0 & 1 \end{array}$$

Addition en base 2

Sur ordinateur : idem mais en base 2 :

$$
\begin{aligned}
&1234_{10} \\
&\quad = \quad 1\times1024+0\times512+0\times256+1\times128+1\times64+0\times32+1\times16+0\times8+0\times4+1\times2+0\times1 \\
&\quad = \quad 10011010010_2 \\
&567_{10} \\
&\quad = \quad 1\times512+0\times256+0\times128+0\times64+1\times32+1\times16+0\times8+1\times4+1\times2+1\times1 \\
&\quad = \quad 1000110111_2
\end{aligned}
$$

			1	1	1	1		1	1		
	1	0	0	1	1	0	1	0	0	1	0
+		1	0	0	0	1	1	0	1	1	1
=	1	1	1	0	0	0	0	1	0	0	1
	×1024	×512	×256	×128	×64	×32	×16	×8	×4	×2	×1

Addition : complexité

Complexité de l'addition de 2 nombres à n chiffres :

$$\mathcal{O}(n)$$

opérations sur des chiffres.

Addition : complexité

Complexité de l'addition de 2 nombres à n chiffres :

$$\mathcal{O}(n)$$

opérations sur des chiffres.

Hypothèse :
additionner deux chiffres quelconques prend toujours le même temps

Addition : complexité

Complexité de l'addition de 2 nombres à n chiffres :

$$\mathcal{O}(n)$$

opérations sur des chiffres.

Hypothèse :
additionner deux chiffres quelconques prend toujours le même temps

ce qui n'est pas le cas du cerveau humain : $1 + 2$ est plus rapide à effectuer que $8 + 9$.

Addition : problème sur ordinateur

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 2 & 3 & 4 & \\ + & & 5 & 6 & 7 & \\ \hline & 1 & 8 & 0 & 1 & \end{array}$$

Addition : problème sur ordinateur

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 2 & 3 & 4 & \\ + & & 5 & 6 & 7 & \\ \hline & 1 & 8 & 0 & 1 & \end{array}$$

Problème : la retenue !

Pourquoi ne pas commencer par la gauche, pour avoir tout de suite l'ordre de grandeur du résultat ?

Addition : problème sur ordinateur

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 2 & 3 & 4 & \\ + & & 5 & 6 & 7 & \\ \hline & 1 & 8 & 0 & 1 & \end{array}$$

Problème : la retenue !

Pourquoi ne pas commencer par la gauche, pour avoir tout de suite l'ordre de grandeur du résultat ?

Pourquoi ne pas additionner chacun des chiffres simultanément ?

Addition : problème sur ordinateur

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & 1 & 2 & 3 & 4 & \\ + & & 5 & 6 & 7 & \\ \hline & 1 & 8 & 0 & 1 & \end{array}$$

Problème : la retenue !

Pourquoi ne pas commencer par la gauche, pour avoir tout de suite l'ordre de grandeur du résultat ?

Pourquoi ne pas additionner chacun des chiffres simultanément ?

Solution : changer de système de numération !

Addition : systèmes redondants de numération

(Avizienis, 1961)

Principe : un même nombre peut avoir plusieurs représentations.
Avantage pour l'addition : on choisit toujours un chiffre qui permettra d'absorber la retenue (**absorber** au sens d'**empêcher de se propager**).

Comment : ne pas se limiter aux chiffres entre 0 et 9 :
- autoriser les chiffres supérieurs à 10 (par ex. de 0 à 19),
- autoriser les chiffres négatifs.

Exemple : en base 10 avec les chiffres de -6 à 6
1546 s'écrit 1546, $2(-5)46$ noté $2\bar{5}46$, $2(-4)(-6)6$ noté $2\bar{4}\bar{6}6$, $2\bar{4}\bar{5}\bar{4}$, $16\bar{6}6$, $16\bar{5}\bar{4}$...

Addition : systèmes redondants de numération

(Avizienis, 1961)

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ + & & 5 & 6 & 7 \\ \hline & 1 & 7 & 9 & 11 \end{array}$$

soit $1790 + 11 = 1801$.

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ + & & 5 & 6 & 7 \\ \hline & 2 & \bar{2} & 0 & 1 \end{array}$$

$2\bar{2}01 = 2000 - 200 + 00 + 1 = 1801$.

Addition : systèmes redondants de numération

(Avizienis, 1961)

En base β, chiffres $\in \{-a, \cdots, +a\}$ avec a entier et $2a \geq \beta + 1$ et $a \leq \beta - 1$,
Le nombre x est représenté par $x_{n-1}x_{n-2}\cdots x_2x_1x_0$ si $x = \sum_{i=0}^{n-1} x_i\beta^i$.

Si $2a + 1 \geq \beta$ alors on peut représenter tous les entiers.
Si $2a + 1 > \beta$ alors un nombre peut avoir **plusieurs** représentations : le système est dit **redondant**.

Exemple : en base $\beta = 10$ avec $a = 6$, 1546 s'écrit 1546, $2(-5)46$ noté $25\bar{4}6$, $2(-4)(-6)6$ noté $2\bar{4}\bar{6}6$, $2\bar{4}5\bar{4}$, $16\bar{6}6$, $165\bar{4}\bar{4}\ldots$

Addition : systèmes redondants de numération

Addition sans propagation de retenue

En base β avec des chiffres compris entre $-a$ et a , calcul de

$$s_n \cdots s_0 = x_{n-1} \cdots x_0 \ + \ y_{n-1} \cdots y_0$$

1. Calculer pour $i = 0 \cdots n-1$

$$\begin{array}{lcl} x_i + y_i & & \\ t_{i+1} & = & \begin{cases} -1 & \text{si} \quad x_i + y_i \leq -a \\ 0 & \text{si} \quad -a+1 \leq x_i + y_i \leq a-1 \\ 1 & \text{si} \quad x_i + y_i \geq a \end{cases} \\ w_i & = & x_i + y_i - \beta t_{i+1} \end{array}$$

2. Calculer pour $i = 0 \cdots n$:

$$s_i = w_i + t_i \text{ avec } w_n = t_0 = 0.$$

Addition : systèmes redondants de numération

Exemple : $\beta = 10$, $a = 6$, calcul de $x_i + y_i$

i		4	3	2	1	0
x_i		1	$\bar{2}$	5	3	$\bar{4}$
y_i		3	5	1	$\bar{5}$	$\bar{7}$
$x_i + y_i$		4	3	6	$\bar{2}$	$\bar{1}\bar{1}$
t_{i+1}						
w_i						
s_i						

Addition : systèmes redondants de numération

Addition sans propagation de retenue

En base 10 avec des chiffres compris entre -6 et 6 , calcul de

$$s_n \cdots s_0 = x_{n-1} \cdots x_0 \; + \; y_{n-1} \cdots y_0$$

1. Calculer pour $i = 0 \cdots n-1$

$$\begin{array}{rcl} t_{i+1} & = & \left\{ \begin{array}{rll} -1 & \text{si} & x_i + y_i \leq -a \\ 0 & \text{si} & -a+1 \leq x_i + y_i \leq a-1 \\ 1 & \text{si} & x_i + y_i \geq a \end{array} \right. \\ w_i & = & x_i + y_i - \beta t_{i+1} \end{array}$$

2. Calculer pour $i = 0 \cdots n$:

$$s_i = w_i + t_i \text{ avec } w_n = t_0 = 0.$$

Addition : systèmes redondants de numération

Exemple : $\beta = 10$, $a = 6$, calcul de t_{i+1}

i		4	3	2	1	0
x_i		1	$\bar{2}$	5	3	$\bar{4}$
y_i		3	5	1	$\bar{5}$	$\bar{7}$
$x_i + y_i$		4	3	6	$\bar{2}$	$\bar{1}\bar{1}$
t_{i+1}	0	0	1	0	$\bar{1}$	
w_i						
s_i						

Addition : systèmes redondants de numération

Addition sans propagation de retenue

En base β avec des chiffres compris entre $-a$ et a , calcul de

$$s_n \cdots s_0 = x_{n-1} \cdots x_0 \ + \ y_{n-1} \cdots y_0$$

1. Calculer pour $i = 0 \cdots n-1$

$$\begin{array}{rcl} t_{i+1} & = & \left\{ \begin{array}{rcl} -1 & \text{si} & x_i + y_i \leq -a \\ 0 & \text{si} & -a+1 \leq x_i + y_i \leq a-1 \\ 1 & \text{si} & x_i + y_i \geq a \end{array} \right. \\ w_i & = & x_i + y_i - \beta t_{i+1} \end{array}$$

