Collège de France
Cours de Xavier Leroy

L'arithmétique des ordinateurs et sa formalisation

Sylvie Boldo

Inria, Université Paris-Saclay

19 décembre 2019

Merci à...

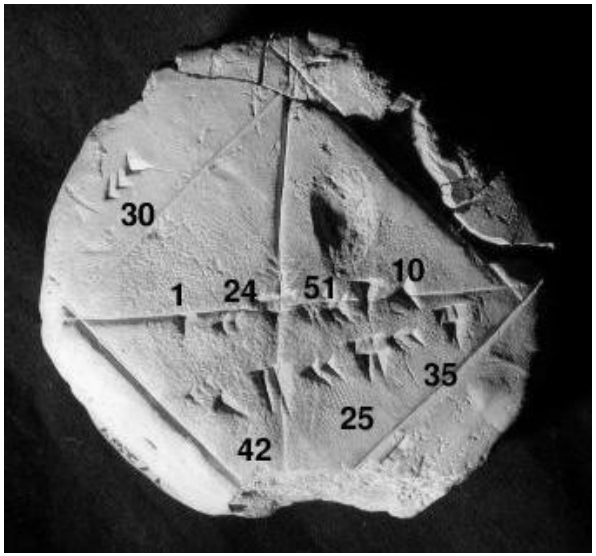
- Xavier Leroy
- Guillaume Melquiond

Plan

Plan

Plan

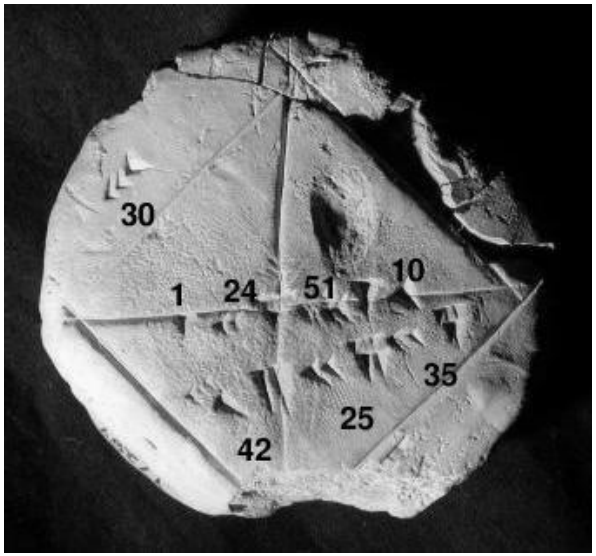
Babyloniens, 2000 avant JC



Tablette YBC 7289
(1600-1800 avant JC)

- les quatre opérations
- extraction de racines carrées, racines cubiques
- résolution d'équations du second degré...

Babyloniens, 2000 avant JC

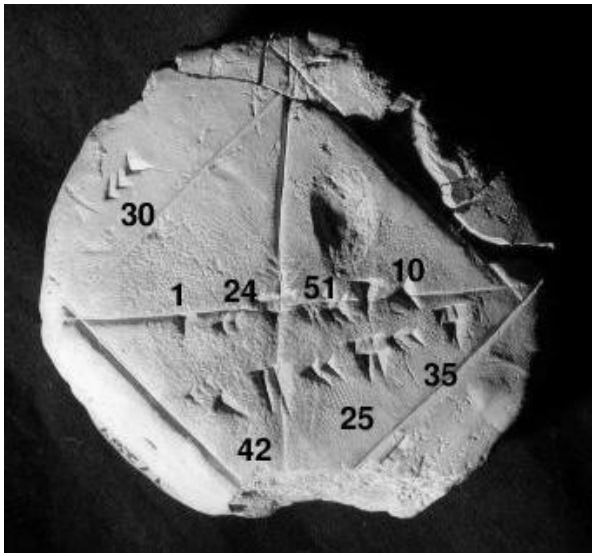


Tablette YBC 7289
(1600-1800 avant JC)

- les quatre opérations
- extraction de racines carrées, racines cubiques
- résolution d'équations du second degré...

En base 60 !

Babyloniens, 2000 avant JC



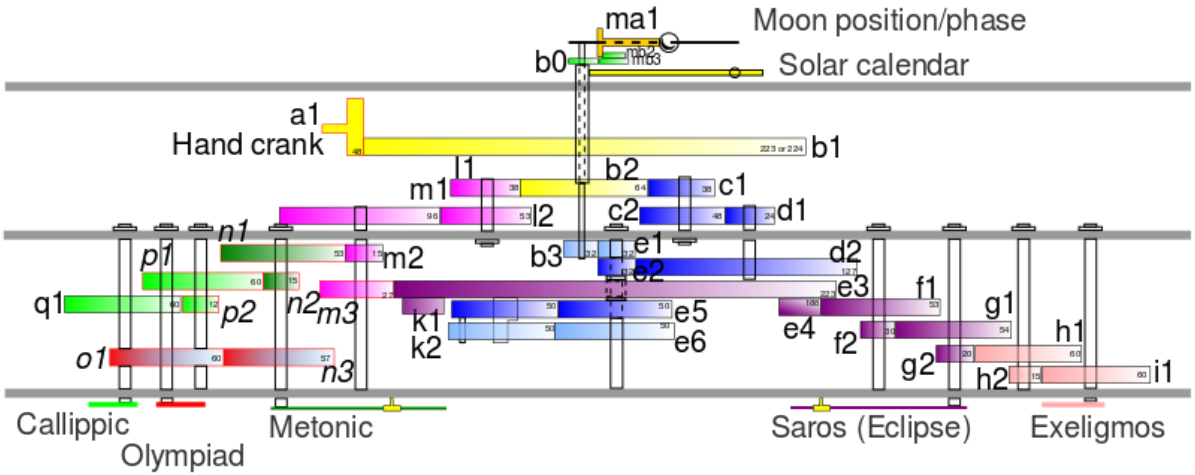
Tablette YBC 7289
(1600-1800 avant JC)

- les quatre opérations
- extraction de racines carrées, racines cubiques
- résolution d'équations du second degré...

En base 60 !

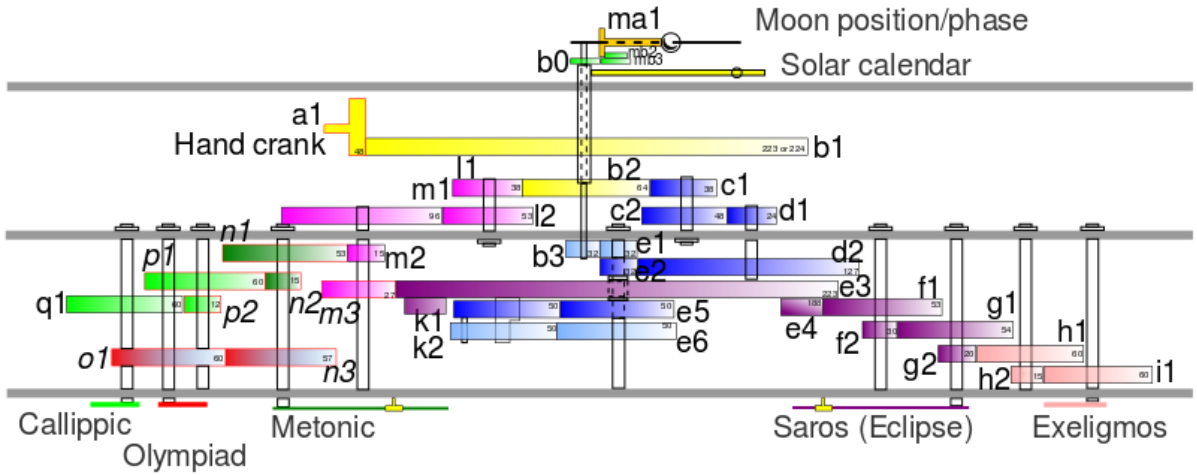
La photo représente $\sqrt{2}$ avec quatre chiffres sexagésimaux significatifs soit près de six chiffres décimaux !
$\Rightarrow$ représentation d'un nombre réel en précision finie

Le calcul mécanisé

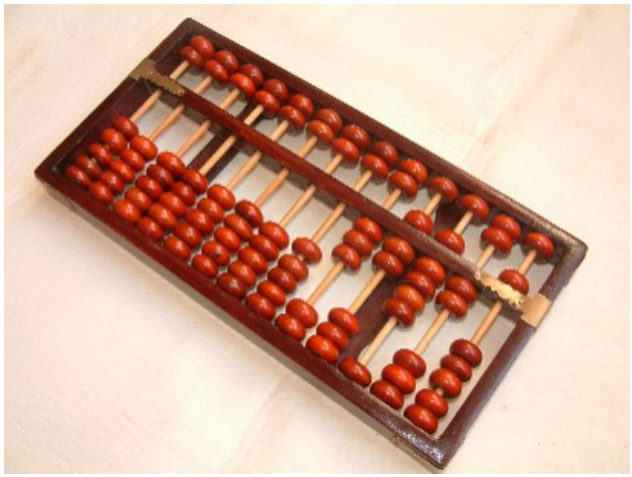


Machine d'Anticythère(100 avant JC)

