De l'implantation de fonctions correctement arrondies aux bibliothèques LIBM IEEE 754-2008 *clef en main*

Présentation pour le groupe de travail PEQUAN

Christoph Quirin Lauter

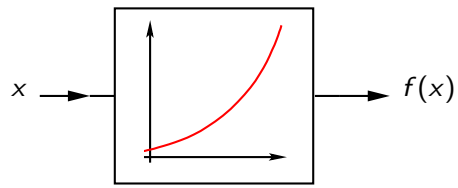
LIP6 - PEQUAN - UPMC Paris VI - CNRS

Paris, 3 février 2011

Généralités sur l'arrondi correct et notations

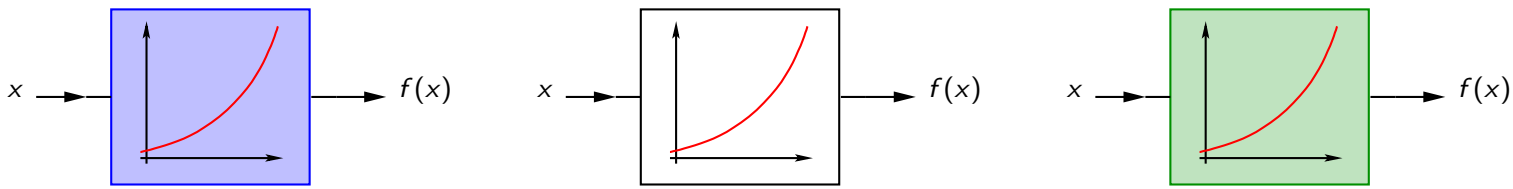
Fonctions mathématiques sur ordinateur

- Réalisation de fonctions mathématiques sur ordinateur
  - Programmes calculant la valeur de $f(x)$ pour un x donné



Fonctions mathématiques sur ordinateur

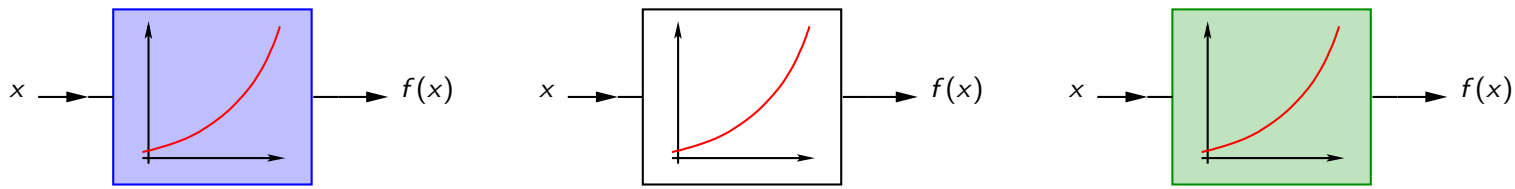
- Réalisation de fonctions mathématiques sur ordinateur
  - Programmes calculant la valeur de $f(x)$ pour un x donné



  - Différentes réalisations d'une fonction f

Fonctions mathématiques sur ordinateur

- Réalisation de fonctions mathématiques sur ordinateur
  - Programmes calculant la valeur de $f(x)$ pour un x donné

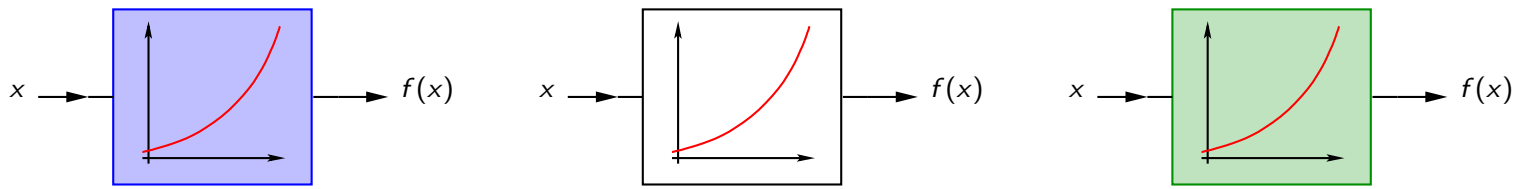


  - Différentes réalisations d'une fonction f

- Les résultats du calcul doivent être les mêmes partout.
  - Des résultats déterministes
  - Des ordinateurs qui se contrôlent l'un l'autre
  - Le montant à payer et celui à percevoir qui sont les mêmes

Fonctions mathématiques sur ordinateur

- Réalisation de fonctions mathématiques sur ordinateur
  - Programmes calculant la valeur de $f(x)$ pour un x donné



  - Différentes réalisations d'une fonction f

- Les résultats du calcul doivent être les mêmes partout.
  - Des résultats déterministes
  - Des ordinateurs qui se contrôlent l'un l'autre
  - Le montant à payer et celui à percevoir qui sont les mêmes

- $\Rightarrow$ L'arrondi correct des fonctions mathématiques

L'arrondi

- Fonctions mathématiques réelles transcendantes :
  Nombre infini de chiffres dans le développement de $f(x)$:

  p.ex. $\exp(1) = 2.71828182845904523536028747 1\ldots$

L'arrondi

- Fonctions mathématiques réelles transcendantes :
  Nombre infini de chiffres dans le développement de $f(x)$:

$$\text{p.ex. } \exp(1) = 2.71828182845904523536028747 1\ldots$$

- Mémoires finies dans nos ordinateurs
  Solution : n'écrire que le début du développement

$$\text{p.ex. } \exp(1) \approx 2.7182818284590459$$

L'arrondi

- Fonctions mathématiques réelles transcendantes :
  Nombre infini de chiffres dans le développement de $f(x)$:

$$\text{p.ex. } \exp(1) = 2.71828182845904523536028747 1\ldots$$

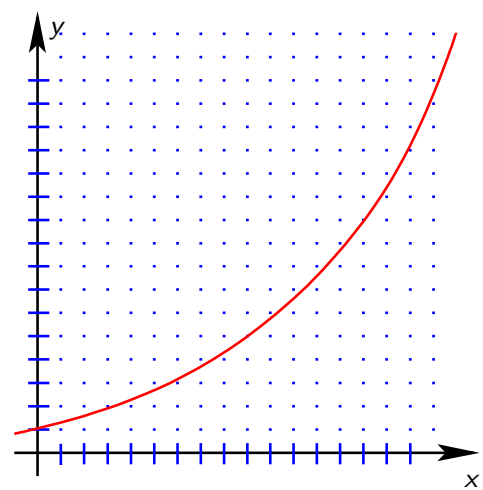
- Mémoires finies dans nos ordinateurs
  Solution : n'écrire que le début du développement

$$\text{p.ex. } \exp(1) \approx 2.718281828459045 9$$

- Il s'agit là d'un arrondi

$$\circ : \mathbb{R} \to \mathbb{F}$$

où $\mathbb{F}$ est une représentation des réels



La virgule flottante

- La virgule flottante est une représentation des réels

La virgule flottante

- La virgule flottante est une représentation des réels
- Il s'agit d'un système semi-logarithmique
  - Un exposant E donne l'ordre de grandeur.
  - Un signifiant m indique les chiffres début du développement.

La virgule flottante

- La virgule flottante est une représentation des réels
- Il s'agit d'un système semi-logarithmique
  - Un exposant E donne l'ordre de grandeur.
  - Un signifiant m indique les chiffres début du développement.
- Exemples :

$$\begin{aligned} 105.415 &= 10^3 \cdot 1.05415 &&= 10^E \cdot m \\ 0.000000124565 &= 10^{-7} \cdot 1.24565 &&= 10^E \cdot m \\ 62 &= 2^5 \cdot 1.11110_2 &&= 2^E \cdot m \end{aligned}$$

La virgule flottante

- La virgule flottante est une représentation des réels
- Il s'agit d'un système semi-logarithmique
  - Un exposant E donne l'ordre de grandeur.
  - Un signifiant m indique les chiffres début du développement.
- Exemples :

$$\begin{aligned} 105.415 &= 10^3 \cdot 1.05415 &&= 10^E \cdot m \\ 0.000000124565 &= 10^{-7} \cdot 1.24565 &&= 10^E \cdot m \\ 62 &= 2^5 \cdot 1.11110_2 &&= 2^E \cdot m \end{aligned}$$

- La représentation se fait dans une base β ; ici $\beta = 2$.
- Le nombre de chiffres indiqués est la précision k.

La virgule flottante

- La virgule flottante est une représentation des réels
- Il s'agit d'un système semi-logarithmique
  - Un exposant E donne l'ordre de grandeur.
  - Un signifiant m indique les chiffres début du développement.
- Exemples :

$$\begin{aligned} 105.415 &= 10^3 \cdot 1.05415 &&= 10^E \cdot m \\ 0.000000124565 &= 10^{-7} \cdot 1.24565 &&= 10^E \cdot m \\ 62 &= 2^5 \cdot 1.11110_2 &&= 2^E \cdot m \end{aligned}$$

- La représentation se fait dans une base β ; ici $\beta = 2$.
- Le nombre de chiffres indiqués est la précision k.
- L'ensemble des nombres flottants en précision k se note $\mathbb{F}_k$.

La norme IEEE 754

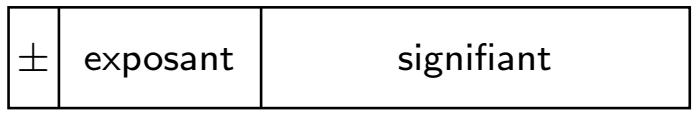
- Calcul en virgule flottante régi par la norme IEEE 754-2008
- Plusieurs formats $\mathbb{F}_k$ spécifiés

Précision double $\mathbb{F}_{53}$

La norme IEEE 754

- Calcul en virgule flottante régi par la norme IEEE 754-2008
- Plusieurs formats $\mathbb{F}_k$ spécifiés

Précision double $\mathbb{F}_{53}$



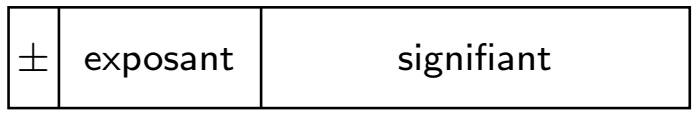
- Un arrondi IEEE 754 $\circ_k$ est une application $\mathbb{R} \to \mathbb{F}_k$:

$$x \mapsto \circ_k(x)$$

La norme IEEE 754

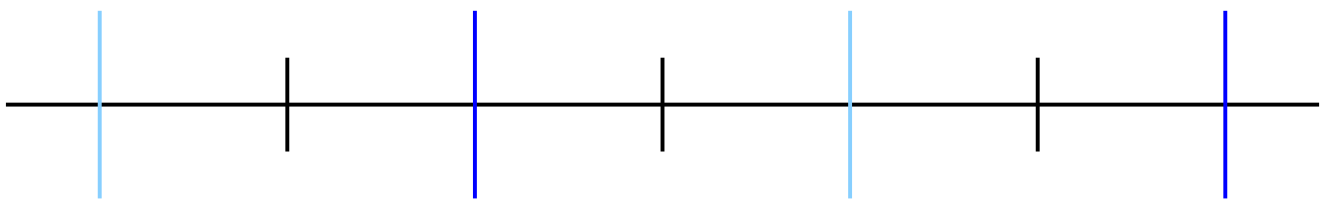
- Calcul en virgule flottante régi par la norme IEEE 754-2008
- Plusieurs formats $\mathbb{F}_k$ spécifiés

Précision double $\mathbb{F}_{53}$



- Un arrondi IEEE 754 $\circ_k$ est une application $\mathbb{R} \to \mathbb{F}_k$:

$$x \mapsto \circ_k(x)$$



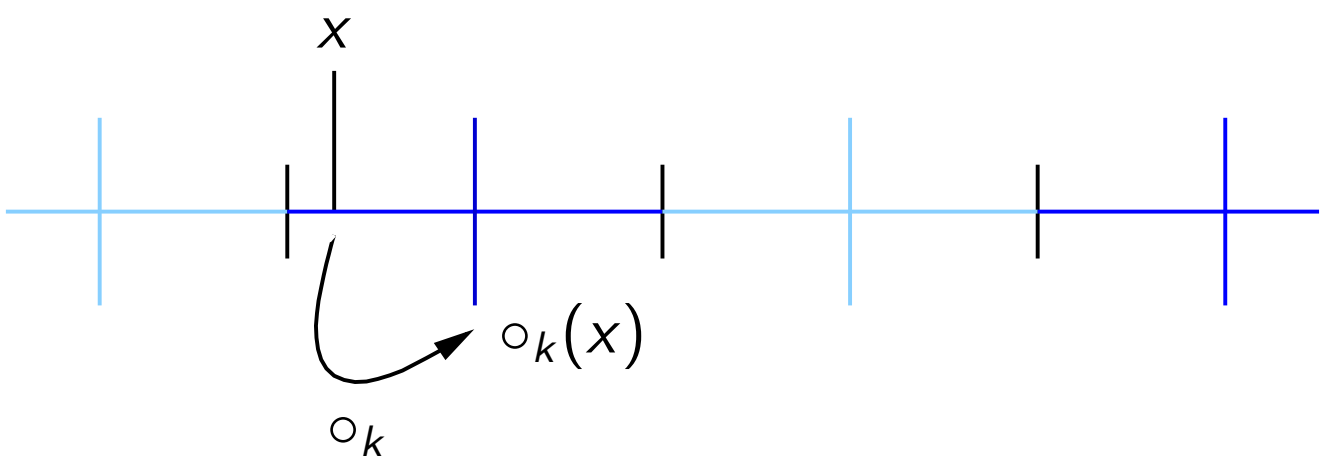
La norme IEEE 754

- Calcul en virgule flottante régi par la norme IEEE 754-2008
- Plusieurs formats $\mathbb{F}_k$ spécifiés