2. Calculer pour $i = 0 \cdots n$:

$$s_i = w_i + t_i \text{ avec } w_n = t_0 = 0.$$

Addition : systèmes redondants de numération

Exemple : $\beta = 10$, $a = 6$, calcul de w_i

i		4	3	2	1	0
x_i		1	$\bar{2}$	5	3	$\bar{4}$
y_i		3	5	1	$\bar{5}$	$\bar{7}$
$x_i + y_i$		4	3	6	$\bar{2}$	$\bar{1}\bar{1}$
t_{i+1}	0	0	1	0	$\bar{1}$	
w_i		4	3	$\bar{4}$	$\bar{2}$	$\bar{1}$
s_i						

Addition : systèmes redondants de numération

Addition sans propagation de retenue

En base β avec des chiffres compris entre $-a$ et a , calcul de

$$s_n \cdots s_0 = x_{n-1} \cdots x_0 \ + \ y_{n-1} \cdots y_0$$

1. Calculer pour $i = 0 \cdots n-1$

$$\begin{array}{rcl} t_{i+1} & = & \left\{ \begin{array}{rcl} -1 & \text{si} & x_i + y_i \leq -a \\ 0 & \text{si} & -a+1 \leq x_i + y_i \leq a-1 \\ 1 & \text{si} & x_i + y_i \geq a \end{array} \right. \\ w_i & = & x_i + y_i - \beta t_{i+1} \end{array}$$

2. Calculer pour $i = 0 \cdots n$:

$$s_i = w_i + t_i \text{ avec } w_n = t_0 = 0.$$

Addition : systèmes redondants de numération

Exemple : $\beta = 10,\ a = 6$

i		4	3	2	1	0
x_i		1	$\bar{2}$	5	3	$\bar{4}$
y_i		3	5	1	$\bar{5}$	$\bar{7}$
$x_i + y_i$		4	3	6	$\bar{2}$	$\bar{1}\bar{1}$
t_{i+1}	0	0	1	0	$\bar{1}$	
w_i		4	3	$\bar{4}$	$\bar{2}$	$\bar{1}$
s_i		4	4	$\bar{4}$	$\bar{3}$	$\bar{1}$

Écriture usuelle : $x = 1\bar{2}53\bar{4} = 8526$, $y = 351\bar{5}\bar{7} = 35043$.
Somme $= 44\bar{4}\bar{2}\bar{1} = 43569$.

Addition : systèmes redondants de numération

Remarques complémentaires

Les $t_i \simeq$ retenues de l'addition usuelle

MAIS pas de dépendance entre t_i et t_{i+1}

$\Rightarrow$ le calcul de chaque chiffre peut se faire en parallèle.

Addition : systèmes redondants de numération

Remarques complémentaires

Les $t_i \simeq$ retenues de l'addition usuelle

MAIS pas de dépendance entre t_i et t_{i+1}

$\Rightarrow$ le calcul de chaque chiffre peut se faire en parallèle.

Complexité : $\mathcal{O}(1)$ si on a assez de ressources de calcul.

Addition : systèmes redondants de numération

Remarques complémentaires

Les $t_i \simeq$ retenues de l'addition usuelle
MAIS pas de dépendance entre t_i et t_{i+1}
$\Rightarrow$ le calcul de chaque chiffre peut se faire en parallèle.

Complexité : $\mathcal{O}(1)$ si on a assez de ressources de calcul.

Remarque : si $\beta = 2$, les conditions $2a \geq b + 1$ et $a \leq b - 1$ ne peuvent pas être satisfaites simultanément.
Autres algorithmes, avec les chiffres $\bar{1}$, 0 et 1 pour la base 2.

Multiplier

À la main :

$$
\begin{array}{cccccccc}
 & & & & 9 & 8 & 7 & 6 \\
\times & & & & 1 & 2 & 3 & 4 \\
\hline
 & & & 3 & 9 & 5 & 0 & 4 \\
 & & 2 & 9 & 6 & 2 & 8 & \\
 & 1 & 9 & 7 & 5 & 2 & & \\
 & 9 & 8 & 7 & 6 & & & \\
\hline
1 & 2 & 1 & 8 & 6 & 9 & 8 & 4
\end{array}
$$

Multiplier

Sur ordinateur :

$$
\begin{array}{rrrrrrrr}
 & & & & 9 & 8 & 7 & 6 \\
\times & & & & 1 & 2 & 3 & 4 \\
\hline
 & & & 3 & 9 & 5 & 0 & 4 \\
 & + & 2 & 9 & 6 & 2 & 8 & \\
\hline
 & & 3 & 3 & 5 & 7 & 8 & 4 \\
+ & 1 & 9 & 7 & 5 & 2 & & \\
\hline
 & 2 & 3 & 1 & 0 & 9 & 8 & 4 \\
+ & 9 & 8 & 7 & 6 & & & \\
\hline
1 & 2 & 1 & 8 & 6 & 9 & 8 & 4
\end{array}
$$

Multiplier

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
172	28

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
172	28
344	14

Multiplier

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
172	28
344	14
688	7

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
172	28
344	14
688	7
1376	3

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
172	28
344	14
688	7
1376	3
2752	1

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
~~172~~	~~28~~
~~344~~	~~14~~
688	7
1376	3
2752	1

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

86	57
~~172~~	~~28~~
~~344~~	~~14~~
688	7
1376	3
2752	1
4902	

Multiplier

Sur ordinateur :

$$
\begin{array}{rrrrrrrr}
 & & & & 9 & 8 & 7 & 6 \\
\times & & & & 1 & 2 & 3 & 4 \\
\hline
 & & & 3 & 9 & 5 & 0 & 4 \\
 & + & 2 & 9 & 6 & 2 & 8 & \\
\hline
 & & 3 & 3 & 5 & 7 & 8 & 4 \\
+ & 1 & 9 & 7 & 5 & 2 & & \\
\hline
 & 2 & 3 & 1 & 0 & 9 & 8 & 4 \\
+ & 9 & 8 & 7 & 6 & & & \\
\hline
1 & 2 & 1 & 8 & 6 & 9 & 8 & 4
\end{array}
$$

Multiplier : complexité

- chacun des chiffres de X est multiplié par chacun des chiffres de Y : n^2 produits
- chaque chiffre de Z est la somme de l produits de 2 chiffres : $O(n^2)$ additions

$$\Rightarrow \textbf{au total, complexité} = O(n^2)$$

En pratique, la différence entre la méthode apprise à l'école et l'algorithme est que la somme du résultat partiel et du produit de X par un nouveau chiffre de Y est effectuée avant de multiplier X par le chiffre suivant de Y, pour réduire la place mémoire utilisée.

Multiplier : multiplication de Karatsuba (version de Knuth)

Supposons n pair et décomposons X et Y :

$$X = X_H \cdot B^{n/2} + X_L \text{ et } Y = Y_H \cdot B^{n/2} + Y_L.$$

$$\begin{aligned} Z &= X \cdot Y \\ &= X_H \cdot Y_H \cdot B^n \\ &\quad + [X_H \cdot Y_H - (X_H - X_L) \cdot (Y_H - Y_L) + X_L \cdot Y_L] \cdot B^{n/2} \\ &\quad + X_L \cdot Y_L. \end{aligned}$$

On a seulement **3** multiplications de nombres de longueur $n/2$.
En appliquant récursivement ce principe, on obtient une complexité de $O(n^{\log_2 3}) = O(n^{1.585})$.

Multiplier : multiplications de Toom-Cook

- décomposer X et Y en k parts
- calculer Z en utilisant $2k-1$ multiplications
- obtenir une complexité de $O(n^{\log_k(2k-1)})$.

