



Vérification numérique des codes industriels avec Verrou

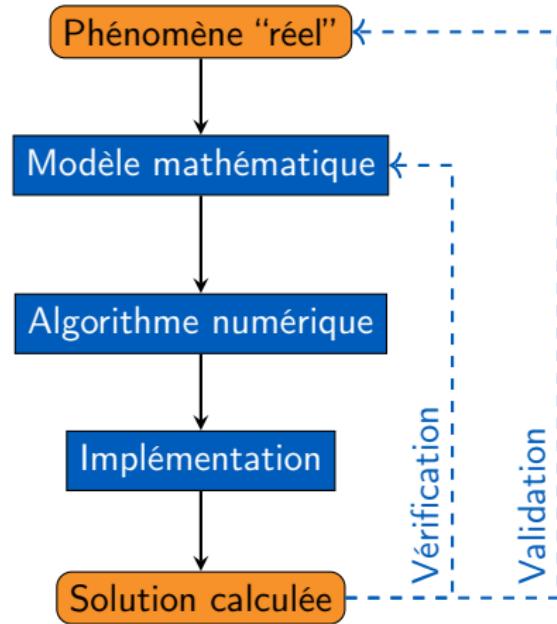
Ecole d'été EDF/CEA/INRIA
26/06/2018

Bruno Lathuilière, François Févotte
EDF R&D



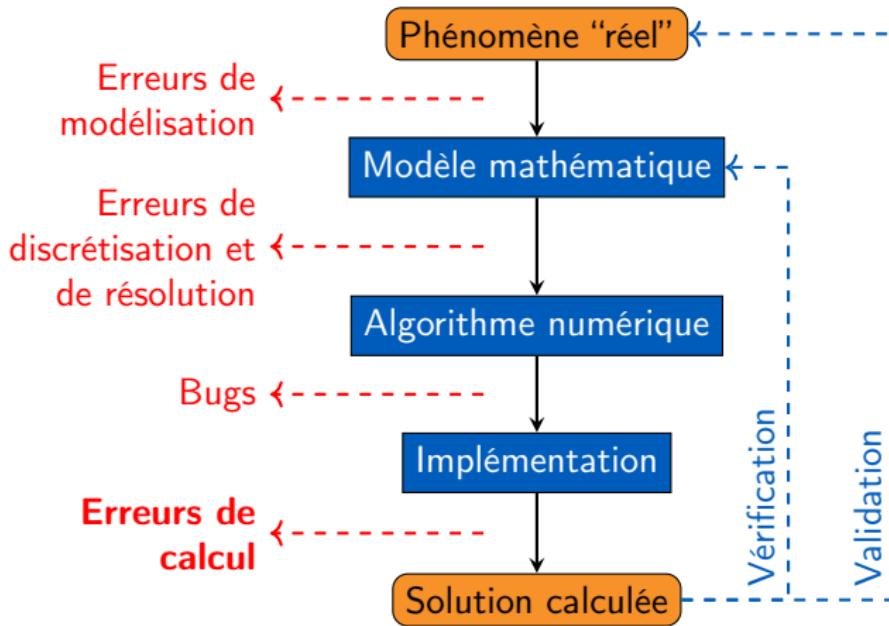
Contexte & enjeux

Vérification & Validation



Contexte & enjeux

Vérification & Validation

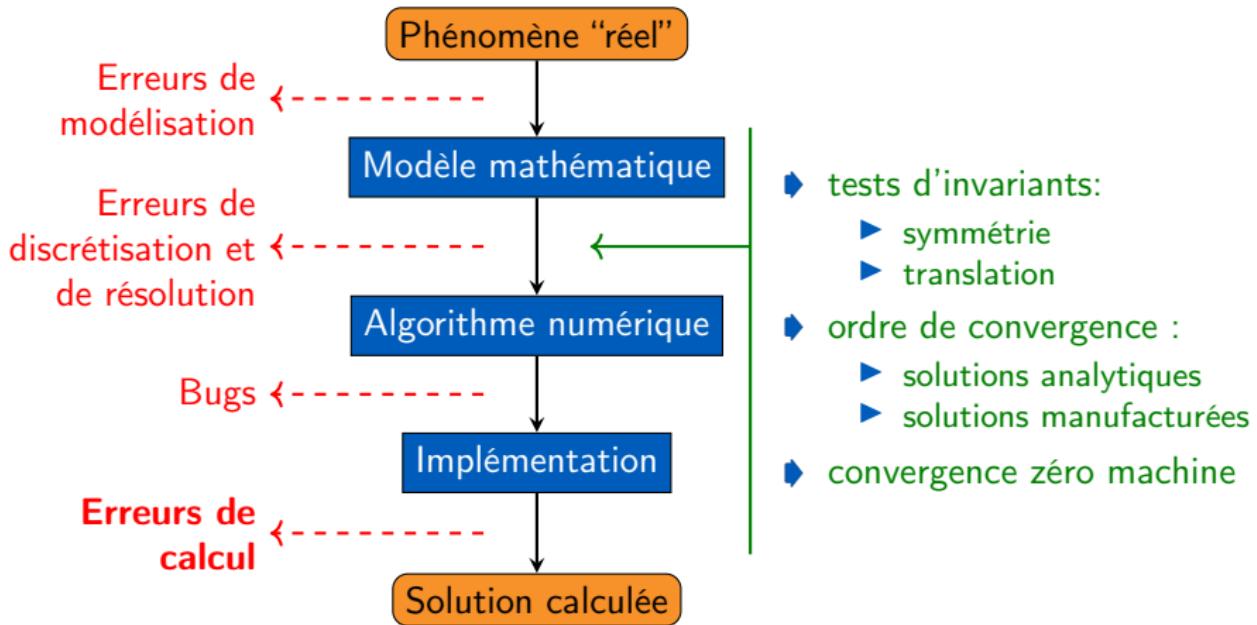


Quantification des erreurs : enjeux

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

Contexte & enjeux

Vérification & Validation

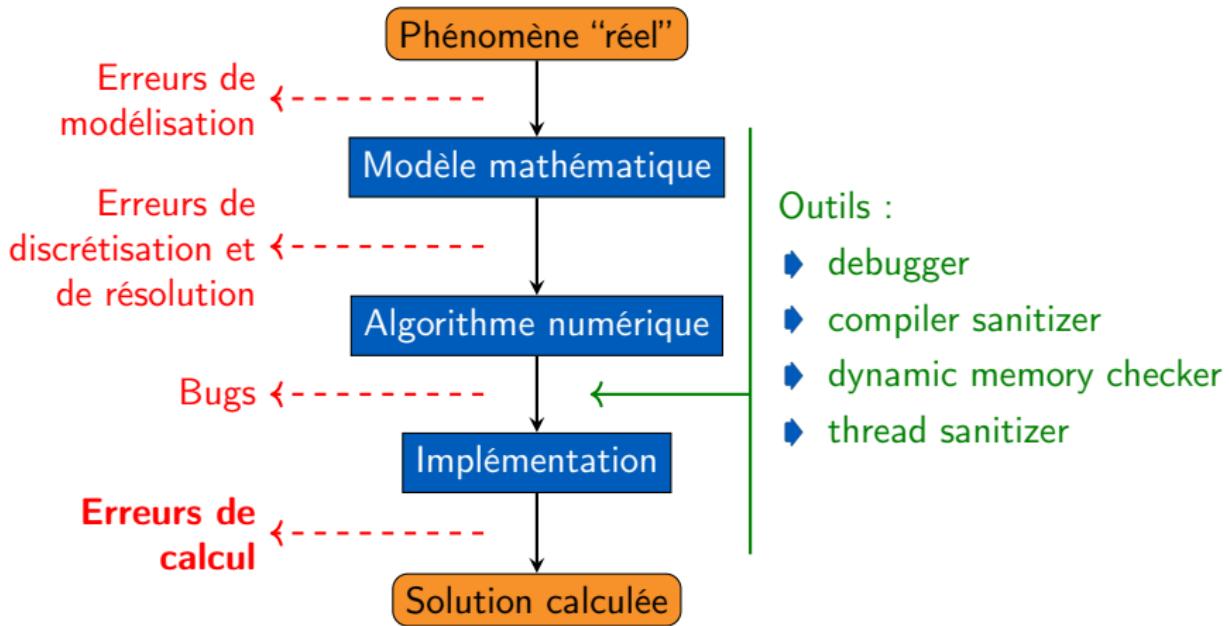


Quantification des erreurs : enjeux

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

Contexte & enjeux

Vérification & Validation

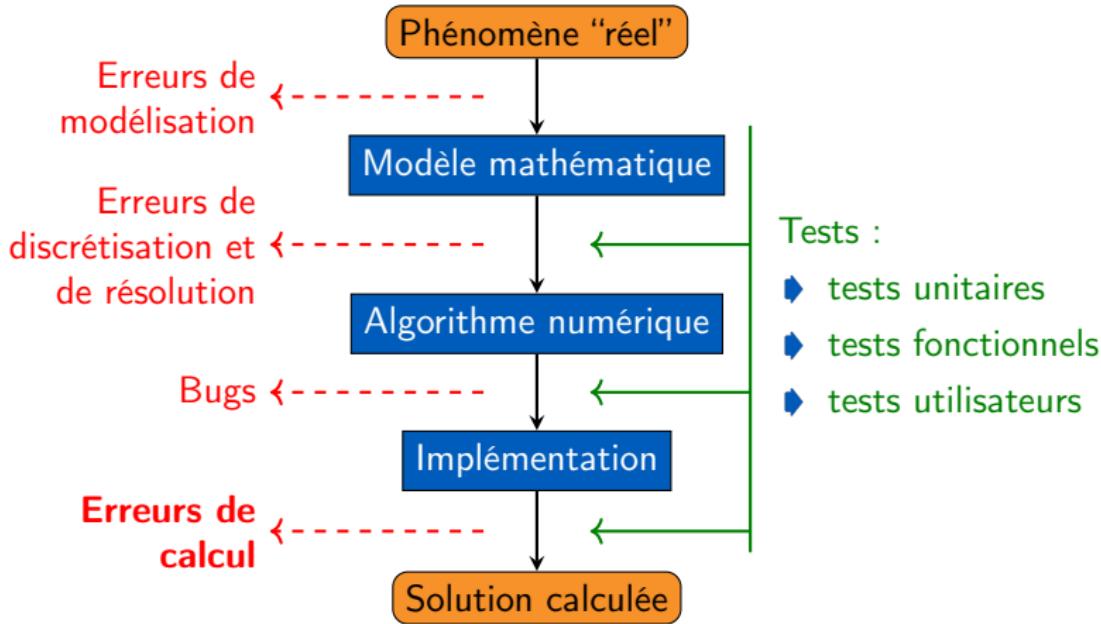


Quantification des erreurs : enjeux