Précision double $\mathbb{F}_{53}$

- Un arrondi IEEE 754 $\circ_k$ est une application $\mathbb{R} \to \mathbb{F}_k$:

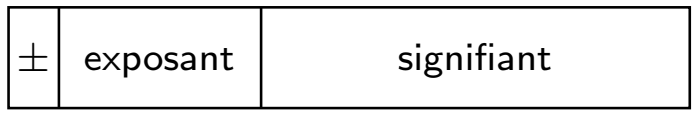
$$x \mapsto \circ_k(x)$$



La norme IEEE 754

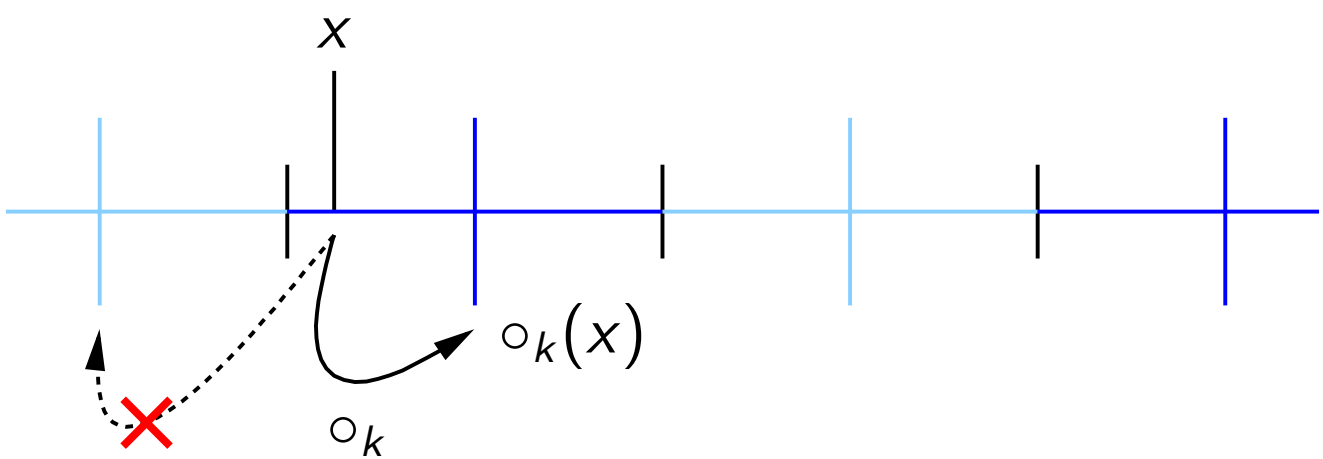
- Calcul en virgule flottante régi par la norme IEEE 754-2008
- Plusieurs formats $\mathbb{F}_k$ spécifiés

Précision double $\mathbb{F}_{53}$

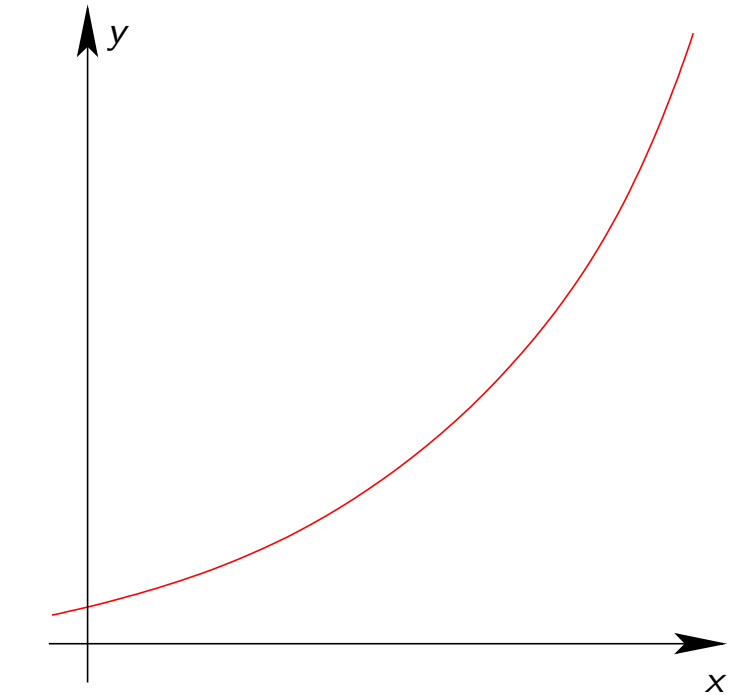


- Un arrondi IEEE 754 $\circ_k$ est une application $\mathbb{R} \to \mathbb{F}_k$:

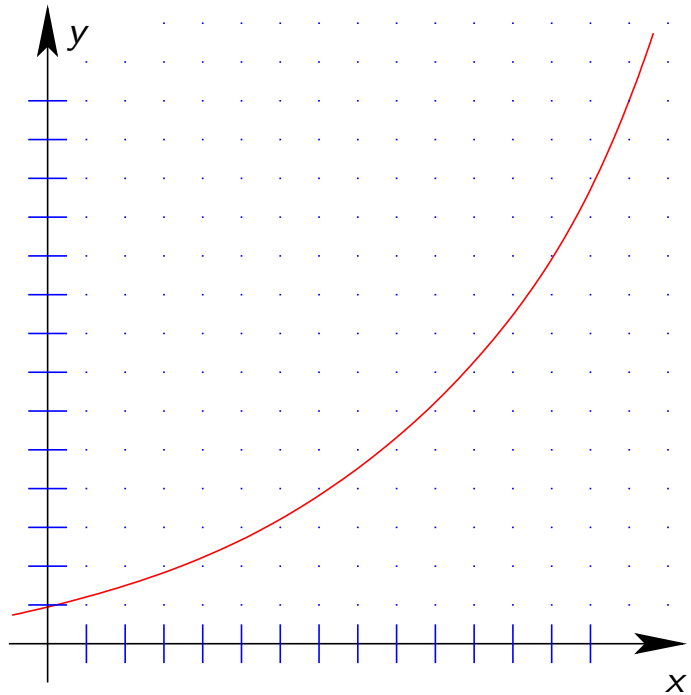
$$x \mapsto \circ_k(x)$$



L'arrondi correct d'une fonction f

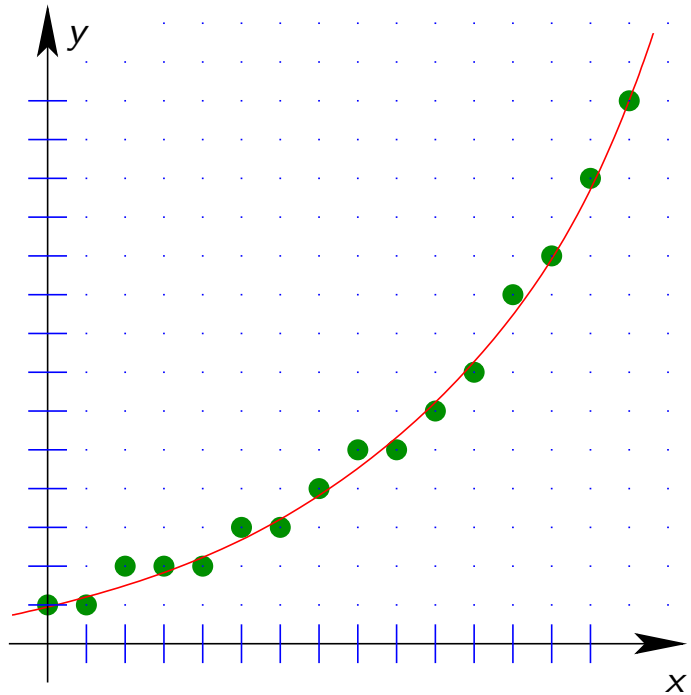


L'arrondi correct d'une fonction f



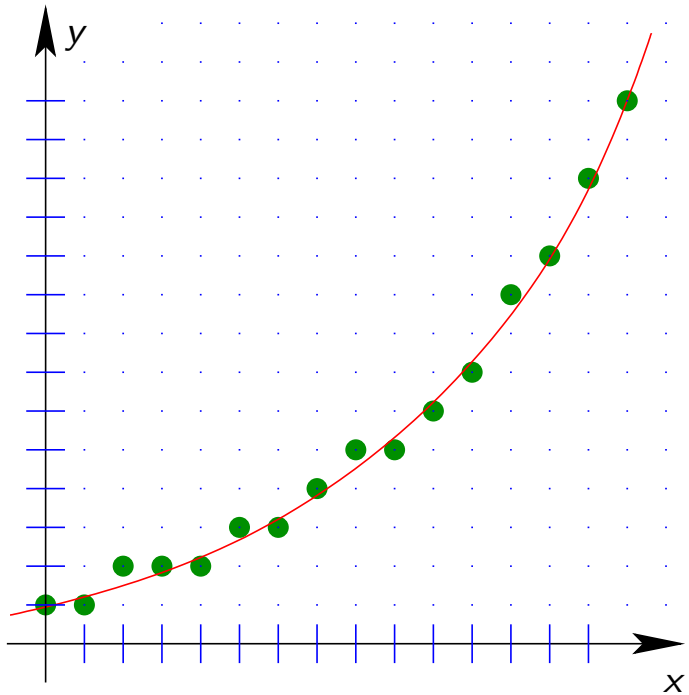
$$\forall x \in \mathbb{F}_k, \qquad x \xmapsto{f} f(x) \xmapsto{\circ_k}$$

L'arrondi correct d'une fonction f



$$\forall x \in \mathbb{F}_k, \qquad x \overset{f}{\longmapsto} f(x) \overset{\circ_k}{\longmapsto} \circ_k(f(x))$$

L'arrondi correct d'une fonction f



$$\forall x \in \mathbb{F}_k, \qquad x \overset{f}{\longmapsto} f(x) \overset{\circ_k}{\longmapsto} \circ_k(f(x))$$

Définition

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction. Soit $\circ_k : \mathbb{R} \to \mathbb{F}_k$ un arrondi. La fonction $F : \mathbb{F}_k^n \to \mathbb{F}_k$ est l'arrondi correct de f ssi

$$\forall x \in \mathbb{F}_k^n, \qquad F(x) = \circ_k(f(x))$$

Voir assez net à l'endroit décisif

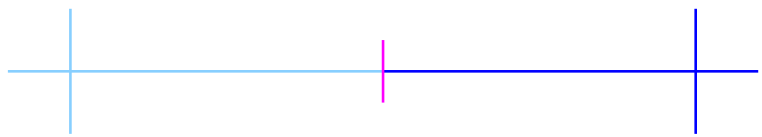
Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

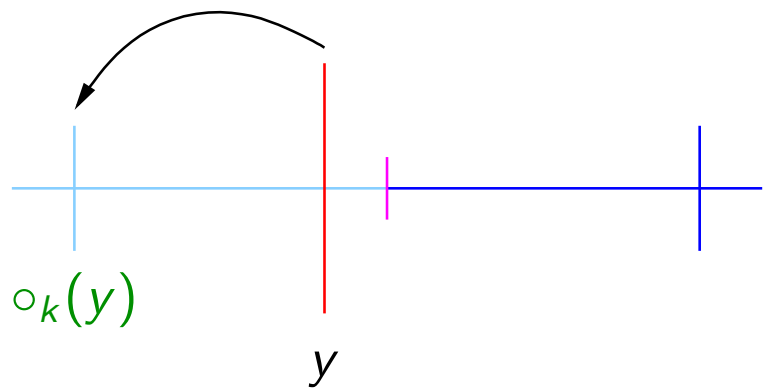


- La valeur $y = f(x)$ ne peut être calculée exactement.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

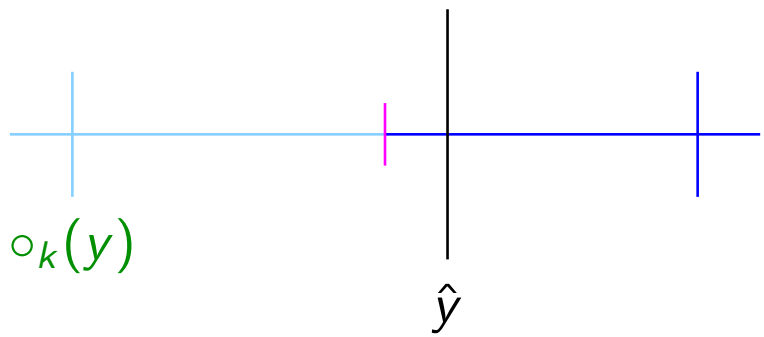


- La valeur $y = f(x)$ ne peut être calculée exactement.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