Multiplier : complexité

- algorithme naïf : $O(n^2)$

Multiplier : complexité

- algorithme naïf : $O(n^2)$
- multiplication de Karatsuba : $O(n^{\log_2 3}) = O(n^{1.585})$

Multiplier : complexité

- algorithme naïf : $O(n^2)$
- multiplication de Karatsuba : $O(n^{\log_2 3}) = O(n^{1.585})$
- multiplications de Toom-Cook : $O(n^{\log_k(2k-1)})$ pour tout k

Multiplier : complexité

- algorithme naïf : $O(n^2)$
- multiplication de Karatsuba : $O(n^{\log_2 3}) = O(n^{1.585})$
- multiplications de Toom-Cook : $O(n^{\log_k(2k-1)})$ pour tout k
- algorithme le plus rapide : dû à Schönhage et Strassen (1971)
  inspiré de la FFT : Fast Fourier Transform complexité : $O(n \log n)$ ou $O(n \log n \log\log n)$

Diviser

990	252
−756	3
234	

252	234
−234	1
18	

$\overset{\frown}{234}$	18
−18	13
54	
−54	
0	

Complexité : c'est la même que celle de la multiplication.
Pour résumer : $O(n^2)$ à la main, $O(n \log n)$ pour l'algorithme le plus rapide.

Diviser : le bug du Pentium en 1994

Bug du Pentium : T. Nicely (Virginie, USA), en octobre 1004, a constaté que son ordinateur donnait un résultat erroné lors du calcul de 1.0/824633702441 : erreur à partir du 12e chiffre.
T. Coe a découvert le pire cas : 41195835.0/3145727.0 donnait 1.333739068902 au lieu de 1.333820449136 (faux à partir du 5e chiffre).

Diviser : le bug du Pentium en 1994

Bug du Pentium : T. Nicely (Virginie, USA), en octobre 1004, a constaté que son ordinateur donnait un résultat erroné lors du calcul de 1.0/824633702441 : erreur à partir du 12e chiffre.
T. Coe a découvert le pire cas : 41195835.0/3145727.0 donnait 1.333739068902 au lieu de 1.333820449136 (faux à partir du 5e chiffre).

Explication : pour aller plus vite, utilisation de notation redondante :
utilisation de chiffres supérieurs à la base pour les restes partiels
utilisation de chiffres négatifs pour le quotient
et erreur en mélangeant tout ça !

Compter
Et votre ordinateur, il compte comment ?

- comment représenter les nombres
- les opérations apprises à l'école primaire
- **le calcul en virgule flottante (ou la notation scientifique, vue au collège)**
- le calcul par intervalles

Résolution d'un système linéaire par (pseudo-)élimination de Gauss

$$\begin{pmatrix} -85 & -55 & -37 & -35 & 97 \\ 50 & 79 & 56 & 49 & 63 \\ 57 & -59 & 45 & -8 & -93 \\ 92 & 43 & -62 & 77 & 66 \\ 54 & -5 & 99 & -61 & -50 \end{pmatrix}$$

$$\begin{pmatrix} -85 & -55 & -37 & -35 & 97 \\ 0 & -3965 & -2910 & -2415 & -10205 \\ 0 & 8150 & -1716 & 2675 & 2376 \\ 0 & 1405 & 8674 & -3325 & -14534 \\ 0 & 3395 & -6417 & 7075 & -988 \end{pmatrix}$$

Résolution d'un système linéaire par (pseudo-)élimination de Gauss

$$\begin{pmatrix} -85 & -55 & -37 & -35 & 97 \\ 0 & -3965 & -2910 & -2415 & -10205 \\ 0 & 0 & 30520440 & 9075875 & 73749910 \\ 0 & 0 & -30303860 & 16576700 & 71965335 \\ 0 & 0 & 35322855 & -19853450 & 38563395 \end{pmatrix}$$

$$\begin{pmatrix} -85 & -55 & -37 & -35 & 97 \\ 0 & -3965 & -2910 & -2415 & -10205 \\ 0 & 0 & 30520440 & 9075875 & 73749910 \\ 0 & 0 & 0 & 780962223125500 & 44313206366000000 \\ 0 & 0 & 0 & -926521846141125 & -142808559389925 0 \end{pmatrix}$$

$$\begin{pmatrix} -85 & -55 & -37 & -35 & 97 \\ 0 & -3965 & -2910 & -2415 & -10205 \\ 0 & 0 & 30520440 & 9075875 & 73749910 \\ 0 & 0 & 0 & 780962223125500 & 44313206366000000 \\ 0 & 0 & 0 & 0 & 29904344768408390283730691250000 \end{pmatrix}$$

Les nombres flottants

Exemple :
$\pi \simeq 3,14 \times 10^0 = 0,314 \times 10^1 = 0,031 \times 10^2 = 314 \times 10^{-2} \ldots$

Convention : on met un chiffre non nul avant la virgule :
$\pi \simeq 3,14 \times 10^0$.

Représentation :
nombre de chiffres utilisés dans la représentation : constant, dans notre exemple, 3 chiffres décimaux.
Population de Lyon : 445 452 habitants en 1999, représenté par $4,45 \times 10^5$.

Dénomination : notation scientifique dans la vie courante, nombre à virgule flottante ou par ellipse **nombre flottant**.

Les nombres flottants : définition

Exemple : $4.67 \times 10^3 + 3,14 \times 10^0$?

$$
\begin{array}{rrr}
 & 4\ ,\ 6\ 7 & \text{x } 10^3 \\
+ & 0\ ,\ 0\ 0\ 3\ 1\ 4 & \text{x } 10^3 \\
\hline
 & 4\ ,\ 6\ 7\ 3\ 1\ 4 & \text{x } 10^3 \\
\frown & 4\ ,\ 6\ 7 & \text{x } 10^3
\end{array}
$$

Les nombres flottants

Nombre de chiffres fixé $\longrightarrow$ arrondi obligatoire.

Comment arrondir ?

- vers le haut
- vers le bas
- vers 0
- en s'éloignant de 0
- au plus près : dans ce cas, comment arrondir 6,785 ?

Besoin d'une norme

- sur certaines machines Cray, on avait

$$\text{parfois, } 1 \times x \Rightarrow \text{ overflow}$$
$$x + y \neq y + x$$
$$0.5 \times x \neq x/2.0$$

- IBM 370, en Fortran, on avait

$$\sqrt{-4} = 2$$
$$\texttt{I = 14.0/7.0} \rightarrow I = 1$$

Les nombres flottants : la norme IEEE-754

Jusqu'en 1985 : anarchie totale dans le monde des processeurs.

En 1985 : adoption de la norme IEEE-754

- formats fixés :
  simple précision et double précision
- arrondis possibles : vers $+\infty$, vers $-\infty$, vers 0 et au plus près (pair)
- opérations arithmétiques : $+$, $-$, $\times$, $/$ et $\sqrt{}$ doivent rendre l'arrondi du résultat exact (toujours possible avec 3 bits supplémentaires pendant les calculs).

Norme IEEE-754 : avantages et inconvénients

Avantages :

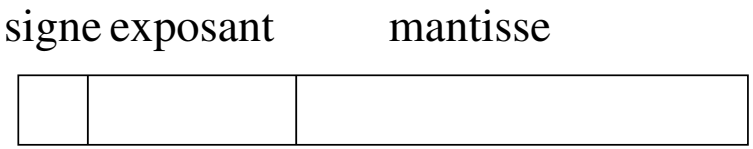
- représentation en ordre de grandeur
- opérations en temps constant (puisque nombre de chiffres constant)
- norme IEEE-754 : opérations le plus précises possible
- norme IEEE-754 : reproductibilité des calculs d'un ordinateur à l'autre
- norme IEEE-754 : calculs bien spécifiés, possibilité de prouver qu'un programme est correct.

Inconvénients :

- résultats toujours arrondis
- précision limitée

Norme IEEE-754 : représentation d'un nombre

(A. Tisserand, cours 2005-2006)



$$x = (-1)^{\text{signe}} \times \text{mantisse} \times 2^{\text{exposant}}$$

Codage :

- **signe** s (codé sur un bit : 0 pour positif et 1 pour négatif)
- **exposant** e entier de k chiffres compris entre $e_{\min}$ et $e_{\max}$
- **mantisse** de $n+1$ chiffres : $m = x_0.x_1x_2\cdots x_n$ où les x_i sont les chiffres (compris entre 0 et $\beta - 1$).