Le calcul mécanisé

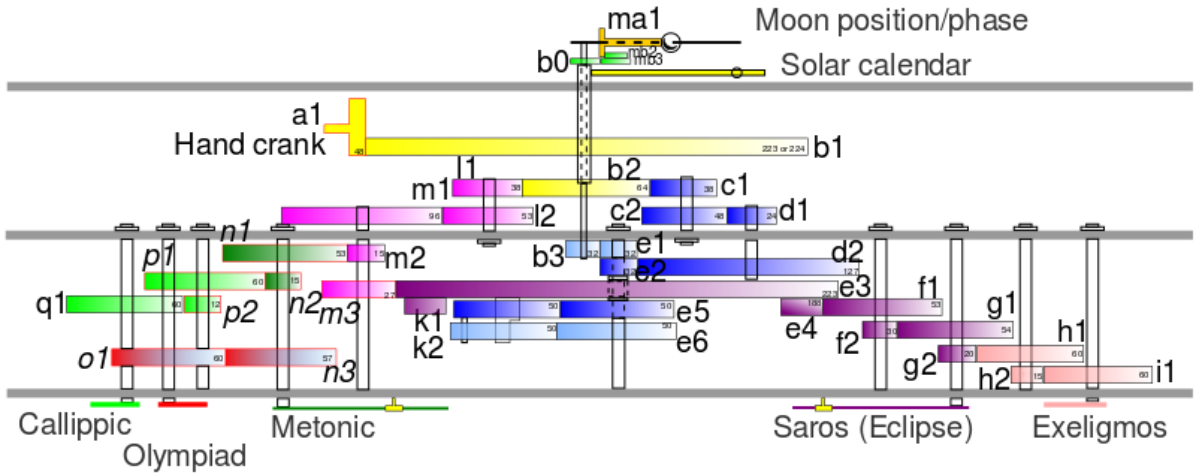


Machine d'Anticythère(100 avant JC)

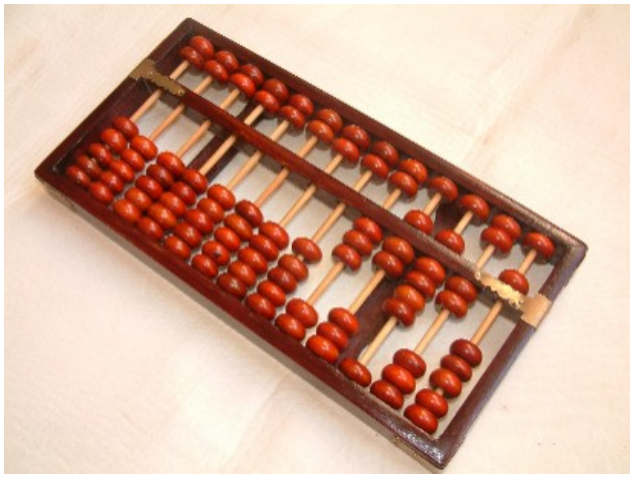


Boulier (2500 avant JC)

Le calcul mécanisé



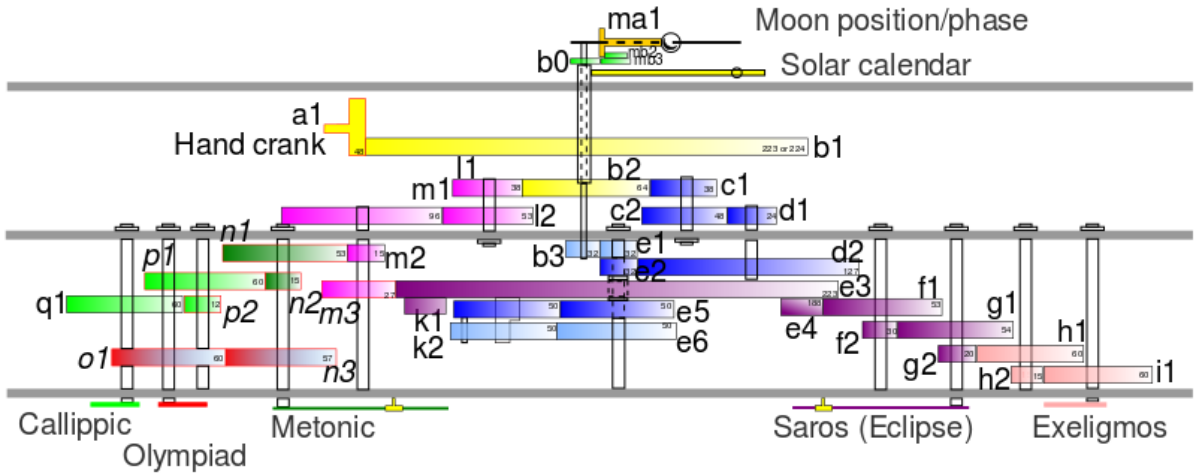
Machine d'Anticythère(100 avant JC)



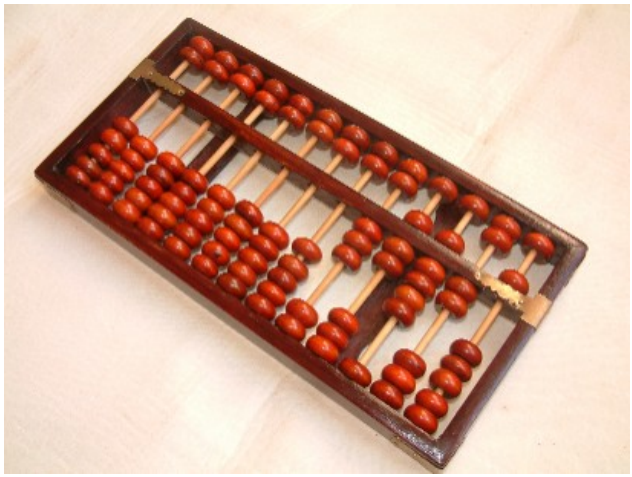
Boulier (2500 avant JC)

Curta (40)

Le calcul mécanisé

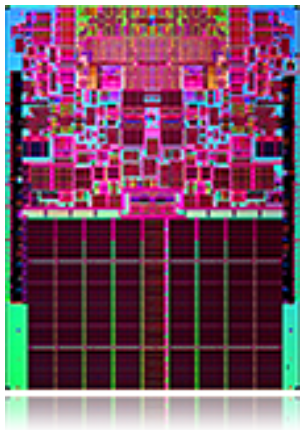


Machine d'Anticythère(100 avant JC)



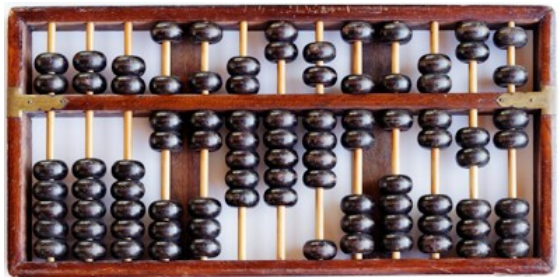
Boulier (2500 avant JC)

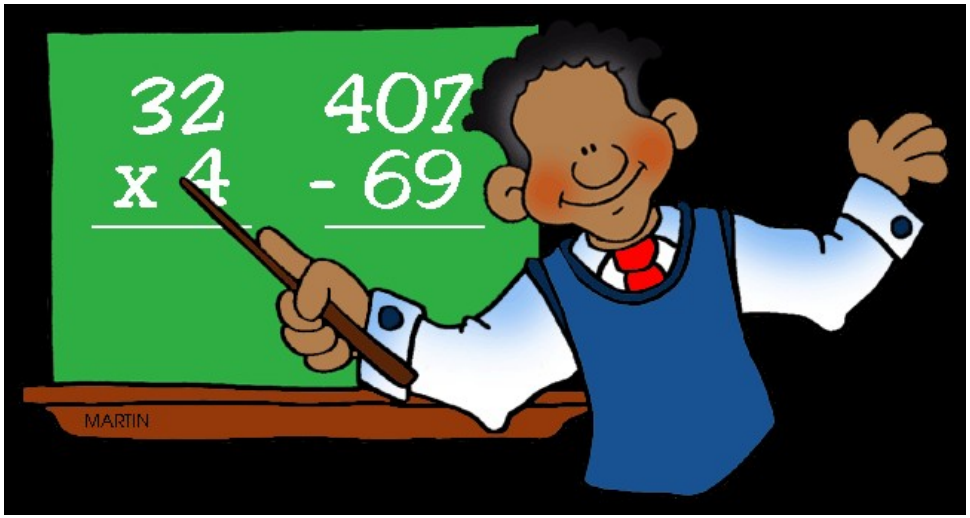
Curta (40)

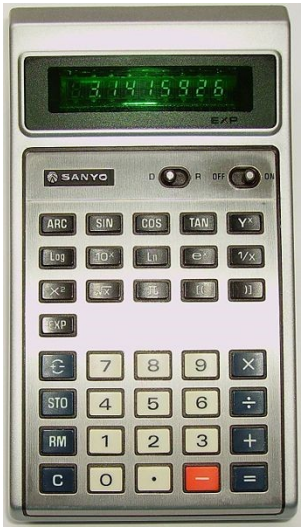


Intel Pentium (90)