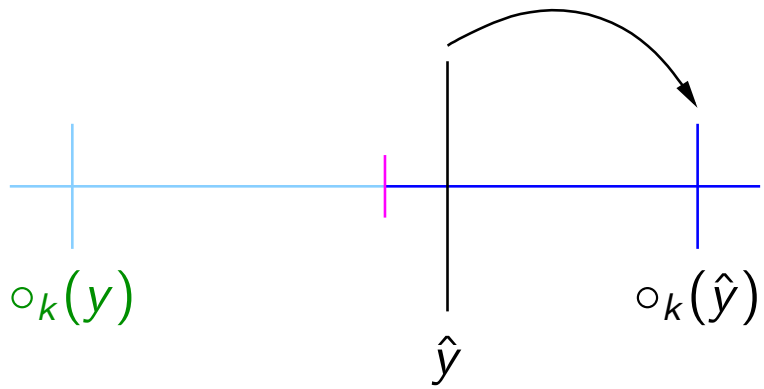


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

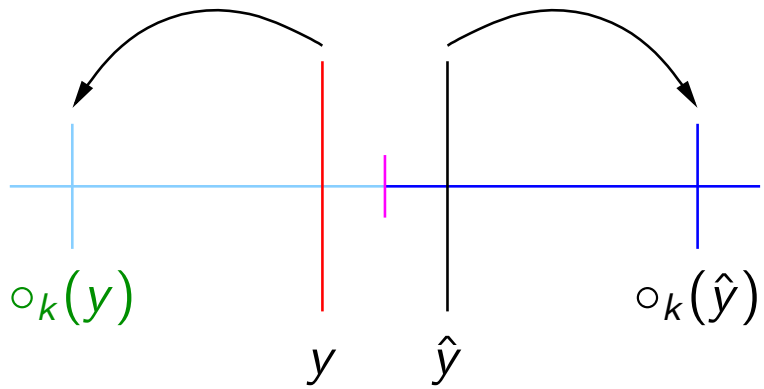


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

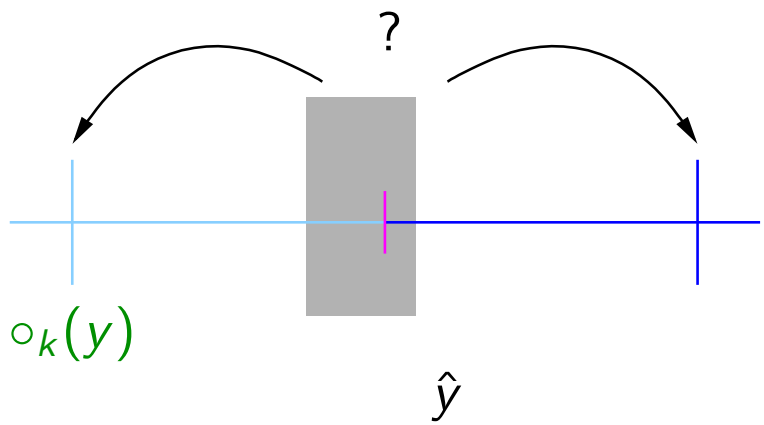


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

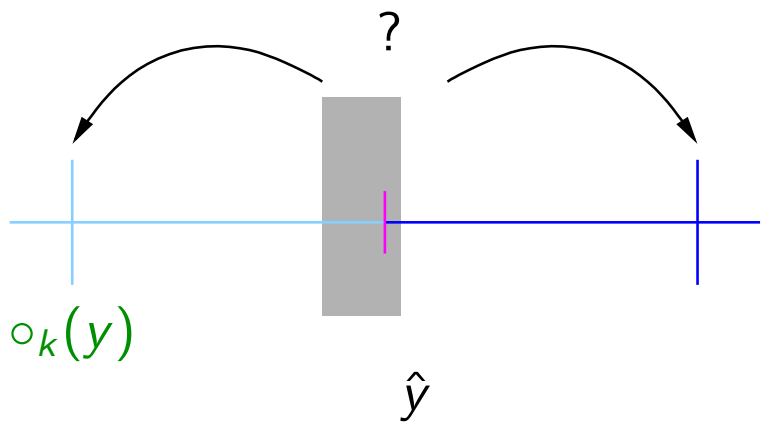


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

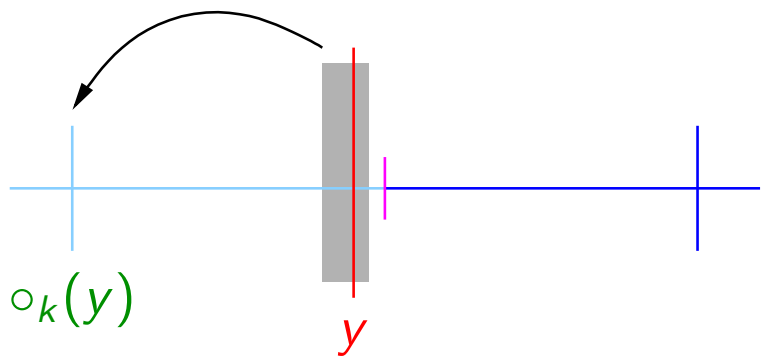


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

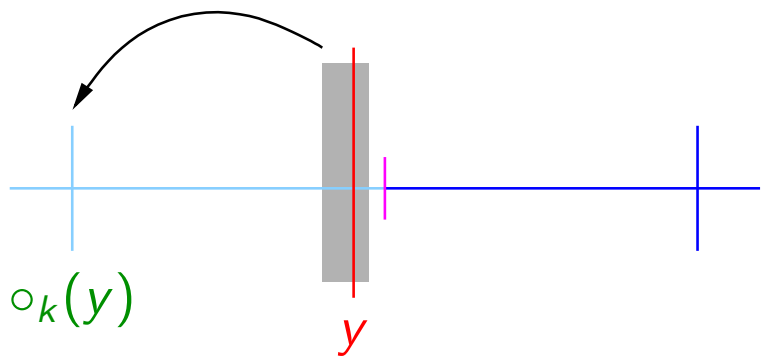


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.

Voir assez net à l'endroit décisif

Arrondi correct $\circ_k(f(x))$:

- Combinaison de deux phénomènes
  - La valeur de $f(x)$, transcendante, ne peut être qu'approchée.
  - L'arrondi $\circ_k$ change brusquement aux frontières d'arrondi.

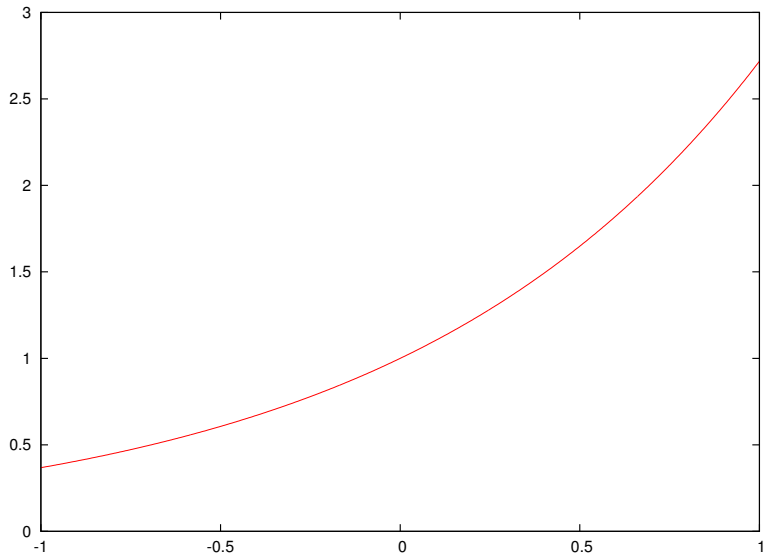


- La valeur $y = f(x)$ ne peut être calculée exactement.
- Une approximation $\hat{y} = f(x) \cdot (1 + \varepsilon)$ peut être calculée.
- La précision $\overline{\varepsilon}$ nécessaire au pire cas est inconnue.
- C'est le dilemme du fabricant de tables.

L'implantation d'une fonction

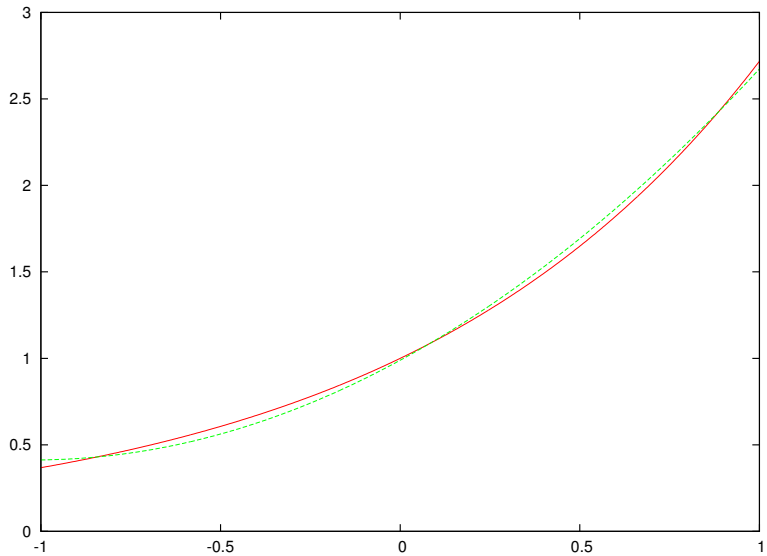
Approximation polynomiale

- Comment donc implanter p.ex. $f = \exp$ ...
- ... sur un domaine restreint d'abord, p.ex. $I = [-1; 1]$?



Approximation polynomiale

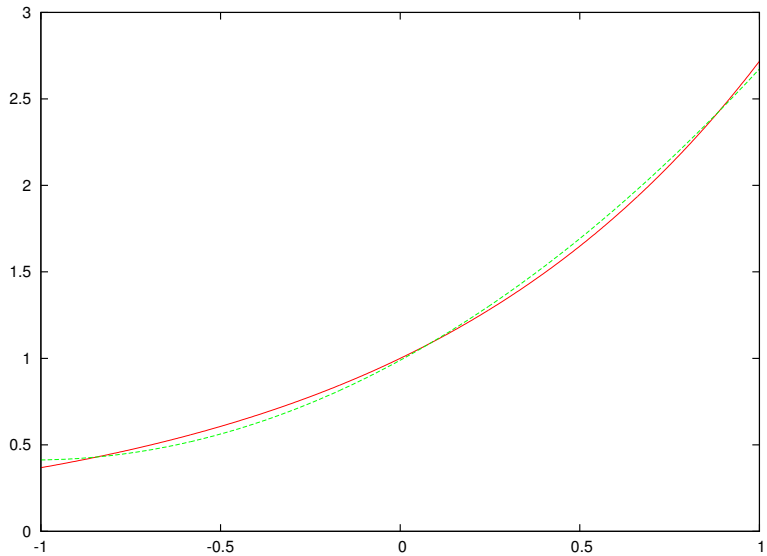
- Comment donc implanter p.ex. $f = \exp$ ...
- ... sur un domaine restreint d'abord, p.ex. $I = [-1; 1]$?



- Solution : remplacer f par un polynôme d'approximation p

Approximation polynomiale

- Comment donc implanter p.ex. $f = \exp$ ...
- ... sur un domaine restreint d'abord, p.ex. $I = [-1; 1]$?



- Solution : remplacer f par un polynôme d'approximation p
- Types de polynômes :
  - Développement de Taylor
  - Polynômes minimisant l'erreur maximale sur le domaine
  - Polynômes quasi-minimax à coefficients flottants (Chevillard)

Pourquoi des polynômes ?

- Pourquoi des polynômes et non d'autres approximations ?
  - Pourquoi pas de fractions continues (tronquées) ?
  - Pourquoi pas d'approximation avec d'autres fonctions de base ?
  - Algorithmes CORDIC etc ?

Pourquoi des polynômes ?

- Pourquoi des polynômes et non d'autres approximations ?
  - Pourquoi pas de fractions continues (tronquées) ?
  - Pourquoi pas d'approximation avec d'autres fonctions de base ?
  - Algorithmes CORDIC etc ?

- Une réponse (presque) purement technologique
  - Matériel très rapide pour $+$, $\times$ et FMA
  - Performance bien moindre pour la division (19 cycles sur i5/i7)
  - Possibilité de calculer l'erreur pour $+$ et $\times$ (et /)...
    ... mais pas pour sin et log

Pourquoi des polynômes ?