Norme IEEE-754 : formats de base

Pour des questions de précision, on exige que le premier chiffre de la mantisse soit non nul : **normalisation**.
En base 2, la normalisation revient à dire que le premier bit de la mantisse est égal à 1 ; on ne le stocke pas : **bit implicite** ou **bit caché**.

	nombre de bits			
format	total	signe	exposant	mantisse
simple	32	1	8	$23+1$
double	64	1	11	$52+1$

Norme IEEE-754 : mantisse

Pour des questions de précision, on exige que le premier chiffre de la mantisse soit non nul : **normalisation**.

La mantisse normalisée est codée sur $n+1$ bits :

$$m = 1.x_1 x_2 \cdots x_n$$

où les x_i sont des **bits** (pour *binary digits*).

Il faut alors un codage spécial pour 0.

Norme IEEE-754 : exposant

Exposant e : entier signé de k chiffres compris entre $e_{\min}$ et $e_{\max}$.

Représentation biaisée : on représente $e + b$ où b est le biais $\Rightarrow e + b \geq 0$.

Avantage : on peut faire des comparaisons entre flottants dans l'ordre lexicographique (sauf pour le signe).

format	taille k	biais b	exp. non biaisé	exp. biaisé
simple	8	127	$[-126, 127]$	$[1, 254]$
double	11	1023	$[-1022, 1023]$	$[1, 2046]$

Remarque : il reste des valeurs réserées, qui sont 0 et $2^k - 1$ en représentation biaisée.

Norme IEEE-754 : représentation de zéro

- **signe** : positif ou négatif
- **exposant** biaisé à 0
- **mantisse** ovec tous ses chiffres à 0

Remarque : on a deux représentations pour 0, selon la valeur du signe.

Norme IEEE-754 : valeurs spéciales

les infinis : $-\infty$ et $+\infty$

codés en utilisant le plus grand exposant possible et que des 0 pour la mantisse.
Le signe correspond à celui de l'infini.

Not a Number : NaN

- permet de représenter le résultat d'une opération invalide telle que $\sqrt{-2}$, $0/0$ ou $\infty - \infty$
- est codé en utilisant le plus grand exposant possible et une mantisse contenant au moins un 1 ;
- se propage dans les calculs, en contaminant les résultats.

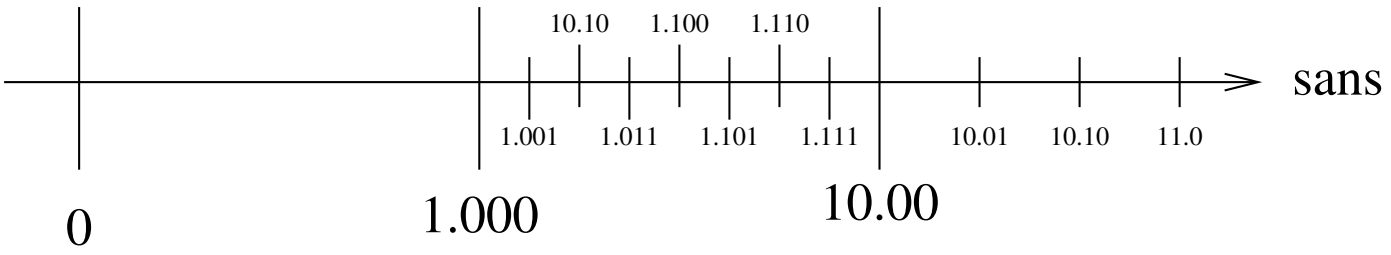
En simple précision

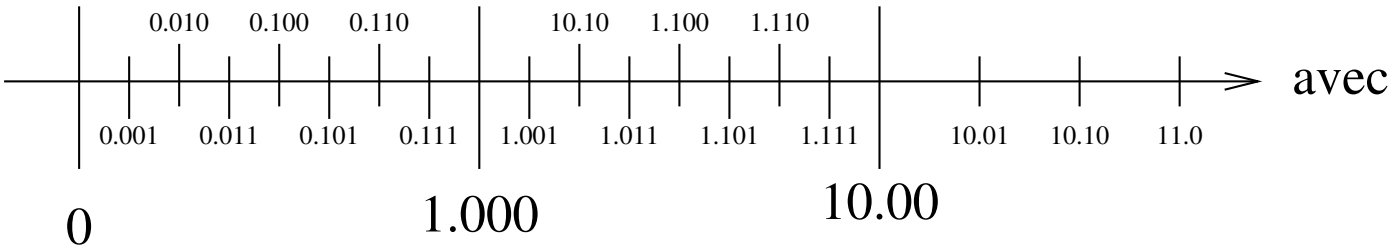
$-\infty$	1	11111111	00000000000000000000000	
$+\infty$	0	11111111	00000000000000000000000	
NaN	0	11111111	00100010000010000001000	(par exemple)

Norme IEEE-754 : subnormals

Pour uniformiser la répartition des nombres représentables autour de 0, on autorise la représentation de nombres qui s'écrivent

$$x = (-1)^{\text{signe}} \times 0.x_1 x_2 \cdots x_n \times 2^{\text{exposant}}$$





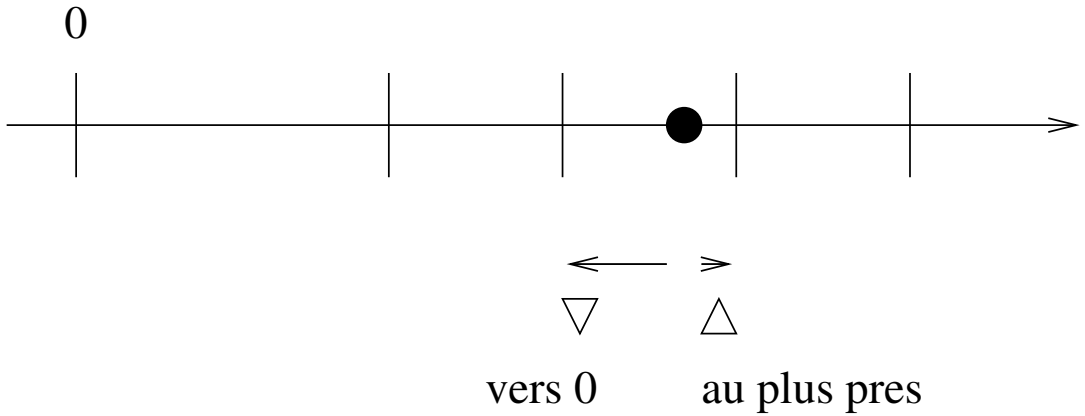
Norme IEEE-754 : modes d'arrondi

- **calculs**

  π : non représentable par un flottant

  produit de 2 nombres flottants : ne rentre pas dans un flottant

- **arrondis**



Norme IEEE-754 : modes d'arrondis

Quatre modes d'arrondi :

- **arrondi vers $+\infty$** ou par excès, noté $\triangle(x)$: retourne le plus petit nombre machine supérieur ou égal au résultat exact x ;
- **arrondi vers $-\infty$** ou par défaut noté $\triangledown(x)$: retourne le plus grand nombre machine inférieur ou égal au résultat exact x ;
- **arrondi vers 0** : retourne $\triangle(x)$ pour les nombres négatifs et $\triangledown(x)$ pour les nombres positifs ;
- **arrondi au plus près** : retourne le nombre machine le plus proche du résultat exact x, celui dont la mantisse se termine par 0 pour le milieu de deux nombres machine consécutifs (on parle d'arrondi pair).

Norme IEEE-754 : arrondi correct

Arrondi correct : soient a et b deux nombres machine et $\odot$ une opération parmi $\{+, -, \times, /\}$ avec o l'un des 4 arrondis prévus par la norme,
le résultat $a \odot b$ doit être $o(a\dot{b})$ l'arrondi du résultat exact.

On a la même exigence pour la racine carrée.