Le calcul humain assisté (G. Berry ©)







Historique

1941 Premier vrai processeur avec unité flottante, le Z3 de K. Zuse

Historique

1941 Premier vrai processeur avec unité flottante, le Z3 de K. Zuse

1980 Coprocesseur flottant Intel 8087 (50 000 flops !)

Historique

1941 Premier vrai processeur avec unité flottante, le Z3 de K. Zuse

1980 Coprocesseur flottant Intel 8087 (50 000 flops !)

1985 Norme IEEE-754 régissant les formats et les calculs flottants

Historique

1941 Premier vrai processeur avec unité flottante, le Z3 de K. Zuse

1980 Coprocesseur flottant Intel 8087 (50 000 flops !)

1985 Norme IEEE-754 régissant les formats et les calculs flottants

2008 Révision de la norme IEEE-754 (dont ajout du décimal)

Plan

Les nombres à virgule flottante

Le mémoire de mon ordinateur étant limitée, on limite la précision (le nombre de chiffres) qu'on utilise. Les valeurs ainsi représentées sont appelées nombres flottants.

$$\pi \quad \hookrightarrow \quad 3{,}14$$

Les nombres à virgule flottante

Le mémoire de mon ordinateur étant limitée, on limite la précision (le nombre de chiffres) qu'on utilise. Les valeurs ainsi représentées sont appelées nombres flottants.

$$\pi \quad \hookrightarrow \quad 3{,}14$$

$$123\ 456 \quad \hookrightarrow \quad 1{,}23 \times 10^5$$

$$\frac{1}{17} = 0{,}058\ 823\ 5\ldots \quad \hookrightarrow \quad 5{,}88 \times 10^{-2}$$

Les nombres à virgule flottante

Le mémoire de mon ordinateur étant limitée, on limite la précision (le nombre de chiffres) qu'on utilise. Les valeurs ainsi représentées sont appelées nombres flottants.

$$\pi \quad \hookrightarrow \quad 3{,}14$$

$$123\ 456 \quad \hookrightarrow \quad 1{,}23 \times 10^5$$

$$\frac{1}{17} = 0{,}058\ 823\ 5\ldots \quad \hookrightarrow \quad 5{,}88 \times 10^{-2}$$

Ces exemples sont en base 10 avec 3 chiffres.
Le processeur utilise la base 2 avec 24 ou 53 chiffres.

Les calculs (version base 10 avec 3 chiffres)

Chaque calcul simple est ensuite fait au mieux : le processeur renvoie le meilleur résultat possible étant donnés ses impératifs.

$$3{,}14^2 = 9{,}859\ 6 \quad \hookrightarrow \quad 9{,}86$$

Chaque résultat de calcul est donc arrondi.

Beaucoup de calculs

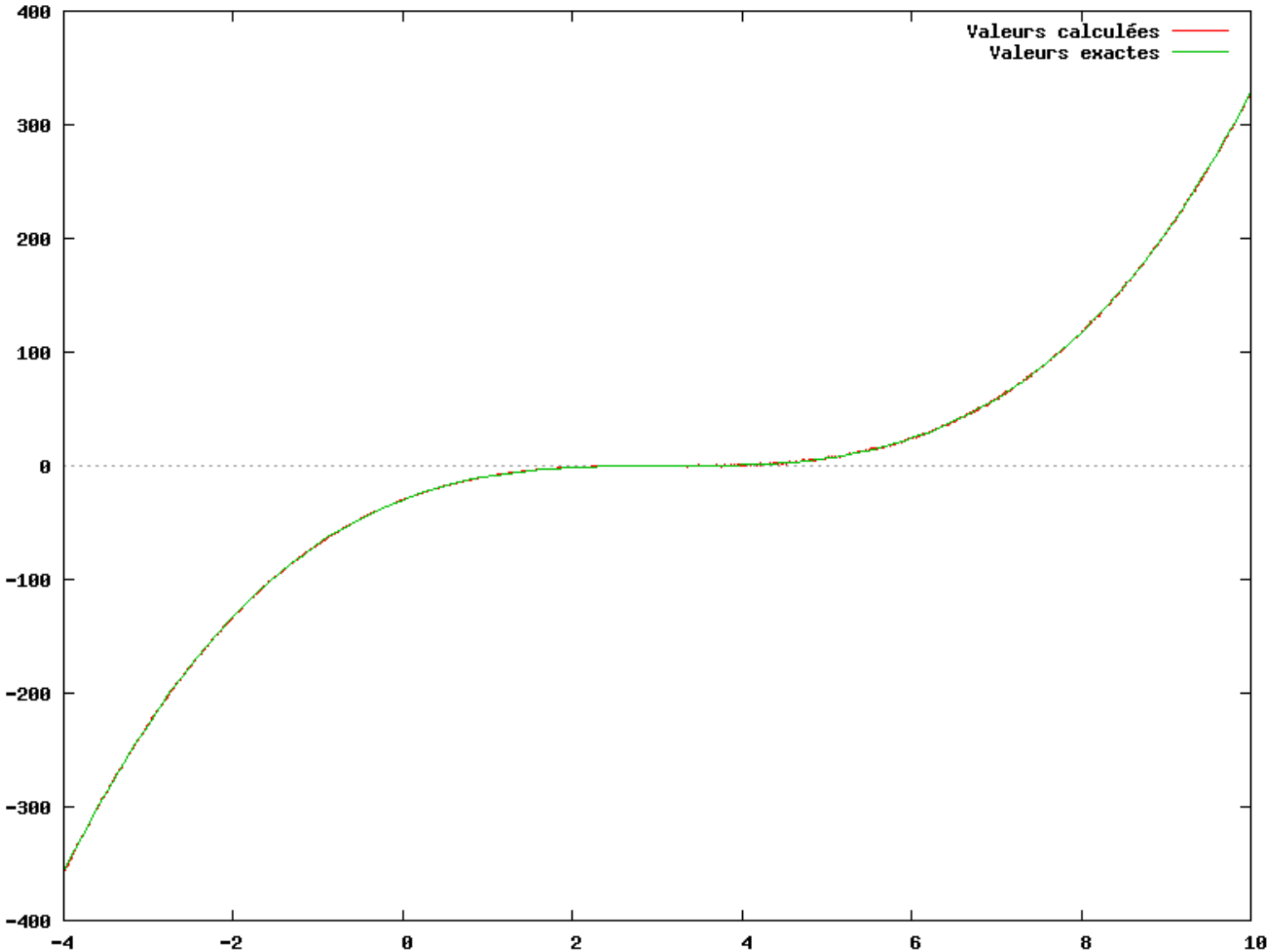
Même si un calcul est presque juste, une succession de calculs est parfois fausse :

$$\pi^2 \quad \hookrightarrow \quad 3{,}14^2 \quad \hookrightarrow \quad 9{,}86$$

Mais $\pi^2 = 9{,}869\ 604\ldots$ donc le nombre flottant le plus proche est 9,87.