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

Contexte & enjeux

Vérification & Validation

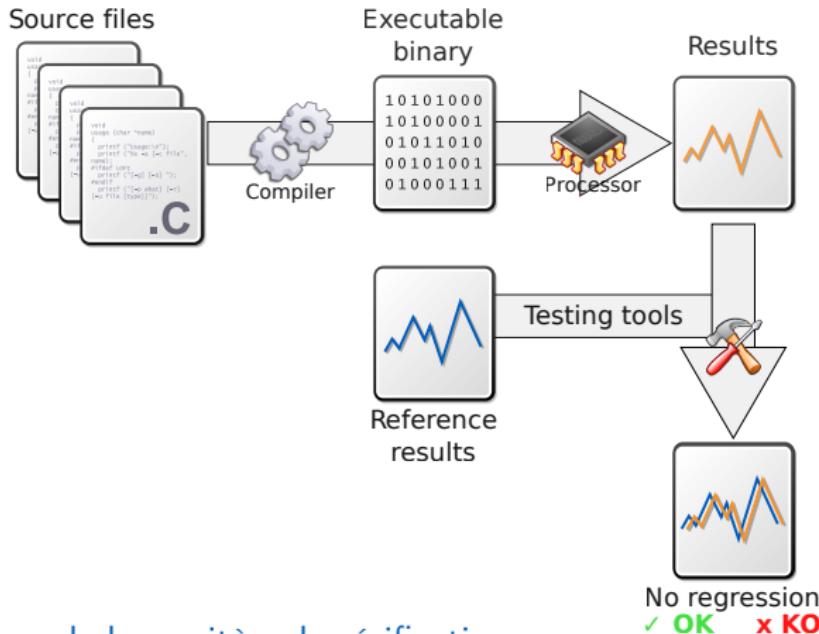


Quantification des erreurs : enjeux

- ▶ 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



Difficulté : trouver le bon critère de vérification

Contexte & enjeux

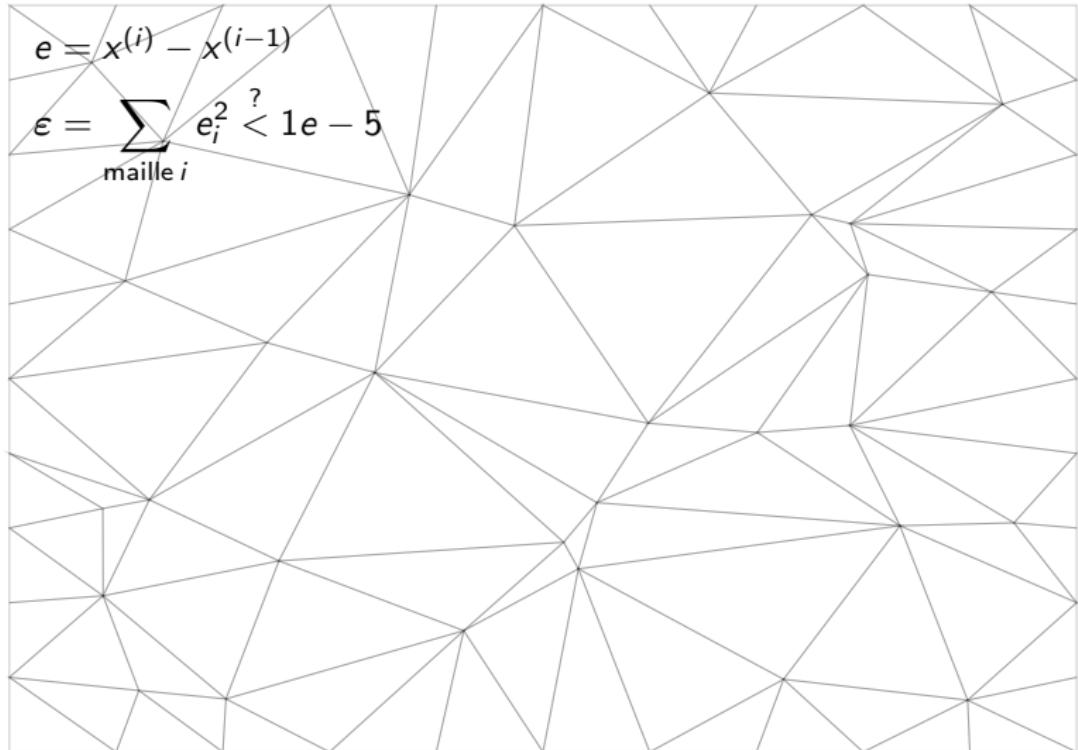
Quelles conséquences ?

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)

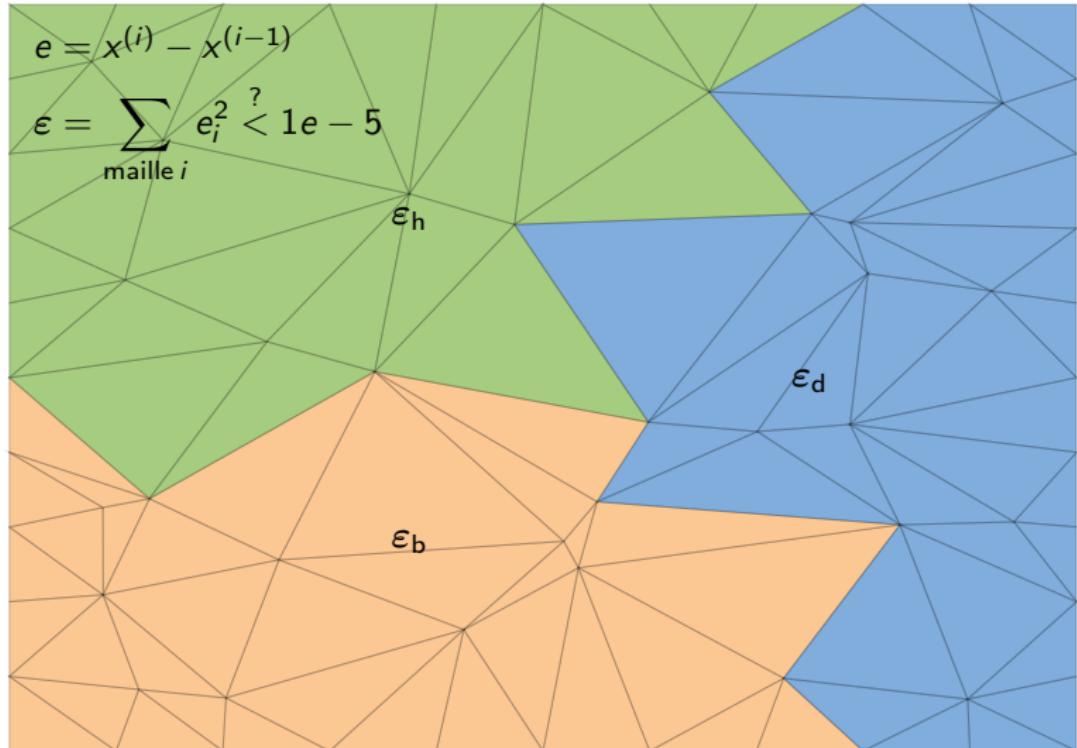
Contexte & enjeux

Calcul bloqué : ex. Telemac2D



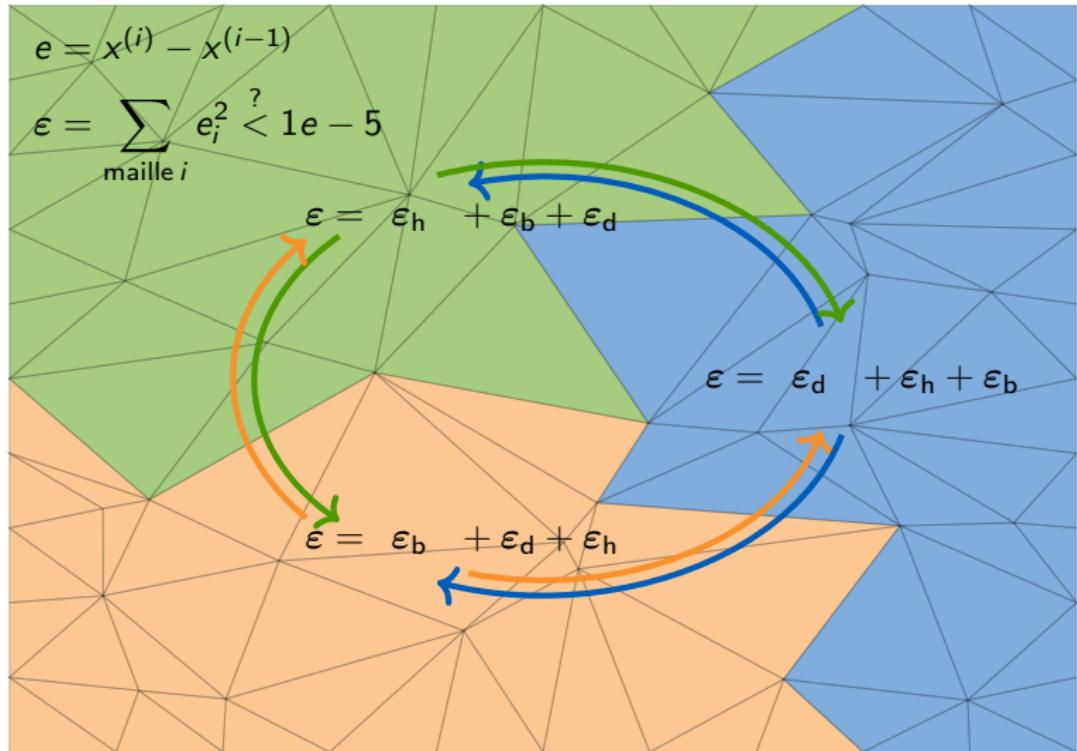
Contexte & enjeux

Calcul bloqué : ex. Telemac2D



Contexte & enjeux

Calcul bloqué : ex. Telemac2D



Contexte & enjeux

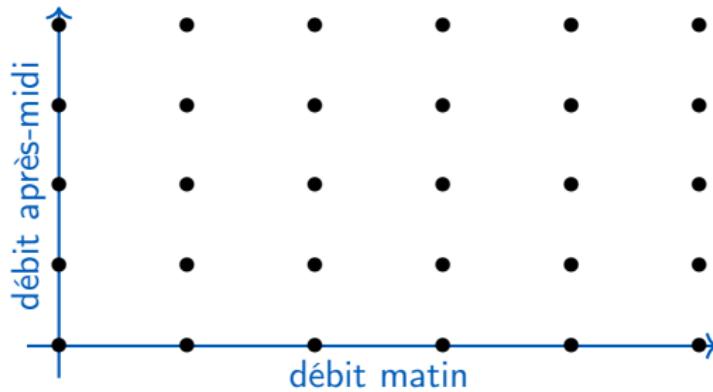
Quelles conséquences ?