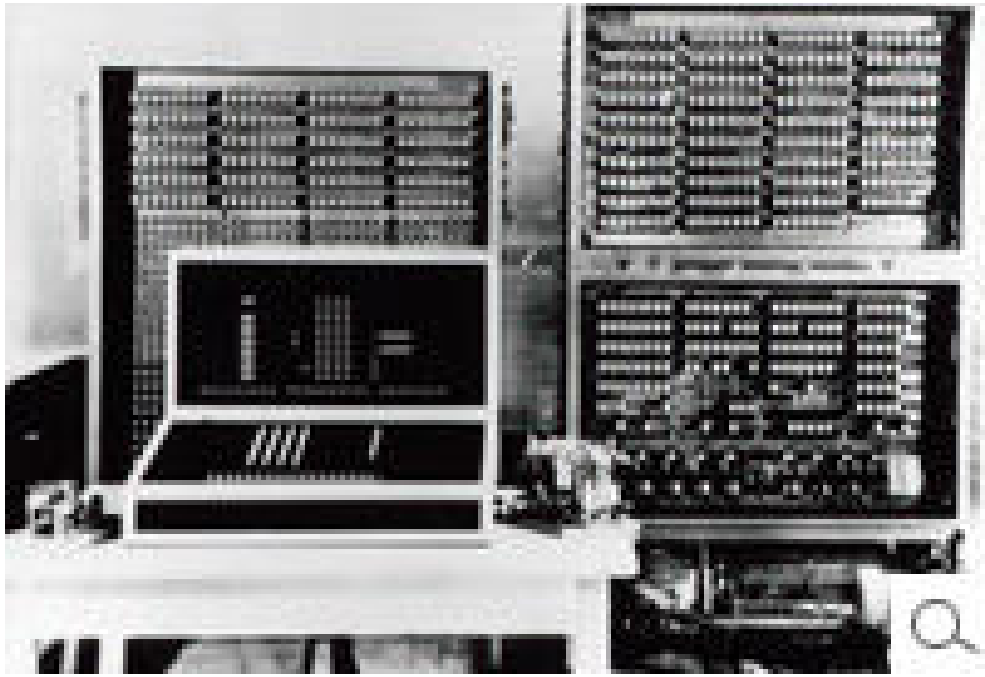
Norme IEEE-754 : conversions

Les conversions normalisées sont :

- flottant vers entier
- entier vers flottant
- flottant vers entier stocké dans un flottant
- entre flottants de différents formats
- entre formats binaire et décimal.

Premier ordinateur flottant ?

Le premier ordinateur avec une arithmétique flottante date de 1941 et a été créé par Konrad Zuse.



Cf. http://www.computerhistory.org/timeline/?year=1941

Norme IEEE-754 : lemme de Sterbenz

Soient a et b deux nombres flottants positifs. Si

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

alors $a - b = a \ominus b$.
Autrement dit, $a - b$ est exactement représentable en arithmétique flottante.

Norme IEEE-754 : une autre propriété valide

Avec le mode d'arrondi au plus près.

Si on calcule en arithmétique flottante

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

a-t-on $-1 \leq z \leq 1$?

Peut-on calculer $t = \sqrt{1 - z^2}$ sans risque d'obtenir un NaN ?

Norme IEEE-754 : une autre propriété valide

Avec le mode d'arrondi au plus près.
Si on calcule en arithmétique flottante

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

a-t-on $-1 \leq z \leq 1$?
Peut-on calculer $t = \sqrt{1 - z^2}$ sans risque d'obtenir un NaN ?

La réponse est **oui**.

Norme IEEE-754 : Computing the roundoff error using FP operations of addition and subtraction

Theorem :
for any pair (x, y) of FP numbers and for rounding to nearest, there exists FP numbers r_+ and r_- such that

$$\begin{aligned} r_+ &= (x+y)-(x \oplus y) \\ r_- &= (x-y)-(x \ominus y) \end{aligned}$$

Furthermore, r_+ and r_- can be computed using FP operations.

Norme IEEE-754 : Computing the roundoff error : + and −

Why with "rounding to nearest" only ?
Counterexample for directed rounding

with basis 2 and at least $p > 4$ bits of mantissa, let's take

$$\begin{aligned} x &= -(2^{2p} + 2^{p+1}) \\ y &= 2^p - 3 \end{aligned}$$

then we have

$$\begin{aligned} x + y &= -2^{2p} - 2^p - 3 \\ x \oplus y &= -2^{2p} \\ (x + y) - (x \oplus y) &= -2^p - 3 \qquad \text{not representable.} \end{aligned}$$

Norme IEEE-754 : Computing roundoff errors using FP operations : $+$

Let x and y be two normal FP numbers such that $|x| \geq |y|$ and the rounding mode be to nearest. Let also assume that $x \oplus y$ does not overflow.

$$\text{Algo Fast2Sum :} \quad \begin{aligned} s &= x \oplus y \\ z &= s \ominus x \\ r &= y \ominus z \end{aligned}$$

The mathematical equality holds :

$$s + r = x + y$$

i.e. r is the roundoff error on the addition of x and y.
Beware of the optimizations done by your compiler...

Norme IEEE-754 : Proof of Fast2Sum

$$\text{Algo Fast2Sum :} \quad \begin{aligned} s &= x \oplus y \\ z &= s \ominus x \\ r &= y \ominus z \end{aligned}$$

- if x and y have the same sign : then $x \leq x + y \leq 2x$ thus $x \leq s \leq 2x$ s since $2x$ is representable and since the rounding mode is monotonic.
  By Sterbenz lemma, z exactly equals $s - x$.
  Since $(x + y) - s$ is exactly representable, then $r = (x \oplus y) \ominus s = (x + y) - s$ and since $y - z = r$, then $b \ominus z = r$ exactly.

Norme IEEE-754 : Proof of Fast2Sum

- if x and y have opposite signs :
  - either $|y| \geq \frac{1}{2}|x|$ and Sterbenz lemma applies : $x \oplus y$ is exact, i.e. $s = x + y$, $z = b$ and $r = 0$;
  - or $|y| < \frac{1}{2}|x|$ and thus $|x + y| > \frac{1}{2}|x|$.
    This implies that $|s| \geq \frac{1}{2}|x|$, since $\frac{1}{2}|x|$ is representable and rounding is monotonic, then Sterbenz implies that $z = s \ominus x = s - x$ exactly.
    Since $(x + y) - s$ is exactly representable, then $r = (x \oplus y) \ominus s = (x + y) - s$ and since $y - z = r$, then $b \ominus z = r$ exactly.

This algo is also correct under the weaker assumption that the exponent of $x \geq$ the exponent of y.

Norme IEEE-754 : Computing roundoff errors using FP operations : $+$

To avoid comparing x and y (a comparison can be costly), let us use

$$\begin{array}{lrcl} \text{Algo TwoSum :} & s & = & x \oplus y \\ & y' & = & s - x \\ & x' & = & s - y' \\ & \delta_y & = & y - y' \\ & \delta_x & = & x - x' \\ & r & = & \delta_x + \delta_y \end{array}$$

The mathematical equality holds : $s + r = x + y$
i.e. r is the roundoff error on the addition of x and y.

Norme IEEE-754 : Computing roundoff errors of $\times$

x, y two normal FP nbs, rounding to nearest, $x \otimes y$ does not overflow.

Theorem : $r = (x \times y) - (x \otimes y)$ is representable.

Denote by $s = \lceil \frac{p}{2} \rceil$.

Algo TwoMult : (aka Dekker)

$$
\begin{aligned}
x' &= x \otimes (2^s \oplus 1) \\
x_h &= (x \ominus x') \oplus x' \\
x_l &= x \ominus x_h \\
&\text{ibid. for } y \\
r_h &= x \otimes y \\
r_l &= (((x_h \otimes y_h \ominus r_h) \\
&\qquad \oplus x_h \otimes y_l) \oplus x_l \otimes y_h) \\
&\qquad \oplus x_l \otimes y_l
\end{aligned}
$$

The mathematical equality holds : $r_h + r_l = x \times y$

i.e. r_l is the roundoff error on the multiplication of x and y.

17 operations : 7 $\otimes$ and 10 $\pm$.

Norme IEEE-754 : Computing roundoff errors using FP operations : $\times$

Let x and y be two normal FP numbers and the rounding mode be to nearest. Let also assume that $x \oplus y$ does not overflow.
If a *fma* is available : *fused multiply-add*, it computes the rounding of $(a \times b + c)$.

$$\text{Algo TwoMultFMA :} \quad \begin{aligned} r_h &= x \otimes y \\ r_l &= fma(x, y, -r) \end{aligned}$$

The mathematical equality holds :

$$r_h + r_l = x \times y$$

i.e. r_l is the roundoff error on the multiplication of x and y.

