Vérification numérique des codes industriels (avec Verrou)

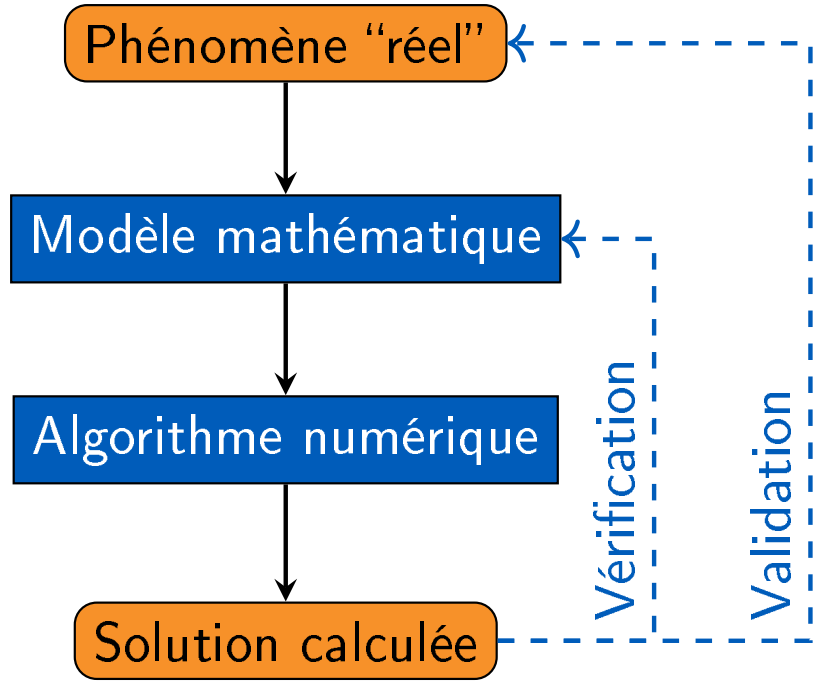
ETSN, Cargèse
24/04/2018

François Févotte[†], Bruno Lathuilière[†],
Eric Petit[*], Pablo de Oliveira Castro[§],
Yohan Chatelain[§]

[†]EDF R&D
[*]Intel, [§]UVSQ

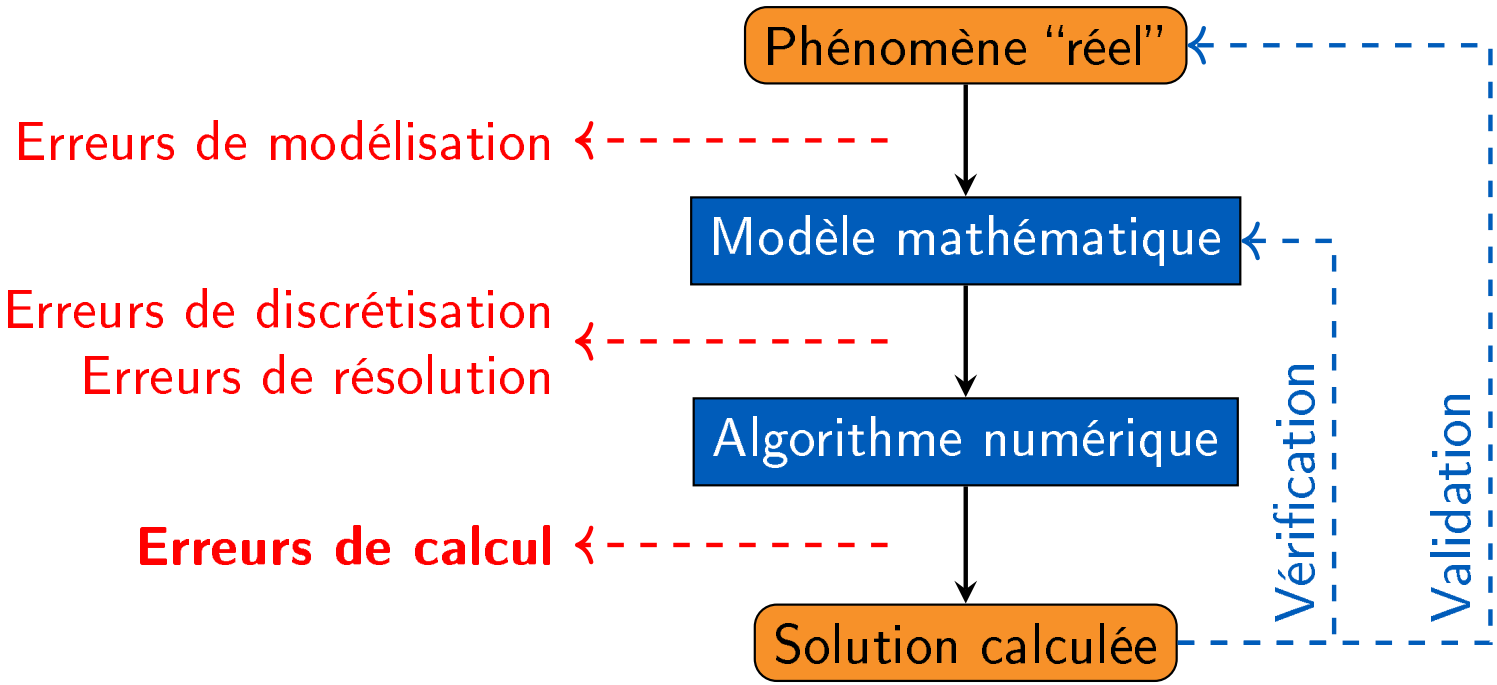
Contexte & enjeux

Vérification & Validation



Contexte & enjeux

Vérification & Validation

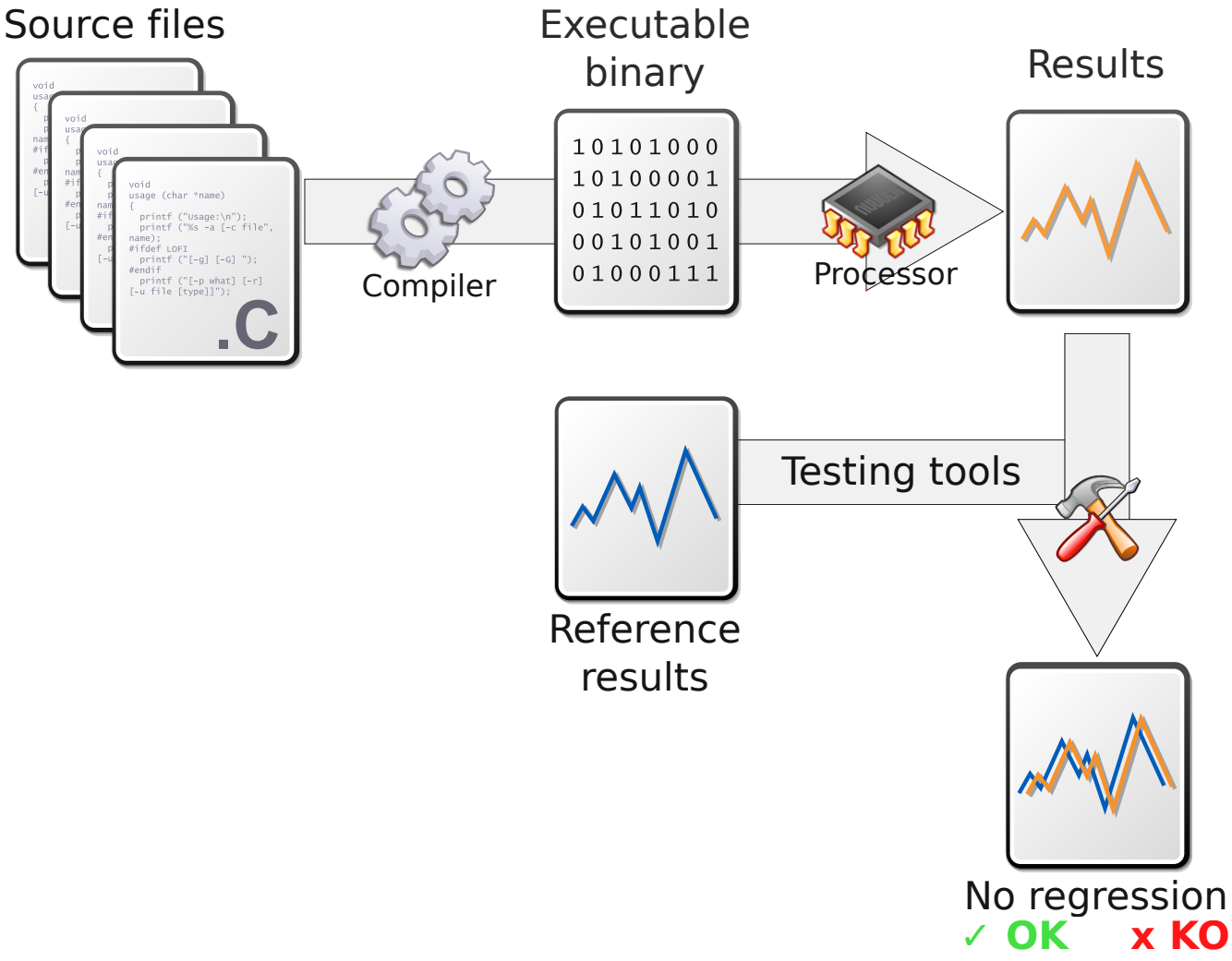


Quantification des erreurs : enjeux

- qualité des résultats produits
- efficacité d'utilisation des ressources (temps de calcul / développement)

Contexte & enjeux

Processus de développement sous AQ : V&V et non-régression



Contexte & enjeux

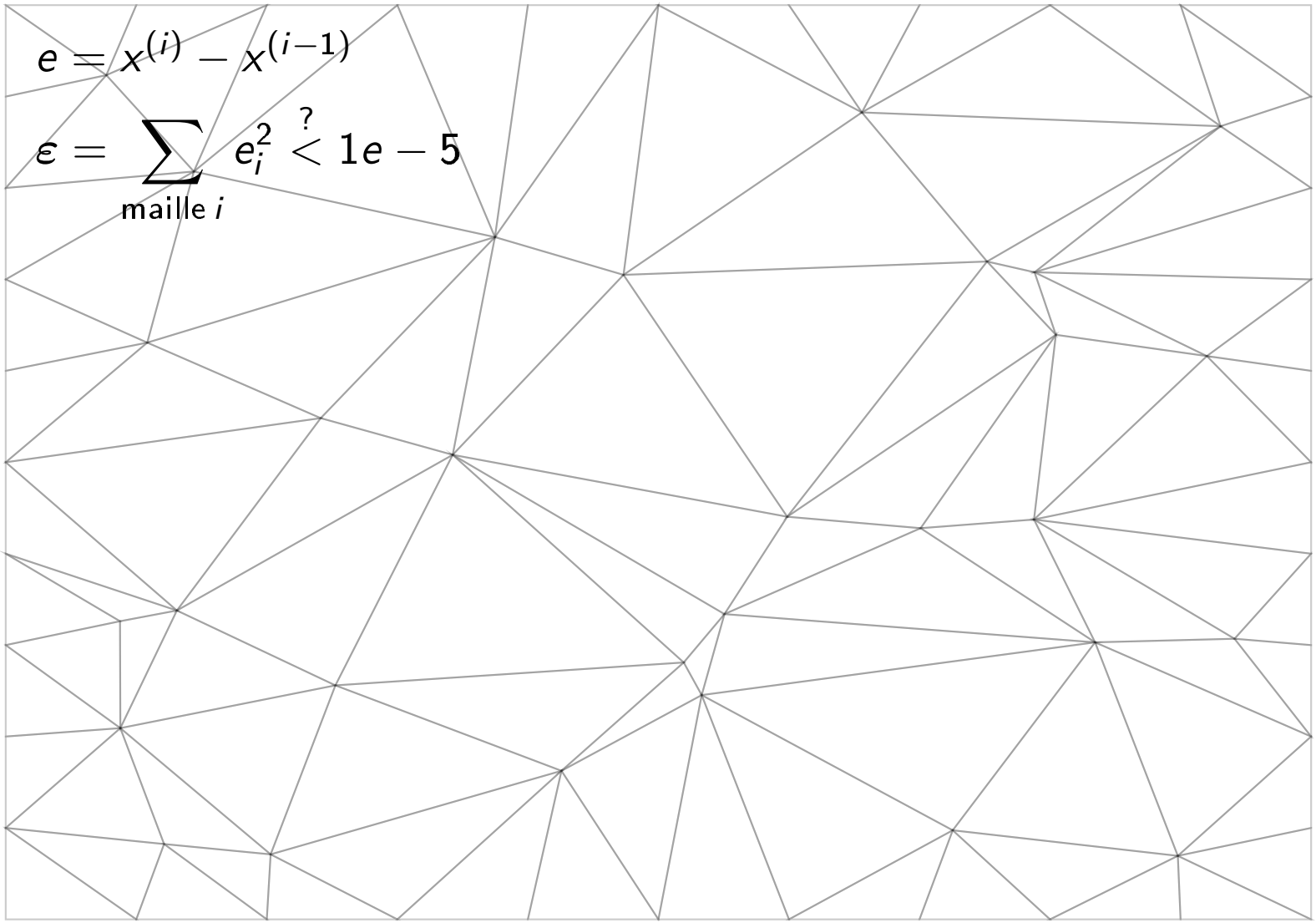
Quelles conséquences ?

Conséquences possibles de l'arithmétique flottante sur nos codes :

- **calcul bloqué (ex. TELEMAC2D)**
- résultat invalide (ex. production hydraulique)
- non-reproductibilité des résultats (ex. ASTER, COCAGNE, ATHENA...)
- performance (ex. SATURNE, Apogene)