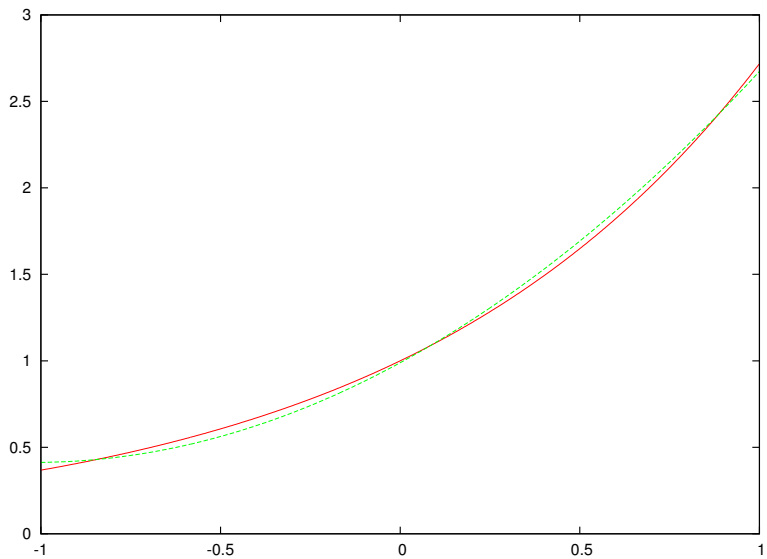
- Pourquoi des polynômes et non d'autres approximations ?
  - Pourquoi pas de fractions continues (tronquées) ?
  - Pourquoi pas d'approximation avec d'autres fonctions de base ?
  - Algorithmes CORDIC etc ?
- Une réponse (presque) purement technologique
  - Matériel très rapide pour $+$, $\times$ et FMA
  - Performance bien moindre pour la division (19 cycles sur i5/i7)
  - Possibilité de calculer l'erreur pour $+$ et $\times$ (et $/$)...
    ... mais pas pour sin et log
- Une réponse pas catégorique
  - Certaines fonctions s'approchent mal par des polynômes : asin
  - CRLibm, c'est un logiciel ; autre chose sur FPGA ou ASIC
  - La chaîne d'outil n'est pas aussi riche pour d'autres approches

Nécessité de la réduction d'argument

- Nécessité d'implanter f sur tout le domaine des flottants
- Pour la double, $f = \exp$ est définie sur $I = [-744.5; 709]$

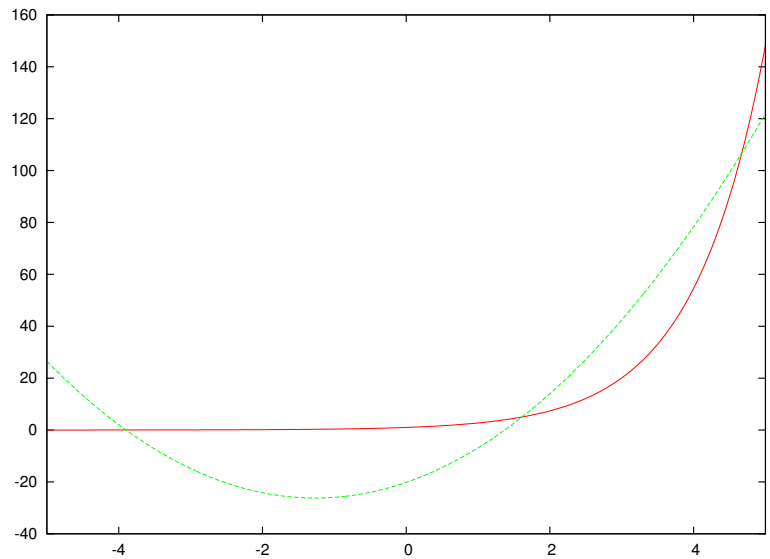
Nécessité de la réduction d'argument

- Nécessité d'implanter f sur tout le domaine des flottants
- Pour la double, $f = \exp$ est définie sur $I = [-744.5; 709]$
- L'approximation polynômiale seule ne suffit pas :



Nécessité de la réduction d'argument

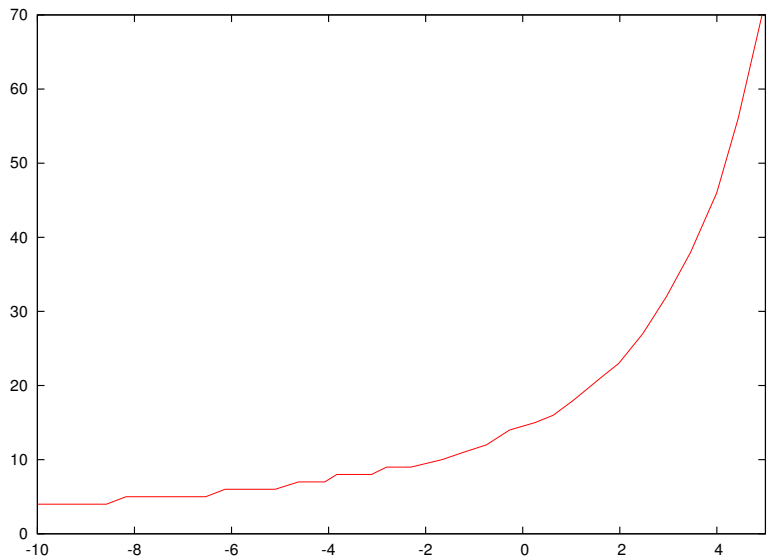
- Nécessité d'implanter f sur tout le domaine des flottants
- Pour la double, $f = \exp$ est définie sur $I = [-744.5; 709]$
- L'approximation polynômiale seule ne suffit pas :



- Quand le domaine est large, l'erreur explose pour un degré

Nécessité de la réduction d'argument

- Nécessité d'implanter f sur tout le domaine des flottants
- Pour la double, $f = \exp$ est définie sur $I = [-744.5; 709]$
- L'approximation polynômiale seule ne suffit pas :



- Quand le domaine est large, l'erreur explose pour un degré
- ou bien : il faut compenser par un degré immense

Réduction et tabulation

- La réduction d'argument ramène tout à un petit domaine
  - Propriétés algébriques de la fonction
    - Périodicité : sin
    - Autosimilarité : exp
    - Symétrie : asin
  - Caractère semi-logarithmique du format flottant
    - Vu de loin, convertToInteger et exp se ressemblent
  - Si rien d'autre : découpage du domaine

Réduction et tabulation

- La réduction d'argument ramène tout à un petit domaine
  - Propriétés algébriques de la fonction
    - Périodicité : sin
    - Autosimilarité : exp
    - Symétrie : asin
  - Caractère semi-logarithmique du format flottant
    - Vu de loin, convertToInteger et exp se ressemblent
  - Si rien d'autre : découpage du domaine

- Exemple pour exp :

$$e^x = 2^{\frac{x}{\log 2}} = 2^{\left\lfloor \frac{x}{\log 2} \right\rceil} \cdot 2^{\frac{x}{\log 2} - \left\lfloor \frac{x}{\log 2} \right\rceil} = 2^E \cdot e^{x - E \log 2} = 2^E \cdot e^r$$

Réduction et tabulation

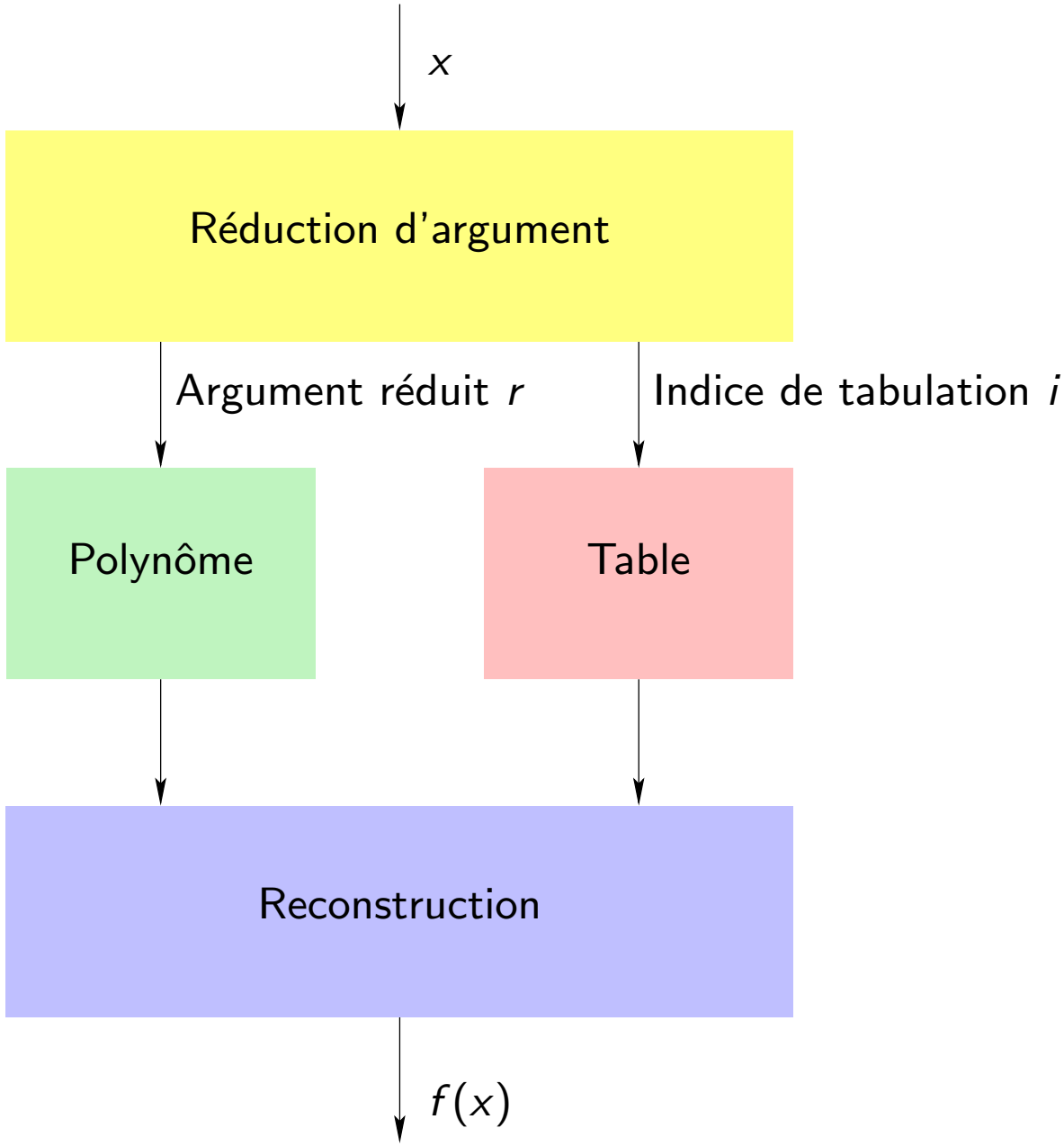
- La réduction d'argument ramène tout à un petit domaine
  - Propriétés algébriques de la fonction
    - Périodicité : sin
    - Autosimilarité : exp
    - Symétrie : asin
  - Caractère semi-logarithmique du format flottant
    - Vu de loin, convertToInteger et exp se ressemblent
  - Si rien d'autre : découpage du domaine

- Exemple pour exp :

$$e^x = 2^{\frac{x}{\log 2}} = 2^{\left\lfloor \frac{x}{\log 2} \right\rceil} \cdot 2^{\frac{x}{\log 2} - \left\lfloor \frac{x}{\log 2} \right\rceil} = 2^E \cdot e^{x - E \log 2} = 2^E \cdot e^r$$

- La réduction est souvent couplée à une tabulation
  - Tabulation : précalcul d'une fonction g à des endroits discrets
  - Réduction : le polynôme p doit être valide seulement entre ces endroits discrets

Schéma de calcul d'une fonction

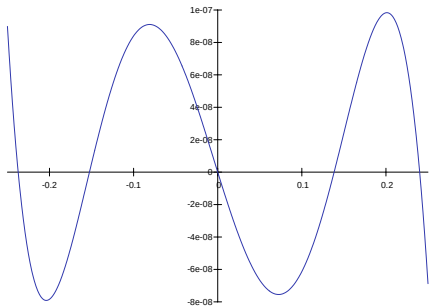


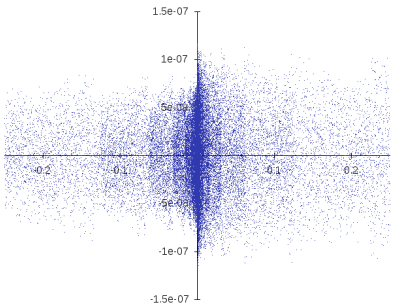
Une exponentielle jouet

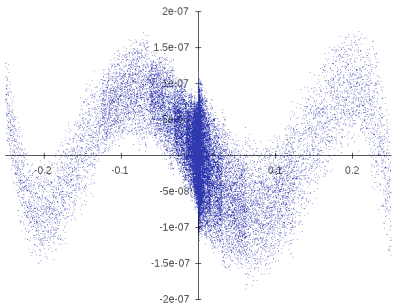
```
// A very crude "toy" implementation of exp(x)
//
// About 45 bits of accuracy. No checks for NaN, Inf whatsoever.
//
double Exp(double x) {
  double z, n, t, r, P, tbl, y;
  uint32_t E, idx, N;
  doubleCaster shiftedN, twoE;

  // Argument reduction
  z = x * TWO_4_RCP_LN_2;          // z = x * 2^4 * 1/ln(2)
  shiftedN.d = z + TWO_52_P_51;    // shiftedN.d = nearestint(z) + 2^52 + 2^51
  n = shiftedN.d - TWO_52_P_51;    // n = nearestint(z) as double
  N = shiftedN.i[LO];              // N = nearestint(z) as integer
  E = N >> 4;                      // E = floor(2^-4 * N)
  idx = N & 0x0f;                  // idx = N - E * 2^4
  t = n * TWO_M_4_LN_2;            // t = n * 2^-4 * ln(2)
  r = x - t;                       // r = x - t

  // Polynomial approximation          p(r) approximates exp(r)
  P = c0 + r * (c1 + r * (c2 + r * (c3 + r * (c4 + r * c5))));

  // Table access
  tbl = table[idx];                // tbl = 2^(2^-4 * idx)

  // Reconstruction
  twoE.i[HI] = (E + 1023) << 20;
  twoE.i[LO] = 0;                  // twoE.d = 2^E
  y = twoE.d * (tbl * P);          // y = 2^E * tbl * P

  return y;
}
```