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)

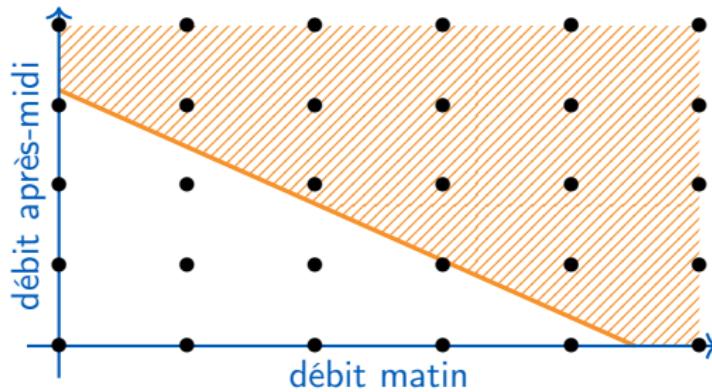
Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



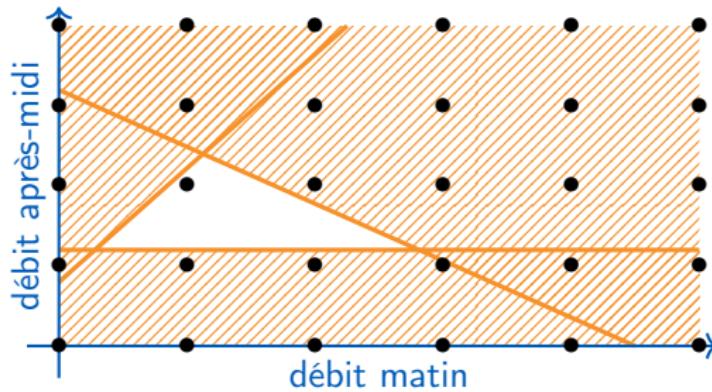
Contexte & enjeux

Résultat incorrect : ex. optimisation de la production hydraulique



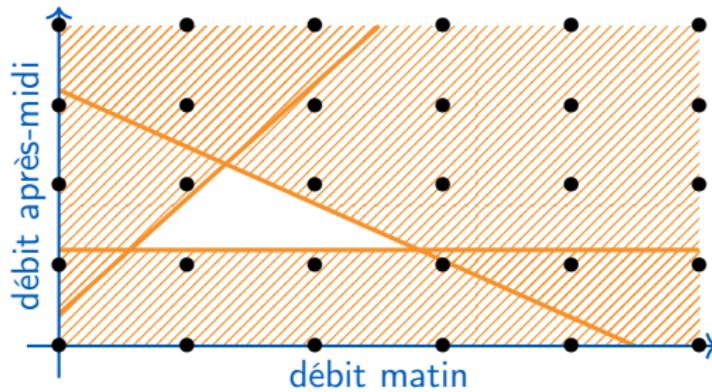
Contexte & enjeux

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

Job	S	W
<u>F3C</u>		
<i>COCAGNE Non Reg</i>		
<u>COCAGNE-NonReg</u>		
<i>Test on multiple configuration</i>		
<u>COCAGNE-NonReg-c7-gcc-dbl-dklib-rel-par</u>		
<u>COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-par</u>		
<u>COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-seq</u>		
<u>COCAGNE-NonReg-c7-intel-dbl-dkzip-rel-par</u>		
<u>COCAGNE-NonReg-c7-gcc-flt-dklib-rel-par</u>		
<u>COCAGNE-NonReg-c7-gcc-flt-dkzip-rel-par</u>		
<u>COCAGNE-NonReg-c9-gcc-dbl-dkzip-rel-par</u>		
<u>COCAGNE-NonReg-c7-gcc-dbl-dkzip-rel-par-reloc</u>		
<u>COCAGNE-NonReg-c9-gcc-dbl-dkzip-rel-par-reloc</u>		

Contexte & enjeux

Non-reproductibilité



- ▶ Architecture du CPU

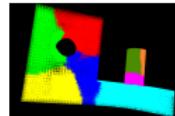


- ▶ Calcul séquentiel ou parallèle



- ▶ Compilateur

(ou même simplement les options)



- ▶ Versions de bibliothèques externes

Contexte & enjeux

Précision... du vocabulaire

(cf. métrologie, ISO-5725)



fidélité
(precision)



justesse
(trueness)



exactitude
(accuracy)

résolution



Objectifs de la vérification numérique
► Déterminer les chiffres significatifs **exacts**

Contexte & enjeux

Précision... du vocabulaire



Expérience d'origine



✓ Répétabilité / fiabilité

Contexte & enjeux

Précision... du vocabulaire



Expérience d'origine



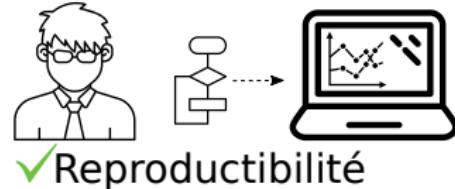
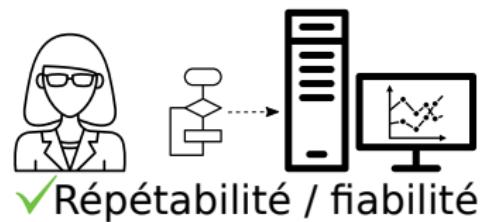
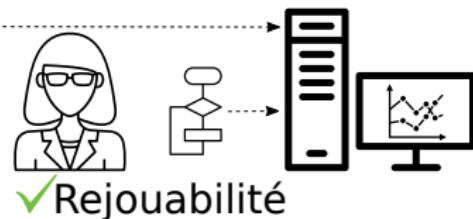
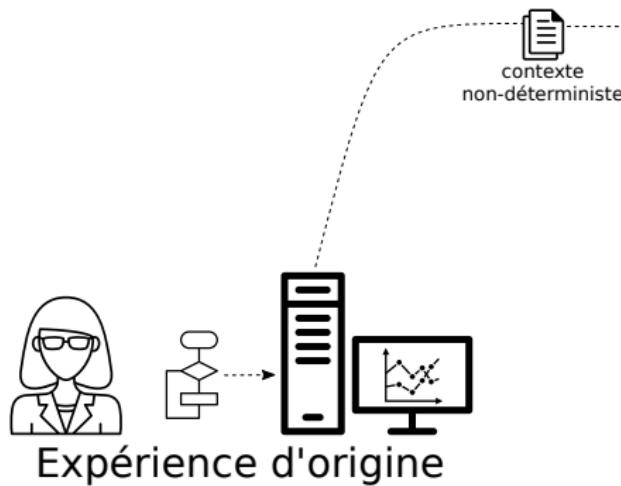
✓ Répétabilité / fiabilité



✓ Reproductibilité

Contexte & enjeux

Précision... du vocabulaire



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

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

1534019.677371745

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

-1534019.677282780

0.000088965 

11 chiffres

Contexte & enjeux

Performances : ex. optimisation production hydraulique

Re-formulation proposée

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

Calcul de la fonction objectif

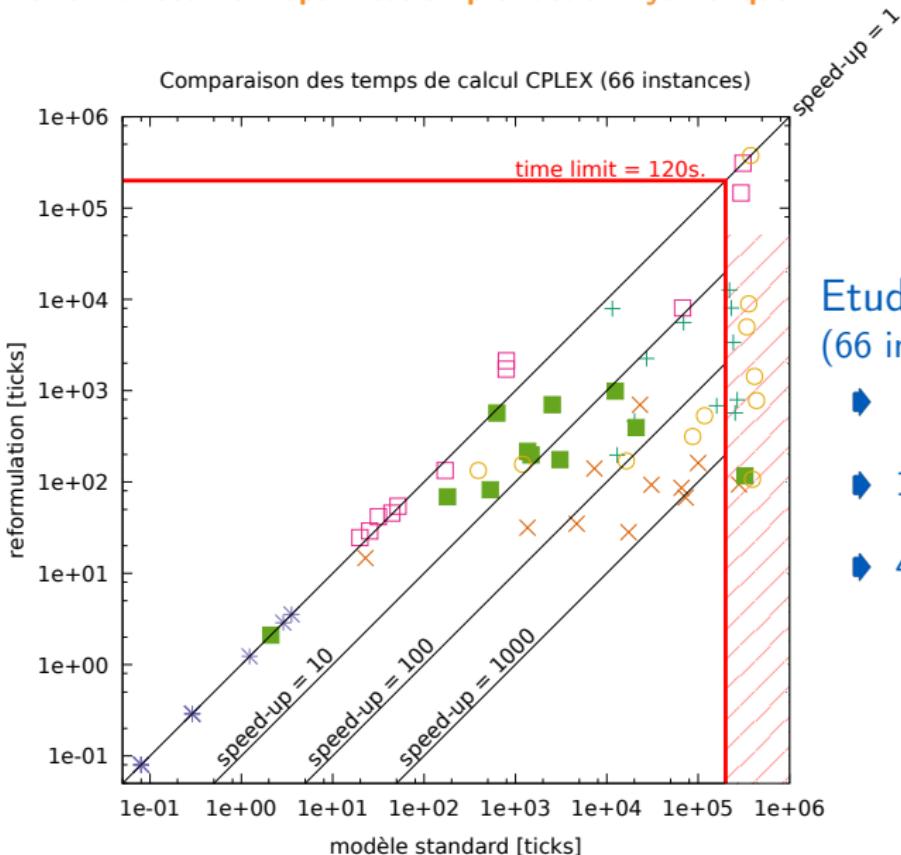
$$\sum_{t \in T} \sum_{i \in I} \lambda_t p_t^i + \sum_{r \in R} \omega_r v_t^r \quad 1534019.677371745$$

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

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

Contexte & enjeux

Performances : ex. optimisation production hydraulique



Etude en PLNE
(66 instances)

- ◆ 2 (3%) ralenties ($\times 2$)
- ◆ 15 (23%) ne changent pas
- ◆ 49 (74%) accélérées ($\times 2-1100$)

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 : athena-2D

3. Localisation & correction

4. Conclusions – perspectives

Méthodes de diagnostic

Tour d'horizon

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

- ① analyse mathématique rigoureuse
- ② comparaison à un calcul en précision supérieure
- ③ **comparaison à des calculs arrondis différemment** (4 modes IEEE-754)
- ④ analyse statistique de calculs arrondis aléatoirement
- ⑤ analyse statistique des calculs avec données d'entrées perturbées
- ⑥ 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]:

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

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

simple précision → 1.172603...

double précision → 1.1726039400531...

précision étendue → 1.172603940053178...

calcul exact → $0.827396059946821 + [3; 4] \times 10^{-16}$

Méthodes de diagnostic

Perturbation des données d'entrée

- ▶ lien fort avec la quantification d'incertitudes → outils disponibles
- ⚠ peu fiable sans connaissance préalable du problème
 - ▶ quel est le conditionnement du système ?
- 💡 certaines méthodes spécifiques peuvent être mises en place
 - ▶ ex: sensibilité à la numérotation du maillage

Arithmétique d'intervalles

ex : MPFI, pyintervals, IntervalArithmetic.jl

idée: $x \rightarrow [\underline{x}, \bar{x}]$ $x - y = [\underline{x} - \bar{y}, \bar{x} - \underline{y}]$