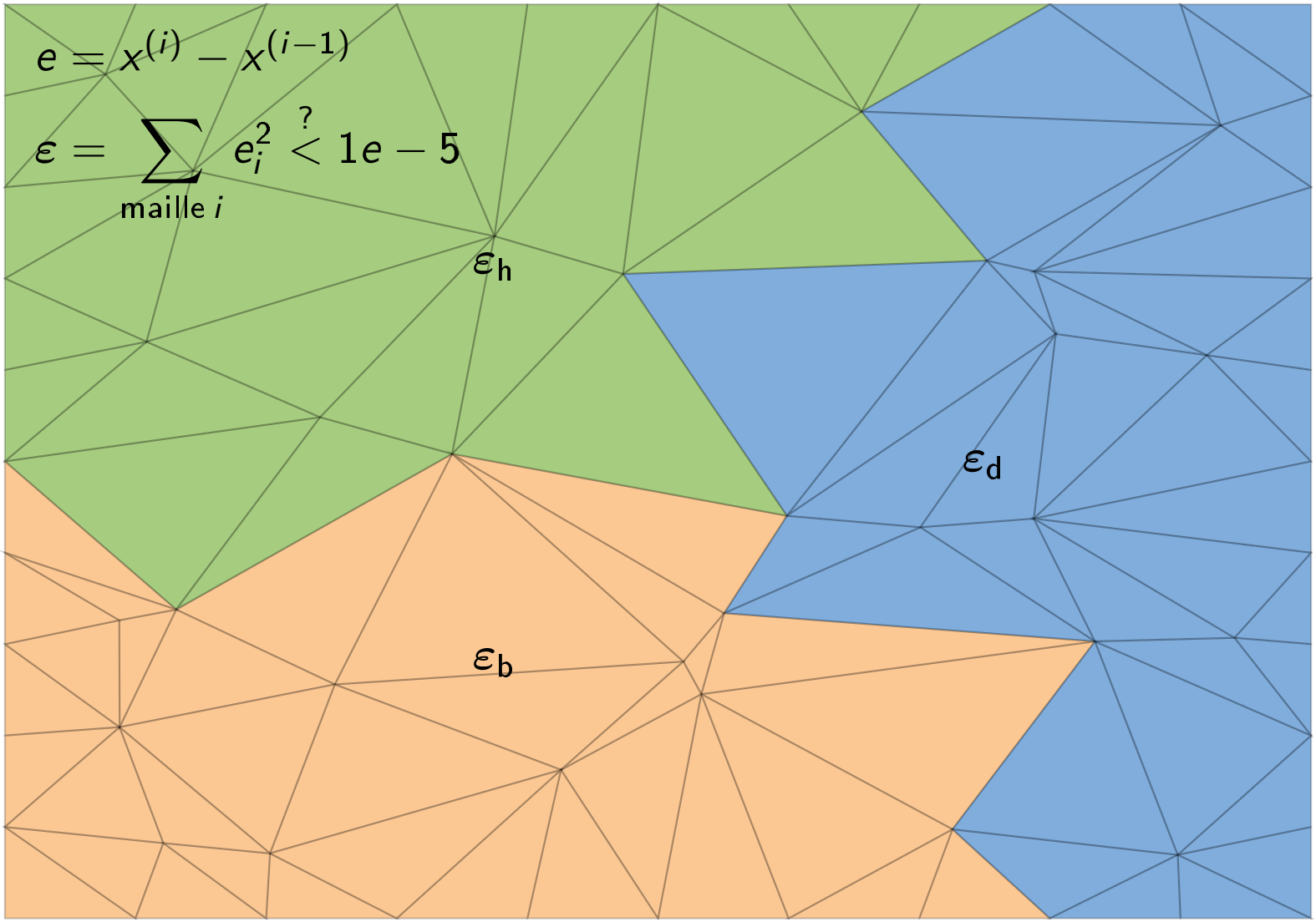
Contexte & enjeux

Calcul bloqué : ex. Telemac2D



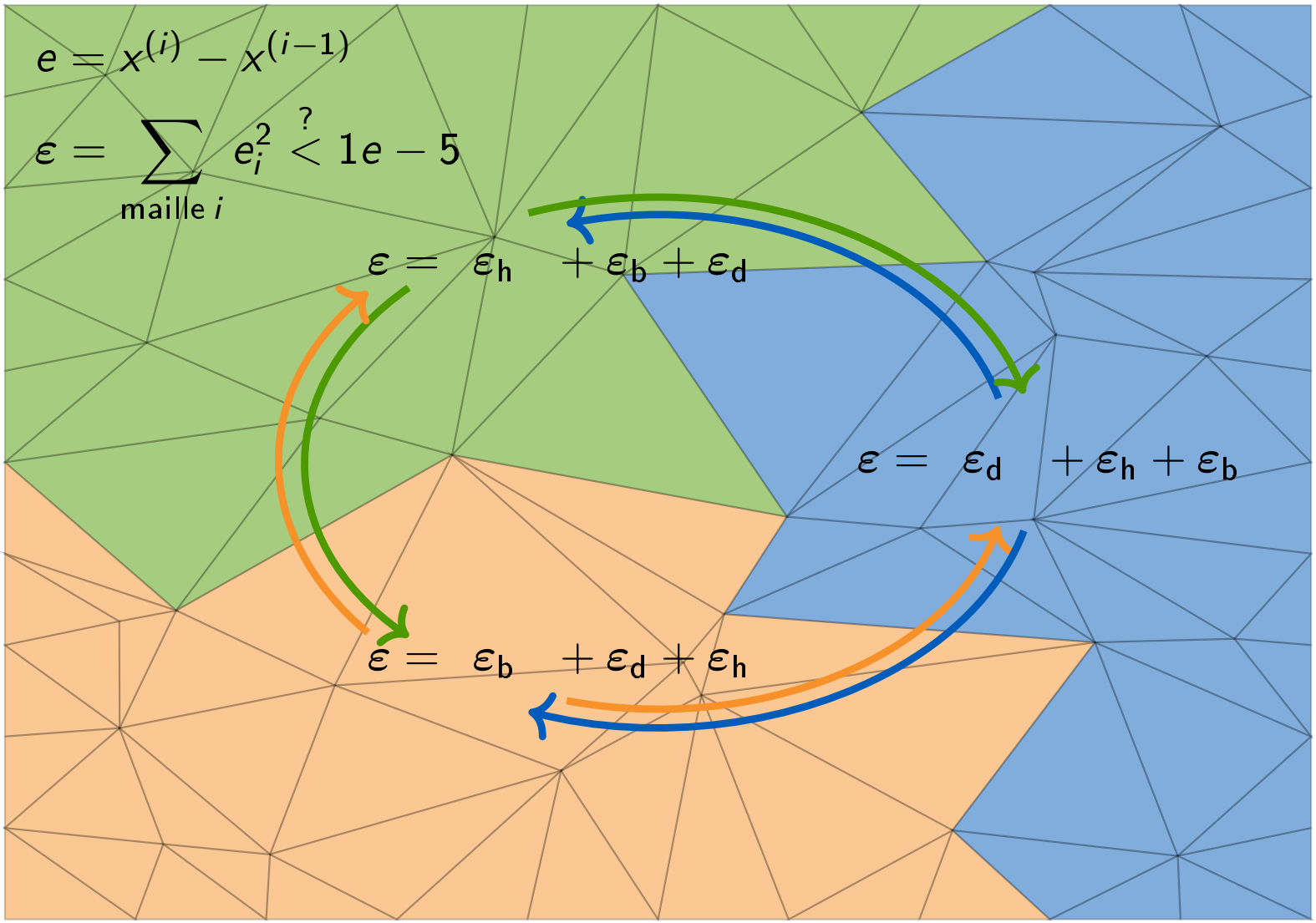
Contexte & enjeux

Calcul bloqué : ex. Telemac2D



Contexte & enjeux

Calcul bloqué : ex. Telemac2D



Contexte & enjeux

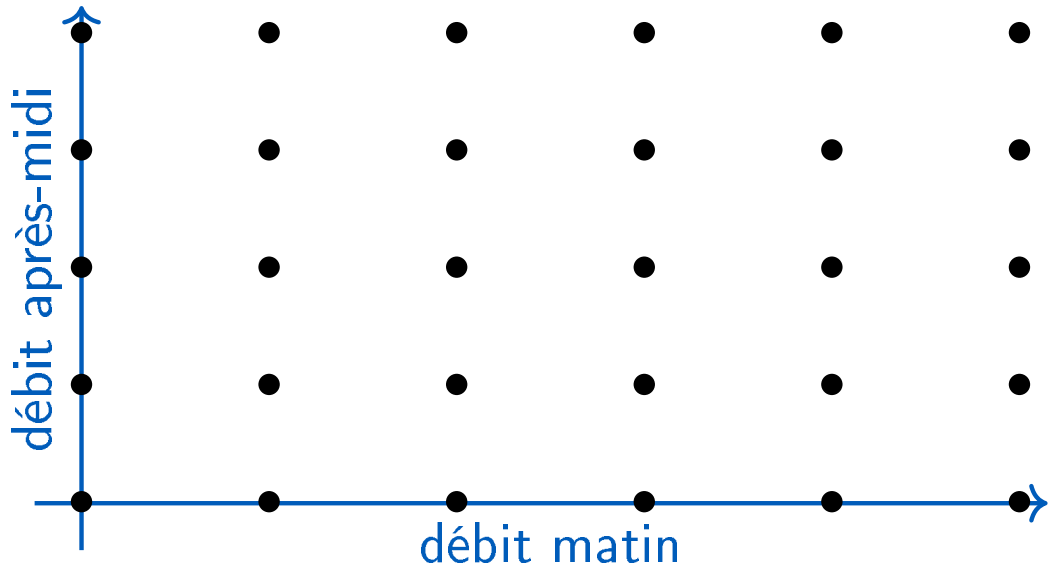
Quelles conséquences ?

Conséquences possibles de l'arithmétique flottante sur nos codes :

- calcul bloqué (ex. TELEMAC2D)
- **résultat invalide (ex. production hydraulique)**
- non-reproductibilité des résultats (ex. ASTER, COCAGNE, ATHENA...)
- performance (ex. SATURNE, Apogene)

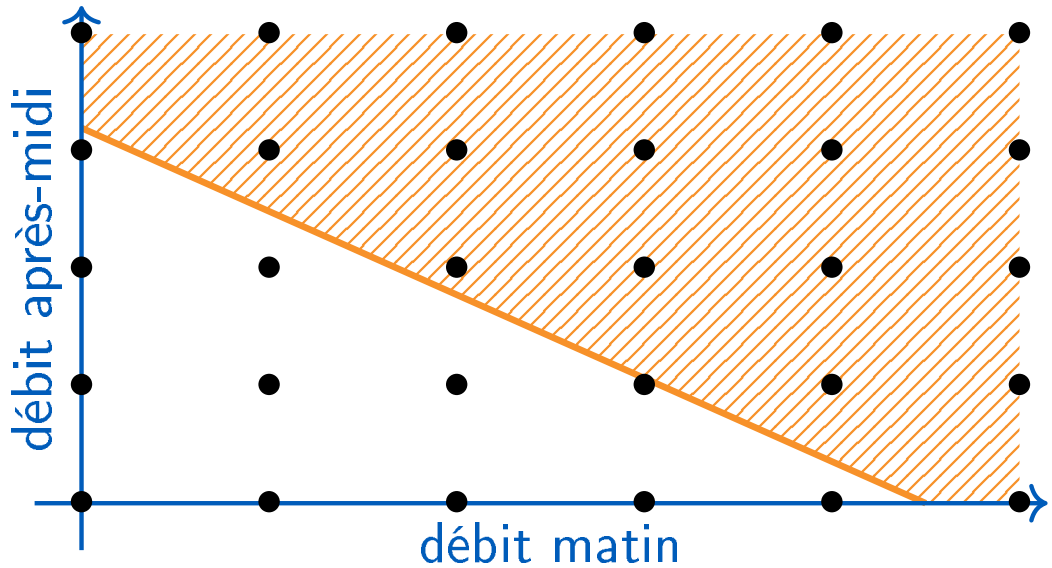
Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



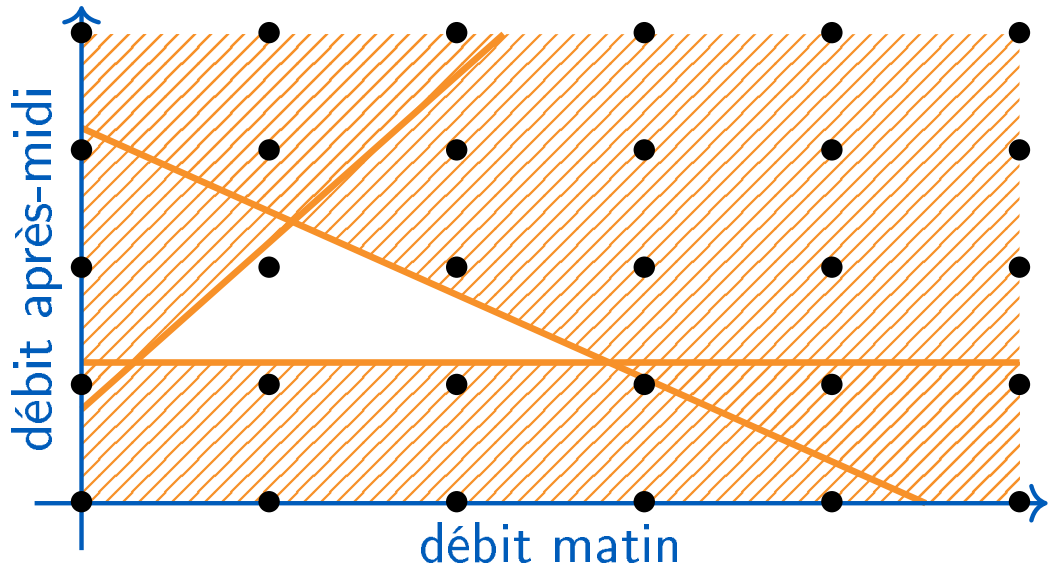
Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



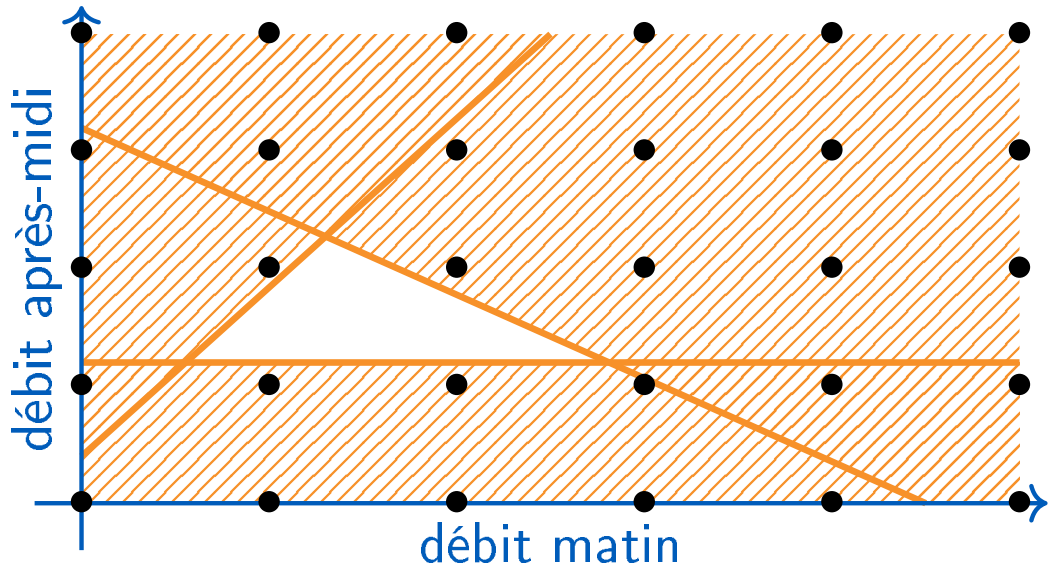
Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



Contexte & enjeux

Quelles conséquences ?

Conséquences possibles de l'arithmétique flottante sur nos codes :

- calcul bloqué (ex. TELEMAC2D)
- résultat invalide (ex. production hydraulique)
- **non-reproductibilité des résultats (ex. ASTER, COCAGNE, ATHENA...)**
- performance (ex. SATURNE, Apogene)

Contexte & enjeux

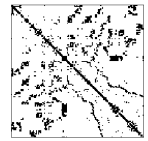
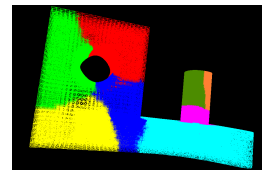
Non-reproductibilité

- Architecture du CPU 

- Calcul séquentiel ou parallèle  

- Compilateur 

(ou même simplement les options)

- Versions de bibliothèques externes  

Contexte & enjeux

Non-reproductibilité

Job	S	W
F3C		
COCAGNE Non Reg		
COCAGNE-NonReg		
Test on mutliple configuration		
COCAGNE-NonReg-c7-gcc-dbl-dklib-rel-par		
COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-par		
COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-seq		
COCAGNE-NonReg-c7-intel-dbl-dkzip-rel-par		
COCAGNE-NonReg-c7-gcc-flt-dklib-rel-par		
COCAGNE-NonReg-c7-gcc-flt-dkzip-rel-par		
COCAGNE-NonReg-c9-gcc-dbl-dkzip-rel-par		
COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-par-reloc		
COCAGNE-NonReg-c9-gcc-dbl-dkzip-rel-par-reloc		

Contexte & enjeux

Quelles conséquences ?

Conséquences possibles de l'arithmétique flottante sur nos codes :

- calcul bloqué (ex. TELEMAC2D)
- résultat invalide (ex. production hydraulique)
- non-reproductibilité des résultats (ex. ASTER, COCAGNE, ATHENA...)
- **performance (ex. SATURNE, Apogene)**

Contexte & enjeux

Performances : ex. optimisation production hydraulique

Formulation du problème

$$\max_{p,v} \sum_{t\in T}\sum_{i\in I} \lambda_t p_t^i + \sum_{r\in R} \omega_r \left(v_t^r - V_0^r - \sum_{t\in T} \Gamma_t^r \right)$$

Calcul de la fonction objectif

$$\sum_{t\in T}\sum_{i\in I} \lambda_t p_t^i + \sum_{r\in R} \omega_r v_t^r \qquad 1534019.677371745$$

$$-\sum_{r\in R} \omega_r \left(V_0^r + \sum_{t\in T} \Gamma_t^r \right) \qquad -1534019.677282780$$

$$0.000088965\underbrace{00000000000}_{\text{11 chiffres}}$$

Contexte & enjeux

Performances : ex. optimisation production hydraulique

Re-formulation proposée

$$\left[\max_{p,v} \sum_{t\in T}\sum_{i\in I} \lambda_t p_t^i + \sum_{r\in R} \omega_r v_t^r\right] - \sum_{r\in R} \omega_r \left(V_0^r + \sum_{t\in T} \Gamma_t^r\right)$$