Sources d'erreur

- Des approximations en flottant, entachées d'erreurs

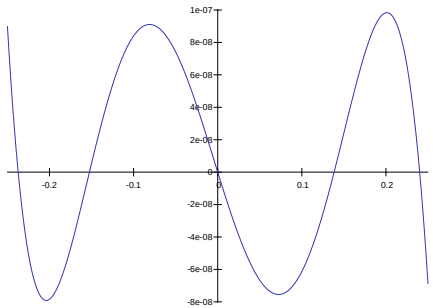


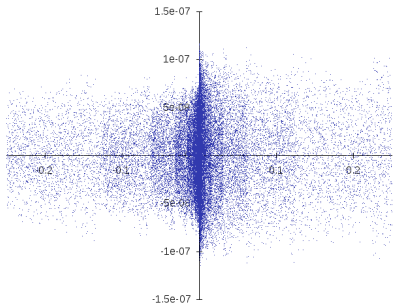


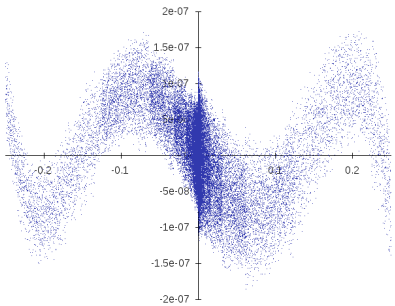


Sources d'erreur

- Des approximations en flottant, entachées d'erreurs



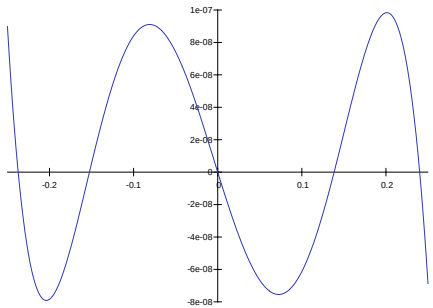


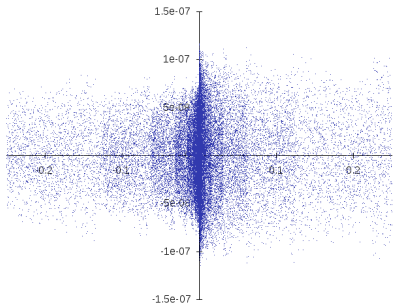


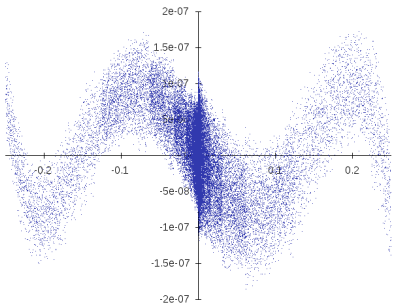
- Cinq sources d'erreur principales :
  - Erreur de réduction d'argument
  - Erreur dans les entrées de table
  - Erreur d'approximation $\|p/f - 1\|_\infty$
  - Erreur d'évaluation du polynôme $|P(x)/p(x) - 1|$
  - Erreur de reconstruction

Sources d'erreur

- Des approximations en flottant, entachées d'erreurs







- Cinq sources d'erreur principales :
  - Erreur de réduction d'argument
  - Erreur dans les entrées de table
  - Erreur d'approximation $\|p/f - 1\|_\infty$
  - Erreur d'évaluation du polynôme $|P(x)/p(x) - 1|$
  - Erreur de reconstruction

- Combinaison des erreurs pour donner l'erreur globale

Méthodes et outils pour l'analyse de précision

- Analyse manuelle
  - Réduction d'argument, tabulation
  - Très fastidieuse
  - Très dangereuse

1.

Méthodes et outils pour l'analyse de précision

- Analyse manuelle
  - Réduction d'argument, tabulation
  - Très fastidieuse
  - Très dangereuse
- L'outil Gappa (Melquiond)
  - Évaluation et reconstruction
  - Rélativement simple (quasi-automatique)
  - Génération d'une preuve Coq possible

1.

Méthodes et outils pour l'analyse de précision

- Analyse manuelle
  - Réduction d'argument, tabulation
  - Très fastidieuse
  - Très dangereuse
- L'outil Gappa (Melquiond)
  - Évaluation et reconstruction
  - Rélativement simple (quasi-automatique)
  - Génération d'une preuve Coq possible
- Algorithmes de calcul de norme infini validée [1]
  - Erreur d'approximation
  - Automatique et rapide
  - Résultats validés par arithmétique d'intervalle
  - Génération d'une preuve HOL est sujet de recherche

---

1. Chevillard, Harrison, Joldeș et Lauter. *Efficient and accurate computation of upper bounds of approximation errors*, TCS, à paraître

Arithmétiques double-double et multi-double

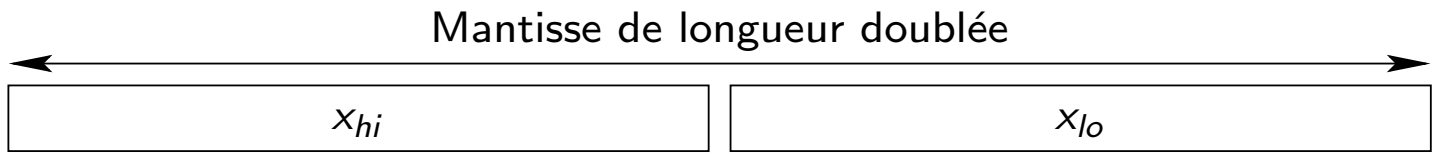
- Précision maximale disponible en matériel : double précision
- But : Arrondi fidèle ou correct en double précision

Arithmétiques double-double et multi-double

- Précision maximale disponible en matériel : double précision
- But : Arrondi fidèle ou correct en double précision
  - *Bits de garde* nécessaires audelà de la double précision
  - Précisions d'environ 120-130 bits pour l'arrondi correct

Arithmétiques double-double et multi-double

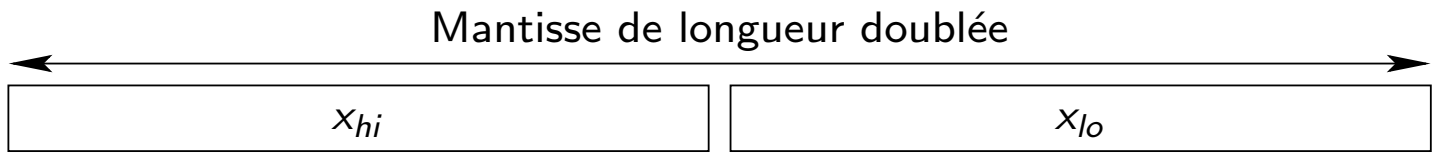
- Précision maximale disponible en matériel : double précision
- But : Arrondi fidèle ou correct en double précision
  - *Bits de garde* nécessaires audelà de la double précision
  - Précisions d'environ 120-130 bits pour l'arrondi correct
- Arithmétique double-double :
  - Représenter x par une somme $x_{hi} + x_{lo}$ de deux doubles



  - Base : errorfree transformations
    - Calcul de l'erreur des opérations flottantes

Arithmétiques double-double et multi-double

- Précision maximale disponible en matériel : double précision
- But : Arrondi fidèle ou correct en double précision
  - *Bits de garde* nécessaires audelà de la double précision
  - Précisions d'environ 120-130 bits pour l'arrondi correct
- Arithmétique double-double :
  - Représenter x par une somme $x_{hi} + x_{lo}$ de deux doubles



  - Base : errorfree transformations
    - Calcul de l'erreur des opérations flottantes
- Un concept extensible : arithmétique multi-double
  - Sommes de 3, 4 . . . doubles
  - Triple-double suffisante pour l'arrondi correct en double
  - Un facteur 10 plus rapide que l'émulation logicielle

Défis de l'arrondi correct

Précision au pire cas

- Pas de calcul *direct* de l'arrondi correct $F(x) = \circ(f(x))$
- Calcul d'une approximation $f(x) \cdot (1 + \varepsilon)$ plus précise que...
  ... la distance relative du pire cas

Précision au pire cas

- Pas de calcul *direct* de l'arrondi correct $F(x) = \circ(f(x))$
- Calcul d'une approximation $f(x) \cdot (1 + \varepsilon)$ plus précise que...
  ... la distance relative du pire cas
- Utilisation d'un lemme comme le suivant :

Lemme

Soit $f = \exp$. *Soit* $\circ_{53} : \mathbb{R} \to \mathbb{F}_{53}$ *un arrondi double précision.*
Soit $\mathbb{D} = \mathbb{F}_{53} \cap [-744.5; 709]$ *le domaine de définition double de* f.
Alors pour tout $x \in \mathbb{D} \backslash \{0\}$ *et tout* $|\varepsilon| \leq 2^{-159}$,

$$\circ_{53}(f(x) \cdot (1 + \varepsilon)) = \circ_{53}(f(x)).$$

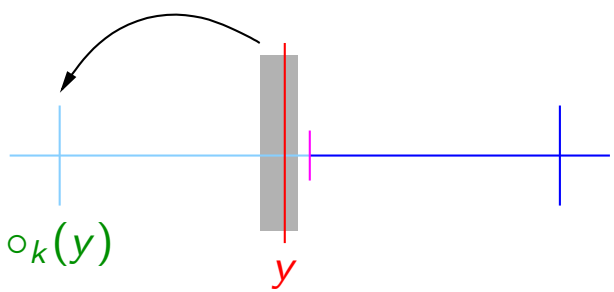
Précision au pire cas

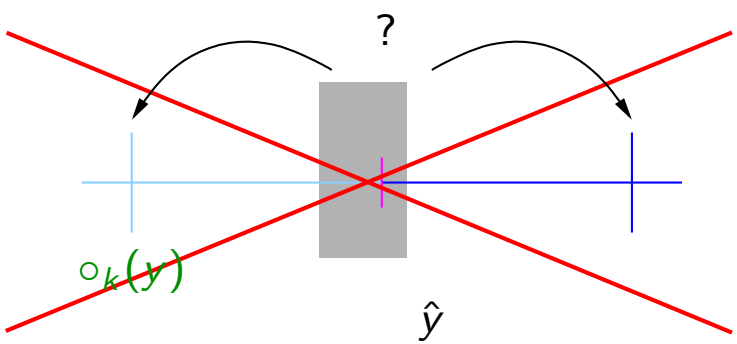
- Pas de calcul *direct* de l'arrondi correct $F(x) = \circ(f(x))$
- Calcul d'une approximation $f(x) \cdot (1 + \varepsilon)$ plus précise que...
  ... la distance relative du pire cas
- Utilisation d'un lemme comme le suivant :