💡 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 : athena-2D

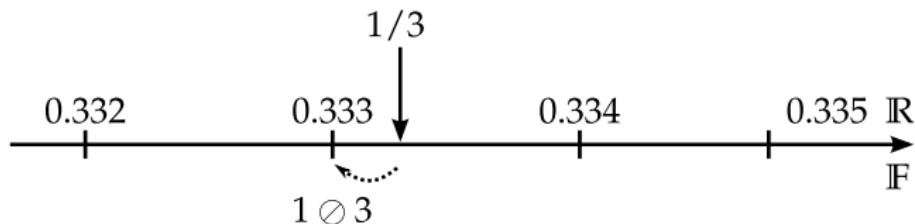
3. Localisation & correction

4. Conclusions – perspectives

Méthodes CESTAC & MCA

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

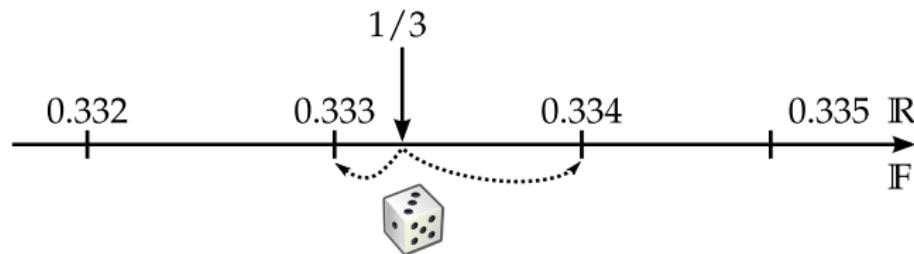
Arrondi au plus proche (défaut IEEE-754)



Méthodes CESTAC & MCA

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

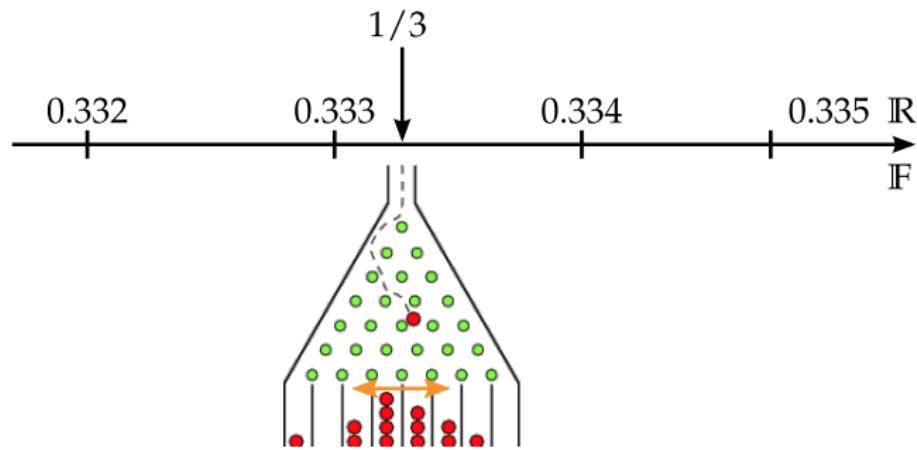
Arrondi aléatoire (CESTAC [Vig93, CV88])



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
$a = 1/3$	$0.333\downarrow$	$0.334\uparrow$	$0.334\uparrow$	0.334
$b = a \times 3$	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 $b = 1$ end $r = \text{asin}(b)$				

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↓	0.334↑	0.334↑	3.34e-1
$b = a \times 3$				
if $b > 1$ then				
$b = 1$				
end				
$r = \text{asin}(b)$				

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↓	0.334↑	0.334↑	3.34e-1
$b = a \times 3$	0.999↓	1.00↓	1.01↑	1.00
if $b > 1$ then $b = 1$ end		False		Warning
$r = \text{asin}(b)$	1.53↑	1.58↑	✗	✗

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↓			
$b = a \times 3$	0.999↓			
if $b > 1$ then	False			
$b = 1$				
end				
$r = \text{asin}(b)$	1.53↑			

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↓	0.334↑	0.334↑	
$b = a \times 3$	0.999↓	1.00↓	1.01↑	
if $b > 1$ then	False	False	True	
$b = 1$			1.00	
end				
$r = \text{asin}(b)$	1.53↑	1.58↑	1.57↓	
print r				1.56

Méthodes CESTAC & MCA

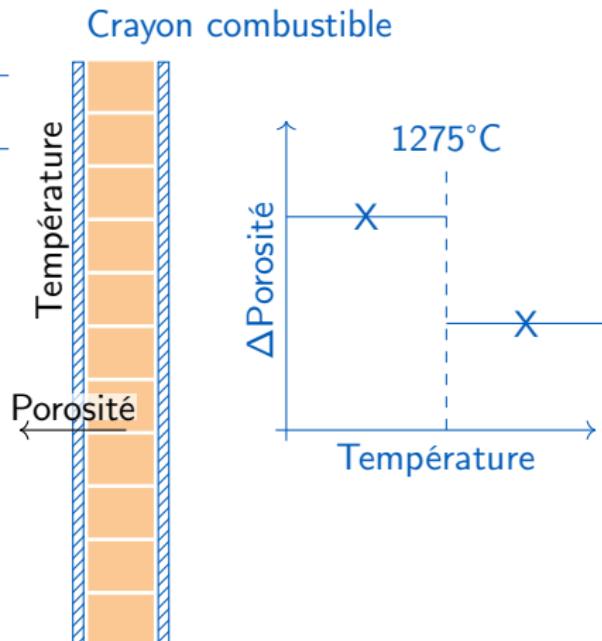
Tests instables : exemple de MAAP

▶ 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 # ...
```



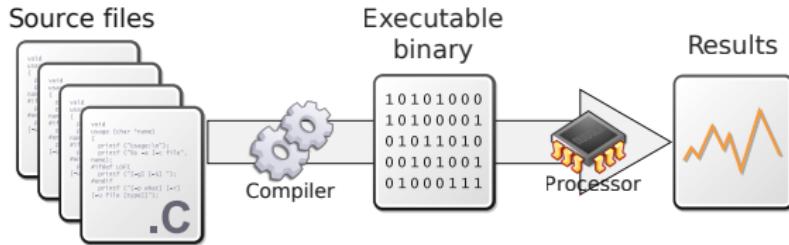
▶ Sorties :

Porosité	0.44282	0.45583	$\sim 10^{-2}$
----------	---------	---------	----------------

Méthodes CESTAC & MCA : Outils

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

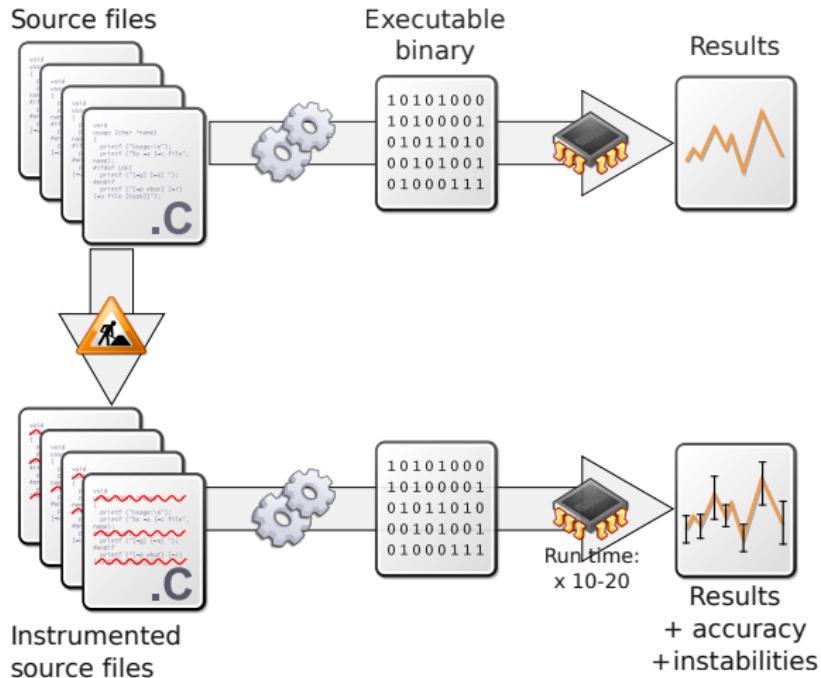
\$ myProg in out



Méthodes CESTAC & MCA : Outils

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

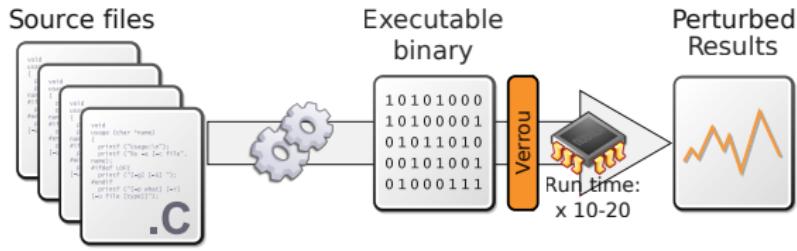
\$ myProg-cadna in out



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

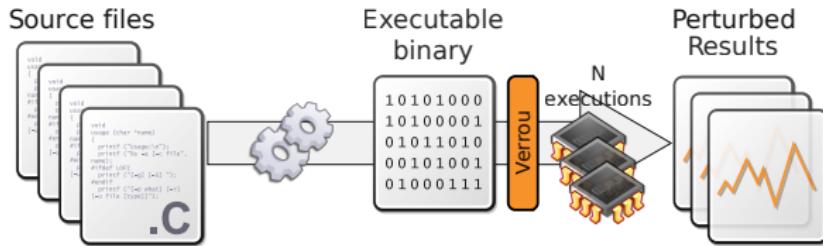
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
```



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

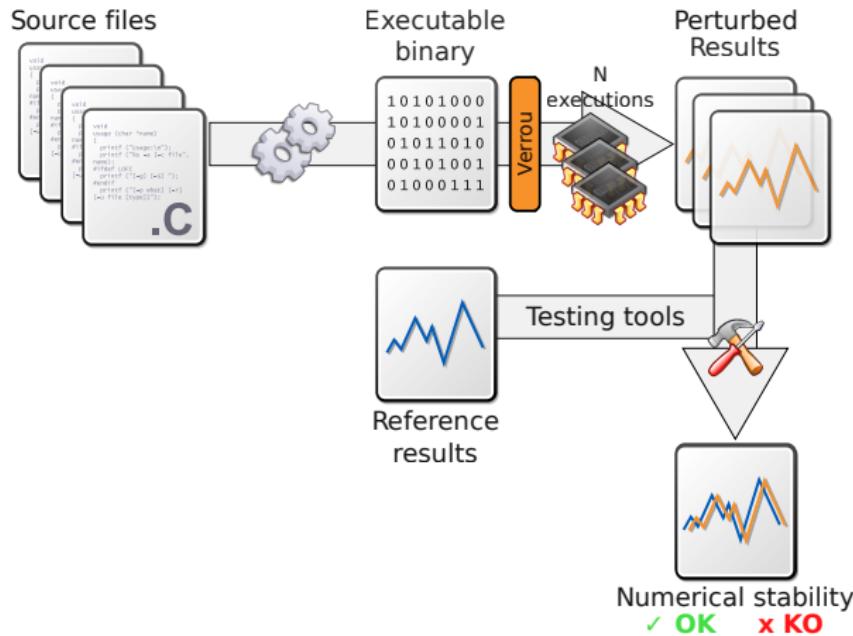
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1  
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



Méthodes CESTAC & MCA : Outils

Verrou : analyse dynamique du binaire [FL16]