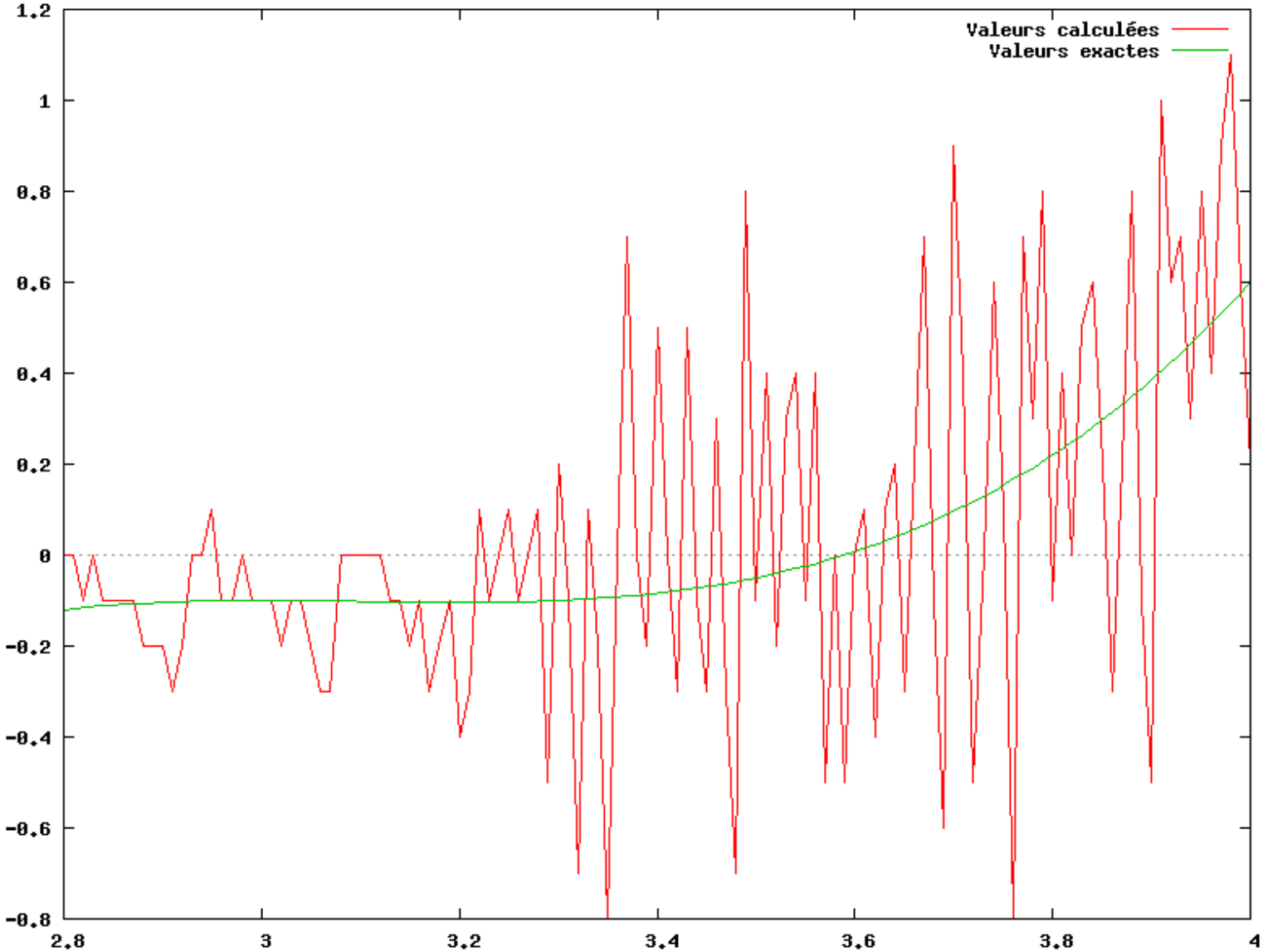
Vraiment beaucoup de calculs

Soit $P(x) = x^3 - 9{,}3x^2 + 28{,}8x - 29{,}8$.



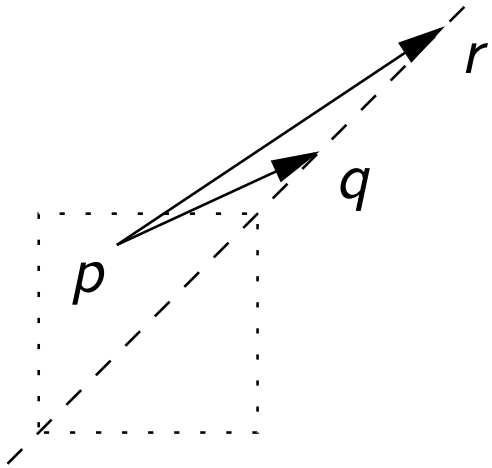
Vraiment beaucoup de calculs

Soit $P(x) = x^3 - 9{,}3x^2 + 28{,}8x - 29{,}8$.



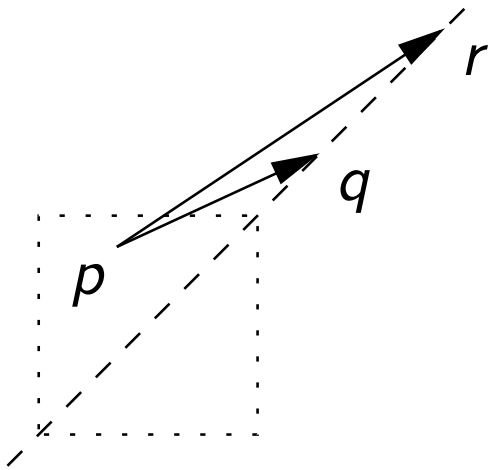
Orientation de 3 points

Étant donnés 3 points du plan p, q et r. On veut savoir si pqr sont alignés dans le sens horaire ou dans le sens anti-horaire.



Orientation de 3 points

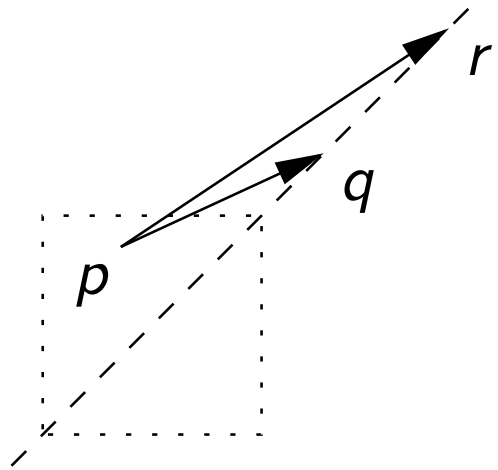
Étant donnés 3 points du plan p, q et r. On veut savoir si pqr sont alignés dans le sens horaire ou dans le sens anti-horaire.



$$\text{orient}_2(p,q,r) = \text{signe} \begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$

Orientation de 3 points

Étant donnés 3 points du plan p, q et r. On veut savoir si pqr sont alignés dans le sens horaire ou dans le sens anti-horaire.

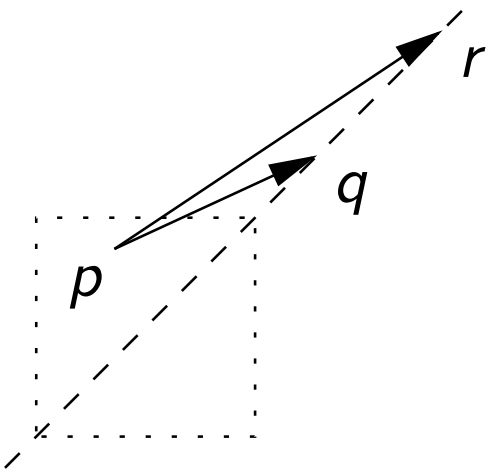


$$\text{orient}_2(p,q,r) = \text{signe} \begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$

```
float det = (qx - px) * (ry - py)
          - (qy - py) * (rx - px);
if (det > 0) return POSITIVE;
if (det < 0) return NEGATIVE;
return ZERO;
```

Orientation de 3 points - calculs exacts

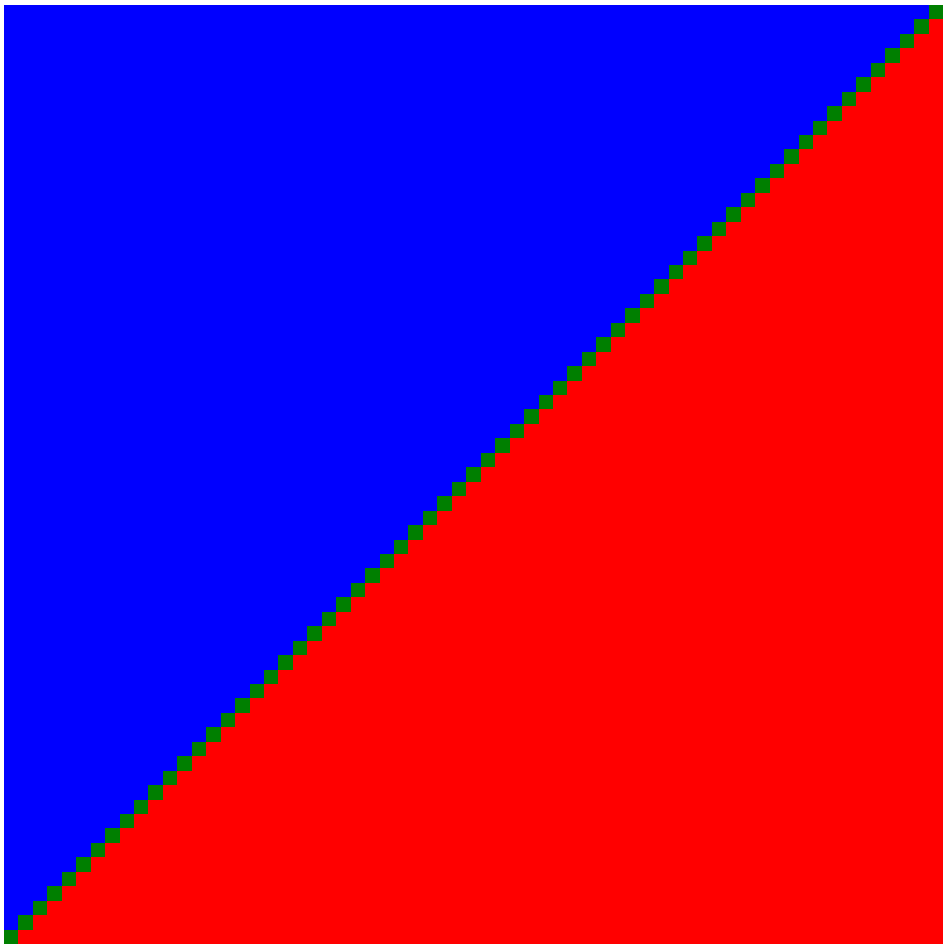
Pour $q = (8.1, 8.1)$ et $r = (12.1, 12.1)$ et p autour de (1,5; 1,5), le signe du déterminant devrait être :