Lemme

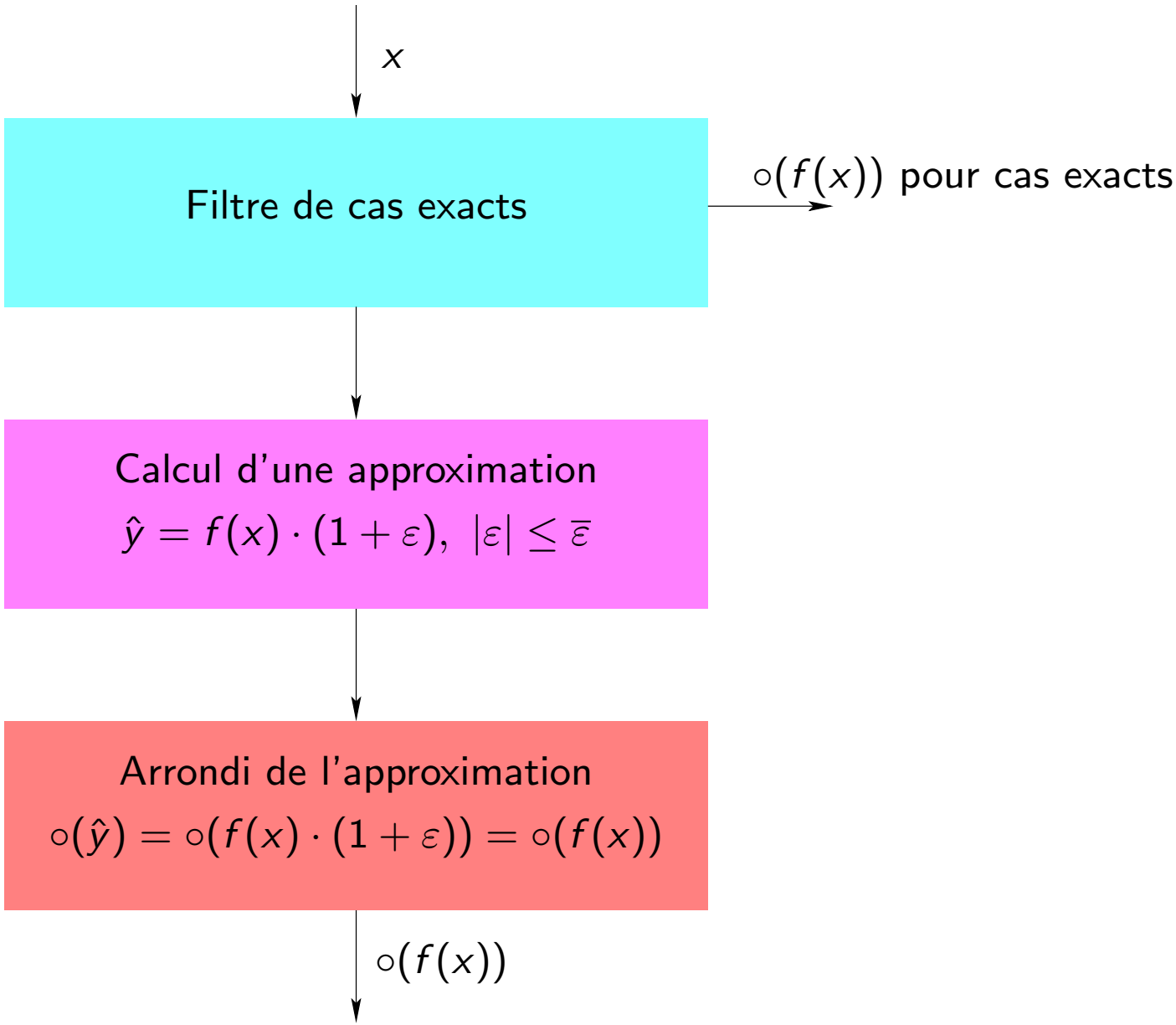
Soit $f = \exp$. *Soit* $\circ_{53} : \mathbb{R} \to \mathbb{F}_{53}$ *un arrondi double précision.*
Soit $\mathbb{D} = \mathbb{F}_{53} \cap [-744.5; 709]$ *le domaine de définition double de* f.
Alors pour tout $x \in \mathbb{D} \backslash \{0\}$ *et tout* $|\varepsilon| \leq 2^{-159}$,