Calcul de la fonction objectif

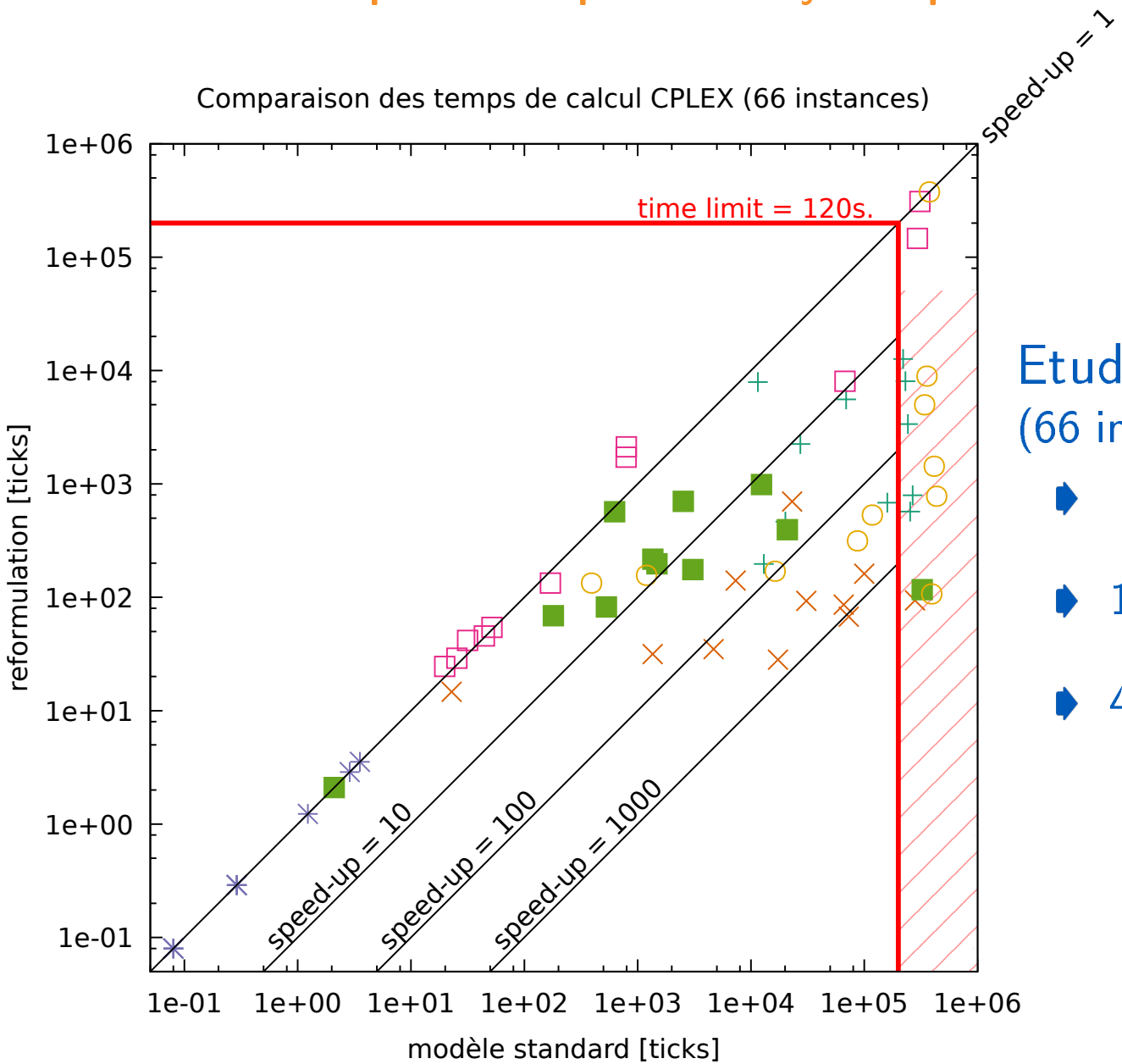
$$\sum_{t\in T}\sum_{i\in I} \lambda_t p_t^i + \sum_{r\in R} \omega_r v_t^r \qquad 1534019.677371745$$

$$-\sum_{r\in R} \omega_r \left(V_0^r + \sum_{t\in T} \Gamma_t^r\right) \qquad -1534019.677282780$$

$$0.000088965\underbrace{00000000000}_{\text{11 chiffres}}$$

Contexte & enjeux

Performances : ex. optimisation production hydraulique



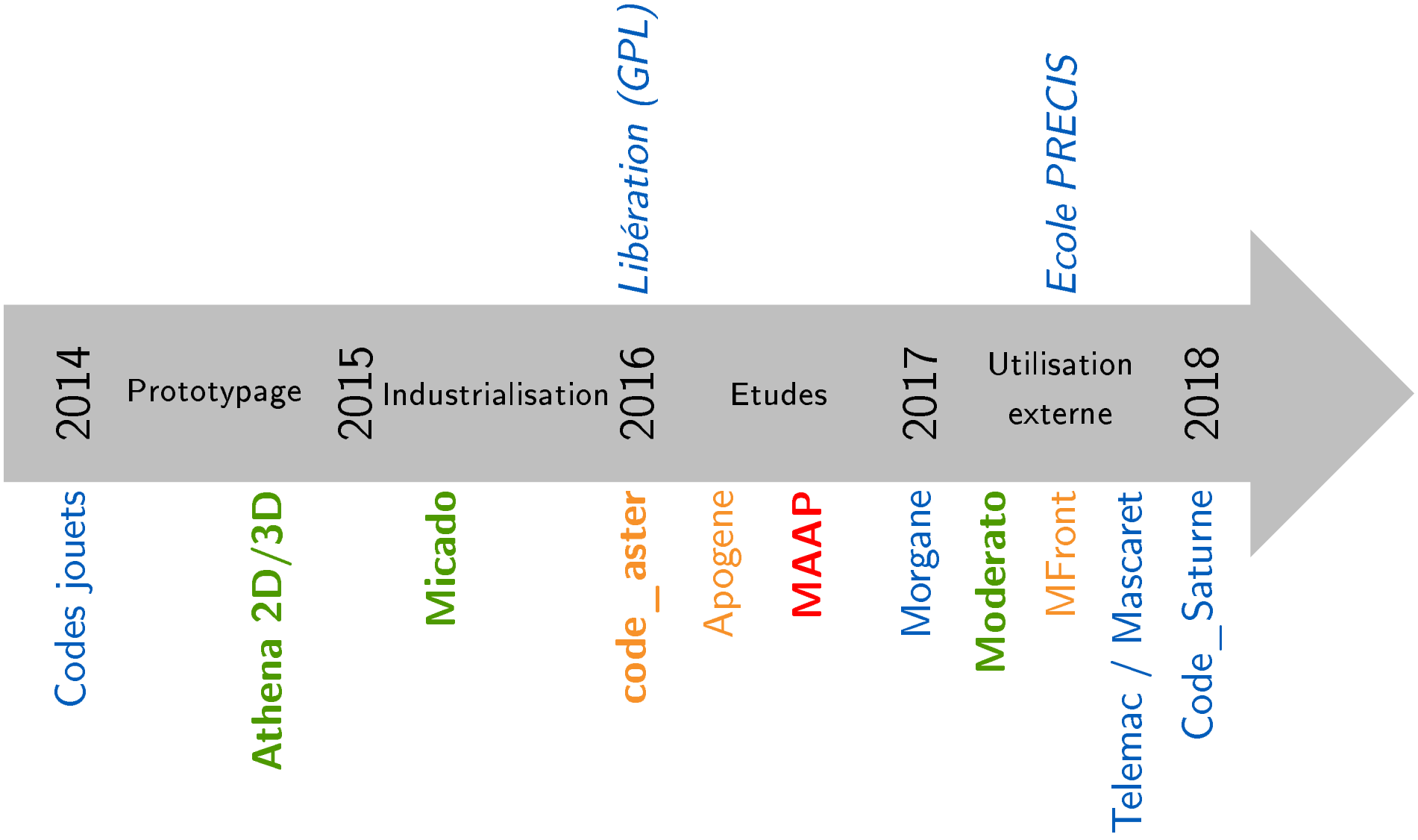
Etude en PLNE

(66 instances)

- 2 (3%) ralenties (×2)
- 15 (23%) ne changent pas
- 49 (74%) accélérées (×2–1100)

Contexte & enjeux

Bref historique de Verrou



Contexte & enjeux

Quels Besoins ?

Diagnostic / Déboguage

- existence des erreurs + lien avec l'arithmétique flottante
- quantification de l'ampleur du problème
- localisation de l'origine dans le code source

Contexte & enjeux

Quels Besoins ?

Diagnostic / Déboguage

- existence des erreurs + lien avec l'arithmétique flottante
- quantification de l'ampleur du problème
- localisation de l'origine dans le code source

Correction

- solutions adaptées aux différentes sources d'erreurs possibles
- sans (trop de) perte de performance
- vérifiables

Contexte & enjeux

Quels Besoins ?

Diagnostic / Déboguage

- existence des erreurs + lien avec l'arithmétique flottante
- quantification de l'ampleur du problème
- localisation de l'origine dans le code source

Correction

- solutions adaptées aux différentes sources d'erreurs possibles
- sans (trop de) perte de performance
- vérifiables

Plan

1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
4. Conclusions – perspectives

Diagnostic

1. Contexte & enjeux

2. Diagnostic
   - Tour d'horizon
   - MCA & CESTAC
   - Verrou
   - Limites
   - Application : code_aster

3. Localisation & correction

4. Conclusions – perspectives

Méthodes de diagnostic

Tour d'horizon

"How futile are Mindless Assessments of Roundoff in F-P Computation?" [Kah06]

0. analyse mathématique rigoureuse
1. comparaison à un calcul en précision supérieure
2. **comparaison à des calculs arrondis différemment** (4 modes IEEE-754)
3. **analyse statistique de calculs arrondis aléatoirement**
4. analsye statistique des calculs avec données d'entrées perturbées
5. reprise du calcul en arithmétique d'intervalles

Méthodes de diagnostic

Comparaison à un calcul en précision supérieure

ex : MPFR, gmpy

- non garanti (*cf* contre-exemple de Rump ci-dessous)
- marche très bien en pratique
- facile à mettre en place (selon langage et âge du code)
- devient coûteux quand on dépasse la double précision

Pas toujours valide [Rum88]:

$$333.75\,y^6 + x^2(11\,x^2\,y^2 - y^6 - 121\,y^4 - 2) + 5.5\,y^8 + \frac{x}{2y}$$

pour $x = 77617$ et $y = 33096.$

simple précision	$\rightarrow$	$1.172603\ldots$
double précision	$\rightarrow$	$1.1726039400531\ldots$
précision étendue	$\rightarrow$	$1.172603940053178\ldots$
calcul exact	$\rightarrow$	$0.827396059946821 + [3;4] \times 10^{-16}$

Méthodes de diagnostic

Perturbation des données d'entrée

- lien fort avec la quantification d'incertitudes $\rightarrow$ outils disponibles
- peu fiable sans connaissance prélable du problème
  - quel est le conditionnement du système ?
- certaines méthode spécifiques peuvent être mises en place
  - ex: sensibilité à la numérotation du maillage

Arithmétique d'intervalles

ex : MPFI, pyintervals, IntervalArithmetic.jl

idée: $x \rightarrow [\underline{x}, \overline{x}]$ $\quad x - y = \left[\underline{x} - \overline{y},\ \overline{x} - \underline{y}\right]$

- résultat garanti... pour le jeu de données testé
- ...mais souvent très large
- pas compatible avec tous les algorithmes (ex : Newton)

Diagnostic
MCA & CESTAC

1. Contexte & enjeux

2. Diagnostic
   - Tour d'horizon
   - MCA & CESTAC
   - Verrou
   - Limites
   - Application : code_aster

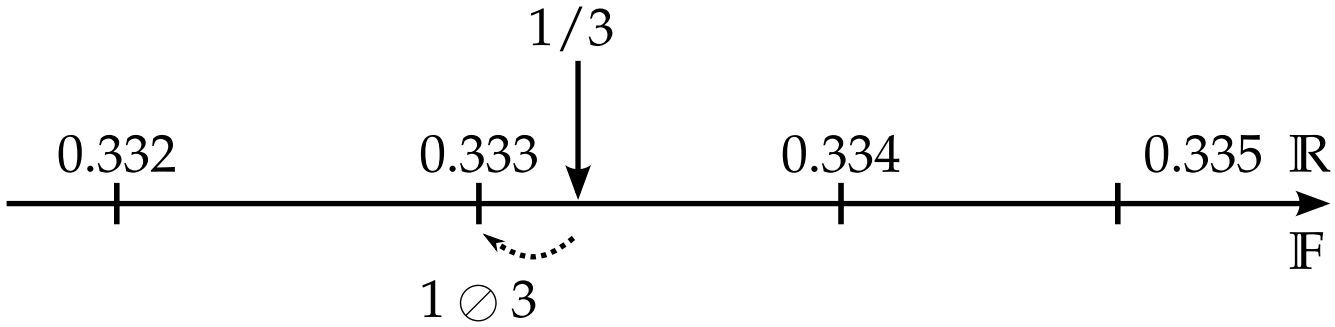
3. Localisation & correction

4. Conclusions – perspectives

Méthodes CESTAC & MCA

Modéliser l'imprécision par un aléa sur le mode d'arrondi

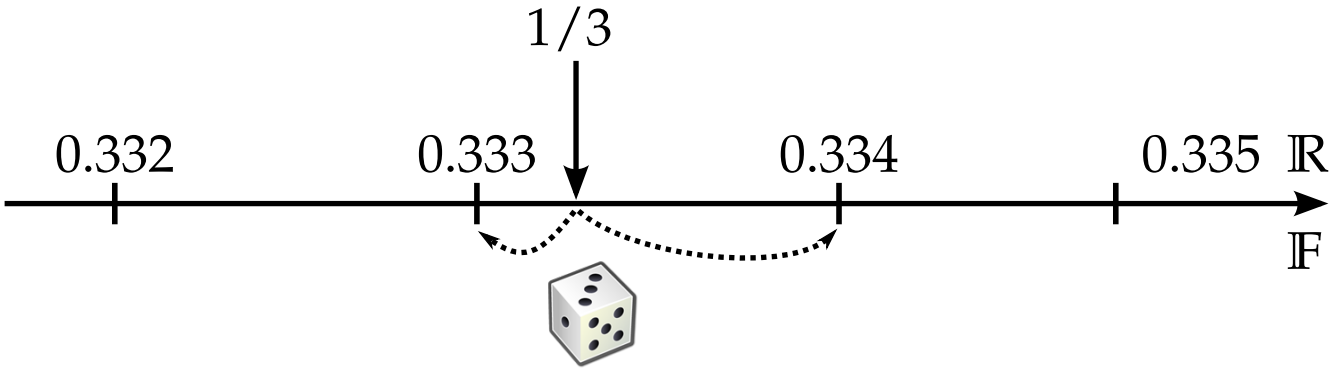
Arrondi au plus proche (défaut IEEE-754)



Méthodes CESTAC & MCA

Modéliser l'imprécision par un aléa sur le mode d'arrondi

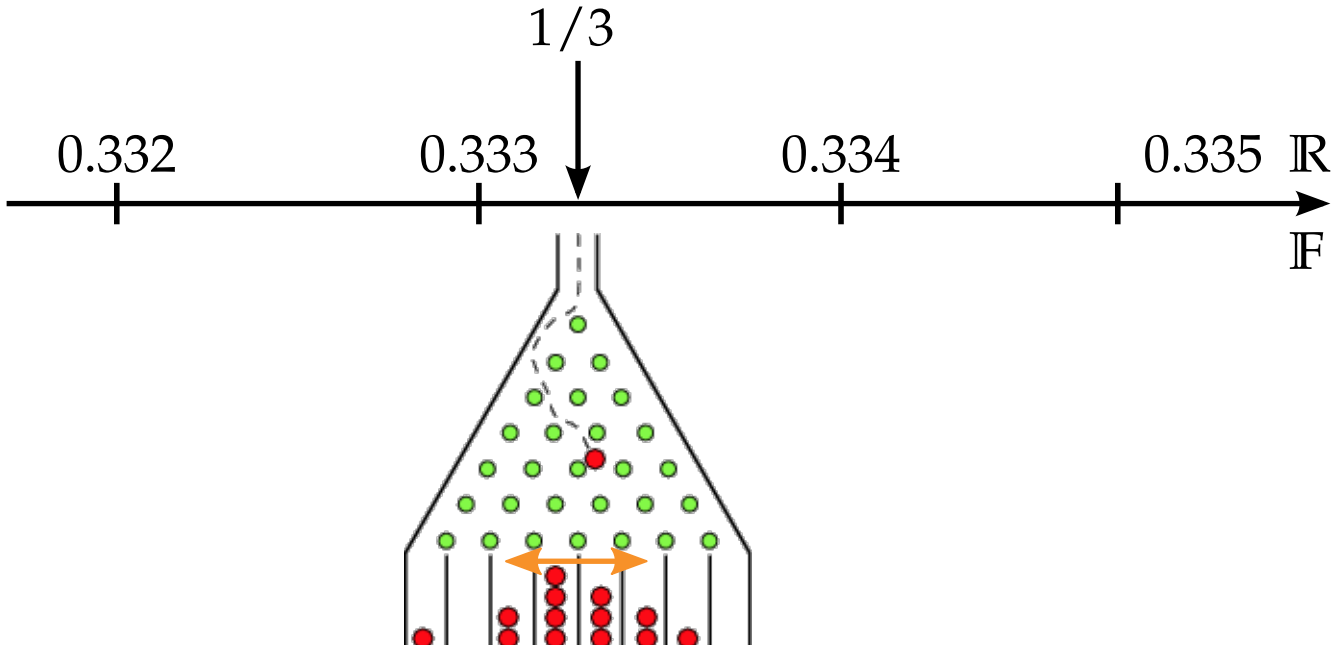
Arrondi aléatoire (CESTAC [Vig93, CV88])



Méthodes CESTAC & MCA

Modéliser l'imprécision par un aléa sur le mode d'arrondi

Arrondi aléatoire (CESTAC [Vig93, CV88])



Instruction	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**0.33**4
$b = a \times 3$	0.999	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.00**

Méthodes CESTAC & MCA

Synchrone *vs* asynchrone

Évaluation :

- synchrone (ex: CADNA)
- asynchrone (ex: Verrou)

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$				
$b = a \times 3$				
if $b > 1$ then				
$\quad b = 1$				
end				
$r = \mathrm{asin}(b)$				

Méthodes CESTAC & MCA

Synchrone *vs* asynchrone

Évaluation :