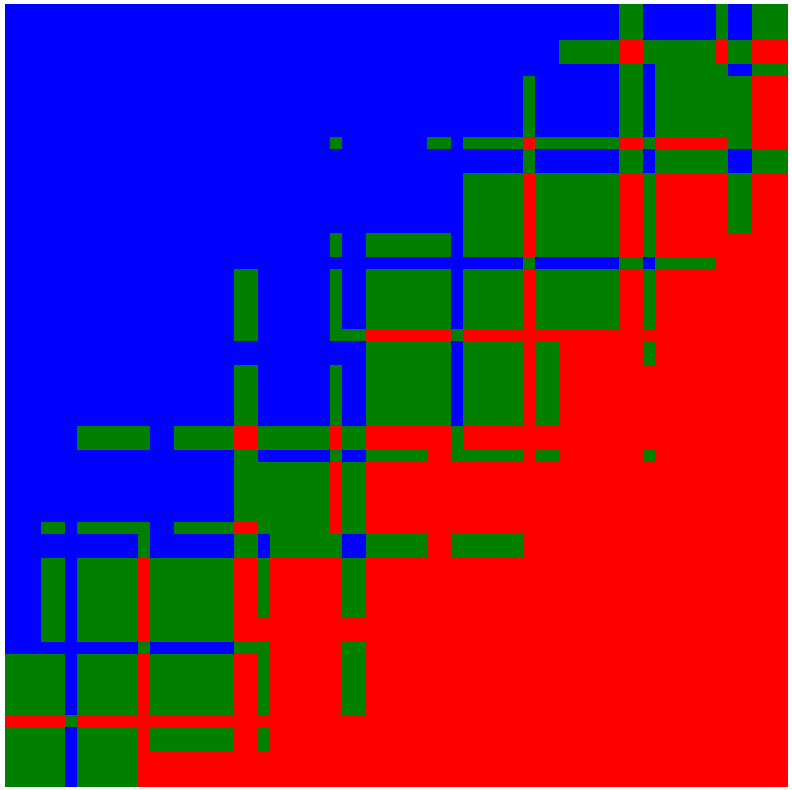
aligné

orienté

orienté



Orientation de 3 points - calculs flottants simple précision



Plan

Nombre à virgule flottante

Ce n'est qu'une suite de bits

11100011010010011110000111000000

Nombre à virgule flottante

Ce n'est qu'une suite de bits

$$11100011010010011110000111000000$$

à laquelle on donne un sens selon des tailles données pour s (signe), e (exposant) et f (fraction)

1	11000110	10010011110000111000000
1	11000110	10010011110000111000000
s	e	f

Nombre à virgule flottante

et une valeur réelle

$$\underset{s}{\boxed{1}} \qquad \underset{e}{\boxed{11000110}} \qquad \underset{f}{\boxed{10010011110000111000000}}$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$(-1)^s \times 2^{e-B} \times 1 \bullet f$$

$$(-1)^1 \times 2^{198-127} \times 1.10010011110000111000000_2$$

$$-2^{54} \times 206727 \approx -3{,}7 \times 10^{21}$$

Nombre à virgule flottante

et une valeur réelle

$$\underset{s}{\boxed{1}} \qquad \underset{e}{\boxed{11000110}} \qquad \underset{f}{\boxed{10010011110000111000000}}$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$(-1)^s \times 2^{e-B} \times 1 \bullet f$$

$$(-1)^1 \times 2^{198-127} \times 1.10010011110000111000000_2$$

$$-2^{54} \times 206727 \approx -3{,}7 \times 10^{21}$$

sauf valeurs spéciales de e : 0 et le maximum $e_{\max}$

Types de nombre à virgule flottante selon la norme IEEE-754

- si $0 < e < e_{\max}$, nombre normal de valeur $(-1)^s \cdot 1.f \cdot 2^{e-B}$
- si $e = 0$, nombre dénormalisé de valeur $(-1)^s \cdot 0.f \cdot 2^{1-B}$
  $\Rightarrow$ on a $+0$ et -0
- si $e = e_{\max}$ et $f = 0$, deux valeurs $+\infty$ et $-\infty$
- si $e = e_{\max}$ et $f \neq 0$, NaN (*Not-a-Number*)

Quelques définitions

précision p : nombre de chiffre de la mantisse.
$\Rightarrow$ double précision IEEE-754 (64 bits) : $p = 53$
$\Rightarrow$ simple précision IEEE-754 (32 bits) : $p = 24$.

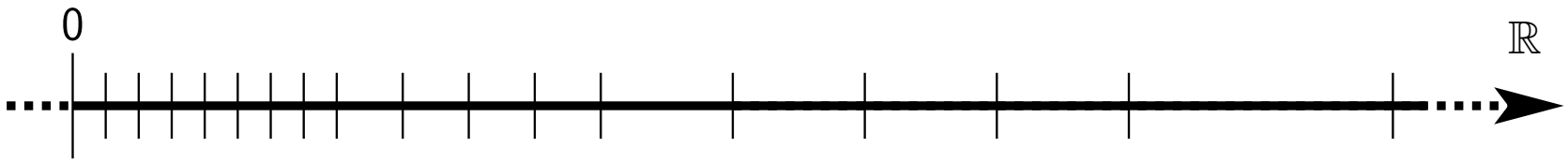
Quelques définitions

précision p : nombre de chiffre de la mantisse.
$\Rightarrow$ double précision IEEE-754 (64 bits) : $p = 53$
$\Rightarrow$ simple précision IEEE-754 (32 bits) : $p = 24$.

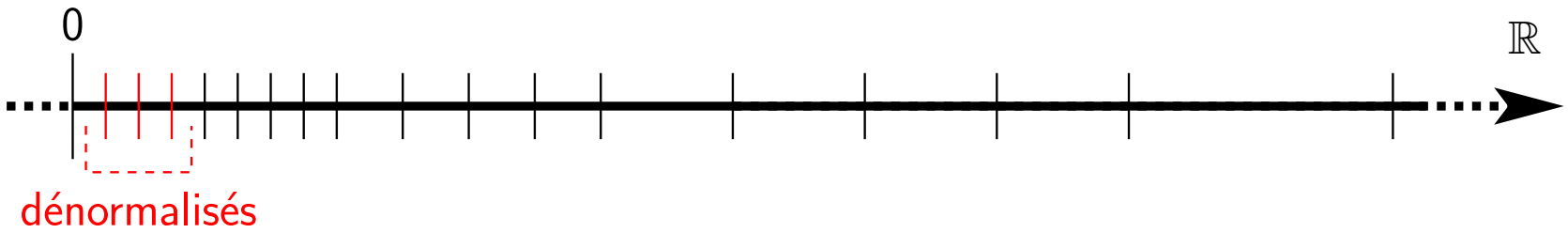
On appelle ulp la valeur du dernier bit de la mantisse d'un flottant.
Ainsi en double précision, on a $\text{ulp}(1) = 2^{-52}$ et
$\text{ulp}\left(2^{-1074}\right) = 2^{-1074}$.

Lorsque l'on obtient un dénormalisé, on parle d'underflow.

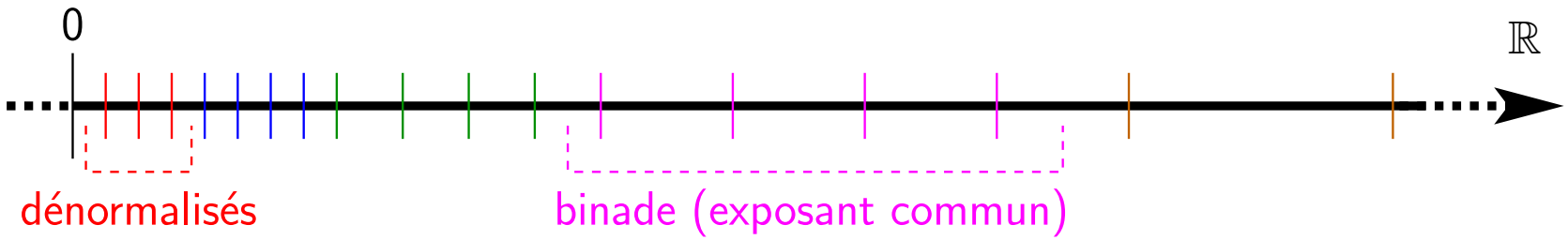
Répartition des flottants



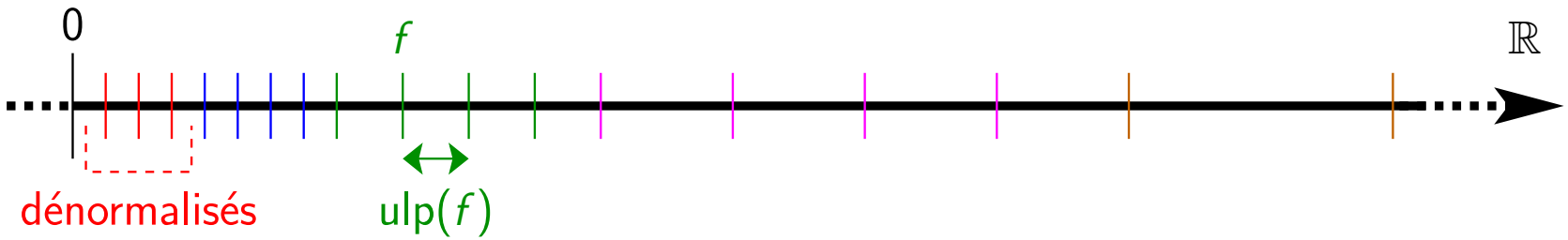
Répartition des flottants



Répartition des flottants



Répartition des flottants



Calcul et arrondis

La norme IEEE-754 définit 5 modes d'arrondi :

- arrondi au plus proche pair, noté $\circ$. Il renvoie le flottant le plus proche. Au milieu, il choisit celui qui a la mantisse paire.
- arrondi vers $-\infty$
- arrondi vers $+\infty$
- arrondi vers zéro
- arrondi au plus proche avec choix loin de zéro. Au milieu, il choisit celui qui a la plus grande valeur absolue.