Encore de la recherche à faire

Fonctions élémentaires

fonctions élémentaires : les calculer vite et bien

système	$sin(10^{22})$ (Ng)
valeur exacte	-0.852200849767...
HP 48 GX	-0.852200849767
HP 700	0.0
IBM 3090/600S-VF AIX 370	0.0
Matlab 4.2c.1 Sparc	-0.8522
Matlab 4.2c.1 MacIntosh	0.8740
SG Indy	0.87402806
Sharp EL5806	-0.090748172
DEC Station 3100	NaN

Encore de la recherche à faire

Fonctions élémentaires :

sinus, cosinus hyperbolique, arc-tangente, exponentielle, logarithme...

- les faire entrer dans la norme IEEE-754
- les calculer correctement

JM Muller's recurrence

Définition de la récurrence :

$$x_{n+1} = 108 - \frac{815 - \frac{1500}{x_n - 1}}{x_n}$$

avec $x_0 = 4$ et $x_1 = 4.25$.
Calculer x_{80}.

Can You "Count" on Your Computer ? up to 6... (N. Higham)

$$2 - 1$$

$$\left(\frac{1}{\cos(100\pi+\pi/4)}\right)^2$$

$$3.0 * (tan(atan(10000000.0))/10000000.0)$$

$$\left(\cdots(\sqrt{\cdots\sqrt{4}})^2\cdots\right)^2 \text{ (20 fois)}$$

$$5 \times \frac{(1+e^{-100})-1}{(1+e^{-100})-1}$$

$$\frac{\ln(e^{6000})}{1000}$$

One, two, three (N. Higham)

$2 - 1$	1.0000000000000000
$\left(\frac{1}{\cos(100\pi+\pi/4)}\right)^2$	2.0000000000001110
$3.0 * (tan(atan(10000000.0))/10000000.0)$	3.0000000030727567

Is this a FLOP ?

Four, five, six (N. Higham)

$\left(\cdots(\sqrt{\cdots\sqrt{4}})^2\cdots\right)^2$ (20 fois)	4.0000000006294343
$5 \times \frac{(1+e^{-100})-1}{(1+e^{-100})-1}$	NaN
$\frac{\ln(e^{6000})}{1000}$	$+\infty$

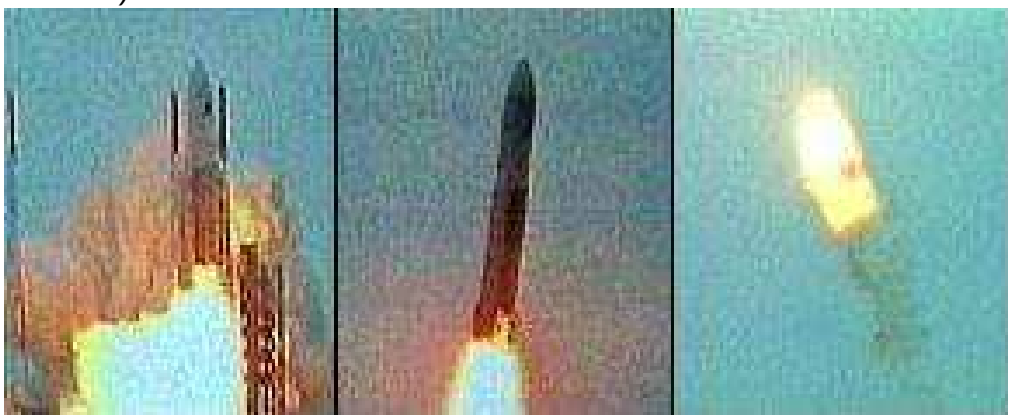
Quelques exemples de ratés

- **bourse de Vancouver :** introduction en 1982 d'un nouvel indice, de valeur de départ 1000.000 recalculé après chaque transaction et tronqué au 3e chiffre après la virgule ;
  valeur calculée 22 mois plus tard : 524.881, alors que la valeur correcte était 1098.811.
- **système de défense anti-missile :** un missile Patriot échoue à intercepter un Scud (février 1991, Dharan, Arabie Saoudite), bilan : 28 morts ;
  explication : une petite erreur d'arrondi qui finit par être accumulée à d'autres petites erreurs d'arrondi, toutes dans le même sens.

Quelques exemples de ratés

Gestion des exceptions

- Nov. 1998, navire lance-missiles américain USS Yorktown, on a par erreur tapé un «zéro» sur un clavier → division par 0. Le programmeur n'avait pas songé que ce problème pourrait arriver → cascade d'erreurs → arrêt du système de propulsion.
- premier envol. . . et premier plongeon d'Ariane 5 (7 millions d'euros)



Limitations de la norme IEEE-754

- besoins en dynamique ?

$$\frac{\text{Diamètre estimé de l'Univers}}{\text{Distance de Planck}} \approx 1.4 \times 10^{62}$$

- besoins en précision ? Certaines prédictions de la relativité générale et de la mécanique quantique vérifiées avec une précision relative de $\sim 10^{-14}$
- calculs intermédiaires : besoin de quadruple précision (J. Laskar, Obs. de Paris) pour stabilité à très long terme du système solaire ;
- record actuel : 1241 milliards de chiffres décimaux de π (Kanada, 2002), en utilisant les deux formules

$$\begin{aligned}\pi &= 48\arctan\tfrac{1}{49} + 128\arctan\tfrac{1}{57} - 20\arctan\tfrac{1}{239} + 48\arctan\tfrac{1}{110443} \\ &= 176\arctan\tfrac{1}{57} + 28\arctan\tfrac{1}{239} - 48\arctan\tfrac{1}{682} + 96\arctan\tfrac{1}{12943}\end{aligned}$$

Norme IEEE-754 : un futur plus performant, mais moins sûr ?

- *The PlayStation 3 will feature the much-vaunted Cell processor, which will run at 3.2GHz, giving the whole system 2 teraflops of overall performance.*

Norme IEEE-754 : un futur plus performant, mais moins sûr ?

- *The PlayStation 3 will feature the much-vaunted Cell processor, which will run at 3.2GHz, giving the whole system 2 teraflops of overall performance.*
- prochaine machine pétaflopique (10^{15}) : *"IBM has announced that they are gearing up to build the world's fastest supercomputer, more than four times faster than the reigning champ, IBM's BlueGene/L. Nicknamed 'Roadrunner,' the new machine will be a hybrid of off-the-shelf CPUs and Cell chips designed for the PS3. [...]According to the BBC : 'The computer will contain 16,000 standard processors working alongside 16,000 Cell processors... each Cell is capable of 256 billion calculations per second.'"*

Norme IEEE-754 : un futur plus performant, mais moins sûr ?

- *The PlayStation 3 will feature the much-vaunted Cell processor, which will run at 3.2GHz, giving the whole system 2 teraflops of overall performance.*
- prochaine machine pétaflopique (10^{15}) : *"IBM has announced that they are gearing up to build the world's fastest supercomputer, more than four times faster than the reigning champ, IBM's BlueGene/L. Nicknamed 'Roadrunner,' the new machine will be a hybrid of off-the-shelf CPUs and Cell chips designed for the PS3. [...]According to the BBC : 'The computer will contain 16,000 standard processors working alongside 16,000 Cell processors... each Cell is capable of 256 billion calculations per second.'"*
- arithmétique flottante : arrondi vers 0 ! ! !

Compter
Et votre ordinateur, il compte comment ?

- comment représenter les nombres
- les opérations apprises à l'école primaire
- le calcul en virgule flottante (ou la notation scientifique, vue au collège)
- **le calcul par intervalles**

Compter sans se tromper : arithmétique par intervalles

(Moore 1966, Kulisch 1983, Neumaier 1990, Rump 1994, Alefeld and Mayer 2000...)

Principe

Nombres remplacés par des intervalles.

π remplacé par $[3.14159, 3.14160]$

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros, $\in [10, 20]$ Euros.

Arithmétique par intervalles : compter

$$[10, 20] + [5, 10] = [15, 30]$$
$$[-2, 3] + [5, 7] = [3, 10]$$

$$[-3, 2] * [-3, 2] = [-6, 9] \text{ est différent de } [-3, 2]^2 = [0, 9]$$

$$[-3, 2]/[0.5, 1] = [-6, 4]$$

$$X \diamond Y = \{x \diamond y \ / \ x \in X,\ y \in Y\}$$

$$\exp[-2, 3] = [\exp(-2), \exp(3)]$$
car exp est une fonction croissante.

$$\sin[\pi/3, \pi] = [0, 1]$$
attention, sin n'est pas monotone.