- **synchrone (ex: CADNA)**
- asynchrone (ex: Verrou)

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**3.34e-1**
$b = a \times 3$				
if $b > 1$ then				
$b = 1$				
end				
$r = \mathrm{asin}(b)$				

Méthodes CESTAC & MCA

Synchrone *vs* asynchrone

Évaluation :

- **synchrone (ex: CADNA)**
- asynchrone (ex: Verrou)

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**3.34e-1**
$b = a \times 3$	$0.999_{\downarrow}$	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.00**
if $b > 1$ then	———	False	———	**Warning**
$b = 1$				
end				
$r = \mathrm{asin}(b)$	$1.53^{\uparrow}$	$1.58^{\uparrow}$	×	×

Méthodes CESTAC & MCA

Synchrone *vs* asynchrone

Évaluation :

- synchrone (ex: CADNA)
- **asynchrone (ex: Verrou)**

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$			
$b = a \times 3$	$0.999_{\downarrow}$			
if $b > 1$ then	False			
$b = 1$				
end				
$r = \mathrm{asin}(b)$	$1.53^{\uparrow}$			

Méthodes CESTAC & MCA

Synchrone *vs* asynchrone

Évaluation :

- synchrone (ex: CADNA)
- **asynchrone (ex: Verrou)**

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	
$b = a \times 3$	$0.999_{\downarrow}$	$1.00_{\downarrow}$	$1.01^{\uparrow}$	
if $b > 1$ then	False	False	True	
$b = 1$			1.00	
end				
$r = \mathrm{asin}(b)$	$1.53^{\uparrow}$	$1.58^{\uparrow}$	$1.57^{\downarrow}$	
print r				**1.5**6

Méthodes CESTAC & MCA

Tests instables : exemple de MAAP

- Entrées:

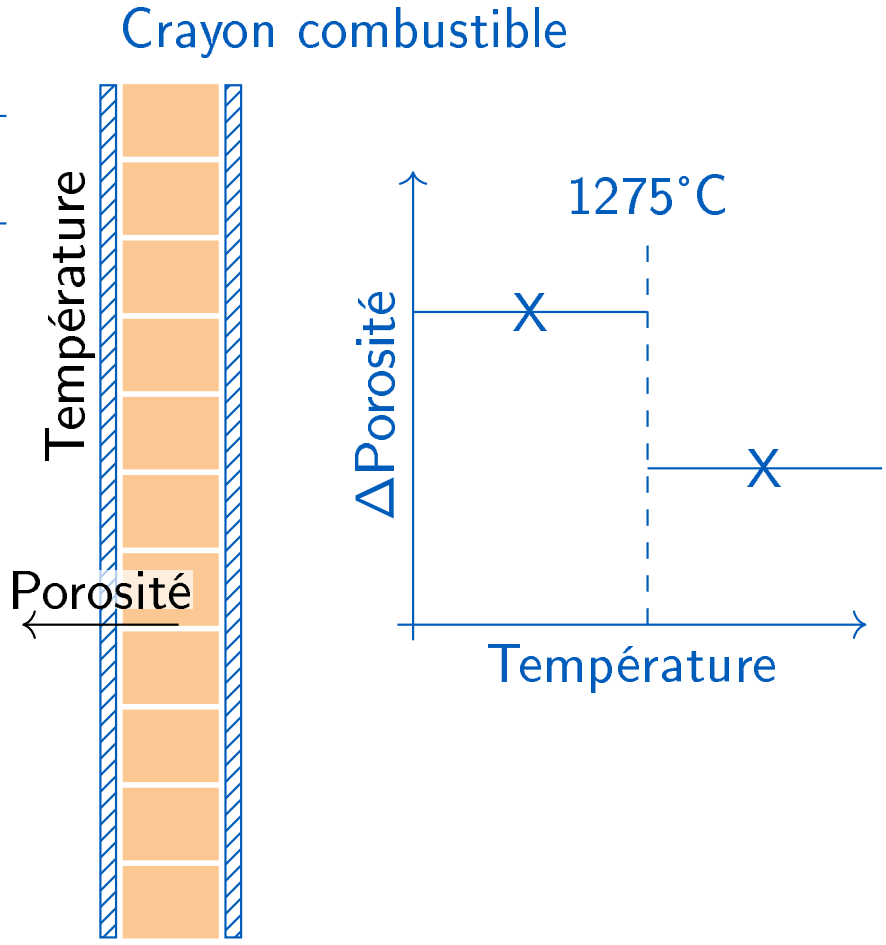
	Eval. 1	Eval. 2	δ_{rel}
Température	**1275**.2	1274.8	$\sim 10^{-4}$
Porosité	**0.42382**	0.42383	$\sim 10^{-5}$

- Algorithme:

```
1 if T <= 1275.:
2   por += 0.032
3
4 elif T <= 1400.:
5   por += 0.019
6
7 # ...
```

- Sorties :

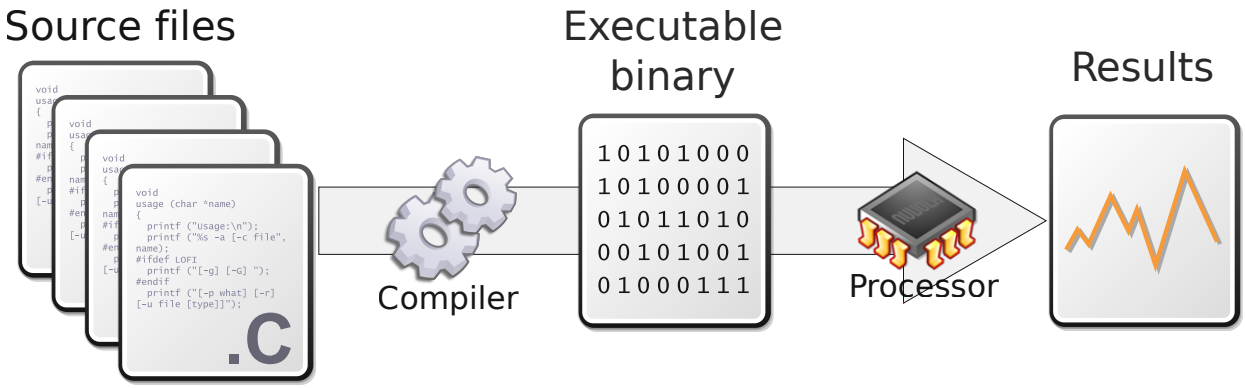
Porosité	**0.4**4282	0.45583	$\sim 10^{-2}$



Méthodes CESTAC & MCA : Outils

CADNA : analyse dynamique par les sources [JCL10, LCJ10]

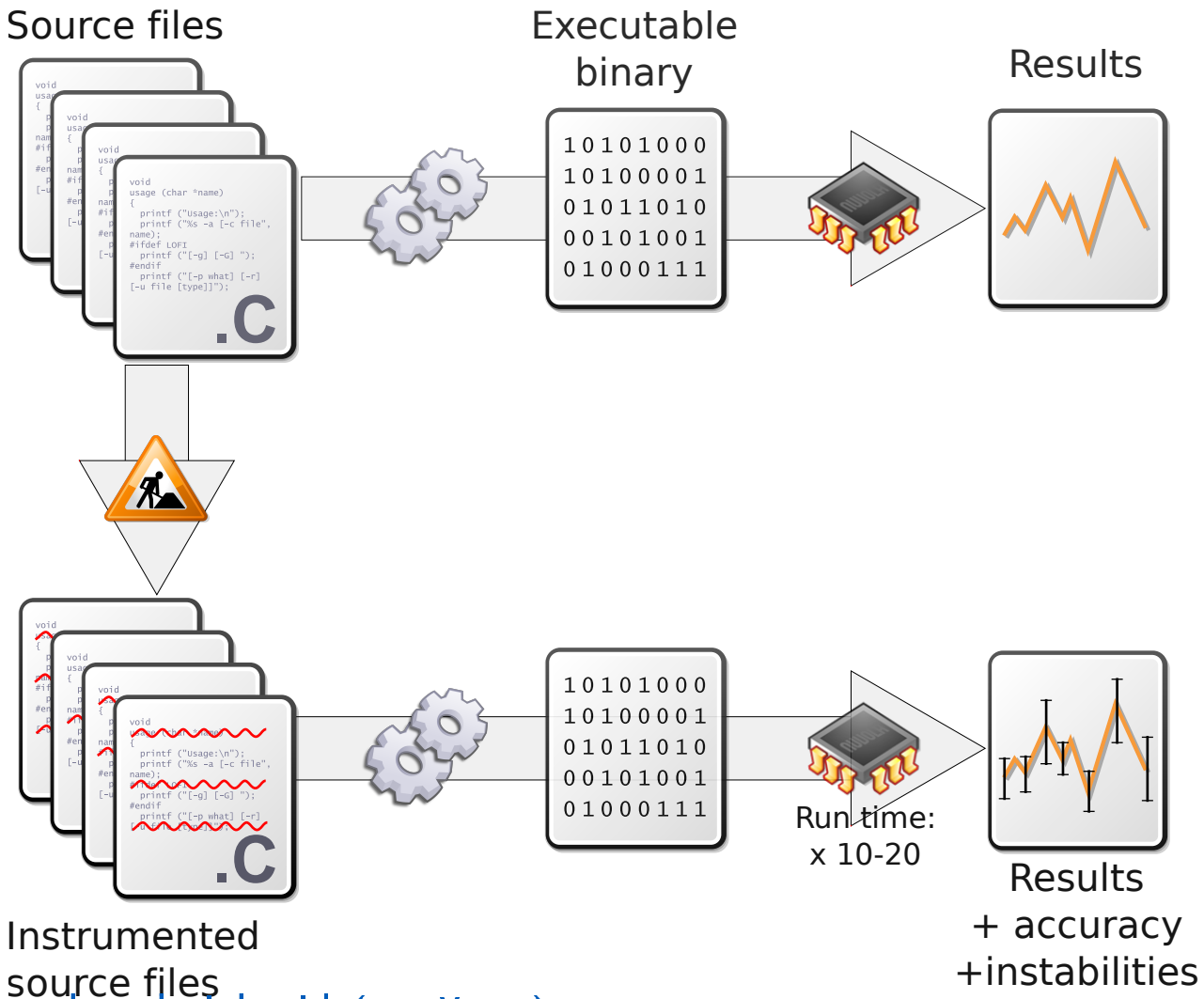
```
$ myProg in out
```



Méthodes CESTAC & MCA : Outils

CADNA : analyse dynamique par les sources [JCL10, LCJ10]

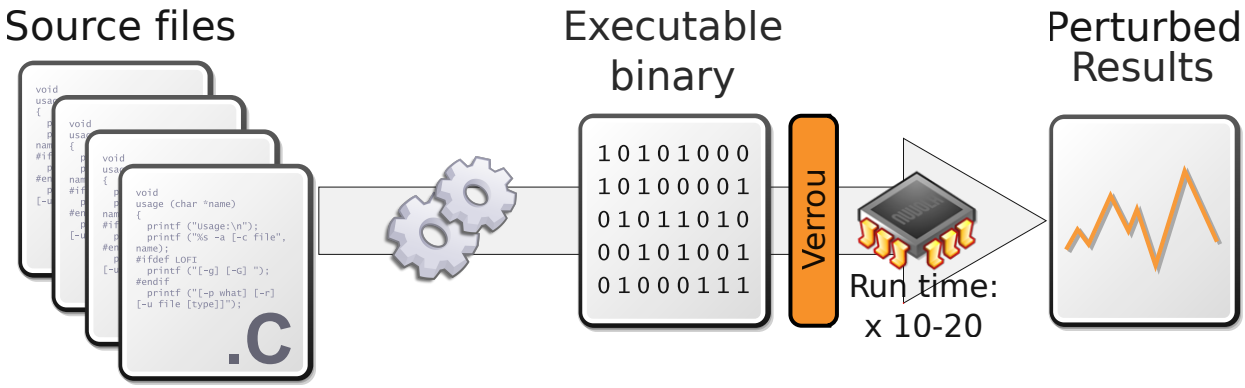
$ myProg-cadna `in out`



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

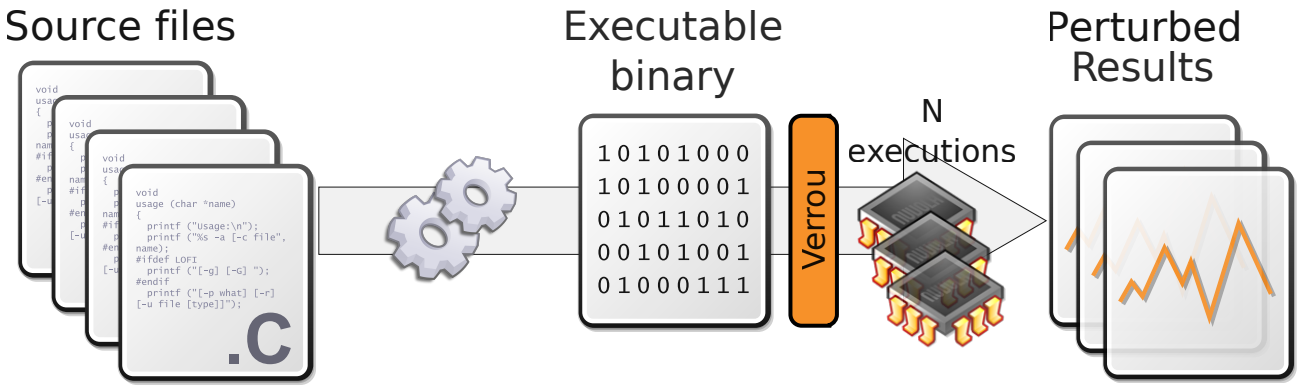
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
```



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

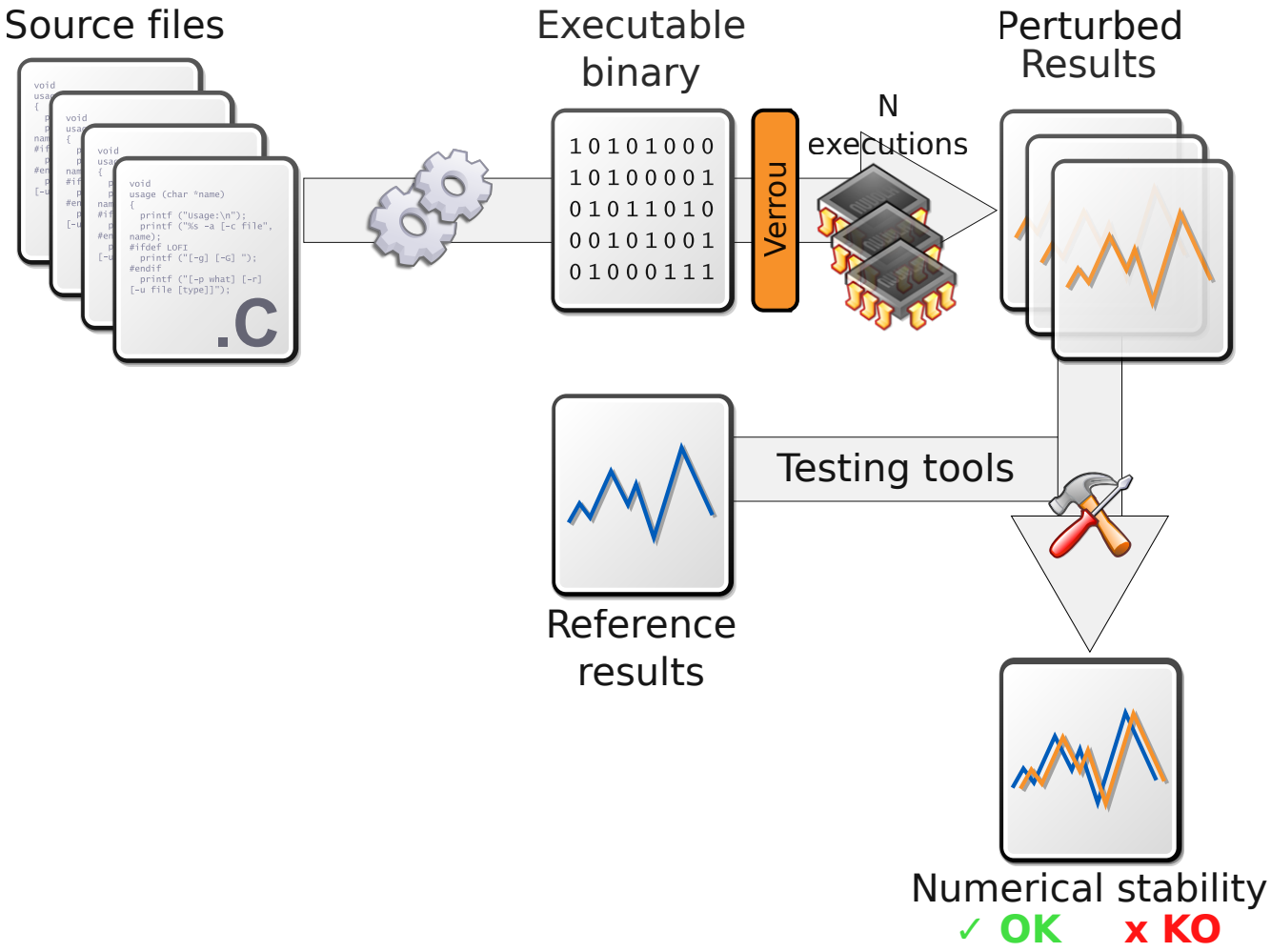
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

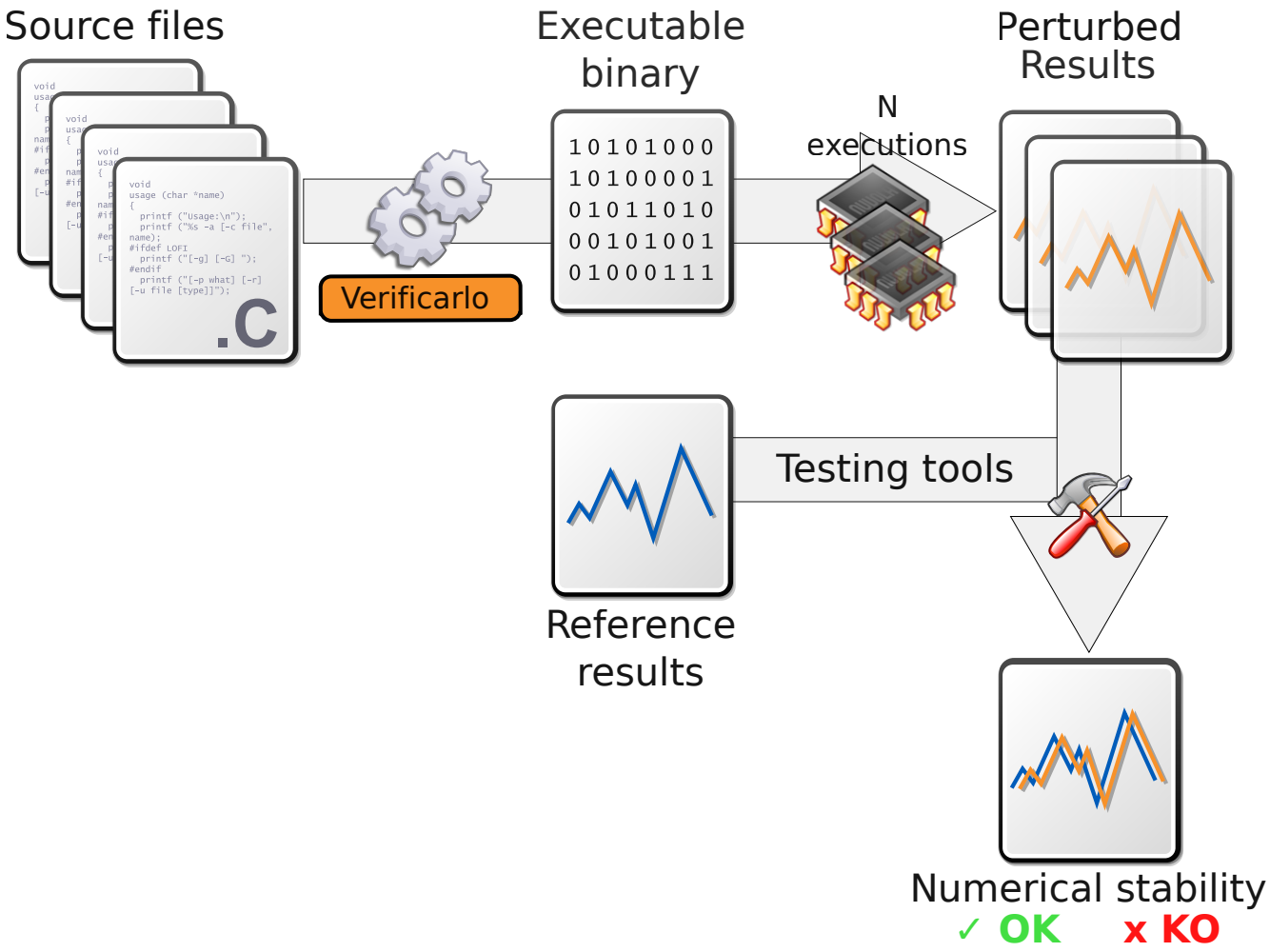
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



Méthodes CESTAC & MCA : Outils

Verificarlo : analyse dynamique de la représentation intermédiaire (IR) [DdOCP16]