Calcul et arrondis

La norme IEEE-754 définit 5 modes d'arrondi :

- arrondi au plus proche pair, noté $\circ$. Il renvoie le flottant le plus proche. Au milieu, il choisit celui qui a la mantisse paire.
- arrondi vers $-\infty$
- arrondi vers $+\infty$
- arrondi vers zéro
- arrondi au plus proche avec choix loin de zéro. Au milieu, il choisit celui qui a la plus grande valeur absolue.

Chaque calcul élémentaire (addition, soustraction, multiplication, division, racine carrée) est parfait : on obtient le même résultat que si l'on avait calculé avec une précision infinie et arrondi ensuite au format choisi.

Vers l'infini et au-delà !

On peut calculer avec les infinis et les zéros de façon intuitive :
Ainsi, $1 + \infty = +\infty$ et $-3 * +\infty = -\infty$

Vers l'infini et au-delà !

On peut calculer avec les infinis et les zéros de façon intuitive :
Ainsi, $1 + \infty = +\infty$ et $-3 * +\infty = -\infty$

Mais

$$\left(10^{100}\right)^2 \hookrightarrow +\infty$$

Vers l'infini et au-delà !

On peut calculer avec les infinis et les zéros de façon intuitive :
Ainsi, $1 + \infty = +\infty$ et $-3 * +\infty = -\infty$

Mais

$$\left(10^{100}\right)^2 \hookrightarrow +\infty$$
$$\frac{1}{\left(10^{100}\right)^2} \hookrightarrow 0$$

Vers l'infini et au-delà !

On peut calculer avec les infinis et les zéros de façon intuitive :
Ainsi, $1 + \infty = +\infty$ et $-3 * +\infty = -\infty$

Mais

$$\left(10^{100}\right)^2 \hookrightarrow +\infty$$

$$\frac{1}{\left(10^{100}\right)^2} \hookrightarrow 0$$

$$\frac{1}{\frac{1}{\left(10^{100}\right)^2}}$$

$\hookrightarrow$ BOUM

Vers l'infini et au-delà !

On peut calculer avec les infinis et les zéros de façon intuitive :
Ainsi, $1 + \infty = +\infty$ et $-3 * +\infty = -\infty$

Mais

$$\left(10^{100}\right)^2 \hookrightarrow +\infty$$

$$\frac{1}{\left(10^{100}\right)^2} \hookrightarrow 0$$

$$\frac{1}{\frac{1}{\left(10^{100}\right)^2}}$$

$\hookrightarrow$ BOUM

(en fait $\hookrightarrow +\infty$ mais le programme s'arrête sur "division by zero").

NaN !

En fait, le processeur me renvoie toujours un résultat, même quand le calcul demandé n'a aucun sens.

Si je calcule $\sqrt{-1}$ ou $+\infty - \infty$, j'obtiens NaN (Not-a-Number).

NaN !

NaN apparaît aux endroits les plus incongrus :

NaN !

NaN apparaît aux endroits les plus incongrus :

NaN !

NaN apparaît aux endroits les plus incongrus :

NaN !

NaN apparaît aux endroits les plus incongrus :

NaN !

NaN apparaît aux endroits les plus incongrus :

Plan

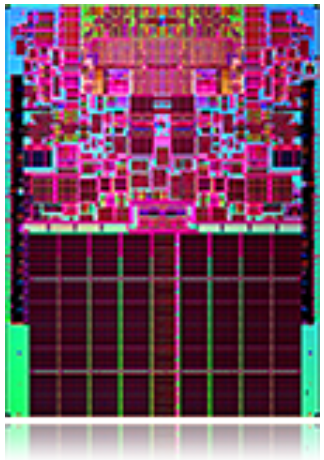
Plan

L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

Il peut y avoir :

- un bug matériel (le processeur est faux).

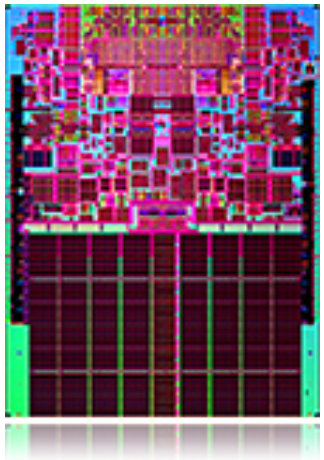


L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

Il peut y avoir :

- un bug matériel (le processeur est faux).

  Ça arrive, mais c'est très rare.



L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

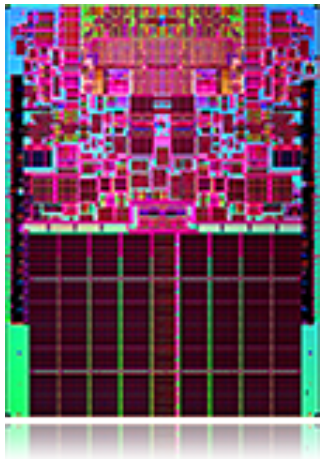
Il peut y avoir :

- un bug matériel (le processeur est faux).

  Ça arrive, mais c'est très rare.

  Bug du Pentium : certaines divisions fausses à partir du 5e chiffre

  475 millions de $.



L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

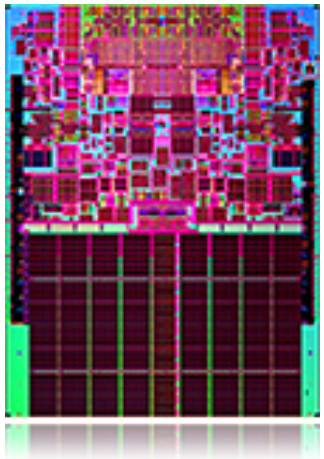
Il peut y avoir :

- un bug matériel (le processeur est faux).

  Ça arrive, mais c'est très rare.

  Bug du Pentium : certaines divisions fausses à partir du 5e chiffre

  475 millions de $.



- un bug logiciel.

L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

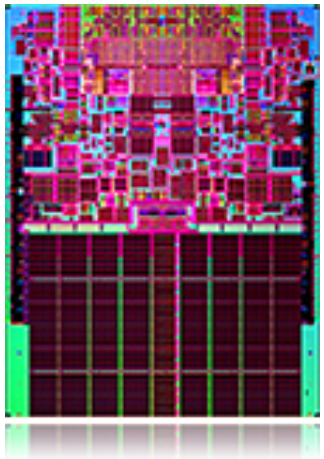
Il peut y avoir :

- un bug matériel (le processeur est faux).

  Ça arrive, mais c'est très rare.

  Bug du Pentium : certaines divisions fausses à partir du 5e chiffre

  475 millions de $.



- un bug logiciel.

  Là, par contre...

L'erreur est humaine, mais pour provoquer une vraie catastrophe, il faut un ordinateur.

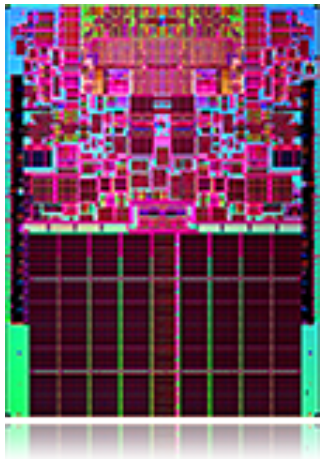
Il peut y avoir :

- un bug matériel (le processeur est faux).

  Ça arrive, mais c'est très rare.

  Bug du Pentium : certaines divisions fausses à partir du 5e chiffre

  475 millions de $.



- un bug logiciel.

  Là, par contre...

  Ariane 5 : explosion due à un dépassement de capacité

  500 millions de $.

Un exemple avec Microsoft Excel (28/09/07)

Prenons Microsoft Excel 2007.

Un exemple avec Microsoft Excel (28/09/07)

Prenons Microsoft Excel 2007.