Arithmétique par intervalles : avantages

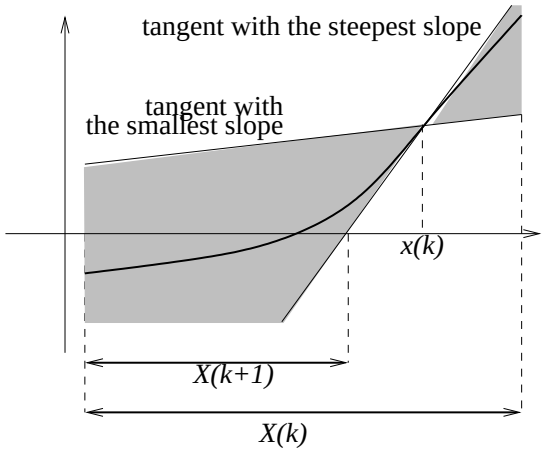
Prise en compte des incertitudes
données expérimentales, résultats de mesures physiques, valeurs non exactement représentables.

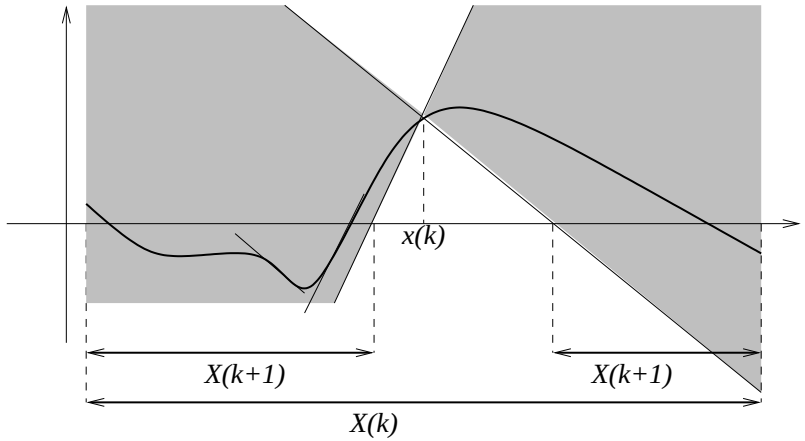
Calcul garanti
le résultat cherché appartient à l'intervalle calculé.

Information globale
on sait encadrer l'image d'une fonction sur tout un intervalle.

Algorithme de Newton par intervalles

(Hansen & Greenberg 1983, Mayer 1995, van Hentenryck et al. 1997. . .)





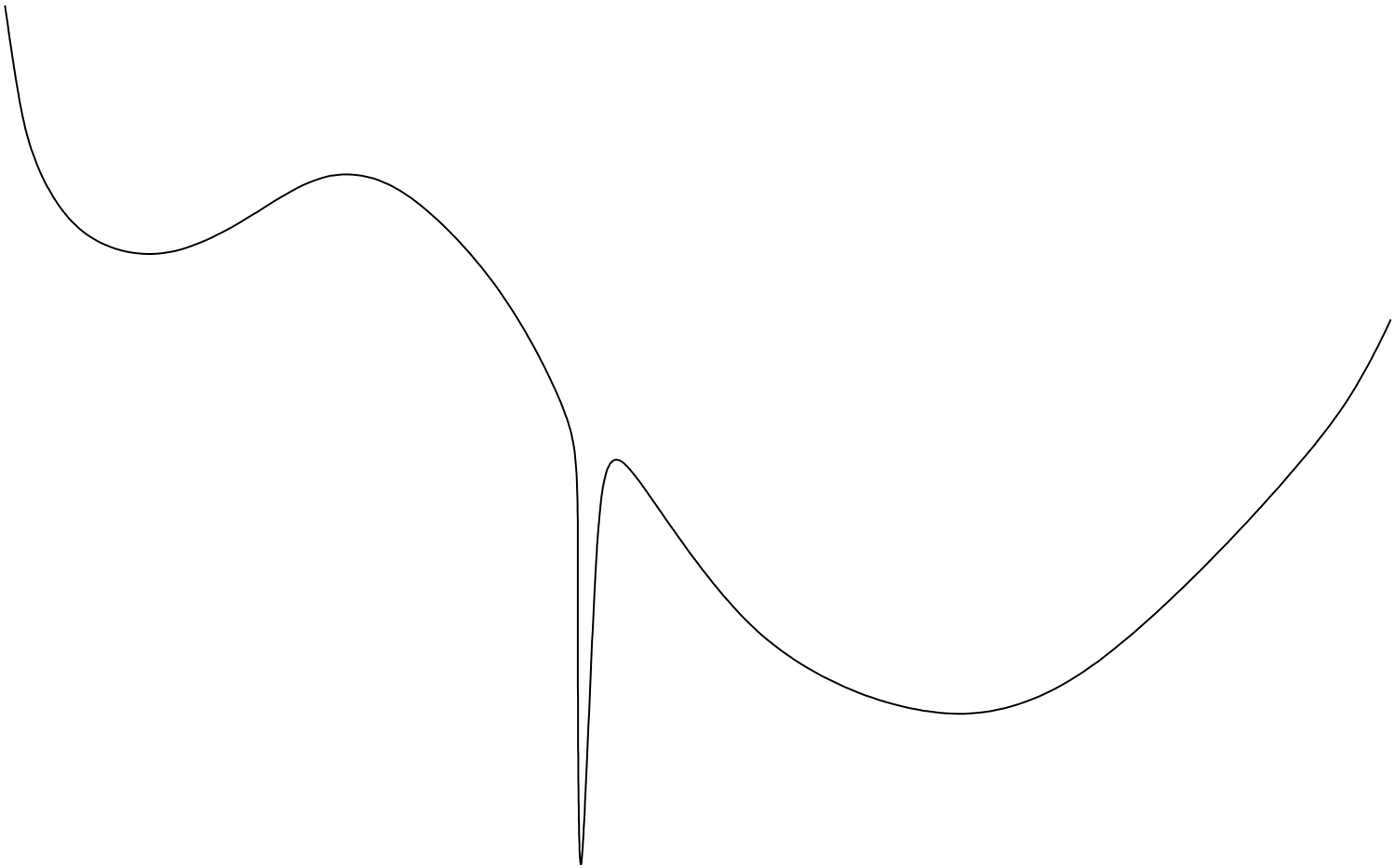
The result will be a list of intervals.

Théorème de Brouwer :