```
$ myProg-verificarlo in out1
$ myProg-verificarlo in out2
```



Méthodes CESTAC & MCA : Outils

Architecture

Front-end : comment remplacer les opérations flottantes ?

Interface utilisateur

- CADNA : instrumentation des sources → bibliothèque
- Verificarlo : passe de compilation LLVM → compilateur
- Verrou : outil Valgrind → environnement d'exécution

Back-end : par quoi remplacer les opérations flottantes ?

Modèle d'arithmétique

- CADNA : CESTAC synchrone (DSA, *Discrete Stochastic Arithmetic*)
- Verificarlo : MCA (*full*, *Random Rounding*)
- Verrou : CESTAC asynchrone, MCA (RR)

Analyse statistique

- CADNA : analyse en ligne, intégrée dans l'outil
- Verificarlo & Verrou : post-traitement laissé à la charge de l'utilisateur

Diagnostic
Verrou

1. Contexte & enjeux

2. Diagnostic
   - Tour d'horizon
   - MCA & CESTAC
   - Verrou
   - Limites
   - Application : code_aster

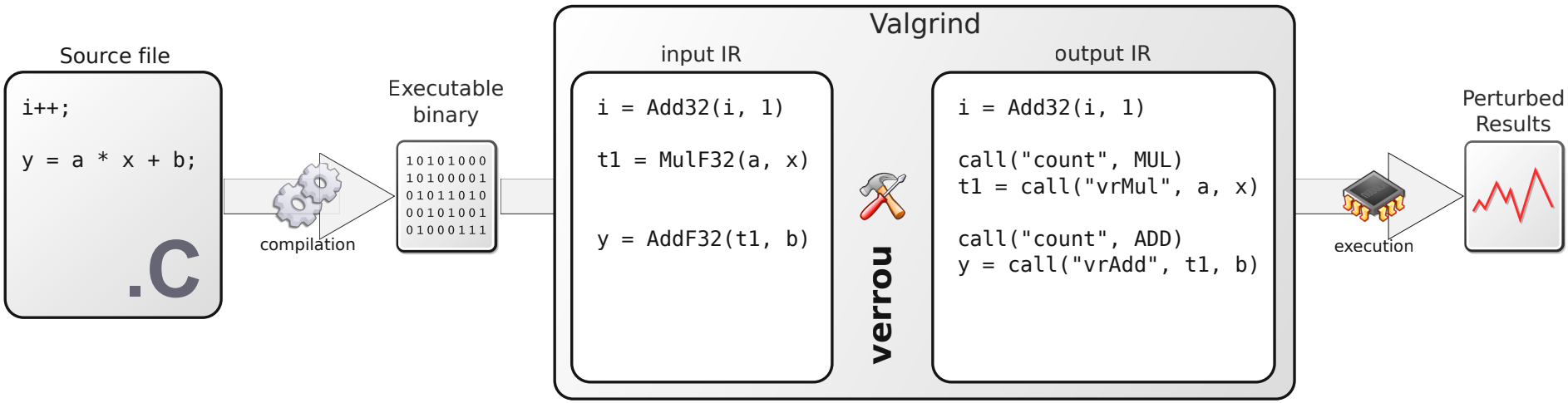
3. Localisation & correction

4. Conclusions – perspectives

L'outil Verrou

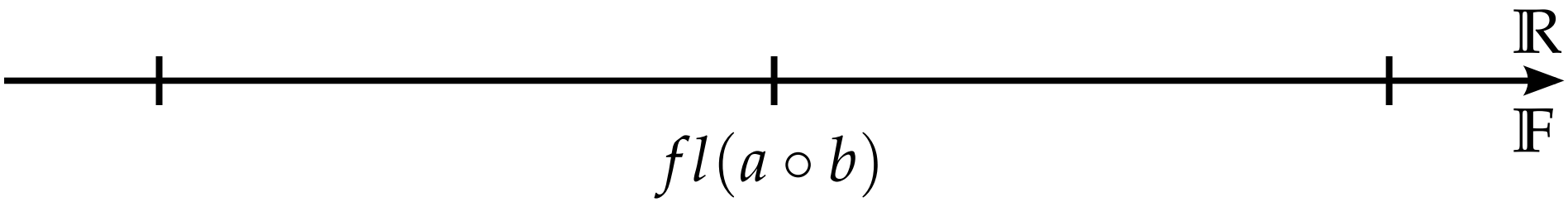
Front-end : Analyse dynamique du binaire avec Valgrind

```
$ valgrind --tool=verrou --rounding-mode=random PROGRAM [ARGS...]
```



L'outil Verrou

Back-end : Simulation des modes d'arrondi

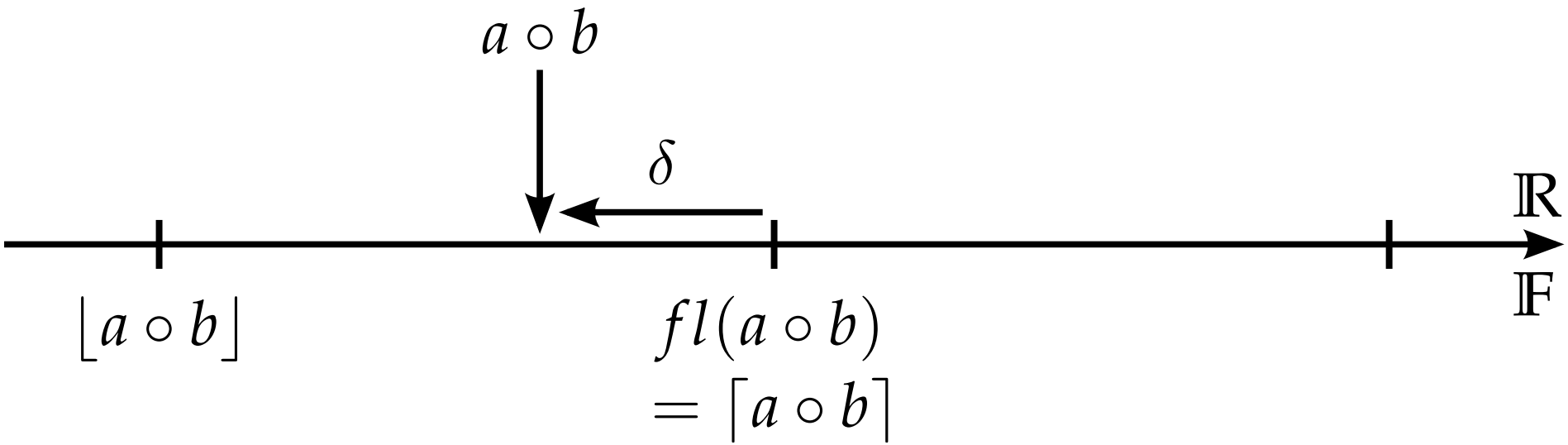


- Transformation exacte
  (la division est un peu plus compliquée):
  - $a \circ b = \sigma + \delta$,
  - $\sigma = fl(a \circ b)$

- Si $\delta < 0$:
  - $\lfloor a \circ b \rfloor = fl(a \circ b) - ulp$,
  - $\lceil a \circ b \rceil = fl(a \circ b)$.

L'outil Verrou

Back-end : Simulation des modes d'arrondi

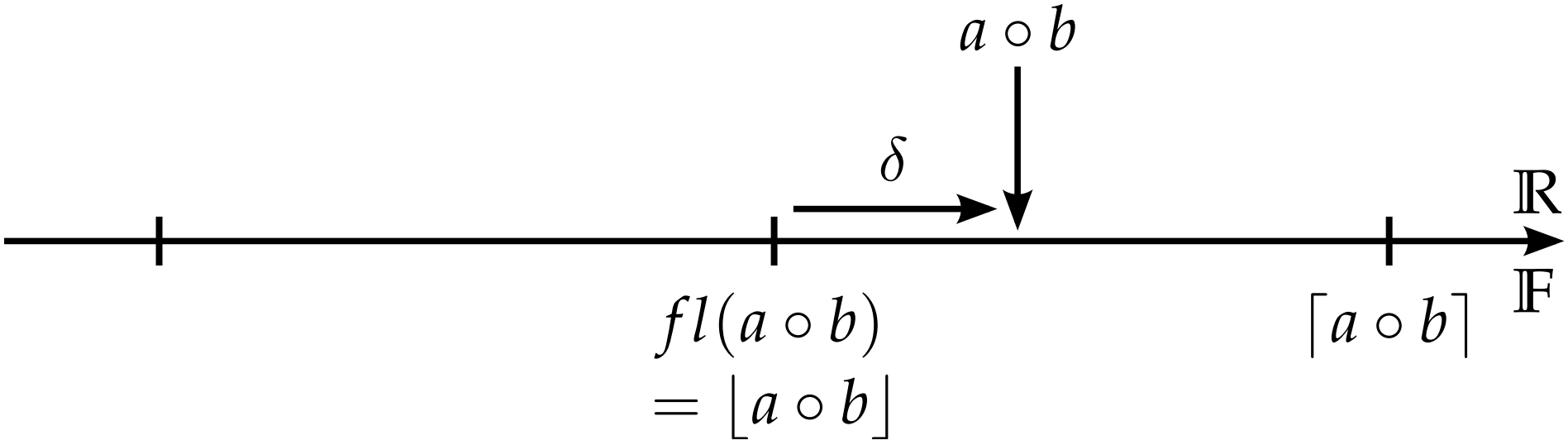


- Transformation exacte
  (la division est un peu plus compliquée):
  - $a \circ b = \sigma + \delta$,
  - $\sigma = fl(a \circ b)$

- Si $\delta > 0$:
  - $\lfloor a \circ b \rfloor = fl(a \circ b)$,
  - $\lceil a \circ b \rceil = fl(a \circ b) + ulp$.

L'outil Verrou

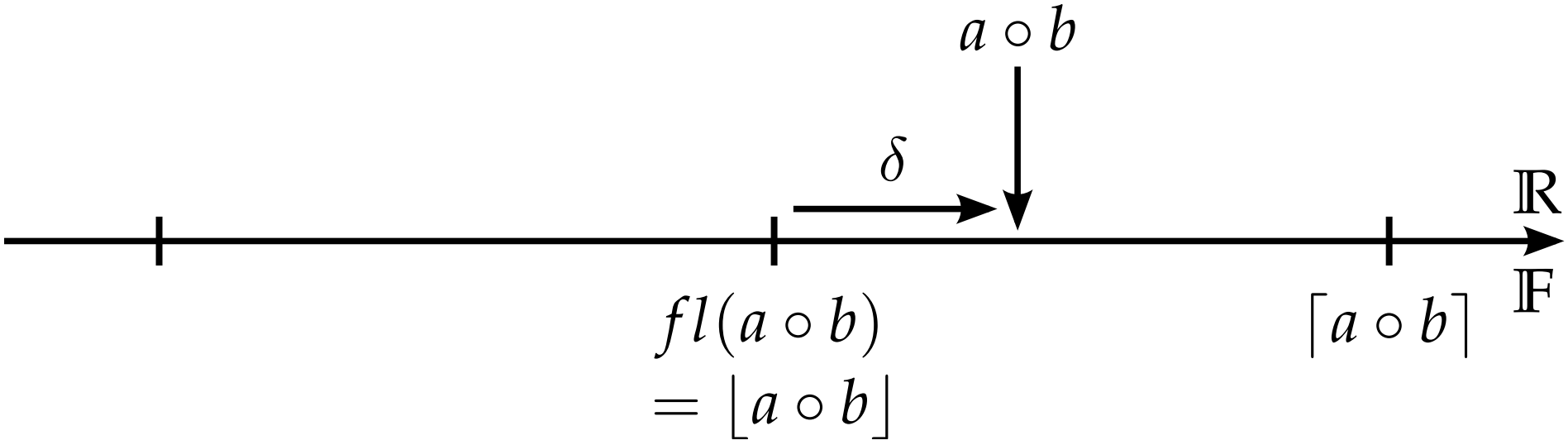
Back-end : Simulation des modes d'arrondi



- Transformation exacte
  (la division est un peu plus compliquée):
  - $a \circ b = \sigma + \delta$,
  - $\sigma = fl(a \circ b)$
- Si $\delta 0$:
  - $\lfloor a \circ b \rfloor = fl(a \circ b)$,
  - $\lceil a \circ b \rceil = fl(a \circ b)$.

L'outil Verrou

Back-end : Simulation des modes d'arrondi



- `average` : arrondi "moyen"
  - $p(\lceil a \circ b \rceil) = \dfrac{\delta}{ulp}$ (si $\delta > 0$)
  - $p(\lfloor a \circ b \rfloor) = \dfrac{ulp - \delta}{ulp}$

L'outil Verrou

Analyse statistique

- Laissée à la charge de l'utilisateur (qui fait ce qu'il veut)

- A minima : comparaison des statuts (OK/KO) de la suite de cas-tests
  - en pratique : 5-10 *runs* max permettent de "stimuler" l'apparition d'un problème
  - beaucoup plus de *runs* nécessaires pour avoir des garanties

- Post-traitement conseillé :
  - N *runs* produisant les résultats $X_i, i = 1 \dots N$
  - moyenne empirique $\hat{\mu}$
  - écart-type empirique $\hat{\sigma}$
  - estimateur du nombre de chiffres significatifs (décimaux) fiables [SP97]:

$$\hat{s}_{\text{mca}} = -\log_{10} \frac{\hat{\sigma}}{|\hat{\mu}|} \qquad \text{ou} \qquad \hat{s}_{\text{mca}} = -\log_{10} \hat{\sigma}$$

Diagnostic
Limites

1. Contexte & enjeux

2. Diagnostic
   - Tour d'horizon
   - MCA & CESTAC
   - Verrou
   - Limites
   - Application : code_aster

3. Localisation & correction

4. Conclusions – perspectives

Limites de CESTAC / MCA

Analyse dynamique

Analyse dynamique ⇒ dépendante du cas testé

- Multiplier les cas-tests
- Utile pour détecter les problèmes (un contre-exemple suffit)
- Utiliser une technique complémentaire pour vérifier la correction

Analyse statistique ⇒ dépendante des tirages aléatoires

- Quel niveau de confiance avoir dans les résultats ?
- Détection de problème
  - petit nombre de tirages
  - assiette de cas aussi larges que possible
- Vérification de la correction :
  - grand nombre de tirages
  - sur les cas problématiques

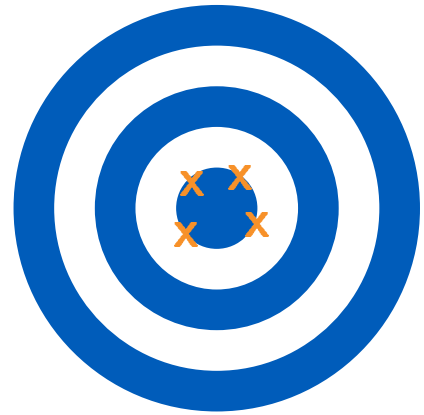
Limites de CESTAC / MCA

Précision... du vocabulaire

(*cf.* métrologie, ISO-5725)

fidélité
(*precision*)

justesse
(*trueness*)



exactitude
(*accuracy*)

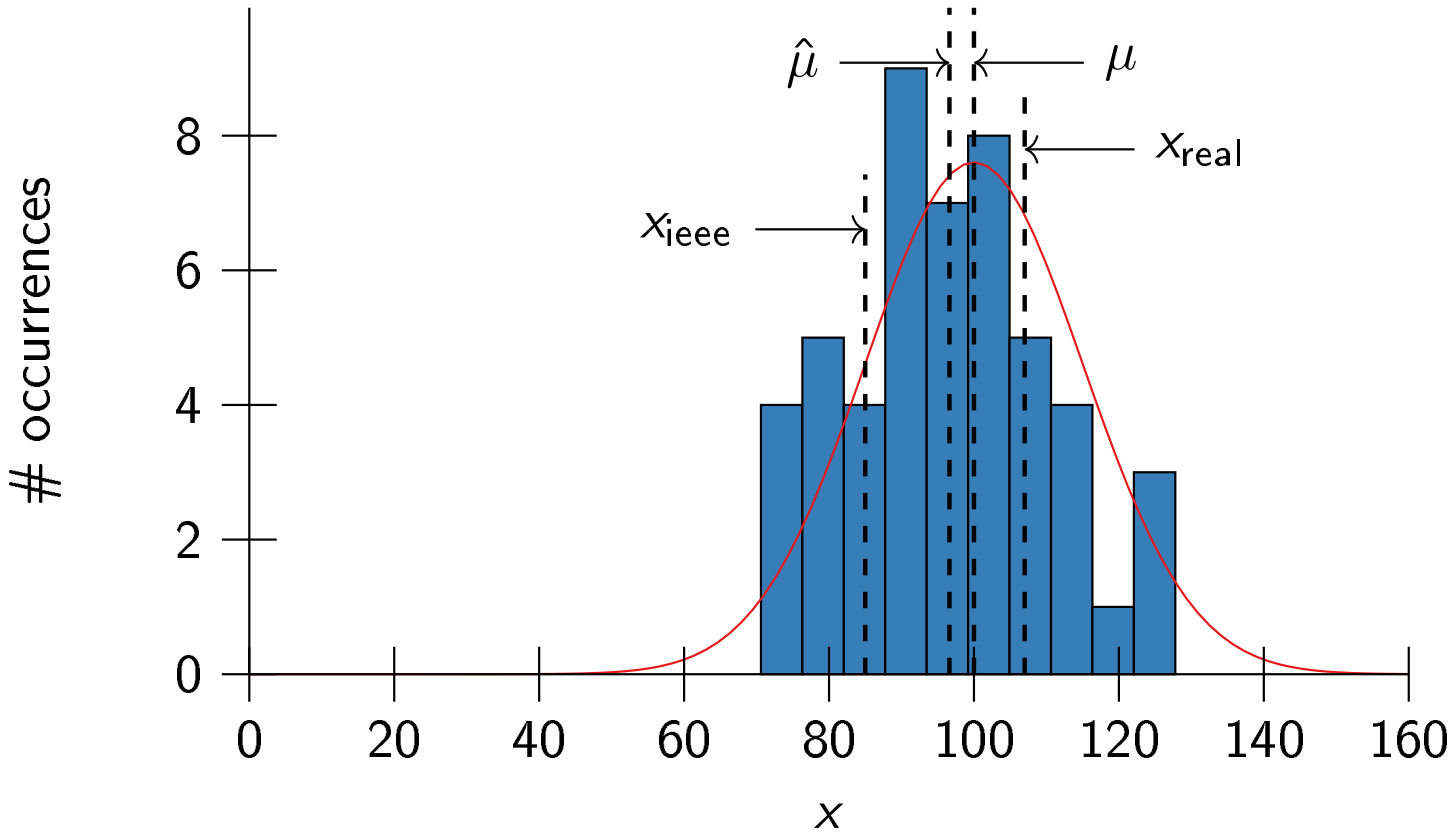
résolution

Objectifs de la vérification numérique

- Déterminer les chiffres significatifs **exacts**

Limites de CESTAC / MCA

Analyse statistique : ne pas confondre exactitude et fidélité



- x_{real} : résultat du calcul exact
- x_{ieee} : résultat du calcul flottant
- μ : espérance distribution
- $\hat{\mu}$: moyenne empirique

- la distribution des résultats est-elle gaussienne ? (Hyp CESTAC)
- biais systématiques : $\mu = x_{real}$?
- x_{ieee} est-il un événement rare ?

Limites de CESTAC / MCA

Algorithmes spécifiques prenant l'arithmétique flottante en compte