- Dans la case A1, inscrivez 850
- Dans la case A2, inscrivez 77,1
- Dans la case A3, inscrivez la formule suivante : =PRODUIT(A1 : A2)

Un exemple avec Microsoft Excel (28/09/07)

Prenons Microsoft Excel 2007.

- Dans la case A1, inscrivez 850
- Dans la case A2, inscrivez 77,1
- Dans la case A3, inscrivez la formule suivante : =PRODUIT(A1 : A2)

Excel vous répond 100 000.

Un exemple avec Microsoft Excel (28/09/07)

Prenons Microsoft Excel 2007.

- Dans la case A1, inscrivez 850
- Dans la case A2, inscrivez 77,1
- Dans la case A3, inscrivez la formule suivante : =PRODUIT(A1 : A2)

Excel vous répond 100 000.
La réponse juste est 65 535.

Un exemple avec Microsoft Excel (28/09/07)

Prenons Microsoft Excel 2007.

- Dans la case A1, inscrivez 850
- Dans la case A2, inscrivez 77,1
- Dans la case A3, inscrivez la formule suivante : =PRODUIT(A1 : A2)

Excel vous répond 100 000.
La réponse juste est 65 535.

Mais ce n'est « que » un bug d'affichage.

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- $x := 0.1\,10^{92233720368547758 07}$.

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{92233720368547758 07}$.

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{92233720368547758 07}$.
- 9.5*x vaut

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{92233720368547758 07}$.
- 9.5*x vaut
  - Avec Maple 11 à 13, **Segmentation fault** !

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^9223372032859295016 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{9223372036854775807}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{9223372036854775807}$.
- 9.5*x vaut
  - Avec Maple 11 à 13, **Segmentation fault** !
  - Avec Maple 14 et 15, $0.10\,10^{-9223372036854775808}$!

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{92233720368547758 07}$.
- 9.5*x vaut
  - Avec Maple 11 à 13, **Segmentation fault** !
  - Avec Maple 14 et 15, $0.10\,10^{-92233720368547758 08}$!
  - Avec Maple 17, 0.10 !

Un exemple avec Maple (25/05/13) de Vincent Lefèvre

Prenons Maple sur une architecture 64 bits.

- Calculons x := 10.^3995480790*10.^92233720328592950 16 ;
- Calculons 10.*x puis 9.4*x puis 9.5*x.

- x := $0.1\,10^{92233720368547758 07}$.
- 10.*x vaut Float(infinity).
- 9.4*x vaut $0.9\,10^{92233720368547758 07}$.
- 9.5*x vaut
  - Avec Maple 11 à 13, **Segmentation fault** !
  - Avec Maple 14 et 15, $0.10\,10^{-92233720368547758 08}$!
  - Avec Maple 17, 0.10 !

Pour une machine 32 bits, on peut considérer 95.*10.^2147483645 ($\rightarrow$ out of memory, petit nombre).

Plan

Flocq

Flocq est une bibliothèque Coq

- développée par Guillaume Melquiond et Sylvie Boldo
- qui définit les nombres flottants, les arrondis et des théorèmes
- générique en terme de base, de format et d'arrondi
- disponible sur :

`http://flocq.gforge.inria.fr/`

Formalisation : Flocq

Définition (Format générique)

- Format flottant : $\phi : \mathbb{Z} \to \mathbb{Z}$.
- Arrondi : (ϕ, rnd) avec $\mathrm{rnd} : \mathbb{R} \to \mathbb{Z}$.
- Nombre flottant : $x \in \mathbb{R}$ tel que $x = \mathrm{rnd}(x \cdot 2^{-\phi(e_x)}) \times 2^{\phi(e_x)}$
  avec $e_x = \lfloor \log_2 |x| \rfloor + 1$.

Formalisation : Flocq

Définition (Format générique)

- Format flottant : $\phi : \mathbb{Z} \to \mathbb{Z}$.
- Arrondi : (ϕ, rnd) avec $\mathrm{rnd} : \mathbb{R} \to \mathbb{Z}$.
- Nombre flottant : $x \in \mathbb{R}$ tel que $x = \mathrm{rnd}(x \cdot 2^{-\phi(e_x)}) \times 2^{\phi(e_x)}$ avec $e_x = \lfloor \log_2 |x| \rfloor + 1$.

Exemple (Arrondis)

- Arrondi vers $-\infty$: $\mathrm{rnd} = \lfloor \cdot \rfloor$.
- Arrondi vers $+\infty$: $\mathrm{rnd} = \lceil \cdot \rceil$.
- ...

Formats usuels en Flocq

Définition (FIX)

Virgule fixe avec exposant $e_{\min}$: $\phi(e) = e_{\min}$.

Formats usuels en Flocq

Définition (FIX)

Virgule fixe avec exposant $e_{\min}$: $\phi(e) = e_{\min}$.

Définition (FL*)

Virgule flottante en précision p :

- non bornée (FLX) : $\phi(e) = e - p$,

Formats usuels en Flocq

Définition (FIX)

Virgule fixe avec exposant $e_{\min}$: $\phi(e) = e_{\min}$.

Définition (FL*)

Virgule flottante en précision p :

- non bornée (FLX) : $\phi(e) = e - p$,
- bornée avec dénormalisés (FLT) : $\phi(e) = \max(e - p, e_{\min})$,

Formats usuels en Flocq

Définition (FIX)

Virgule fixe avec exposant $e_{\min}$: $\phi(e) = e_{\min}$.

Définition (FL*)

Virgule flottante en précision p :

- non bornée (FLX) : $\phi(e) = e - p$,
- bornée avec dénormalisés (FLT) : $\phi(e) = \max(e - p, e_{\min})$,
- bornée sans dénormalisés (FTZ) :

$$\phi(e) = \begin{cases} e - p & \text{si } e - p \geq e_{\min}, \\ e_{\min} + p - 1 & \text{sinon.} \end{cases}$$

Plan

Propriétés des modes d'arrondi

Lemme (Arrondi fidèle)

- $\triangledown(x) = \max\{y \in \mathbb{F} \mid y \leq x\}$,
- $\triangle(x) = \min\{y \in \mathbb{F} \mid y \geq x\}$,
- $\square(x) = \triangledown(x)$ *ou* $\square(x) = \triangle(x)$.

Propriétés des modes d'arrondi

Lemme (Arrondi fidèle)

- $\triangledown(x) = \max\{y \in \mathbb{F} \mid y \leq x\}$,
- $\triangle(x) = \min\{y \in \mathbb{F} \mid y \geq x\}$,
- $\square(x) = \triangledown(x)$ *ou* $\square(x) = \triangle(x)$.

Lemme (Idempotence)

$$\forall x \in \mathbb{F},\ \square(x) = x.$$

Propriétés des modes d'arrondi

Lemme (Arrondi fidèle)

- $\triangledown(x) = \max\{y \in \mathbb{F} \mid y \leq x\}$,
- $\triangle(x) = \min\{y \in \mathbb{F} \mid y \geq x\}$,
- $\square(x) = \triangledown(x)$ *ou* $\square(x) = \triangle(x)$.

Lemme (Idempotence)

$$\forall x \in \mathbb{F},\ \square(x) = x.$$

Lemme (Monotonie)

$$\forall x, y \in \mathbb{R},\ x \leq y \Rightarrow \square(x) \leq \square(y).$$

Bornes d'erreur

Lemme (Erreur d'arrondi en arrondi au plus près)

Pour tout $x \in \mathbb{R}$, il existe ε et δ tel que

$$\circ(x) = x \cdot (1 + \varepsilon) + \delta \quad \textit{et} \quad |\varepsilon| \leq 2^{-p} = u \quad \textit{et} \quad |\delta| \leq 2^{e_{\min}-1}.$$

Qui plus est, $\delta = 0$ ou $\varepsilon = 0$.

Addition dénormalisée

Lemme (Exactitude de l'addition dénormalisée)

$$\forall x, y \in \mathbb{F},\ |x + y| \leq 2^{e_{\min}+p} \Rightarrow x + y \in \mathbb{F}.$$

Addition dénormalisée

Lemme (Exactitude de l'addition dénormalisée)

$$\forall x, y \in \mathbb{F},\ |x + y| \leq 2^{e_{\min}+p} \Rightarrow x + y \in \mathbb{F}.$$

Corollaire (Erreur d'arrondi pour l'addition)

$$\forall x, y \in \mathbb{F},\ \exists \varepsilon,\ \circ(x + y) = (x + y) \cdot (1 + \varepsilon) \quad \textit{et} \quad |\varepsilon| \leq 2^{-p}.$$

Sterbenz

Théorème (Sterbenz)

Soient a et b dans $\mathbb{F}$ tels que

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

Alors le réel $a - b$ est représentable.
En particulier, il est calculé sans erreur par la soustraction flottante.

Sterbenz

Théorème (Sterbenz)

Soient a et b dans $\mathbb{F}$ tels que

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

Alors le réel $a - b$ est représentable.
En particulier, il est calculé sans erreur par la soustraction flottante.