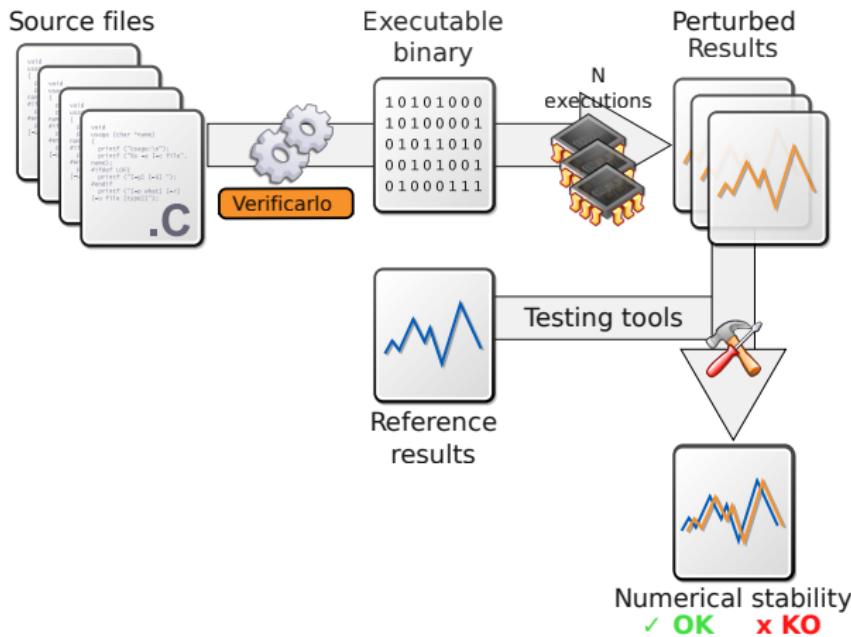
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1  
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



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 : athena-2D

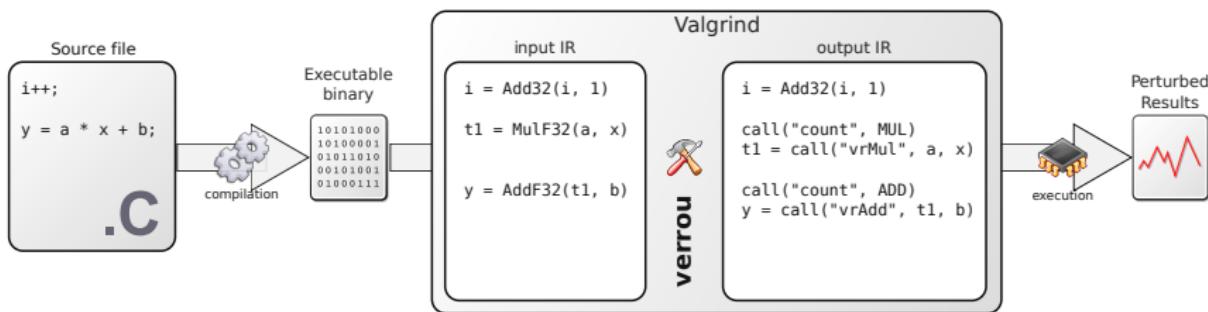
3. Localisation & correction

4. Conclusions – perspectives

L'outil Verrou

Front-end : Analyse dynamique du binaire avec Valgrind

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



L'outil Verrou

Exemple de sortie

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

==4683== Verrou, Check floating-point rounding errors
==4683== Copyright (C) 2014, F. Fevotte & B. Lathuiliere.
...
==4683== First seed : 1430818339
==4683== Simulating AVERAGE rounding mode
==4683== Instrumented operations :
==4683==   add : yes
...
==4683== -----
==4683==          Operation           Instructions count
==4683==            '- Precision      Instrumented          Total
==4683== -----
==4683==   add                  500869335                500869335      (100%)
==4683==   '- flt                 400695468                400695468      (100%)
==4683==   '- dbl                 100173867                100173867      (100%)
==4683== -----
==4683==   sub                  763127658                763127658      (100%)
==4683==   '- flt                 763127658                763127658      (100%)
==4683== -----
==4683==   mul                  1202086563               1202086563      (100%)
==4683==   '- flt                 1101912537               1101912537      (100%)
==4683==   '- dbl                 100174026                100174026      (100%)
==4683== -----
...

```

L'outil Verrou

Exemple de sortie

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

```
==4683== Verrou, Check floating-point rounding errors
==4683== Copyright (C) 2014, F. Fevotte & B. Lathuiliere.
```

```
...
==4683== First seed : 1430818339
```

```
==4683== Simulating AVERAGE rounding mode
```

```
==4683== Instrumented operations :
```

```
==4683== add . v8s
```

```
...
    ↪
```

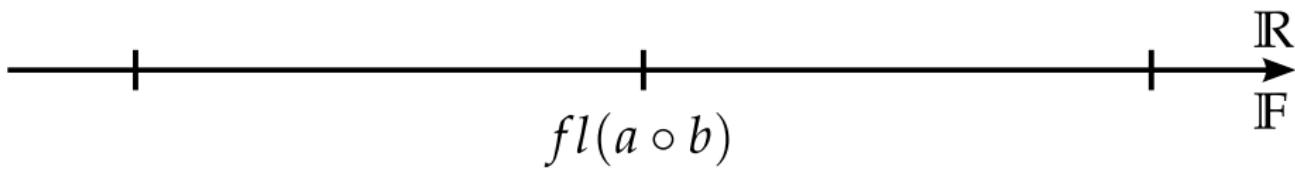
Sorties normales du programme

+ Warnings pour les instructions “dangereuses”
(ex : x87, NaN)

==4683==	- abi	100173867	100173867	(100%)
==4683==	-			
==4683==	sub	763127658	763127658	(100%)
==4683==	' - flt	763127658	763127658	(100%)
==4683==	-			
==4683==	mul	1202086563	1202086563	(100%)
==4683==	' - flt	1101912537	1101912537	(100%)
==4683==	' - dbl	100174026	100174026	(100%)
==4683==	-			

L'outil Verrou

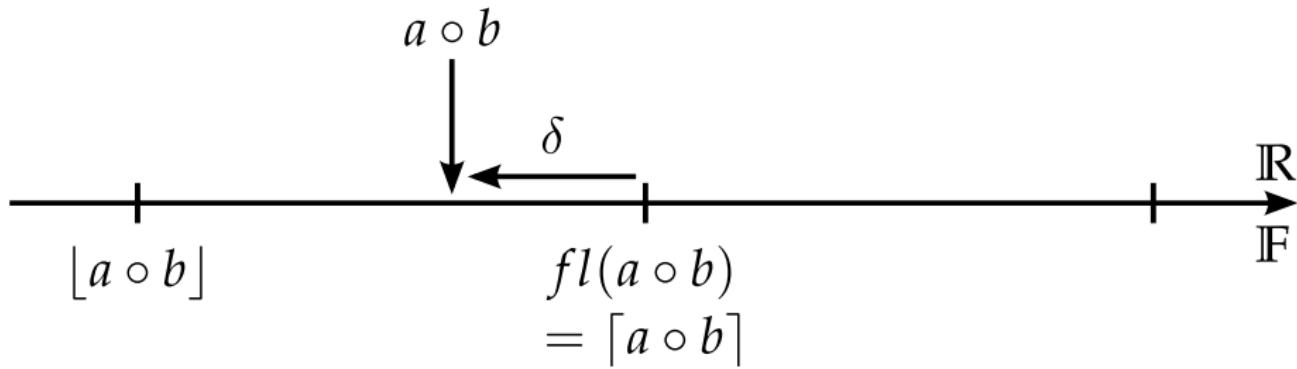
Back-end : Simulation des modes d'arrondi



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

L'outil Verrou

Back-end : Simulation des modes d'arrondi

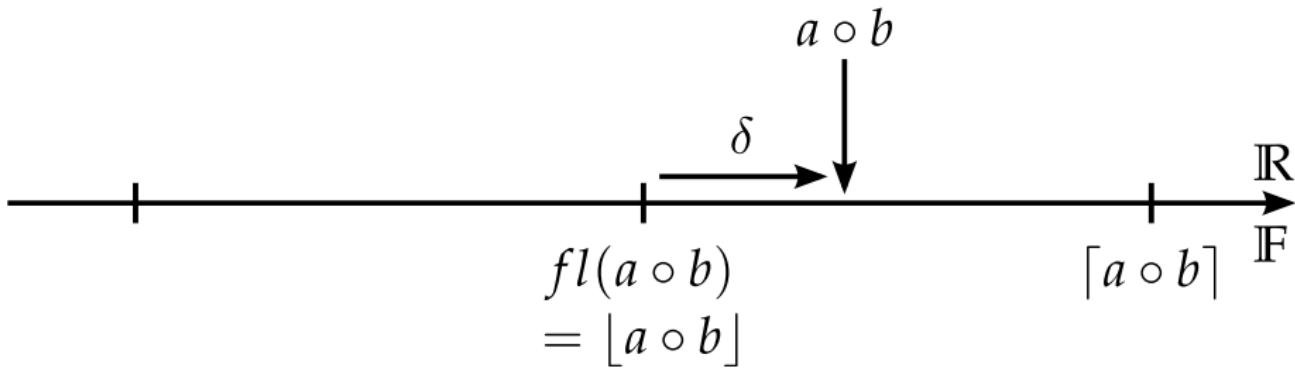


- ▶ Transformation exacte
(la division est un peu plus compliquée):
 - ▶ $a \circ b = \sigma + \delta$,
 - ▶ $\sigma = fl(a \circ b)$

- ▶ 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

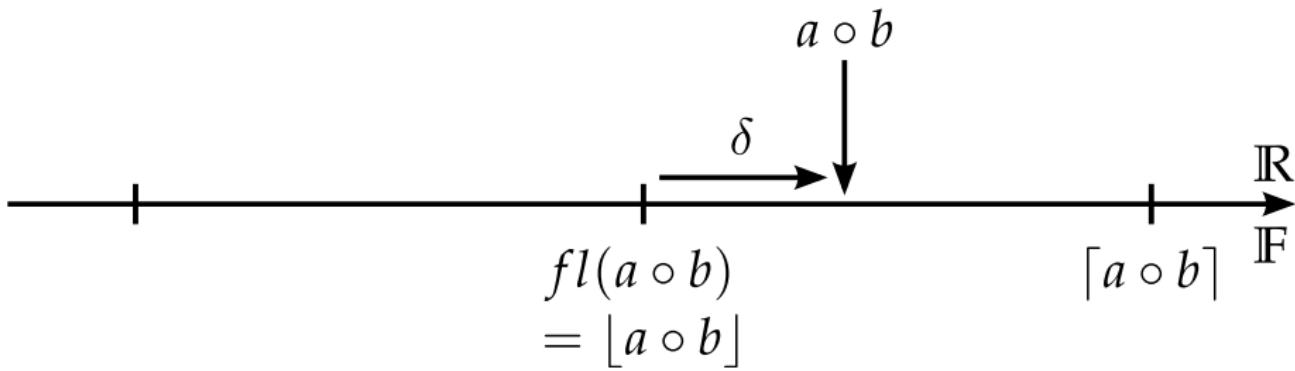


- ▶ Transformation exacte
(la division est un peu plus compliquée):
 - ▶ $a \circ b = \sigma + \delta$,
 - ▶ $\sigma = fl(a \circ b)$

- ▶ Si $\delta > 0$:
 - ▶ $[a \circ b] = fl(a \circ b)$,
 - ▶ $[a \circ b] = fl(a \circ b) + ulp.$

L'outil Verrou

Back-end : Simulation des modes d'arrondi



▶ random : arrondi équiprobable (CESTAC)

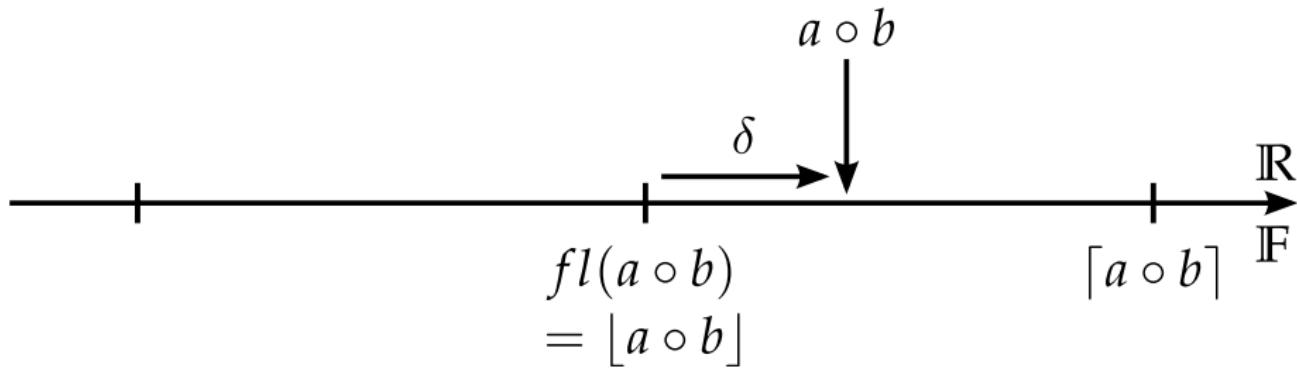
$$\blacktriangleright p(\lceil a \circ b \rceil) = 0.5$$

$$\blacktriangleright p(\lfloor a \circ b \rfloor) = 0.5$$



L'outil Verrou

Back-end : Simulation des modes d'arrondi



▶ average : arrondi “moyen” (MCA RR-ulp)

$$\triangleright p(\lceil a \circ b \rceil) = \frac{\delta}{ulp} \quad (\text{si } \delta > 0)$$

$$\triangleright p(\lfloor a \circ b \rfloor) = \frac{ulp - \delta}{ulp}$$



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]:

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



Diagnostic Limites

1. Contexte & enjeux

2. Diagnostic

Tour d'horizon

MCA & CESTAC

Verrou

Limites

Application : athena-2D

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

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
```