```
$ valgrind --tool=verrou --rounding-mode=random  python
```

```
» import math
» math.cos(42.)
-4.5847217124585136
» math.cos(42.)
-4.6689026578736614
» math.cos(42.)
-0.39998531498835133
```

Limites de CESTAC / MCA

Algorithmes spécifiques prenant l'arithmétique flottante en compte

```
$ valgrind --tool=verrou --rounding-mode=random  \
          --exclude=libmath.exclude python
```

```
» import math
» math.cos(42.)
==17509== Using exclusion rule: * /lib/libm-2.11.3.so
-0.39998531498835127
» math.cos(42.)
-0.39998531498835127
» math.cos(42.)
-0.39998531498835127
```

- Fichier `libmath.exclude`:

```
#sym   lib
*      /lib/libm-2.11.3.so
```

Diagnostic

Application : code_aster

1. Contexte & enjeux

2. Diagnostic
   - Tour d'horizon
   - MCA & CESTAC
   - Verrou
   - Limites
   - Application : code_aster

3. Localisation & correction

4. Conclusions – perspectives

Application : code_aster

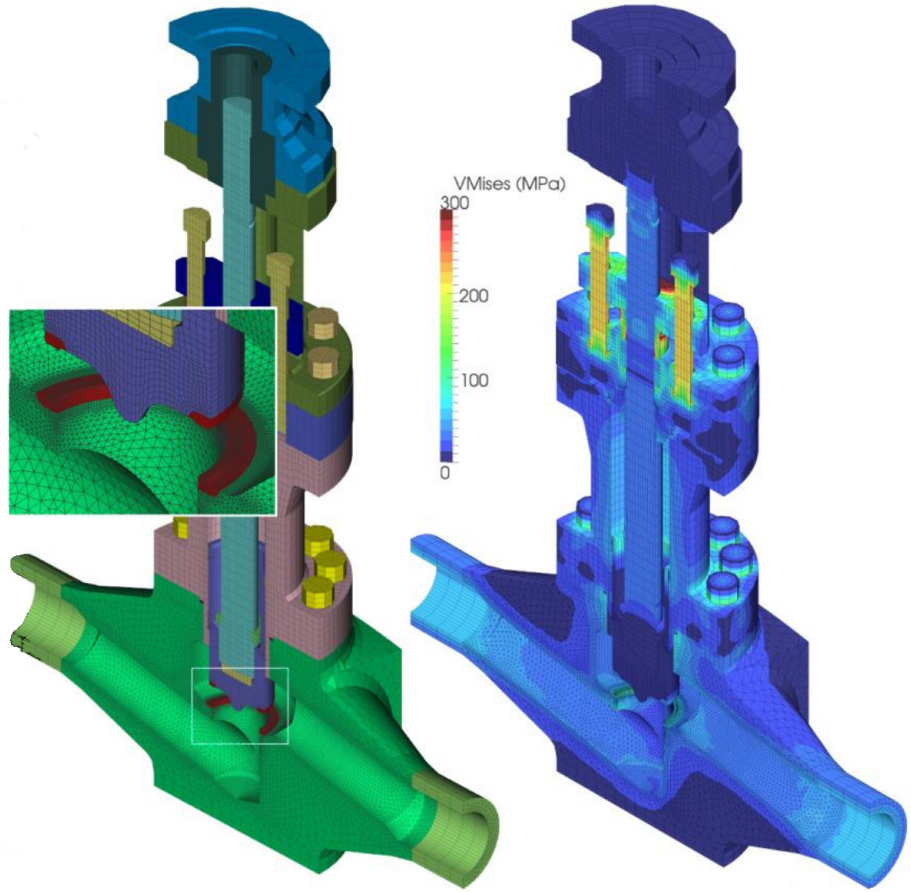
description générale

Mécanique au sens large

- Sismique
- Acoustique
- Thermique

Code_Aster

- 1.2M lignes de code
- Fortran 90, C, Python
- Nombreuses dépendances :
  - solveurs linéaires (MUMPS...)
  - mailleurs et partitionneurs (Metis, Scotch...)



Objectifs de l'étude

- investiguer des problèmes de non-reproductibilité entre machines de test

Application : code_aster

Travaux préliminaires

(étude complète détaillée dans [FL17])

Sélection de 72 cas-tests

- certains (considérés comme) stables
- certains (connus pour être) instables

"Méta-vérification": vérification du processus de vérification lui-même

- Verrou pourrait avoir des *bugs*
- Problème des "Heisenbugs": ils disparaissent lors de l'instrumentation
  - introspection
  - instructions matérielles problématiques (ex : jeu d'instructions `x87`)
  - algorithmes spécifiques conscients des arrondis

Application : code_aster

Travaux préliminaires

(étude complète détaillée dans [FL17])

Sélection de 72 cas-tests

- certains (considérés comme) stables
- certains (connus pour être) instables

"Méta-vérification": vérification du processus de vérification lui-même

- Verrou **avait un *bug***, pourrait en avoir d'autres
- Problème des "Heisenbugs": ils disparaissent lors de l'instrumentation
  - introspection
  - instructions matérielles problématiques (ex : jeu d'instructions `x87`)
  - algorithmes spécifiques conscients des arrondis

Application : code_aster

Analyse des instabilités numériques : Verrou en arrondi aléatoire

Cas test	nearest
ssls108i	OK
ssls108j	OK
ssls108k	OK
ssls108l	OK
sdnl112a	OK
ssnp130a	OK
ssnp130b	OK
ssnp130c	OK
ssnp130d	OK

Application : code_aster

Analyse des instabilités numériques : Verrou en arrondi aléatoire

Cas test	nearest	rnd_1
ssls108i	OK	OK
ssls108j	OK	OK
ssls108k	OK	OK
ssls108l	OK	OK
sdnl112a	OK	KO
ssnp130a	OK	OK
ssnp130b	OK	OK
ssnp130c	OK	OK
ssnp130d	OK	OK

Application : code_aster

Analyse des instabilités numériques : Verrou en arrondi aléatoire

Cas test	Statut nearest	rnd_1	rnd_2	rnd_3
sslsl08i	OK	OK	OK	OK
sslsl08j	OK	OK	OK	OK
sslsl08k	OK	OK	OK	OK
sslsl08l	OK	OK	OK	OK
sdnl112a	OK	KO	KO	KO
ssnp130a	OK	OK	OK	OK
ssnp130b	OK	OK	OK	OK
ssnp130c	OK	OK	OK	OK
ssnp130d	OK	OK	OK	OK

10 minutes

20 minutes chacun

(72 cas tests)

Application : code_aster

Analyse des instabilités numériques : Verrou en arrondi aléatoire

Cas test	Statut nearest	rnd_1	rnd_2	rnd_3	# chiffres décimaux en commun $C(rnd_1, rnd_2, rnd_3)$
ssls108i	OK	OK	OK	OK	11 10
ssls108j	OK	OK	OK	OK	10 10
ssls108k	OK	OK	OK	OK	11 10
ssls108l	OK	OK	OK	OK	10 9
sdnl112a	OK	KO	KO	KO	6 6 6 * 3 (0 expected)
ssnp130a	OK	OK	OK	OK	* * 10 10 10 10 9 * * * 9 9 9 9 *
ssnp130b	OK	OK	OK	OK	* * 11 11 * 12 9 * * * 9 9 9 9 9
ssnp130c	OK	OK	OK	OK	* 11 11 11 11 10 9 11 11 10 10
ssnp130d	OK	OK	OK	OK	* 9 * * * 10 9 9 9 9 9 9 9 * 9 *

10 minutes (72 cas tests)

20 minutes chacun

$$C(x) = \log_{10} \left| \frac{\mu(x)}{\sigma(x)} \right|$$

Localisation & correction

1. Contexte & enjeux

2. Diagnostic

3. Localisation & correction
   - Tour d'horizon
   - Couverture de code
   - Bisection

4. Conclusions – perspectives

Localisation & correction

Tour d'horizon

1. Contexte & enjeux

2. Diagnostic

3. Localisation & correction
   - Tour d'horizon
   - Couverture de code
   - Bisection

4. Conclusions – perspectives

Localisation

Tour d'horizon

Détection synchrone

Suivi en ligne de l'erreur / écart et de sa croissance

- arithmétique d'intervalles
- arithmétique stochastique discrète (ex : CADNA)
- exécution *"shadow"* en précision supérieure (ex : fpDebug, CRAFT, Herbgrind)

Détection asynchrone

Autopsie, analyse *post mortem*

- ajout de sorties intermédiaires + post-traitement (ex : Verificarlo + Veritracer)
- **comparaison de couvertures d'exécutions** (ex : Verrou + gprof)
- **delta-debugging** (ex : Verrou)

Correction

Tests instables

- Attention aux tests d'égalité de nombres flottants
  - il est rare qu'un test sans tolérance soit correct
  - comment dimensionner la tolérance ? ($\rightarrow$ estimation MCA)

- Prendre du recul : il s'agit parfois d'une fonction "élémentaire" continue
  - Exemple : maximum des éléments d'un vecteur : $m = \max_i x_i$

```
double m = x[0];
for (int i=1 ; i<N ; ++i)
  if (x[i] > m) // <--Instabilité sans conséquence
    m = x[i];
return m;
```

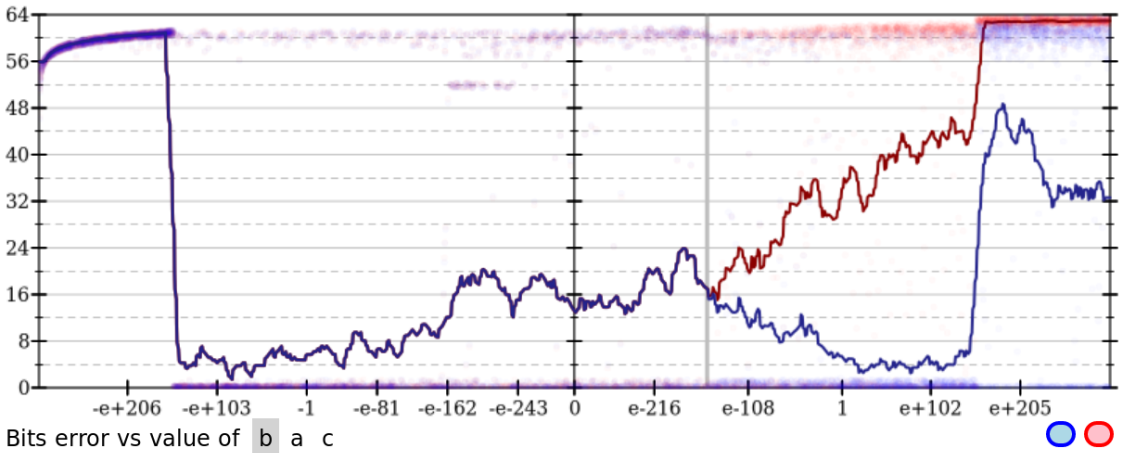
Correction

Formules inexactes

- Beaucoup de formules trouvées dans les livres sont écrites pour simplifier la lecture...
  - pas pour optimiser la justesse du résultat !
  - l'outil Herbie peut aider à récrire la formule de manière optimale

- Exemple : racines du trinôme $a\,x^2 + b\,x + c$

$$x^{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,a\,c}}{2\,a}$$



$$x^{+} = \begin{cases} \dfrac{-b + \sqrt{b^2 - 4\,a\,c}}{2\,a} & \text{si } b \leqslant 0 \\ \dfrac{2\,c}{-b - \sqrt{b^2 - 4\,a\,c}} & \text{sinon} \end{cases}$$