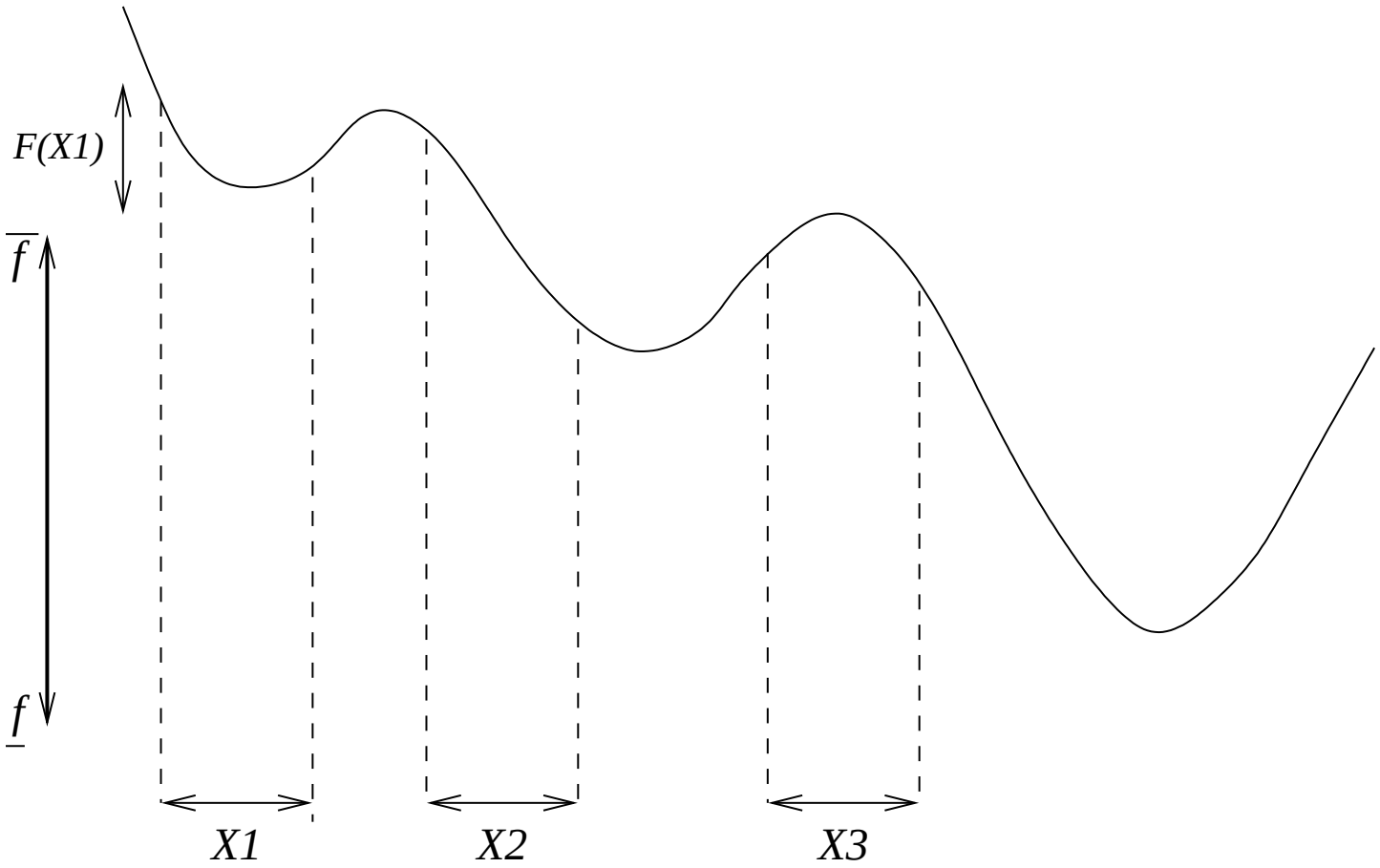
si $f(I) \subset I$ alors f admet un point fixe dans I.

Algorithme de Hansen par intervalles

(Hansen 1992)



Algorithme de Hansen par intervalles



f trop haute : F(X1) > $\overline{f}$

f non convexe sur X3

0 n'est pas dans G(X2)

Algorithme de Hansen

$\mathcal{L}$ = liste des pavés en attente := $\{X_0\}$

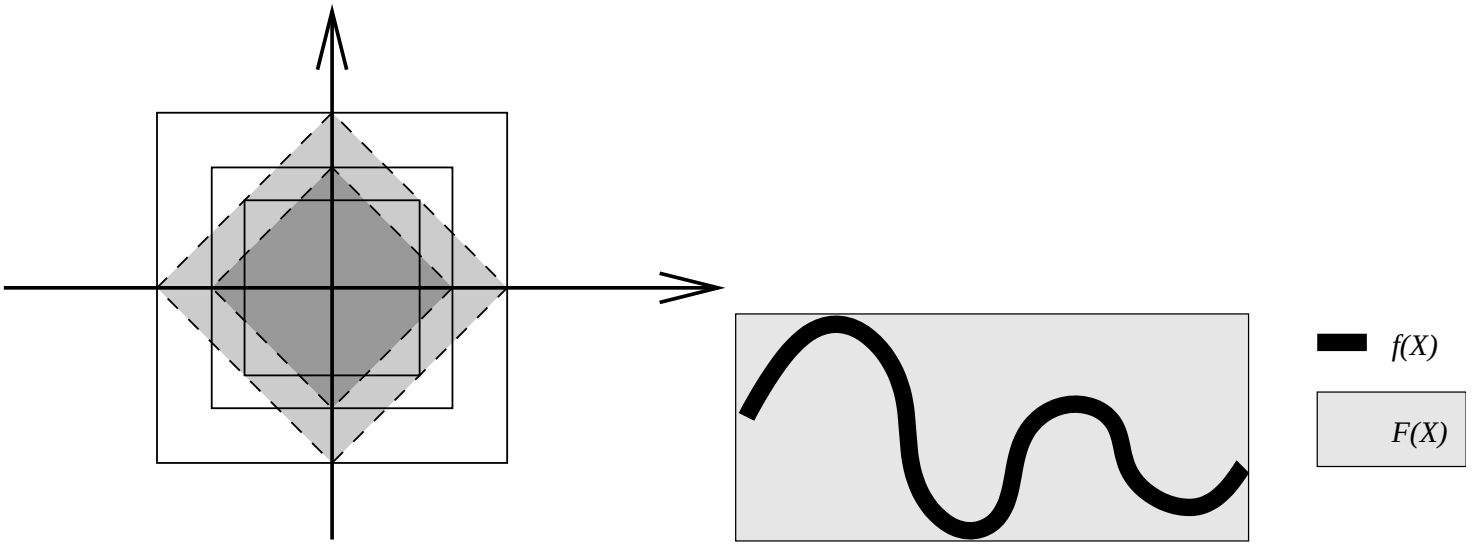
tant que $\mathcal{L} \neq \emptyset$ faire
- sortir X de $\mathcal{L}$
- **rejeter** X **?**
  - oui si $F(X) > \bar{f}$
  - oui si Grad$F(X) \not\ni 0$
  - oui si $HF(X)$ à diagonale non > 0
- **réduire** X
  - Newton sur le gradient
  - résoudre $Y \subset X$ tel que $F(Y) \leq \bar{f}$
- **couper** Y **en 2** : Y_1 et Y_2
  - ranger Y_1 et Y_2 dans $\mathcal{L}$

Arithmétique par intervalles : problèmes

Décorrélation des variables ou *variable dependency*

$$I - I = \{x - y \; ; \; x \in I, \; y \in I\} \supset \{0\} = \{x - x \; ; \; x \in I\}$$

Effet enveloppant ou *wrapping effect*



Arithmétique par intervalles : plus de précision

Idée :
calculer par intervalles pour conserver la garantie sur les résultats
utiliser des flottants en précision arbitraire pour gagner en précision.

Exemple : π

10 bits	[3.140,	3.145]
24 bits	[3.1415925,	3.1415928]
53 bits	[3.141592653589793,	3.141592653589794]
100 bits	[3.14159265358979323846264338327	9, 3.14159265358979323846264338

Bilan

On sait calculer

comme on a appris à l'école
les opérations arithmétiques exactes et flottantes

Pour calculer plus vite :

changer de système de numération
oublier les algorithmes qu'on a appris à l'école
et ne pas se tromper !

Pour calculer plus fiable

avoir une arithmétique bien spécifiée
avoir une arithmétique par intervalles

Perspectives

Il reste encore de la recherche à mener :

- en arithmétique proprement dite : représentation des nombres, implantation matérielle (spécifique ou non) ou logicielle (calcul exact ou flottant en précision arbitraire)
- sur la normalisation : arithmétique flottante, arithmétique par intervalles
- sur l'utilisation de ces arithmétiques : analyse numérique pour l'arithmétique flottante, calcul par intervalles...
- sur l'automatisation de mécanismes qui garantissent le bon fonctionnement d'un code.

Il reste encore de l'enseignement à donner sur les utilisations de ces différentes arithmétiques.

Bibliographie

1. Stanislas Dehaene : *La bosse des maths*, Odile Jacob, 1997, 2003.
2. Georges Ifrah : *Histoire des chiffres*, Robert Laffont, 1994
3. Geneviève Guitel : Histoire comparée des numérations écrites, Flammarion, 1975.
4. *Histoire d'algorithmes - Du caillou à la puce*, J.-L. Chabert et al., Belin 1994
5. *Logique, informatique et paradoxes*, J.-P. Delahaye, Belin 1995
6. *Histoire des codes secrets*, S. Singh, Le livre de poche 2001
7. http://www.col-camus-soufflenheim.ac-strasbourg.fr/Page.php?IDP=79&IDD=0 la page des maths du collège Albert Camus à Soufflenheim
8. http://perso.orange.fr/therese.eveilleau/pages/hist_mat/indexF.htm le site d'une formatrice à l'IUFM