Limites de CESTAC / MCA

Algorithmes spécifiques prenant l'arithmétique flottante en compte

Autres faux positifs connus

- ▶ Conversion binaire décimal (affichage)
- ▶ Formule $\arcsin\left(\frac{y}{\sqrt{x^2+y^2}}\right)$
 - ▶ Solution : formule à remplacer par $\arcsin(\max(-1, \min(1, \frac{y}{\sqrt{x^2+y^2}})))$
 - ▶ Cf. présentation S. Boldo
- ▶ Besoin de déterminisme au sein du calcul
 - ▶ Re-calculation plutôt que stockage
 - ▶ Structure de donnée dépendant d'un calcul flottant (cf. code athena)



Diagnostic

Application : athena-2D

1. Contexte & enjeux

2. Diagnostic

Tour d'horizon

MCA & CESTAC

Verrou

Limites

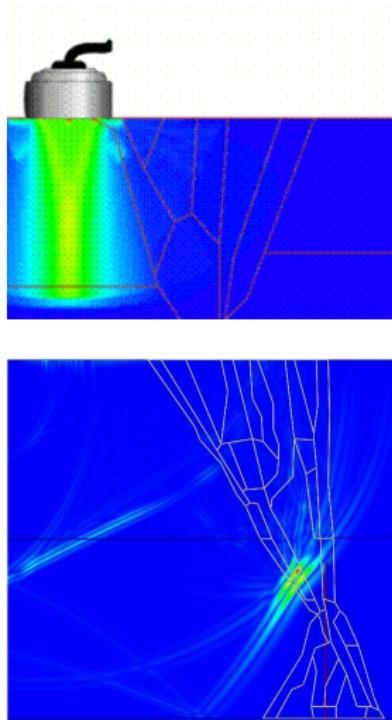
Application : athena-2D

3. Localisation & correction

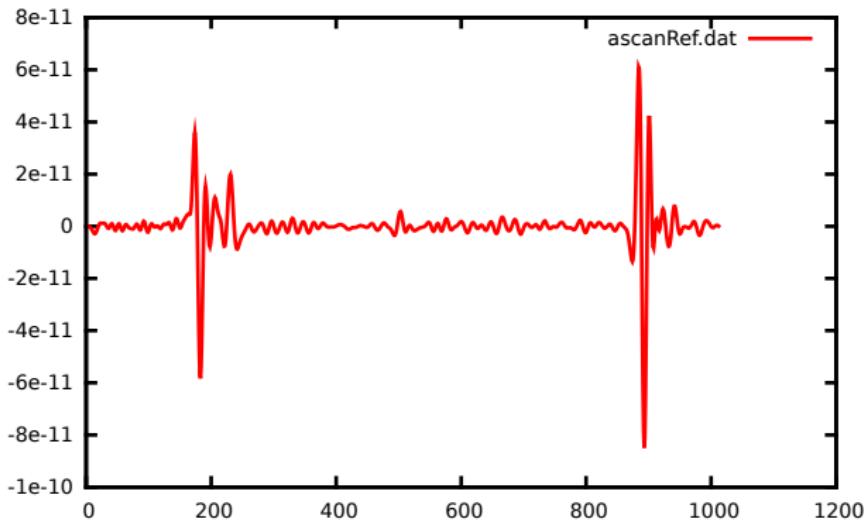
4. Conclusions – perspectives

Application : CND par ultra-sons

Code Athena 2D

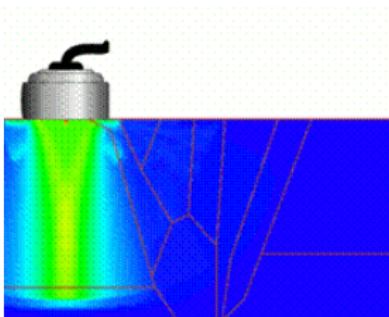


Résultat produit : "A-scan"



Application : CND par ultra-sons

Code Athena 2D



Contrôle Non Destructif par Ultra-Sons

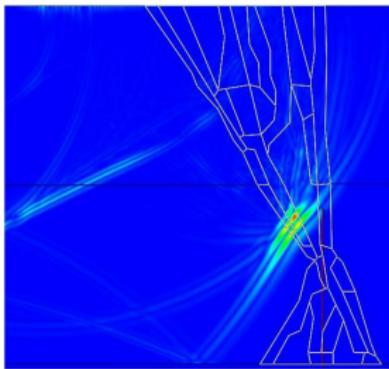
- ◆ études réalisées par la R&D + ingénierie
- ◆ sous contrôle de l'ASN

Code Athena 2D

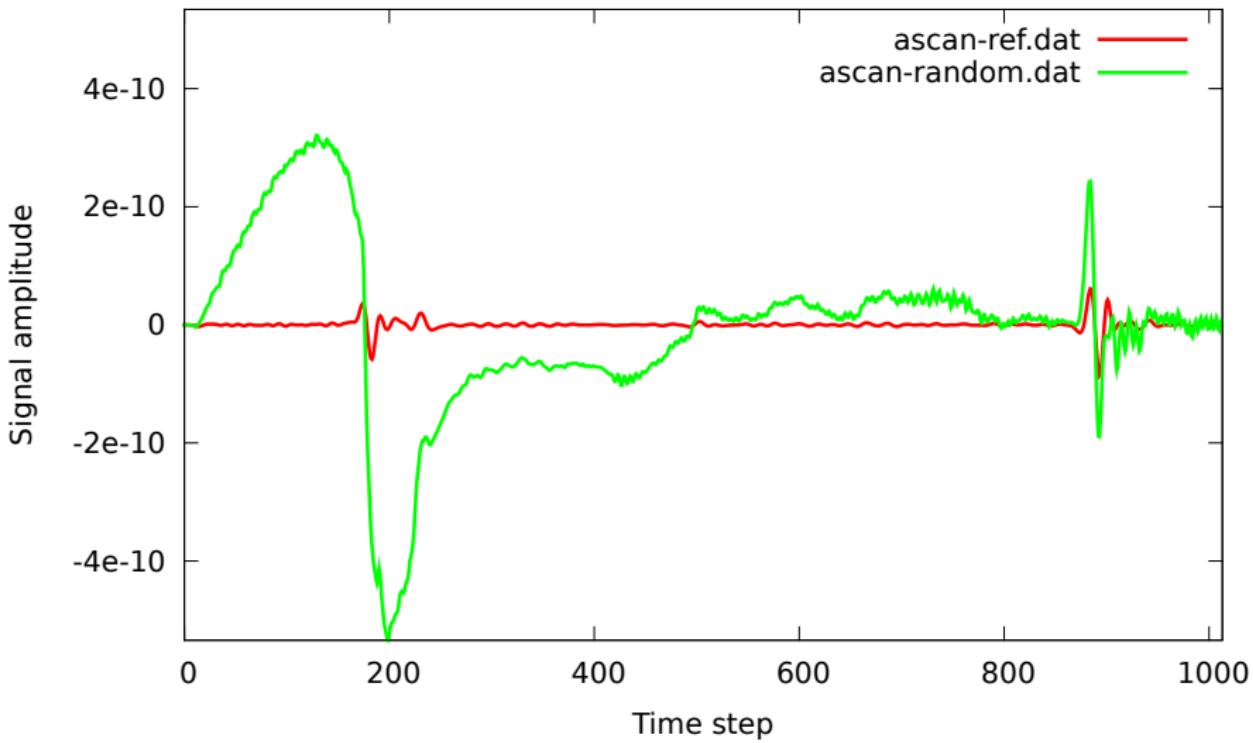
- ◆ 36k lignes de code
- ◆ Fortran 77 + Fortran 90
- ◆ Dépendances : BLAS, LAPACK
- ◆ Code "connu"