$$\circ_{53}(f(x) \cdot (1 + \varepsilon)) = \circ_{53}(f(x)).$$

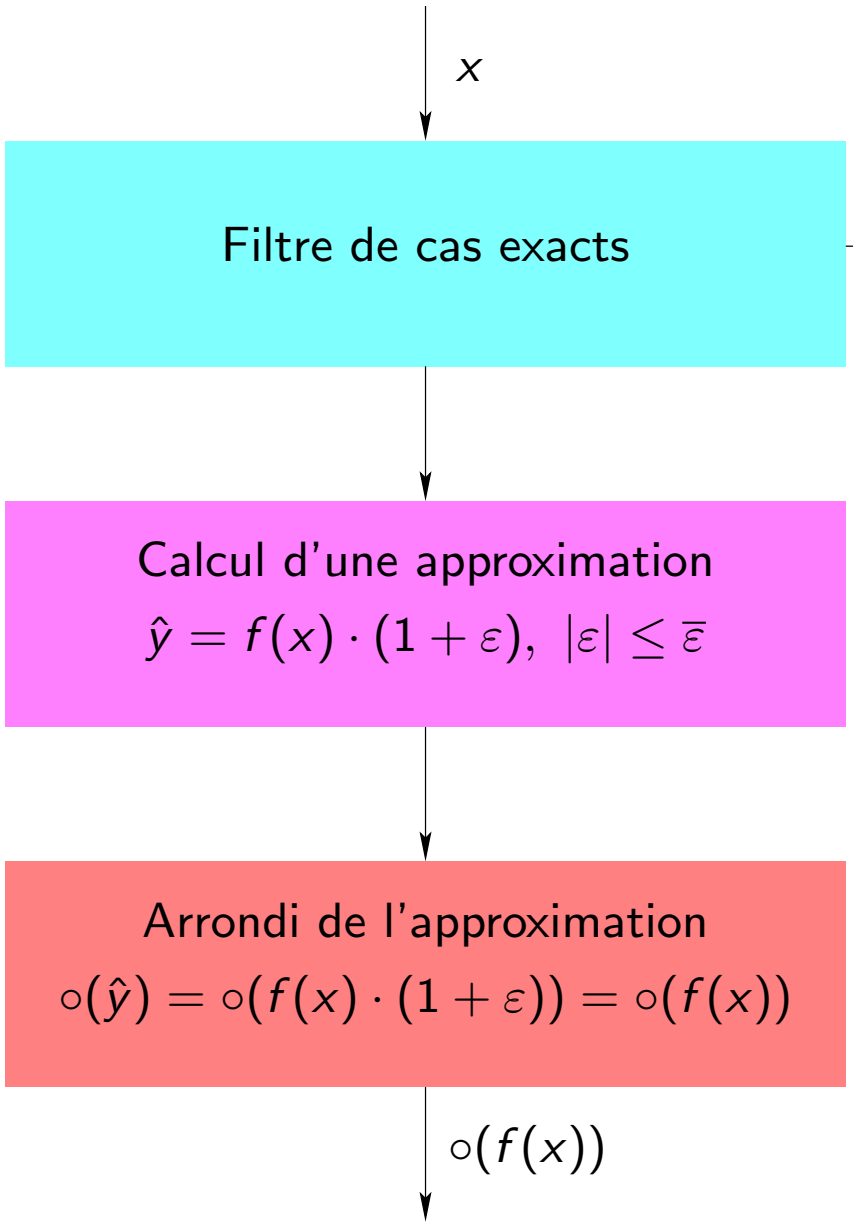




Un schéma simple pour l'arrondi correct

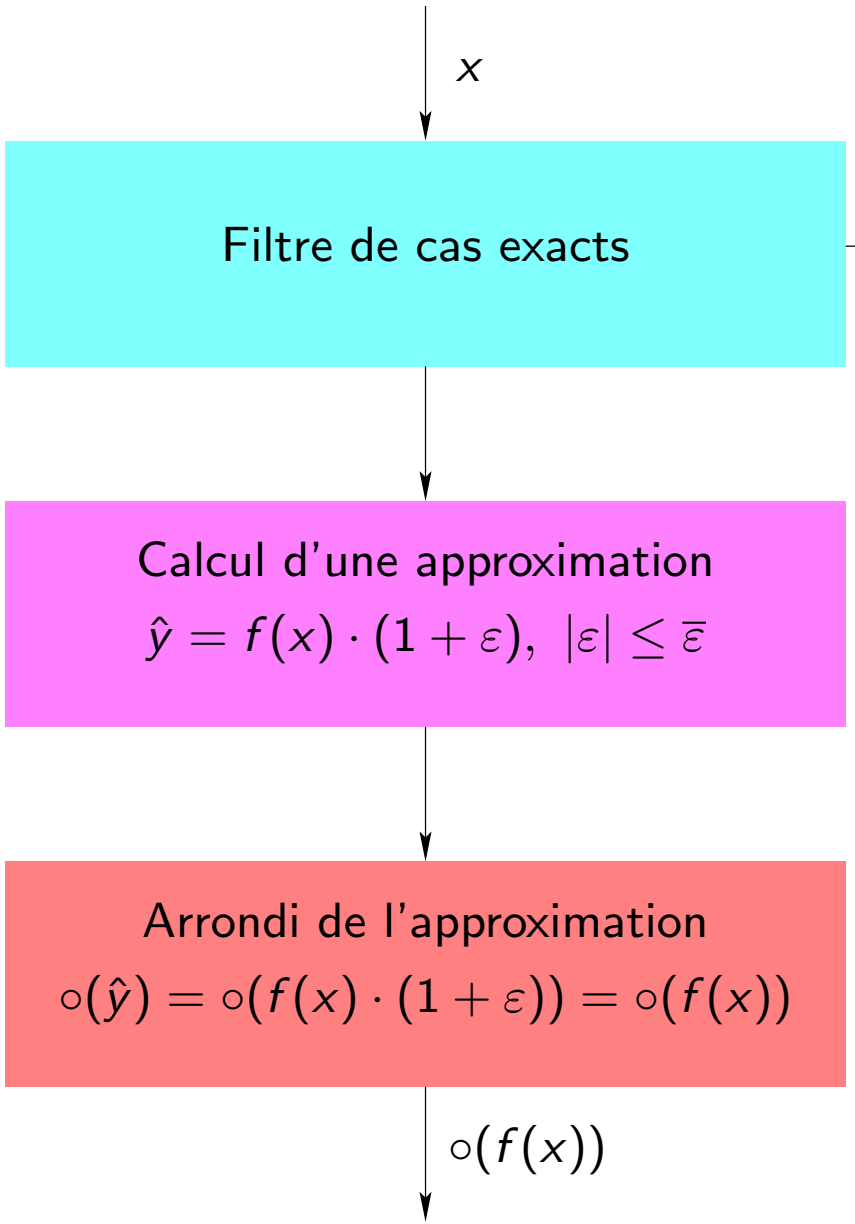


Un schéma simple pour l'arrondi correct



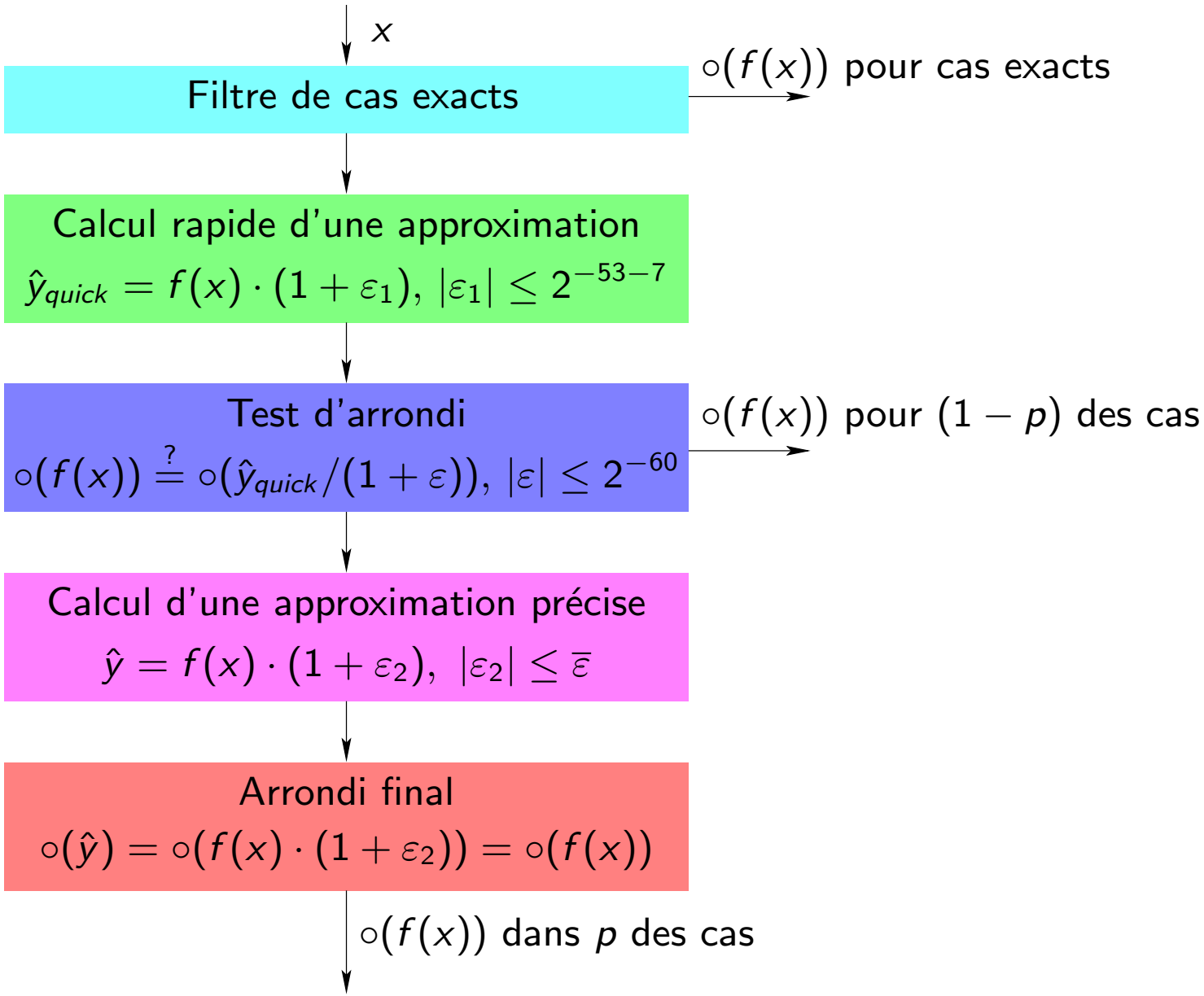
- Ça marche, très bien,

Un schéma simple pour l'arrondi correct

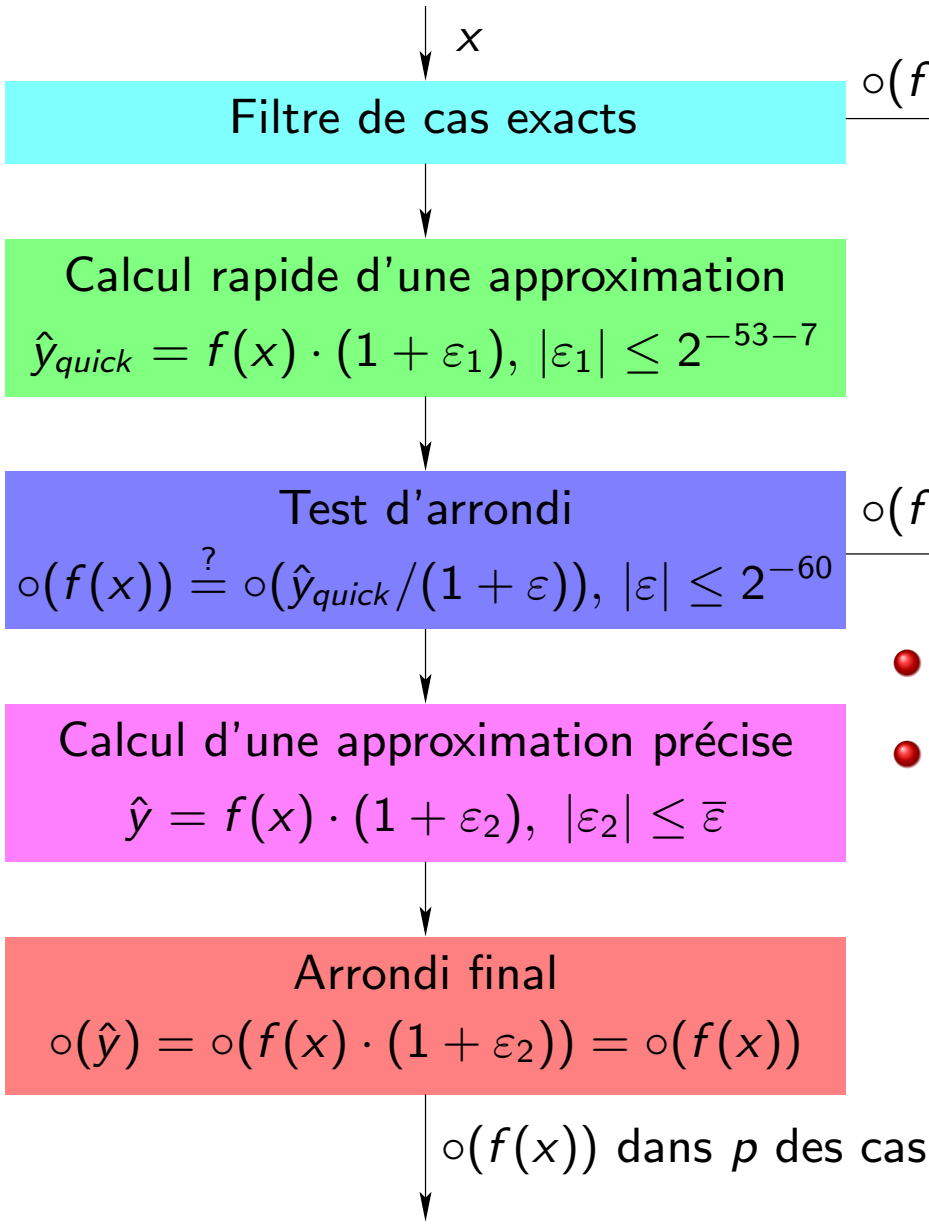


- Ça marche, très bien, mais...
- Ce n'est pas compétitif du tout
  - Arrondi fidèle : $53 + 5$ bits
  - Arrondi correct : 159 bits
  - Facteur 10 en temps de calcul

L'approche en deux étapes



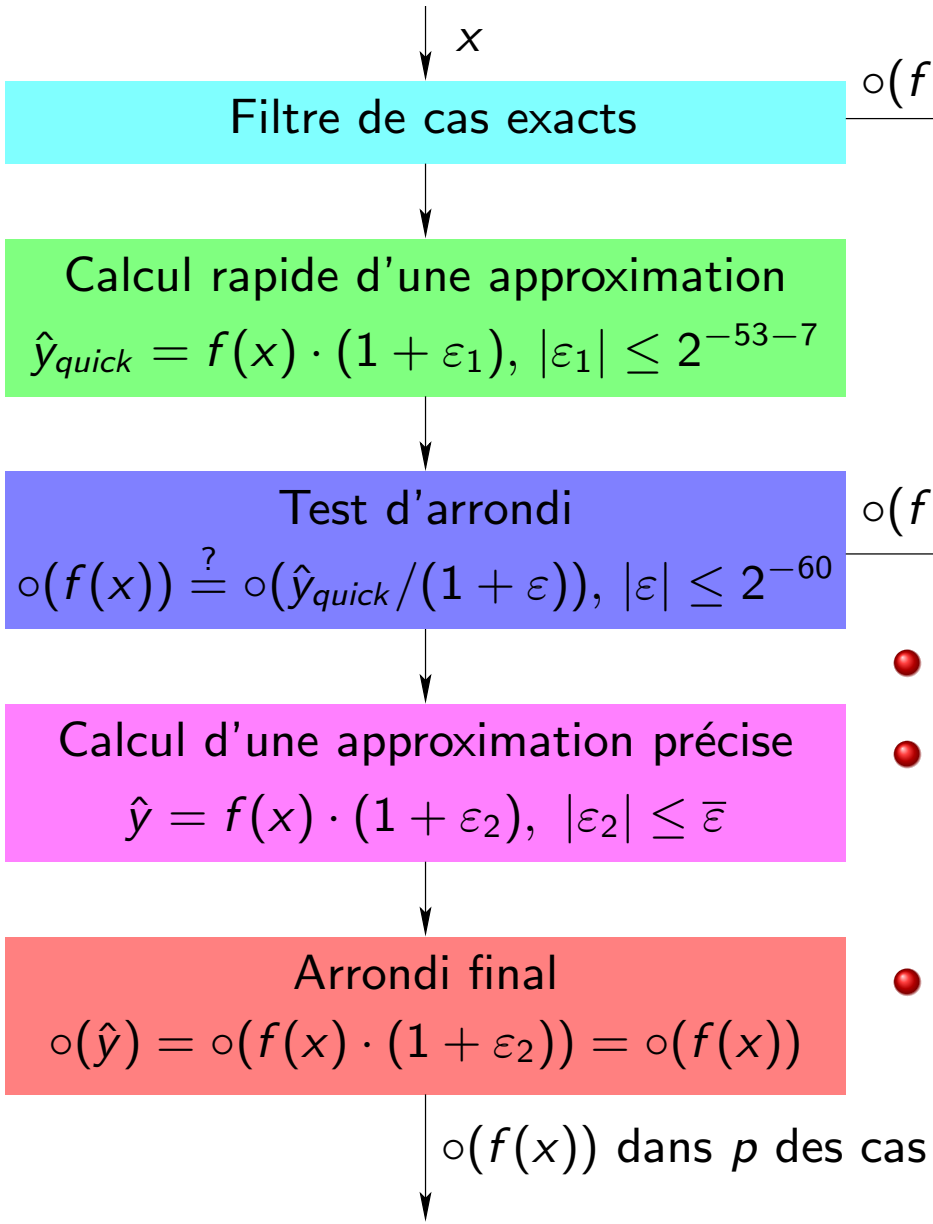
L'approche en deux étapes



- Maintenant, c'est compétitif
- Temps en moyenne

$$T_{average} = T_{quick} + p \cdot T_{accurate}$$

L'approche en deux étapes



- Maintenant, c'est compétitif
- Temps en moyenne

$$T_{average} = T_{quick} + p \cdot T_{accurate}$$

- Valeurs typiques
  - $T_{quick} = 1.01 \cdot T_{faithful}$
  - $T_{accurate} = 10 \cdot T_{quick}$
  - $p = 2^{-7}$

Performances CRLibm

Opteron (cycles)	moyenne	pire
CRLibm en double-étendue	118	862
libm par défaut (sans arrondi correct)	189	8050

Pentium 4 (cycles)	moyenne	pire
CRLibm en double-étendue	339	4824
libm par défaut (sans arrondi correct)	332	8424

Pentium 3 (cycles)	moyenne	pire
CRLibm en double-étendue	150	891
libm par défaut (sans arrondi correct)	172	1286

Power 5 (unités de mésure)	moyenne	pire
CRLibm (sans FMA)	50	259
libm par défaut (avec arrondi correct)	52	28881

Itanium (cycles)	moyenne	pire
CRLibm en double-étendue avec FMA	73	2150
libm par défaut (sans arrondi correct)	54	8077

Preuves de correction

- Preuve de correction d'une fonction CRLibm
  - Preuve : analyse fine de l'erreur
    - Réduction d'argument
    - Erreur d'approximation
    - Erreur d'arrondi
    - etc.
  - Pour chacune des phases à part :
    - Phase rapide : hypothèses du test d'arrondi vérifiées
    - Phase précise : utilisation du lemme pire cas

Preuves de correction

- Preuve de correction d'une fonction CRLibm
  - Preuve : analyse fine de l'erreur
    - Réduction d'argument
    - Erreur d'approximation
    - Erreur d'arrondi
    - etc.
  - Pour chacune des phases à part :
    - Phase rapide : hypothèses du test d'arrondi vérifiées
    - Phase précise : utilisation du lemme pire cas
- Structure générale de la preuve d'arrondi correct

$$\frac{\begin{array}{cc}\text{Gappa} & \text{Norme inf}\end{array}}{\dfrac{\hat{y} = f(x)(1+\varepsilon),\ |\varepsilon| \leq \overline{\varepsilon} \qquad |\varepsilon| \leq \overline{\varepsilon} \Rightarrow \circ(f(x)) = \circ(f(x)\cdot(1+\varepsilon))}{\circ(\hat{y}) = \circ(f(x))}}$$

Le lemme clef qui manque...

- Méthodologie générale de preuve – 1 thèse
- Gappa : analyse formelle des erreurs d'arrondi – 1 thèse
- Normes infinies rigoreuses et validées – 1 thèse

Le lemme clef qui manque...

- Méthodologie générale de preuve – 1 thèse
- Gappa : analyse formelle des erreurs d'arrondi – 1 thèse
- Normes infinies rigoreuses et validées – 1 thèse

- Rien de formel disponible pour les lemmes clefs :

Lemme

Soit $f = \exp$. *Soit* $\circ_{53} : \mathbb{R} \to \mathbb{F}_{53}$ *un arrondi double précision.*
Soit $\mathbb{D} = \mathbb{F}_{53} \cap [-744.5; 709]$ *le domaine de définition double de* f.
Alors pour tout $x \in \mathbb{D}\backslash\{0\}$ *et tout* $|\varepsilon| \leq 2^{-159}$,

$$\circ_{53}(f(x) \cdot (1 + \varepsilon)) = \circ_{53}(f(x)).$$

Le lemme clef qui manque...

- Méthodologie générale de preuve – 1 thèse
- Gappa : analyse formelle des erreurs d'arrondi – 1 thèse
- Normes infinies rigoreuses et validées – 1 thèse

- Rien de formel disponible pour les lemmes clefs :

Lemme

Soit $f = \exp$. *Soit* $\circ_{53} : \mathbb{R} \to \mathbb{F}_{53}$ *un arrondi double précision.*
Soit $\mathbb{D} = \mathbb{F}_{53} \cap [-744.5; 709]$ *le domaine de définition double de* f.
Alors pour tout $x \in \mathbb{D}\backslash\{0\}$ *et tout* $|\varepsilon| \leq 2^{-159}$,

$$\circ_{53}(f(x) \cdot (1 + \varepsilon)) = \circ_{53}(f(x)).$$

- Ben, bon, c'est la tâche de l'ANR TaMaDi...