```
(-b + sqrt(b^2 - 4*a*c))/(2*a)
```

Correction

Algorithmes classiques

- Pour toute une classe d'algorithmes classiques, il existe des variantes "compensées" :
  - somme des éléments d'un vecteur
  - produit scalaire
  - évaluation polynômiale
  - ...

- fondées sur des transformations exactes (*Error Free Transformation*, EFT) :

$$x \underset{\text{eft}}{\circ} y = (r, \delta)$$

tels que

$$r = \lfloor x \circ y \rceil$$
$$r + \delta = x \circ y$$

ex : LibEFT implémente

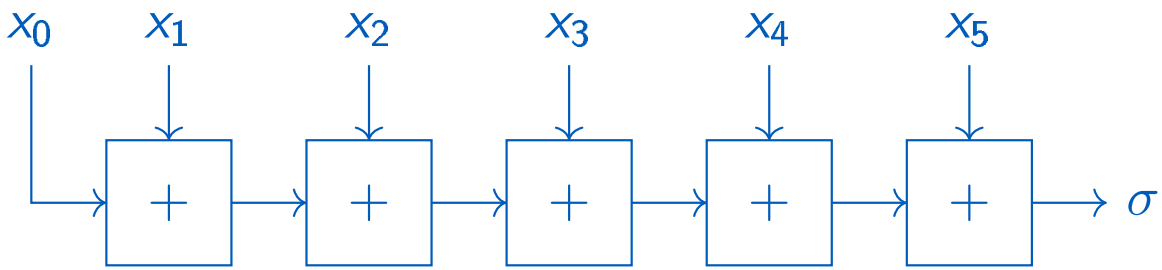
- des transformations exactes
- pour $+$, $-$ et $\times$
- en simple et double précision

Correction

Algorithmes compensés, exemple : somme des éléments d'un vecteur

Sum(x) : algorithme naïf

$\sigma \leftarrow x_0$;
for $i = 1 \ldots n-1$ **do**
    $\sigma \leftarrow \sigma + x_i$;
end
return σ ;



Correction

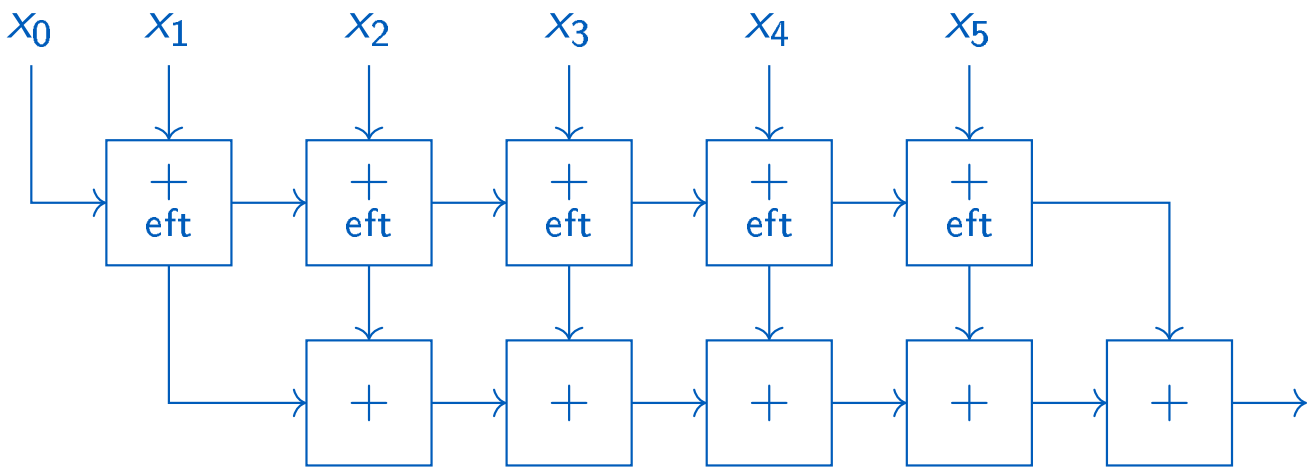
Algorithmes compensés, exemple : somme des éléments d'un vecteur

CompSum(x) : algorithme compensé

$\sigma \leftarrow x_0$;
$\pi \leftarrow 0$;
for $i = 1 \ldots n-1$ **do**
    $(\sigma, e) \leftarrow \sigma \underset{\text{eft}}{+} x_i$;
    $\pi \leftarrow \pi + e$;
end
return $\sigma + \pi$;

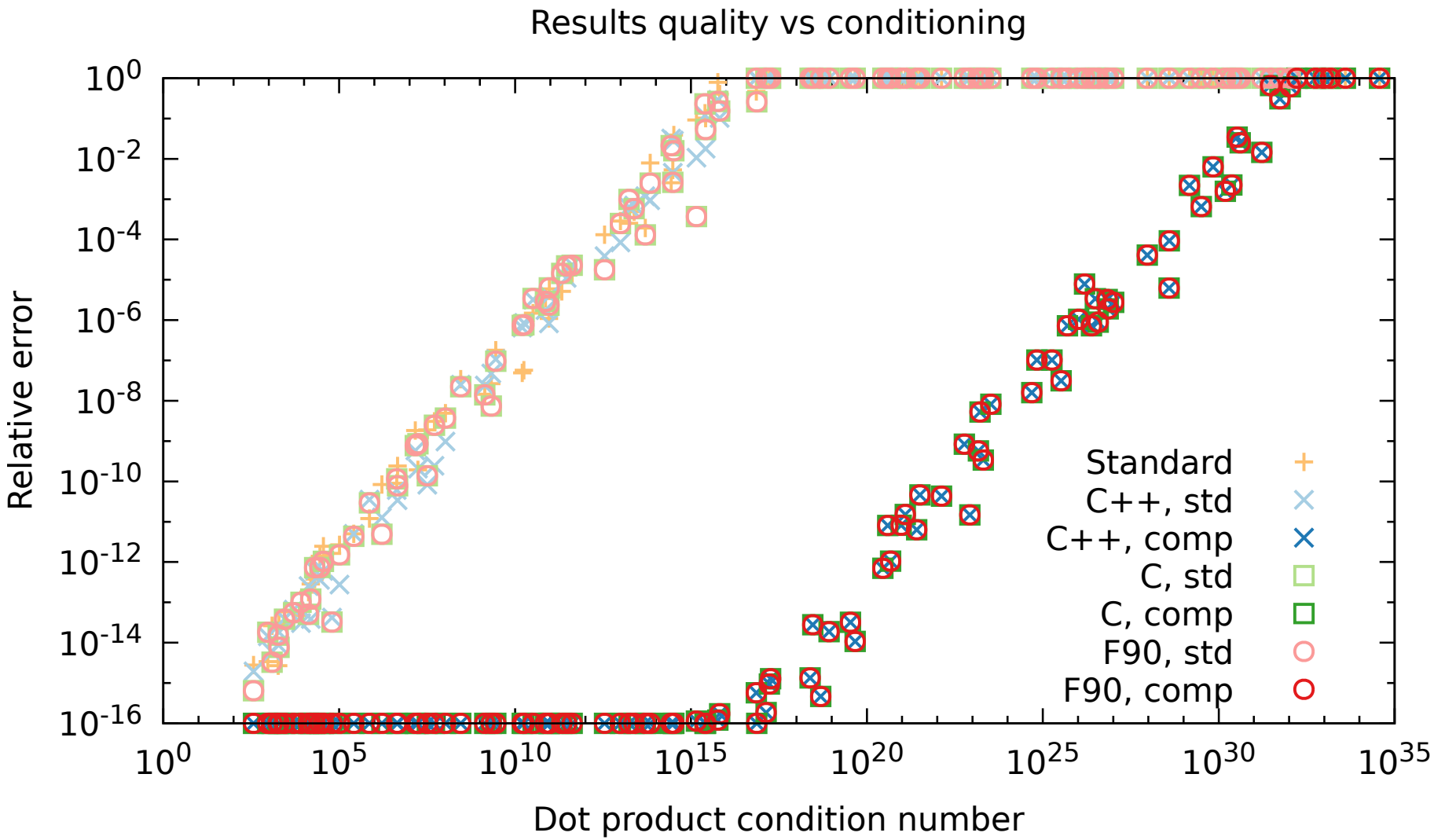
FastTwoSum(a, b) : EFT pour l'addition

if $|b| > |a|$ **then**
    $(a, b) \leftarrow (b, a)$;
end
$c \leftarrow a + b$;
$z \leftarrow c - a$;
$d \leftarrow b - z$;
return (c, d) ;



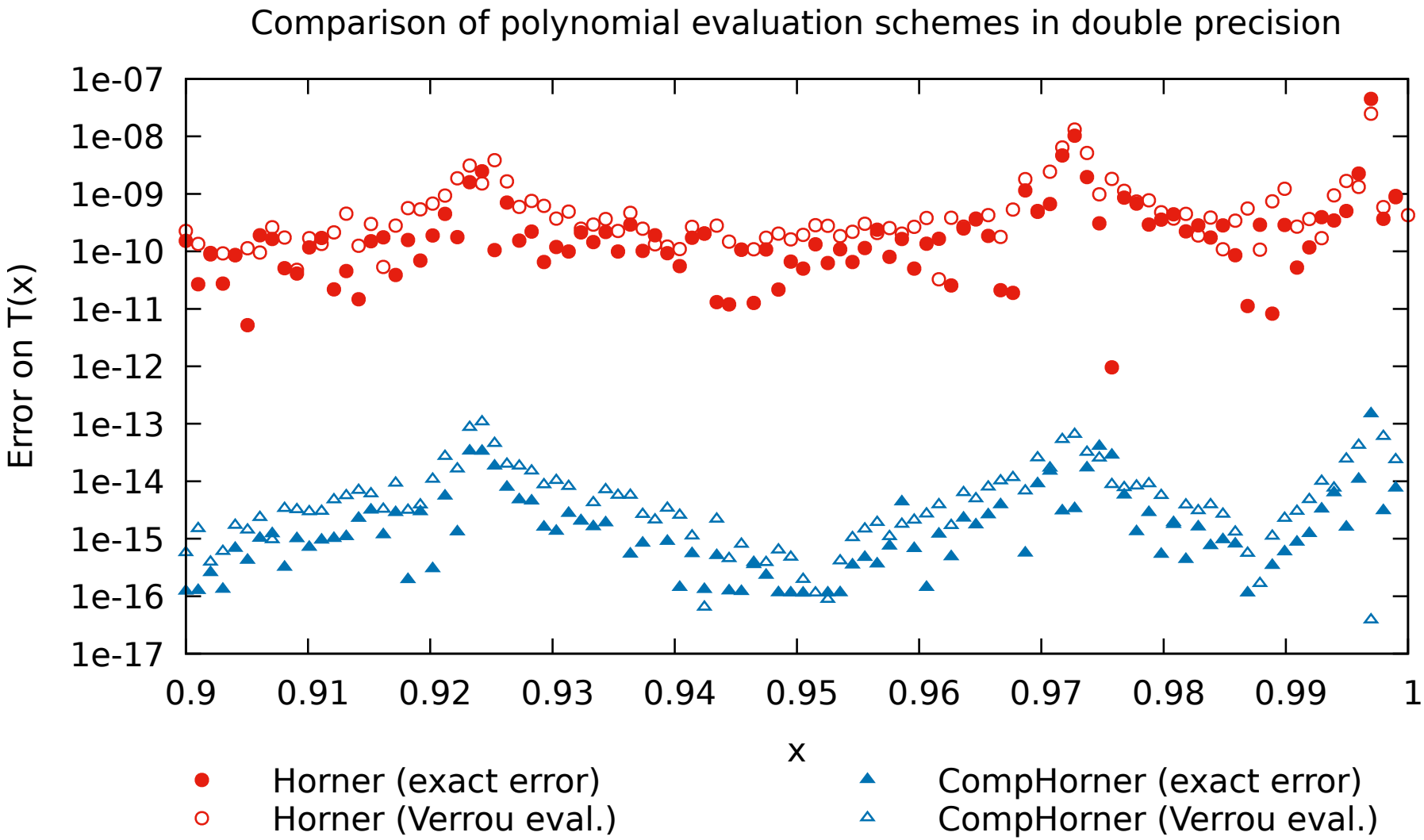
Correction

Algorithmes compensés, exemple : somme des éléments d'un vecteur



Correction

Algorithmes compensés, exemple : évaluation polynômiale



Localisation & correction

Couverture de code

1. Contexte & enjeux

2. Diagnostic

3. Localisation & correction
   - Tour d'horizon
   - Couverture de code
   - Bisection

4. Conclusions – perspectives

Localisation : couverture de code

Détection des tests instables

```
$ make CFLAGS="-fprofile-arcs -ftest-coverage"
$ make check
$ gcov *.c *.f
```

Couverture "standard"

```
120:subroutine fun1(area, a1, a2, n)
  -:    implicit none
  -:    integer :: n
  -:    real(kind=8) :: area, a1, a2
120:    if (a1 .eq. a2) then
 13:        area = a1
  -:    else
107:        if (n .lt. 2) then
107:            area = (a2-a1) / (log(a2)-log(a1))
###:        else if (n .eq.2) then
###:            area = sqrt (a1*a2)
  -:        else
###:            ! ...
  -:        endif
  -:    endif
120:end subroutine
```

Localisation : couverture de code

Détection des tests instables

```
$ make CFLAGS="-fprofile-arcs -ftest-coverage"
$ make check
$ gcov *.c *.f
```

Couverture "standard"

```
120:subroutine fun1(area, a1, a2, n)
  -:    implicit none
  -:    integer :: n
  -:    real(kind=8) :: area, a1, a2
120:    if (a1 .eq. a2) then
 13:        area = a1
  -:    else
107:        if (n .lt. 2) then
107:            area = (a2-a1) / (log(a2)-log(a1))
###:        else if (n .eq.2) then
###:            area = sqrt (a1*a2)
  -:        else
###:            ! ...
  -:        endif
  -:    endif
120:end subroutine
```

Couverture "Verrou"