Objectifs de l'étude

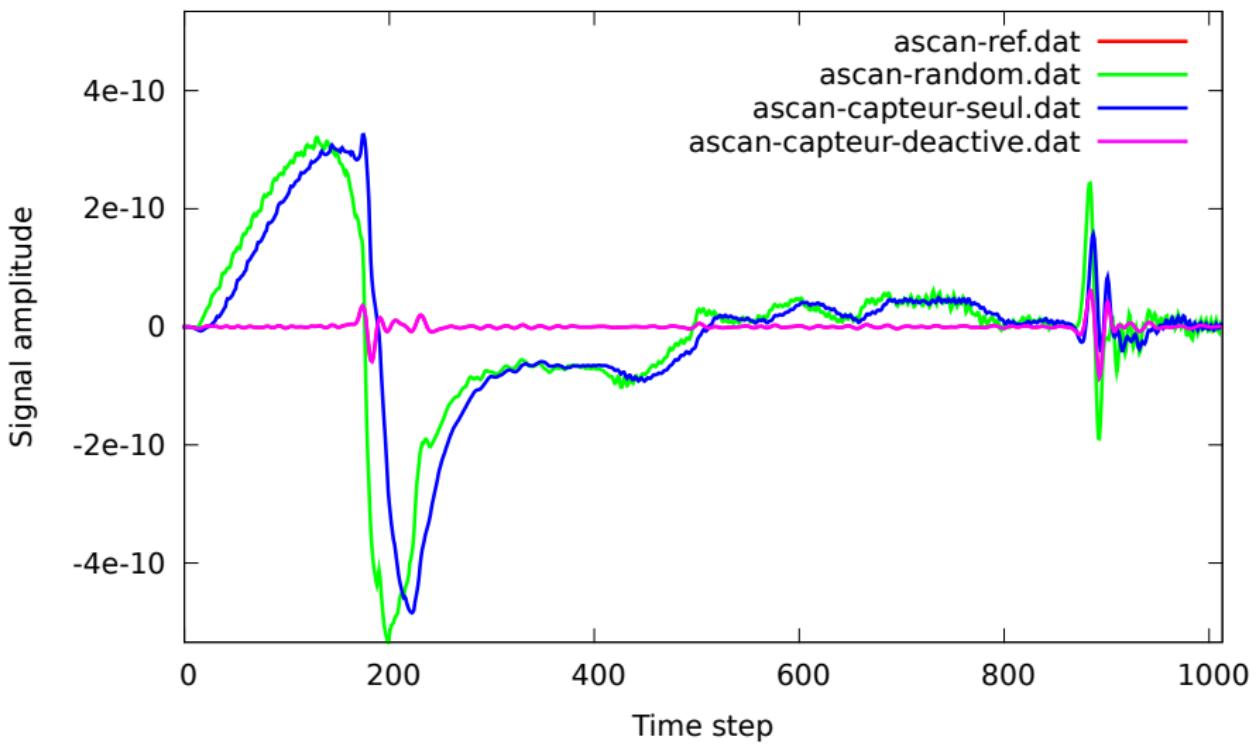
- ◆ Pas de problème identifié
- ◆ Test de "routine"



Premier calcul



Diagnostic



Explication

```
Nx,Nt=calculDimensionStockage(...);
```

```
tab1D=enregistrementTab2D(Nt, Nx, ...);
```

```
Nx,Nt=calculDimensionStockage(...);
```

```
calculAscan(tab1D, Nt,Nx, ...);
```

Explication

```
Nx,Nt=calculDimensionStockage(...);
```

```
tab1D=enregistrementTab2D(Nt, Nx, ...);
```

```
Nx,Nt=calculDimensionStockage(...);
```

```
calculAscan(tab1D, Nt,Nx, ...);
```

Enregistrement



Vérification numérique des cycles industriels avec Verrou

Relecture



29/63

Explication

```
Nx,Nt=calculDimensionStockage(...);  
tab1D=enregistrementTab2D(Nt, Nx, ...);  
//Nx,Nt=calculDimensionStockage(...);  
calculAscan(tab1D, Nt,Nx, ...);
```

Solutions :

- ① Restructurer le code pour éviter le second appel à calculDimensionStockage ;

Explication

```
VERROU_STOP_INSTRUMENTATION;  
Nx,Nt=calculDimensionStockage(...);  
VERROU_START_INSTRUMENTATION;  
tab1D=enregistrementTab2D(Nt, Nx, ...);  
VERROU_STOP_INSTRUMENTATION;  
Nx,Nt=calculDimensionStockage(...);  
VERROU_START_INSTRUMENTATION;  
calculAscan(tab1D, Nt,Nx, ...);
```

Solutions :

- ① Restructurer le code pour éviter le second appel à calculDimensionStockage ;
- ② Déactiver l'arithmétique dans la fonction calculDimensionStockage (via exclude ou client request);

Explication

```
Nx,Nt=calculDimensionStockage(...);  
tab1D=enregistrementTab2D(Nt, Nx, ...);  
Nx,Nt=calculDimensionStockage(...);  
calculAscan(tab1D, Nt,Nx, ...);
```

Solutions :

- ① Restructurer le code pour éviter le second appel à calculDimensionStockage ;
- ② Déactiver l'arithmétique dans la fonction calculDimensionStockage (via exclude ou client request);
- ③ Rendre la fonction calculDimensionStockage déterministe au sein d'un exécutable (client request);

Application : CND par ultra-sons

Tests de non-régression dans verrou

random 1

Cas-Test A

ins1.dat	0
ascan.dat	1.8e-12

Cas-Test B

sismo.dat	7.9e-69
ascan.dat	1.2e-10

Cas-Test C

ins1.dat	4.6e-06
sismo.dat	8.0e-28
ascan.dat	2.0e-11

Cas-Test D

ins1.dat	1.5e-18
enerloc.dat	0
sismo.dat	0
ascan.dat	0

Application : CND par ultra-sons

Tests de non-régression dans verrou

	random 1	random 2
Cas-Test A		
ins1.dat	0	6.1e-06
ascan.dat	1.8e-12	5.9e-12
Cas-Test B		
sismo.dat	7.9e-69	7.9e-69
ascan.dat	1.2e-10	2.0e-11
Cas-Test C		
ins1.dat	4.6e-06	4.6e-06
sismo.dat	8.0e-28	2.8e-28
ascan.dat	2.0e-11	1.2e-11
Cas-Test D		
ins1.dat	1.5e-18	4.1e-01
enerloc.dat	0	2.3e-01
sismo.dat	0	1.6e-01
ascan.dat	0	1.5e-01

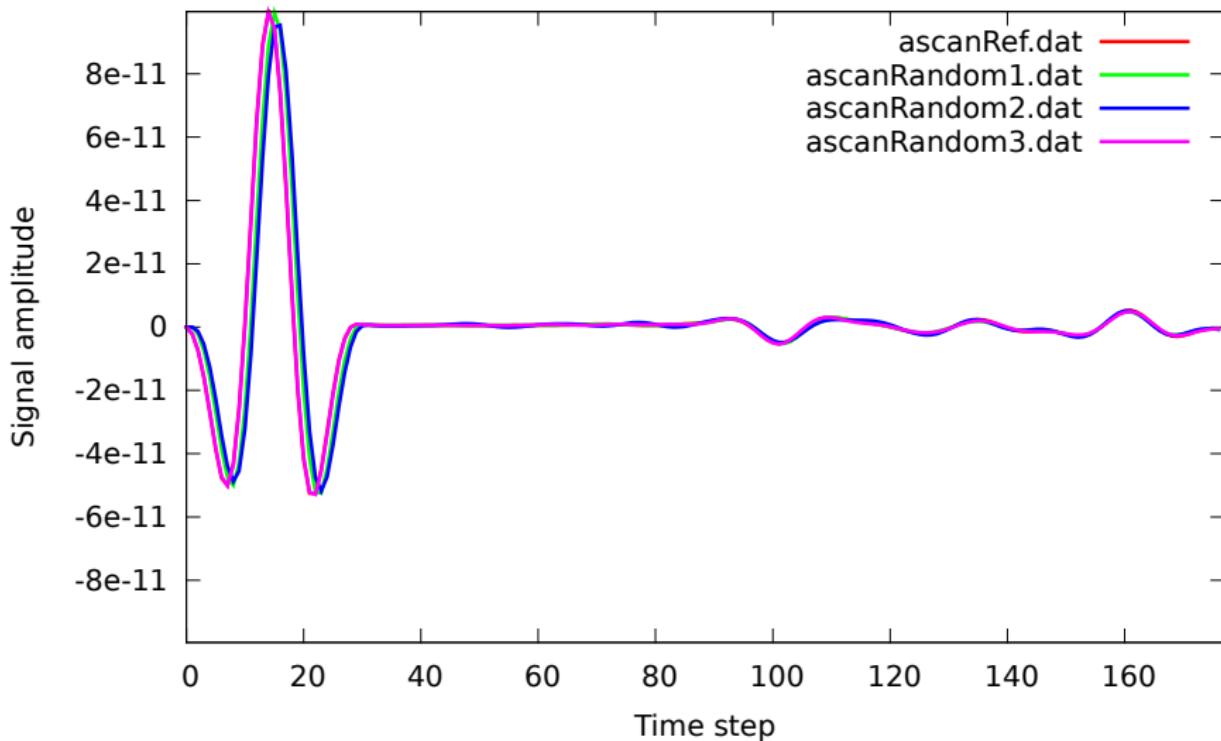
Application : CND par ultra-sons

Tests de non-régression dans verrou

	random 1	random 2	random 3	random 4
Cas-Test A				
ins1.dat	0	6.1e-06	6.1e-06	6.1e-06
ascan.dat	1.8e-12	5.9e-12	5.9e-12	5.9e-12
Cas-Test B				
sismo.dat	7.9e-69	7.9e-69	4.3e-69	4.3e-69
ascan.dat	1.2e-10	2.0e-11	2.8e-10	1.1e-11
Cas-Test C				
ins1.dat	4.6e-06	4.6e-06	4.6e-06	0
sismo.dat	8.0e-28	2.8e-28	8.0e-28	0
ascan.dat	2.0e-11	1.2e-11	1.8e-11	0
Cas-Test D				
ins1.dat	1.5e-18	4.1e-01	2.0e-01	0
enerloc.dat	0	2.3e-01	1.2e-01	0
sismo.dat	0	1.6e-01	3.2e-02	0
ascan.dat	0	1.5e-01	3.6e-01	6.5e-03

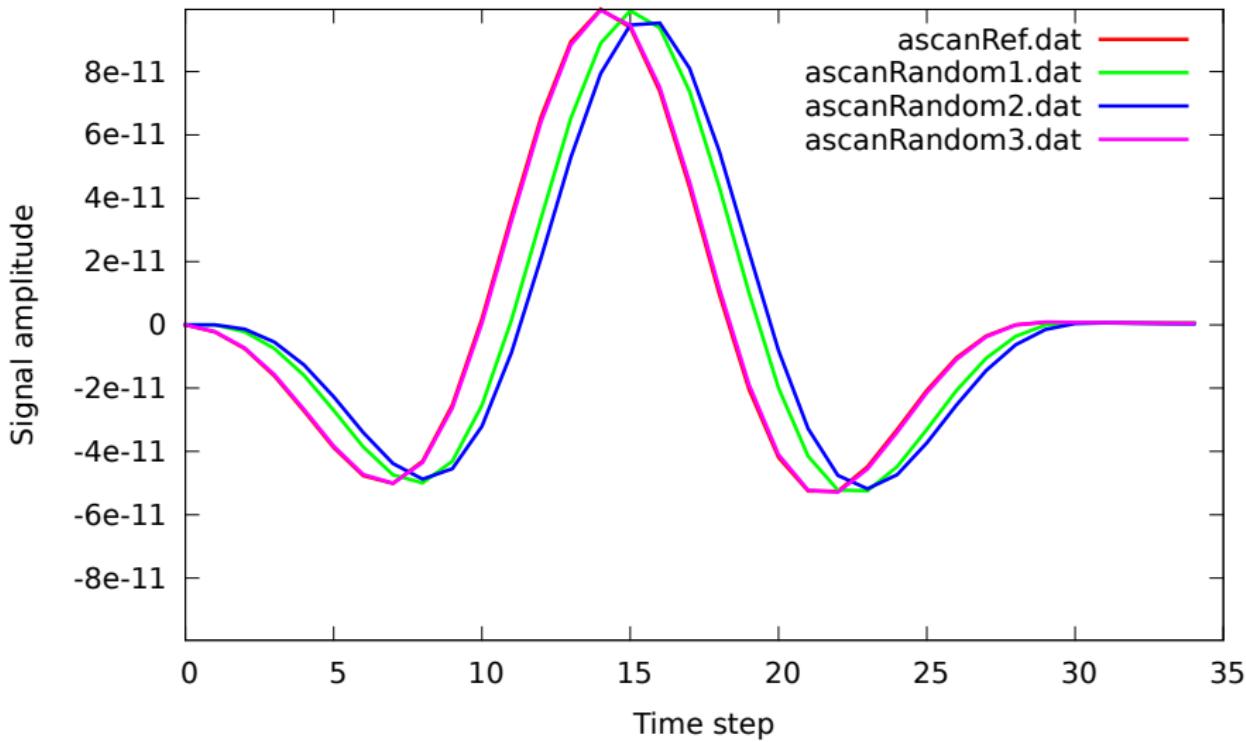
Application : CND par ultra-sons

Cas-test problématique



Application : CND par ultra-sons

Cas-test problématique



Application : CND par ultra-sons

Besoin de définir des normes adaptées

- ▶ Les écarts en amplitude et en temps de vol sont acceptables pour les utilisateurs.