Vers une vraie industrialisation de l'arrondi correct

Coût d'une bibliothèque LIBM

- Coût d'une fonction en termes de développement
  - Début de CRLibm : 3-18 mois homme par fonction
  - Coût moyen CRLibm : 1 mois homme
  - En industrie : 3 semaines homme

Coût d'une bibliothèque LIBM

- Coût d'une fonction en termes de développement
  - Début de CRLibm : 3-18 mois homme par fonction
  - Coût moyen CRLibm : 1 mois homme
  - En industrie : 3 semaines homme
- Nombre de fonctions dans une LIBM
  - IEEE 754-2008 : 17 fonctions
  - LIBM standard : à peu près 70 fonctions

Coût d'une bibliothèque LIBM

- Coût d'une fonction en termes de développement
  - Début de CRLibm : 3-18 mois homme par fonction
  - Coût moyen CRLibm : 1 mois homme
  - En industrie : 3 semaines homme
- Nombre de fonctions dans une LIBM
  - IEEE 754-2008 : 17 fonctions
  - LIBM standard : à peu près 70 fonctions
- Maintenance à coût élevé
  - Plusieurs codes adaptés aux différentes architectures
  - Optimisation à jamais pour de nouvelles architectures
  - Codes prouvés : synchroniser preuve et code

Coût d'une bibliothèque LIBM

- Coût d'une fonction en termes de développement
  - Début de CRLibm : 3-18 mois homme par fonction
  - Coût moyen CRLibm : 1 mois homme
  - En industrie : 3 semaines homme
- Nombre de fonctions dans une LIBM
  - IEEE 754-2008 : 17 fonctions
  - LIBM standard : à peu près 70 fonctions
- Maintenance à coût élevé
  - Plusieurs codes adaptés aux différentes architectures
  - Optimisation à jamais pour de nouvelles architectures
  - Codes prouvés : synchroniser preuve et code
- Problèmes des LIBM correctement arrondies
  - Calcul du pire cas nécessaire par fonction
  - Codes précisement analysés et prouvés :
    - Un travail de spécialistes

Le problème avec la preuve

- L'arrondi correct demande une grande précision.
- Rares sont les cas où la précision maximale est nécessaire.

exemple : exp, $\geq 2^{58}$ entrées valides

bits nécessaires	nombre de cas
159	1
150	6
140	36
130	121
⋮	⋮

Le problème avec la preuve

- L'arrondi correct demande une grande précision.
- Rares sont les cas où la précision maximale est nécessaire.

exemple : exp, $\geq 2^{58}$ entrées valides

bits nécessaires	nombre de cas
159	1
150	6
140	36
130	121
⋮	⋮

$\Rightarrow$ une détection de bogues par tests est insuffisante

Le problème avec la preuve

- L'arrondi correct demande une grande précision.
- Rares sont les cas où la précision maximale est nécessaire.

exemple : exp, $\geq 2^{58}$ entrées valides

bits nécessaires	nombre de cas
159	1
150	6
140	36
130	121
⋮	⋮

$\Rightarrow$ une détection de bogues par tests est insuffisante

- Mais : les ingénieurs vivent par les tests et haïssent les preuves

Le problème avec la preuve

- L'arrondi correct demande une grande précision.
- Rares sont les cas où la précision maximale est nécessaire.

exemple : exp, $\geq 2^{58}$ entrées valides

bits nécessaires	nombre de cas
159	1
150	6
140	36
130	121
⋮	⋮

$\Rightarrow$ une détection de bogues par tests est insuffisante

- Mais : les ingénieurs vivent par les tests et haïssent les preuves
- $\Rightarrow$ nos *solutions* pour l'arrondi correct ne sont pas prêtes

Vers la génération de codes LIBM

- L'assembleur est fastidieux ⇒ compilateurs

2. disponible sur `http://lipforge.ens-lyon.fr/www/metalibm/`

Vers la génération de codes LIBM

- L'assembleur est fastidieux ⇒ compilateurs
- Une LIBM est difficile à écrire ⇒ génération automagique !

2. disponible sur `http://lipforge.ens-lyon.fr/www/metalibm/`

Vers la génération de codes LIBM

- L'assembleur est fastidieux $\Rightarrow$ compilateurs
- Une LIBM est difficile à écrire $\Rightarrow$ génération automagique !
- MetaLibm [2] – un générateur prototypé :
  - Entrées :
    - Une fonction $f : \mathbb{R} \to \mathbb{R}$
    - Un petit domaine $dom = [a; b]$, après réduction d'argument
    - Une précision cible $\varepsilon \in \mathbb{R}^+$

2. disponible sur `http://lipforge.ens-lyon.fr/www/metalibm/`

Vers la génération de codes LIBM

- L'assembleur est fastidieux ⇒ compilateurs
- Une LIBM est difficile à écrire ⇒ génération automagique !
- MetaLibm [2] – un générateur prototypé :
  - Entrées :
    - Une fonction $f : \mathbb{R} \to \mathbb{R}$
    - Un petit domaine $dom = [a; b]$, après réduction d'argument
    - Une précision cible $\varepsilon \in \mathbb{R}^+$
  - Sorties :
    - Un code F tel que

$$\forall x \in dom, \quad \left| \frac{F(x) - f(x)}{f(x)} \right| \leq \varepsilon$$

    - Une preuve Gappa pour cette propriété
  - Ingrédients :
    - Approximation polynomiale à coefficients flottants et *à trous*
    - Évaluation en arithmétique multi-double

2. disponible sur `http://lipforge.ens-lyon.fr/www/metalibm/`

Génération de réductions d'argument

- Générer du code pour exp dans $\left[-2^{-5}; 2^{-5}\right]$ n'est pas tout
- $\Rightarrow$ il faut générer des codes de réduction d'argument

Génération de réductions d'argument

- Générer du code pour exp dans $\left[-2^{-5}; 2^{-5}\right]$ n'est pas tout
- $\Rightarrow$ il faut générer des codes de réduction d'argument
- La réduction d'argument – un vrai défi :
  - Une multitude de techniques hétérogènes
    - Abramowitz Stegun à coder en dur ?
    - Comment choisir les bons compromis ?
  - Codes très irréguliers
    - Mélange de flottant et de virgule fixe
    - Astuces sordides de manipulation de flottants
  - Preuve manuelle déjà mal comprise

Génération de réductions d'argument

- Générer du code pour exp dans $\left[-2^{-5}; 2^{-5}\right]$ n'est pas tout
- $\Rightarrow$ il faut générer des codes de réduction d'argument
- La réduction d'argument – un vrai défi :
  - Une multitude de techniques hétérogènes
    - Abramowitz Stegun à coder en dur ?
    - Comment choisir les bons compromis ?
  - Codes très irréguliers
    - Mélange de flottant et de virgule fixe
    - Astuces sordides de manipulation de flottants
  - Preuve manuelle déjà mal comprise
- La carotte : plein de possibilités ensuite
  - Recherche de compromis optimal tabulation/ approximation
  - Codes générés pour des précisions bizarres
  - Meilleure compréhension théorique de la réduction d'argument

Des transformations numériques de code

- Génération de codes : la fonction f est une boîte noire
  - Approximation polynomiale : indépendante de f
  - Réduction d'argument : analyse numérique des propriétés

3. Équipe-projet Algorithms, INRIA Rocquencourt

Des transformations numériques de code

- Génération de codes : la fonction f est une boîte noire
  - Approximation polynomiale : indépendante de f
  - Réduction d'argument : analyse numérique des propriétés
- Une fonction f boîte noire est...

3. Équipe-projet Algorithms, INRIA Rocquencourt

Des transformations numériques de code

- Génération de codes : la fonction f est une boîte noire
  - Approximation polynomiale : indépendante de f
  - Réduction d'argument : analyse numérique des propriétés
- Une fonction f boîte noire est...
  - un code aussi, peut-être pas efficace ;
  - p.ex. une inversion de Newton, une intégration d'équa diff.

3. Équipe-projet Algorithms, INRIA Rocquencourt

Des transformations numériques de code

- Génération de codes : la fonction f est une boîte noire
  - Approximation polynomiale : indépendante de f
  - Réduction d'argument : analyse numérique des propriétés
- Une fonction f boîte noire est...
  - un code aussi, peut-être pas efficace ;
  - p.ex. une inversion de Newton, une intégration d'équa diff.
- La génération automatique est une transformation de codes
  - Pas d'analyse de code traditionnelle
  - Purement un échantillonage (intervalle) numérique

3. Équipe-projet Algorithms, INRIA Rocquencourt

Des transformations numériques de code

- Génération de codes : la fonction f est une boîte noire
  - Approximation polynomiale : indépendante de f
  - Réduction d'argument : analyse numérique des propriétés
- Une fonction f boîte noire est...
  - un code aussi, peut-être pas efficace ;
  - p.ex. une inversion de Newton, une intégration d'équa diff.
- La génération automatique est une transformation de codes
  - Pas d'analyse de code traditionnelle
  - Purement un échantillonage (intervalle) numérique
- Recherche en cours :
  - Mezzarobba [3] a un code (en Maple) intégrant des équas diffs
  - On combine ce code avec MetaLibm
  - $\Rightarrow$ on passe d'une équa diff à un code en double

3. Équipe-projet Algorithms, INRIA Rocquencourt

Et d'autres codes ?

- Génération déjà bien présente en analyse numérique
  - BLAS : ATLAS
  - FFTs : SPIRAL

Et d'autres codes ?

- Génération déjà bien présente en analyse numérique
  - BLAS : ATLAS
  - FFTs : SPIRAL
- Codes moins réguliers :
  - Codes financiers
    - un Black-Scholes à 17 bits ?
  - Solutions d'équations différentielles implantées
    - L'évaluation remplace l'intégration
  - Traitement de signal
  - Un nouveau Formula Translator...

Et d'autres codes ?

- Génération déjà bien présente en analyse numérique
  - BLAS : ATLAS
  - FFTs : SPIRAL
- Codes moins réguliers :
  - Codes financiers
    - un Black-Scholes à 17 bits ?
  - Solutions d'équations différentielles implantées
    - L'évaluation remplace l'intégration
  - Traitement de signal
  - Un nouveau Formula Translator...
- Arithmétiques différentes
  - Virgule fixe
  - Systèmes flottants et logarithmiques sur FPGA
  - Arithmétique flottante vectorisée

Conclusions et extensions

Conclusions

- IEEE 754-2008 encourage les LIBMs avec arrondi correct
  - Des performances tout à fait comparables

Conclusions

- IEEE 754-2008 encourage les LIBMs avec arrondi correct
  - Des performances tout à fait comparables
- Approche comprise pour l'implantation et la preuve
  - L'implantation suit des principes classiques
    - Réduction d'argument
    - Tabulation
    - Approximation polynomiale
  - La preuve est aborable : Gappa et normes sup validées

Conclusions

- IEEE 754-2008 encourage les LIBMs avec arrondi correct
  - Des performances tout à fait comparables
- Approche comprise pour l'implantation et la preuve
  - L'implantation suit des principes classiques
    - Réduction d'argument
    - Tabulation
    - Approximation polynomiale
  - La preuve est aborable : Gappa et normes sup validées
- Il y a un immense trou dans la preuve
  - Tout est réduit à un lemme de pire cas
  - Ce lemme n'est pas prouvé
  - $\Rightarrow$ ANR TaMaDi

Conclusions

- IEEE 754-2008 encourage les LIBMs avec arrondi correct
  - Des performances tout à fait comparables
- Approche comprise pour l'implantation et la preuve
  - L'implantation suit des principes classiques
    - Réduction d'argument
    - Tabulation
    - Approximation polynomiale
  - La preuve est aborable : Gappa et normes sup validées
- Il y a un immense trou dans la preuve
  - Tout est réduit à un lemme de pire cas
  - Ce lemme n'est pas prouvé
  - $\Rightarrow$ ANR TaMaDi
- L'arrondi correct reste un domaine de spécialistes
  - Coût trop élevé d'une LIBM avec arrondi correct
  - Besoin de former des ingénieurs spécialement pour cela
  - $\Rightarrow$ Il faut des solutions *clef en main*

Extensions

- Solution proposée :
  - Au lieu des LIBM, écrire des générateurs de LIBM
  - Générateur optimise pour chaque architecture
  - Preuve générée automatiquement

Extensions

- Solution proposée :
  - Au lieu des LIBM, écrire des générateurs de LIBM
  - Générateur optimise pour chaque architecture
  - Preuve générée automatiquement
- État actuel
  - Code fonctionnel pour des domaines petits
  - Utilisé pour passer d'une équa diff à un code

Extensions

- Solution proposée :
  - Au lieu des LIBM, écrire des générateurs de LIBM
  - Générateur optimise pour chaque architecture
  - Preuve générée automatiquement
- État actuel
  - Code fonctionnel pour des domaines petits
  - Utilisé pour passer d'une équa diff à un code
- Défis et opportunités
  - Réduction d'argument nécessaire pour des domaines larges
  - Génération de preuve non-triviale
  - Nécessité de repenser les implantations traditionnelles
  - Approches transposables à d'autres types de code
  - Des codes mieux validés à un moindre coût

Extensions

- Solution proposée :
  - Au lieu des LIBM, écrire des générateurs de LIBM
  - Générateur optimise pour chaque architecture
  - Preuve générée automatiquement
- État actuel
  - Code fonctionnel pour des domaines petits
  - Utilisé pour passer d'une équa diff à un code
- Défis et opportunités
  - Réduction d'argument nécessaire pour des domaines larges
  - Génération de preuve non-triviale
  - Nécessité de repenser les implantations traditionnelles
  - Approches transposables à d'autres types de code
  - Des codes mieux validés à un moindre coût

Merci !

Merci pour votre attention !

Des questions ?