Plan

Discriminant précis

Il est très difficile de calculer précisément $b^2 - ac$.

Discriminant précis

Il est très difficile de calculer précisément $b^2 - ac$.

Théorème (Kahan)

S'il n'y a ni overflow, ni underflow, il y a un algorithme qui calcule $b^2 - a \times c$ à 2 ulps près.

Discriminant précis – programme

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discriminant(double a, double b, double c) {
  double p,q,d,dp,dq;
  p=b*b;
  q=a*c;

  if (p+q <= 3*fabs(p-q))
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

Discriminant précis – programme

Underflow

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discriminant(double a, double b, double c) {
  double p,q,d,dp,dq;
  p=b*b;
  q=a*c;

  if (p+q <= 3*fabs(p-q))
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

Discriminant précis – programme

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discriminant(double a, double b, double c) {
  double p,q,d,dp,dq;
  p=b*b;
  q=a*c;

  if (p+q <= 3*fabs(p-q))
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

Overflow

Discriminant précis – programme

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discriminant(double a, double b, double c) {
  double p,q,d,dp,dq;
  p=b*b;
  q=a*c;

  if (p+q <= 3*fabs(p-q))
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

2 ulps

Discriminant précis – programme

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discriminant(double a, double b, double c) {
  double p,q,d,dp,dq;
  p=b*b;
  q=a*c;
  i
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

Appels de fonctions

⇒ pré-conditions à prouver

⇒ post-conditions garanties

$p + dp = b^2$ et $q + dq = ac$

Discriminant précis – programme

```
/*@ requires (b==0.   || 0x1.p-916 <= \abs(b*b)) &&
  @          (a*c==0. || 0x1.p-916 <= \abs(a*c)) &&
  @          \abs(b) <= 0x1.p510 &&
  @          \abs(a) <= 0x1.p995 && \abs(c) <= 0x1.p995 &&
  @          \abs(a*c) <= 0x1.p1021;
  @ ensures  \result ==0. || \abs(\result -(b*b-a*c)) <= 2.*ulp(\result);
  @ */
double discrimi                                          {
  double p,q,d;
  p=b*b;
  q=a*c;

  if (p+q <= 3*fabs(p-q))
    d=p-q;
  else {
    dp=Dekker(b,b,p);
    dq=Dekker(a,c,q);
    d=(p-q)+(dp-dq);
  }
  return d;
}
```

Dans la preuve initiale, test supposé correct

⇒ Preuve supplémentaire quand le test est incorrect

Plan

Parenthésage : le cas Fortran

Norme Fortran 77 (6.6.4)

Once the interpretation has been established in accordance with those rules, the processor may evaluate any mathematically equivalent expression, provided that the integrity of parentheses is not violated.

Parenthésage : le cas Fortran

Norme Fortran 77 (6.6.4)

Once the interpretation has been established in accordance with those rules, the processor may evaluate any mathematically equivalent expression, provided that the integrity of parentheses is not violated.

Exemple (Addition)

L'expression `a+b+c+d` peut être compilée en

- `((a+b)+c)+d` : interprétation naturelle,
- `(a+b)+(c+d)` : parallélisme,
- `a+(b+(c+d))` : pourquoi pas ?

Non-associativité des calculs flottants

Exemple

Évaluation de `delta + 1. + (-1.)` avec `delta` $= 2^{-p}$.

- $\circ(\circ(\delta + 1) - 1) = \circ(1 - 1) = 0$.
- $\circ(\delta + \circ(1 - 1)) = \circ(\delta + 0) = \delta$.

Précision intermédiaire : le cas C

Norme C99 (5.2.4.2.2 §8)

The values of operations with floating operands and values subject to the usual arithmetic conversions and of floating constants are evaluated to a format whose range and precision may be greater than required by the type.

Précision intermédiaire : le cas C

Norme C99 (5.2.4.2.2 §8)

The values of operations with floating operands and values subject to the usual arithmetic conversions and of floating constants are evaluated to a format whose range and precision may be greater than required by the type.

Exemple (Précision étendue)

```
double x = 1.0;
double y = 0x1p-53 + 0x1p-64;
double z = x + y;
```

Précision intermédiaire : le cas C

Norme C99 (5.2.4.2.2 §8)

The values of operations with floating operands and values subject to the usual arithmetic conversions and of floating constants are evaluated to a format whose range and precision may be greater than required by the type.

Exemple (Précision étendue)

```
double x = 1.0;
double y = 0x1p-53 + 0x1p-64;
double z = x + y;
```

peut produire les valeurs suivantes :

- $z = 1 + 2^{-52}$: addition en `binary64`,
- $z = 1$: addition en `binary80` puis stockage en `binary64`.

Précision intermédiaire : le cas C

Norme C99 (5.2.4.2.2 §8)

The values of operations with floating operands and values subject to the usual arithmetic conversions and of floating constants are evaluated to a format whose range and precision may be greater than required by the type.

Exemple (Précision étendue)

```
double x = 1.0;
double y = 0x1p-53 + 0x1p-64;
double z = x + y;
```

peut produire les valeurs suivantes :

- $z = 1 + 2^{-52}$: addition en `binary64`,
- $z = 1$: addition en `binary80` puis stockage en `binary64`. Précision supérieure $\not\Rightarrow$ résultat plus précis !

Optimisations

Celles qui sont valides :

$$x \otimes 2.0 \quad \rightarrow \quad x \oplus x$$
$$2.0 \otimes x \quad \rightarrow \quad x \oplus x$$

Optimisations

Celles qui sont valides :

$$\begin{aligned} x \otimes 2.0 &\rightarrow x \oplus x \\ 2.0 \otimes x &\rightarrow x \oplus x \end{aligned}$$

Celles qui sont plausibles, mais fausses :

$$\begin{aligned} x \oplus 0.0 &\not\rightarrow x \\ x \oplus (-0.0) &\not\rightarrow x \\ x \ominus 0.0 &\not\rightarrow x \\ x \ominus (-0.0) &\not\rightarrow x \\ -(-x) &\not\rightarrow x \\ (-x) \oplus y &\not\rightarrow y \ominus x \\ y \oplus (-x) &\not\rightarrow y \ominus x \\ y \ominus (-x) &\not\rightarrow y \oplus x \end{aligned}$$

Conclusion

Les calculs sur ordinateur,

Conclusion

Les calculs sur ordinateur,

- c'est rapide,

Conclusion

Les calculs sur ordinateur,

- c'est rapide,
- mais ce n'est pas exact.

Conclusion

Les calculs sur ordinateur,

- c'est rapide,
- mais ce n'est pas exact.

Il faut donc

Conclusion

Les calculs sur ordinateur,

- c'est rapide,
- mais ce n'est pas exact.

Il faut donc

- garder un œil critique sur les réponses fournies,

Conclusion

Les calculs sur ordinateur,

- c'est rapide,
- mais ce n'est pas exact.

Il faut donc

- garder un œil critique sur les réponses fournies,
- ou attendre que votre programme favori soit prouvé.

Dernier exemple. . .

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n,

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n,
- Pour récupérer mon argent, il y a 1€ de frais.

Il était une fois, dans une galaxie très très lointaine

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2{,}718\ 28\ldots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n,
- Pour récupérer mon argent, il y a 1€ de frais.

Dans 50 ans, pour ma retraite, combien d'argent aurai-je ?

Combien ?

Machine	Valeur

Combien ?

Machine	**Valeur**
HP-48S	$+2{,}903\ 83 \quad 10^{52}$ €

$\Rightarrow$ Super !

Combien ?

Machine	**Valeur**	
HP-48S	$+2{,}903\ 83$	10^{52} € ⇒ Super !
C (format `double`)	$-4{,}396\ 80$	10^{48} € ⇒ Oups !

Combien ?

Machine	**Valeur**	
HP-48S	$+2{,}903\ 83$	10^{52} € ⇒ Super !
C (format `double`)	$-4{,}396\ 80$	10^{48} € ⇒ Oups !
C (format `float`)	$-\infty$	⇒ Oups ! ! ! !

Combien ?

Machine	**Valeur**		
HP-48S	$+2{,}903\ 83$	10^{52} €	$\Rightarrow$ Super !
C (format `double`)	$-4{,}396\ 80$	10^{48} €	$\Rightarrow$ Oups !
C (format `float`)	$-\infty$		$\Rightarrow$ Oups ! ! ! !
Maple (10 chiffres)	$-1{,}396\ 14$	10^{55} €	
Maple (20 chiffres)	$+1{,}207\ 82$	10^{45} €	$\Rightarrow$ Hein ?

Combien ?

Machine	**Valeur**	
HP-48S	$+2{,}903\ 83$	10^{52} € ⇒ Super !
C (format `double`)	$-4{,}396\ 80$	10^{48} € ⇒ Oups !
C (format `float`)	$-\infty$	⇒ Oups ! ! ! !
Maple (10 chiffres)	$-1{,}396\ 14$	10^{55} €
Maple (20 chiffres)	$+1{,}207\ 82$	10^{45} € ⇒ Hein ?
Valeur exacte	$\approx 0{,}02$€	