```
120:subroutine fun1(area, a1,...
  -:    implicit none
  -:    integer :: n
  -:    real(kind=8) :: area,...
120:    if (a1 .eq. a2) then
  4:        area = a1
  -:    else
116:        if (n .lt. 2) then
116:            area = (a2-a1...
###:        else if (n .eq.2)...
###:            area = sqrt (...
  -:        else
###:            ! ...
  -:        endif
  -:    endif
120:end subroutine
```

Correction

Formule de code_aster

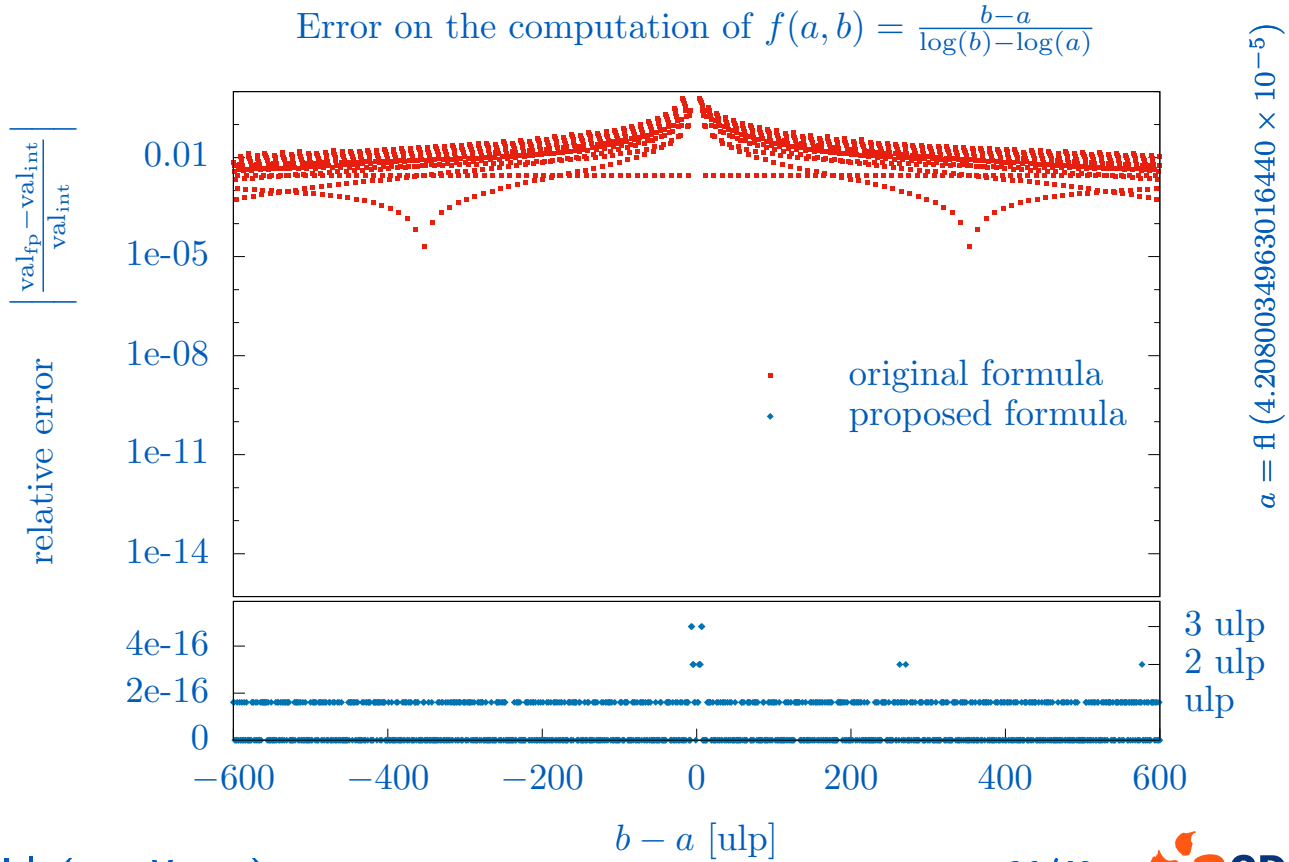
$$f(a,b) = \begin{cases} a & \text{si } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{sinon} \end{cases} \xrightarrow[\text{manuelle}]{\text{récriture}} f(a,b) = \begin{cases} a & \text{si } a = b \\ a\,\frac{\frac{b}{a}-1}{\log(\frac{b}{a})} & \text{sinon} \end{cases}$$

Étude empirique

- en dehors du code
- autour du point problématique
- référence = arithmétique d'intervalles

Preuve

- erreur bornée par 10 ulps



Localisation & correction

Bisection

1. Contexte & enjeux

2. Diagnostic

3. Localisation & correction
   - Tour d'horizon
   - Couverture de code
   - Bisection

4. Conclusions – perspectives

Localisation par bisection

Delta-Debugging

```
log.L                 .../apogene.release
volum2_               .../apogene.release
bilpla_               .../apogene.release
ecrval_               .../apogene.release
print_plath_          .../apogene.release
classer_groupes_      .../apogene.release
etupla_               .../apogene.release
couhyd_pi_            .../apogene.release
ecrplr_               .../apogene.release
imovi_                .../apogene.release
resopt_               .../apogene.release
getgrp_marginal_      .../apogene.release
ecrpla_               .../apogene.release
fin_exec_main_        .../apogene.release
decopt_pi_            .../apogene.release
paraend_              .../apogene.release
resopt_cnt_zones_     .../apogene.release
apstop_               .../apogene.release
ihyd_                 .../apogene.release
impression_info_      .../apogene.release
coupla_               .../apogene.release
gere_print_plath_     .../apogene.release
log                   .../apogene.release
thepla_               .../apogene.release
coutot_               .../apogene.release
iprit_                .../apogene.release
```

- Delta-Debugging [Zel99]

Localisation par bisection

Delta-Debugging

```
# log.L                 .../apogene.release
# volum2_               .../apogene.release
# bilpla_               .../apogene.release
# ecrval_               .../apogene.release
# print_plath_          .../apogene.release
# classer_groupes_      .../apogene.release
# etupla_               .../apogene.release
# couhyd_pi_            .../apogene.release
# ecrplr_               .../apogene.release
# imovi_                .../apogene.release
# resopt_               .../apogene.release
# getgrp_marginal_      .../apogene.release
# ecrpla_               .../apogene.release
# fin_exec_main_        .../apogene.release
# decopt_pi_            .../apogene.release
# paraend_              .../apogene.release
# resopt_cnt_zones_     .../apogene.release
# apstop_               .../apogene.release
# ihyd_                 .../apogene.release
# impression_info_      .../apogene.release
# coupla_               .../apogene.release
# gere_print_plath_     .../apogene.release
# log                   .../apogene.release
# thepla_               .../apogene.release
# coutot_               .../apogene.release
# iprit_                .../apogene.release
```

- Delta-Debugging [Zel99]

Localisation par bisection

Delta-Debugging

```
# log.L                 .../apogene.release
# volum2_               .../apogene.release
# bilpla_               .../apogene.release
# ecrval_               .../apogene.release
# print_plath_          .../apogene.release
# classer_groupes_      .../apogene.release
# etupla_               .../apogene.release
# couhyd_pi_            .../apogene.release
# ecrplr_               .../apogene.release
# imovi_                .../apogene.release
# resopt_               .../apogene.release
# getgrp_marginal_      .../apogene.release
# ecrpla_               .../apogene.release
fin_exec_main_          .../apogene.release
decopt_pi_              .../apogene.release
paraend_                .../apogene.release
resopt_cnt_zones_       .../apogene.release
apstop_                 .../apogene.release
ihyd_                   .../apogene.release
impression_info_        .../apogene.release
coupla_                 .../apogene.release
gere_print_plath_       .../apogene.release
log                     .../apogene.release
thepla_                 .../apogene.release
coutot_                 .../apogene.release
iprit_                  .../apogene.release
```

- Delta-Debugging [Zel99]

Localisation par bisection

Delta-Debugging

```
# log.L                 .../apogene.release
# volum2_               .../apogene.release
# bilpla_               .../apogene.release
# ecrval_               .../apogene.release
# print_plath_          .../apogene.release
# classer_groupes_      .../apogene.release
# etupla_               .../apogene.release
couhyd_pi_              .../apogene.release
ecrplr_                 .../apogene.release
imovi_                  .../apogene.release
resopt_                 .../apogene.release
getgrp_marginal_        .../apogene.release
ecrpla_                 .../apogene.release
fin_exec_main_          .../apogene.release
decopt_pi_              .../apogene.release
paraend_                .../apogene.release
resopt_cnt_zones_       .../apogene.release
apstop_                 .../apogene.release
ihyd_                   .../apogene.release
impression_info_        .../apogene.release
coupla_                 .../apogene.release
gere_print_plath_       .../apogene.release
log                     .../apogene.release
thepla_                 .../apogene.release
coutot_                 .../apogene.release
iprit_                  .../apogene.release
```

- Delta-Debugging [Zel99]

Localisation par bisection

Delta-Debugging

```
log.L                 .../apogene.release
volum2_               .../apogene.release
bilpla_               .../apogene.release
ecrval_               .../apogene.release
print_plath_          .../apogene.release
classer_groupes_      .../apogene.release
etupla_               .../apogene.release
# couhyd_pi_          .../apogene.release
# ecrplr_             .../apogene.release
# imovi_              .../apogene.release
# resopt_             .../apogene.release
# getgrp_marginal_    .../apogene.release
# ecrpla_             .../apogene.release
fin_exec_main_        .../apogene.release
decopt_pi_            .../apogene.release
paraend_              .../apogene.release
resopt_cnt_zones_     .../apogene.release
apstop_               .../apogene.release
ihyd_                 .../apogene.release
impression_info_      .../apogene.release
coupla_               .../apogene.release
gere_print_plath_     .../apogene.release
log                   .../apogene.release
thepla_               .../apogene.release
coutot_               .../apogene.release
iprit_                .../apogene.release
```

- Delta-Debugging [Zel99]

Localisation par bisection

Delta-Debugging

```
log.L                 .../apogene.release
volum2_               .../apogene.release
bilpla_               .../apogene.release
ecrval_               .../apogene.release
print_plath_          .../apogene.release
classer_groupes_      .../apogene.release
etupla_               .../apogene.release
# couhyd_pi_          .../apogene.release
ecrplr_               .../apogene.release
imovi_                .../apogene.release
resopt_               .../apogene.release
getgrp_marginal_      .../apogene.release
ecrpla_               .../apogene.release
fin_exec_main_        .../apogene.release
# decopt_pi_          .../apogene.release
paraend_              .../apogene.release
resopt_cnt_zones_     .../apogene.release
apstop_               .../apogene.release
# ihyd_               .../apogene.release
impression_info_      .../apogene.release
coupla_               .../apogene.release
gere_print_plath_     .../apogene.release
log                   .../apogene.release
thepla_               .../apogene.release
# coutot_             .../apogene.release
# iprit_              .../apogene.release
```

- Delta-Debugging [Zel99]

- Entrées :
  - script de lancement
  - script de comparaison
- Sortie :
  - DDmax: ensemble maximal de fonctions générant une erreur

Localisation par bisection

Delta-Debugging sur code_aster (cas-test `sdnl112a`)

- 2:20' run time =

(Herbgrind serait-il compétitif?)

- 86 configurations testées
- max. 15 exécutions perturbées par test
- 10s. par exécution (*vs* 4s. sans Verrou)

```
      do 60 jvec = 1, nbvect
          do 30 k = 1, neq
              vectmp(k)=vect(k,jvec)
30        continue
          if (prepos) call mrconl('DIVI', lmat, 0, 'R', vectmp,1)
          xsol(1,jvec)=xsol(1,jvec)+zr(jvalms-1+1)*vectmp(1)
          do 50 ilig = 2, neq
              kdeb=smdi(ilig-1)+1
              kfin=smdi(ilig)-1
              do 40 ki = kdeb, kfin
                  jcol=smhc(ki)
                  xsol(ilig,jvec)=xsol(ilig,jvec) + zr(jvalmi-1+ki) * vectmp(jcol)
                  xsol(jcol,jvec)=xsol(jcol,jvec) + zr(jvalms-1+ki) * vectmp(ilig)
40            continue
              xsol(ilig,jvec)=xsol(ilig,jvec) + zr(jvalms+kfin) * vectmp(ilig)
50        continue
          if (prepos) call mrconl('DIVI', lmat, 0, 'R', xsol(1, jvec),1)
60    continue
```

Localisation par bisection

Delta-Debugging sur code_aster (cas-test `sdnl112a`)

- 2:20' run time =

(Herbgrind serait-il compétitif?)

- 86 configurations testées
- max. 15 exécutions perturbées par test
- 10s. par exécution (*vs* 4s. sans Verrou)

```
      do 60 jvec = 1, nbvect
          do 30 k = 1, neq
              vectmp(k)=vect(k,jvec)
30        continue
          if (prepos) call mrconl('DIVI', lmat, 0, 'R', vectmp,1)
          xsol(1,jvec)=xsol(1,jvec)+zr(jvalms-1+1)*vectmp(1)
          do 50 ilig = 2, neq
```

- Correction : algorithmes compensés [ORO05]
  - CompSum ne suffit pas
  - CompDot corrige le problème !

```
40            continue
              xsol(ilig,jvec)=xsol(ilig,jvec) + zr(jvalms+kfin) * vectmp(ilig)
50        continue
          if (prepos) call mrconl('DIVI', lmat, 0, 'R', xsol(1, jvec),1)
60    continue
```

Conclusions

Estimation de la fidélité après correction

`sdnl112a`

Version	Statut nearest	rnd_1	rnd_2	rnd_3	# chiffres en commun $C(\mathrm{rnd}_1, \mathrm{rnd}_2, \mathrm{rnd}_3)$
Avant correction	OK	KO	KO	KO	6 6 6 * 3 0
Après correction	OK	OK	OK	OK	10 10 9 * 6 0

Question : vérifiabilité des algorithmes compensés avec MCA ?
Réponse : oui (mais pas tout le temps) [GJP18]

Conclusions – Perspectives

Vérification numérique

Un sujet ré-émergent...

- unités de calcul de plus en plus complexes
- calculs de plus en plus gros
- modèles de plus en plus précis
- études paramétriques plus denses
- apparition d'outils / méthodologies

... à démocratiser

- intégrer ces démarches dans le processus AQ
- aller vers une meilleure ergonomie
- veille importante (nombreux nouveaux acteurs)
- renforcer (créer ?) des liens avec la communauté "UQ"

Conclusions – Perspectives

Verrou

Conclusions

Verrou semble convenir à nos besoins:

- coût d'entrée (quasiment) nul,
- quantification des pertes de précision flottante,
- localisation semi-automatique des parties instables (à gros grain).

Perspectives

- gérer toutes les instructions
  - AVX & instructions vectorielles SSE simple précision,
  - instructions scalaires x87 ;
- conforter les fonctionnalités de Delta-Debugging ;
- conforter les fonctionnalités de localisation de branchements instables ;
- gérer "proprement" la libmath.

Perspectives : Interflop

Logiciel

- Interface commune pour l'interopérabilité des "briques" logicielles
  - *front-ends* : comment intercepter les opérations flottantes ?
  - *back-ends* : par quoi les remplacer ?
- Outils visés
  - Verificarlo (*front-end* LLVM, *back-ends* Arithmétique de Monte Carlo)
  - Verrou (*front-end* Valgrind, *back-end* CESTAC asynchrone)
  - CADNA (*back-end* CESTAC synchrone)
  - ...

Consortium

- Objectifs :
  - structurer les activités : FUI, ANR
  - point d'entrée unique
- Partenaires :
  - UVSQ, Intel, EDF, Aneo, UPMC, CEA

Merci !
Des questions ?

Récupérez `verrou` sur github :
`http://github.com/edf-hpc/verrou`

Documentation :
`http://edf-hpc.github.io/verrou/vr-manual.html`

Outils mentionnés

CADNA	http://www-pequan.lip6.fr/cadna/
CRAFT HPC	https://github.com/crafthpc/craft/
Herbgrind	http://herbgrind.ucsd.edu/
Herbie	http://herbie.uwplse.org/
IntervalArithmetic.jl	https://github.com/JuliaIntervals/IntervalArithmetic.jl
LibEFT	http://github.com/ffevotte/libeft
MPFI	http://perso.ens-lyon.fr/nathalie.revol/software.html
MPFR	http://www.mpfr.org/
Verificarlo	https://github.com/verificarlo/verificarlo
Verrou	https://github.com/edf-hpc/verrou
fpDebug	https://github.com/fbenz/FpDebug
gmpy	https://pypi.org/project/gmpy2/
pyintervals	https://github.com/taschini/pyintervalpyintervals

Bibliographie I

- Jean-Marie Chesneaux and Jean Vignes, *On the robustness of the cestac method*, C. R. Acad.Sci. Paris **1** (1988), 855–860.
- Christophe Denis, Pablo de Oliveira Castro, and Eric Petit, *Verificarlo: checking floating point accuracy through Monte Carlo Arithmetic*, 23rd IEEE Internatinal Symposium on Computer Arithmetic (ARITH'23), 2016.
- François Févotte and Bruno Lathuilière, *VERROU: Assessing Floating-Point Accuracy Without Recompiling*, October 2016.
- François Févotte and Bruno Lathuilière, *Studying the numerical quality of an industrial computing code: A case study on code_aster*, 10th International Workshop on Numerical Software Verification (NSV) (Heidelberg, Germany), July 2017, pp. 61–80.
- Stef Graillat, Fabienne Jézéquel, and Romain Picot, *Numerical validation of compensated algorithms with stochastic arithmetic*, Applied Mathematics and Computation **329** (2018), 339 – 363.

Bibliographie II

- Fabienne Jézéquel, Jean-Marie Chesneaux, and Jean-Luc Lamotte, *A new version of the CADNA library for estimating round-off error propagation in Fortran programs*, Computer Physics Communications **181** (2010), no. 11, 1927–1928.
- William Kahan, *How futile are mindless assessments of roundoff in floating-point computations?*, 2006.
- Jean-Luc Lamotte, Jean-Marie Chesneaux, and Fabienne Jézéquel, *CADNA_C: A version of CADNA for use with C or C++ programs*, Computer Physics Communications **181** (2010), no. 11, 1925–1926.
- Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi, *Accurate sum and dot product*, SIAM J. Sci. Comput. **26** (2005), 1955–1988.
- Siegfried M. Rump, *Algorithms for verified inclusions: theory and practice*, Reliability in Computing: The Role of Interval Methods in Scientific Computing, Academic Press, 1988, pp. 109–126.

Bibliographie III

- Douglas Stott Parker, *Monte Carlo arithmetic: exploiting randomness in floating-point arithmetic*, Tech. Report CSD-970002, University of California, Los Angeles, 1997.
- Jean Vignes, *A stochastic arithmetic for reliable scientific computation*, Mathematics and Computers in Simulation **35** (1993), 233–261.
- Andreas Zeller, *Yesterday, My Program Worked. Today, It Does Not. Why?*, SIGSOFT Softw. Eng. Notes **24** (1999), no. 6, 253–267.