Mise en avant d'un défaut de discrétisation

- ▶ Discontinuité dans le positionnement du capteur.

Objectif de la correction à venir

- ▶ Gagner en confiance,
- ▶ Gagner du temps pour les phases de vérification,
- ▶ Etre prêt pour de nouvelles utilisations demandant plus de précision.

Application : CND par ultra-sons

Performances de verrou

	reference	random	average
Cas-Test A	4.70s	83.23s (x17)	90.49s (x19)
Cas-Test B	29.79s	969.54s (x32)	1042.02s (x34)
Cas-Test C	21.15s	326.81s (x15)	358.08s (x16)
Cas-Test D	1.99s	24.20s (x12)	25.87s (x12)
Cas-Test E	0.46s	7.88s (x17)	8.88s (x19)
Cas-Test F	0.38s	4.54s (x11)	4.95s (x12)
Cas-Test G	6.16s	100.31s (x16)	109.70s (x17)
Cas-Test H	14.09s	503.90s (x35)	549.50s (x39)
Cas-Test I	1.48s	14.34s (x9)	14.85s (x10)

Surcoût compris entre ×9 et ×39



Localisation & correction

1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
 - Tour d'horizon
 - Bisection
 - Couverture de code
 - Corrections
4. Conclusions – perspectives



Localisation & correction

Tour d'horizon

1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
Tour d'horizon
Bisection
Couverture de code
Corrections
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)
- ▶ delta-debugging (ex : Verrou)
- ▶ comparaison de couvertures d'exécutions (ex : Verrou + gcov)



Localisation & correction

Bisection

1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
 - Tour d'horizon
 - Bisection
 - Couverture de code
 - Corrections
4. Conclusions – perspectives

Application : planification de production

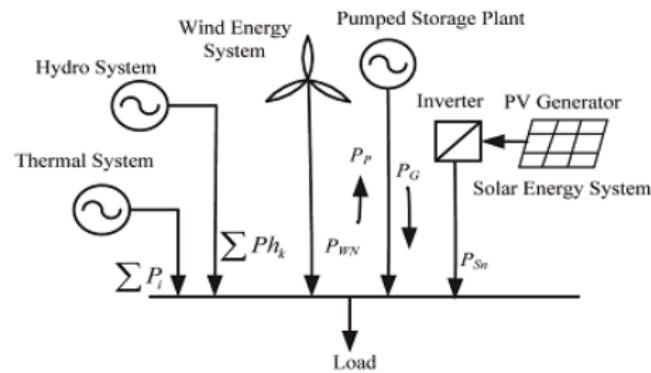
Code Apogee

Planification court terme

- ▶ assurer l'équilibre production = consommation,
- ▶ minimiser les coûts de production.

Code Apogee v1

- ▶ 300k+ lignes de Fortran,
- ▶ "boîte noire":
 - ▶ aucune connaissance préalable,
 - ▶ difficile à recompiler ;
- ▶ dépendance à IBM ILOG CPLEX.



Objectifs

- ▶ investiguer un problème de non-reproductibilité en cas de re-numérotation des vallées hydrauliques

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 ddmax
ecrpl_         ./apogene.release
imovi_         ./apogene.release
resopt_         ./apogene.release
getgrp_marginal_ ./apogene.release
ecrpla_        ./apogene.release
fin_exec_main_ ./apogene.release
# deceipt_pi_  ./apogene.release ddmax
paraend_       ./apogene.release
resopt_cnt_zones_ ./apogene.release
apstop_        ./apogene.release
# ihyd_         ./apogene.release ddmax
impression_info_ ./apogene.release
coupla_        ./apogene.release
gere_print_plath_ ./apogene.release
log            ./apogene.release
thepla_        ./apogene.release
# coutot_       ./apogene.release ddmax
# iprit_        ./apogene.release ddmax
```

► Delta-Debugging [Zel99]



► Entrées :

- ▶ script de lancement
- ▶ script de comparaison

► Sortie :

- ▶ DDmax:
 - ▶ ensemble maximal de fonctions ne générant pas d'erreur
 - ▶ renvoie le complémentaire

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 ddmin
# 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 ddmin
# 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 ne générant pas d'erreur
 - ▶ renvoie le complémentaire
- ▶ DDmin:
 - ▶ ensemble minimal de fonctions générant une erreur

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 ddmin0
# 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 ddmin0
# paraend_       ./apogene.release
# resopt_cnt_zones_ ./apogene.release
# apstop_        ./apogene.release
# ihyd_          ./apogene.release ddmin1
# impression_info_ ./apogene.release
# coupla_        ./apogene.release
# gere_print_plath_ ./apogene.release
# log            ./apogene.release
# thepla_        ./apogene.release
coutot_          ./apogene.release ddmin2
iprit_           ./apogene.release ddmin2
```

► Delta-Debugging [Zel99]



► Entrées :

- ▶ script de lancement
- ▶ script de comparaison

► Sortie :

- ▶ DDmax:
 - ▶ ensemble maximal de fonctions ne générant pas d'erreur
 - ▶ renvoie le complémentaire
- ▶ DDmin:
 - ▶ ensemble minimal de fonctions générant une erreur
- ▶ rDDmin:
 - ▶ appel récursif à DDmin

Application : planification de production

Apogene : Verrou + Delta Debugging

▶ Symboles instables :

```
couhyd_pi_
coutot_
decopt_pi_
ihyd_
iprit_
matctr_pi_
nzsv1_
opti_un_grth_
pildef_
prosca_
proxmyqn_
recrea_pi_
relax_vol_
remise_grad_
scale_hyd_
thpdyn_
```

▶ Lignes source instables :

```
COUHYD_PI.f:196
COUHYD_PI.f:197
COUHYD_PI.f:198
COUHYD_PI.f:199
COUHYD_PI.f:200
COUHYD_PI.f:203
COUHYD_PI.f:204
COUHYD_PI.f:205
COUHYD_PI.f:206
COUHYD_PI.f:208
COUHYD_PI.f:211
COUHYD_PI.f:212
COUHYD_PI.f:213
COUHYD_PI.f:215

COUTOT.f:61
COUTOT.f:64
COUTOT.f:65
COUTOT.f:70
COUTOT.f:91
COUTOT.f:96
COUTOT.f:98

:
```

Application : planification de production

Apogène : Verrou + Delta Debugging

▶ Symboles instables :

```
couhyd_pi_
coutot_
decopt_pi_
ihyd_
iprit_
matctr_pi_
nzsv1_
opti_un_grth_
pildef_
prosca_
proxmyqn_
recrea_pi_
relax_vol_
remise_grad_
scale_hyd_
thpdyn_
```

▶ Lignes source instables :

```
COUHYD_PI.f:196
COUHYD_PI.f:197
COUHYD_PI.f:198
COUHYD_PI.f:199
COUHYD_PI.f:200
COUHYD_PI.f:203
COUHYD_PI.f:204
```

- ▶ 1/2 journée pour mettre en place l'étude
- ▶ 4 jours de calcul (slow down = 9×)
- ▶ **instabilités trouvées !**

```
COUTOT.f:61
COUTOT.f:64
COUTOT.f:65
COUTOT.f:70
COUTOT.f:91
COUTOT.f:96
COUTOT.f:98
```

:



Localisation & correction

Couverture de code

1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
 - Tour d'horizon
 - Bisection
 - Couverture de code
 - Corrections
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
sndl112a	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
ssls108i	OK	OK	OK	OK
ssls108j	OK	OK	OK	OK
ssls108k	OK	OK	OK	OK
ssls108l	OK	OK	OK	OK
sndl112a	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	nearest	Statut rnd ₁	rnd ₂	rnd ₃	# chiffres décimaux en commun $C(\text{rnd}_1, \text{rnd}_2, \text{rnd}_3)$
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 20 minutes chacun
(72 cas tests)

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

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
```

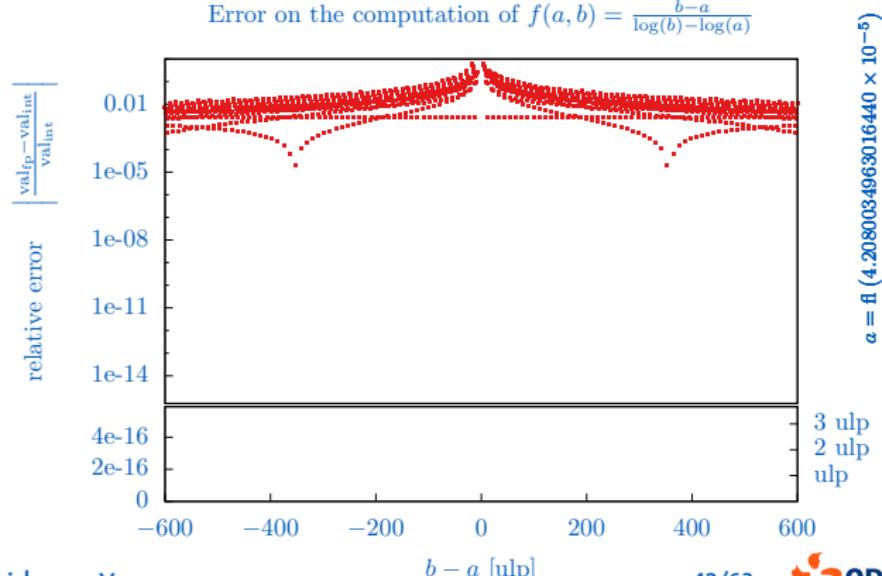
Correction

Formule de code_aster

$$f(a, b) = \begin{cases} a & \text{si } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{sinon} \end{cases}$$

Étude empirique

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



Correction

Formule de code_aster

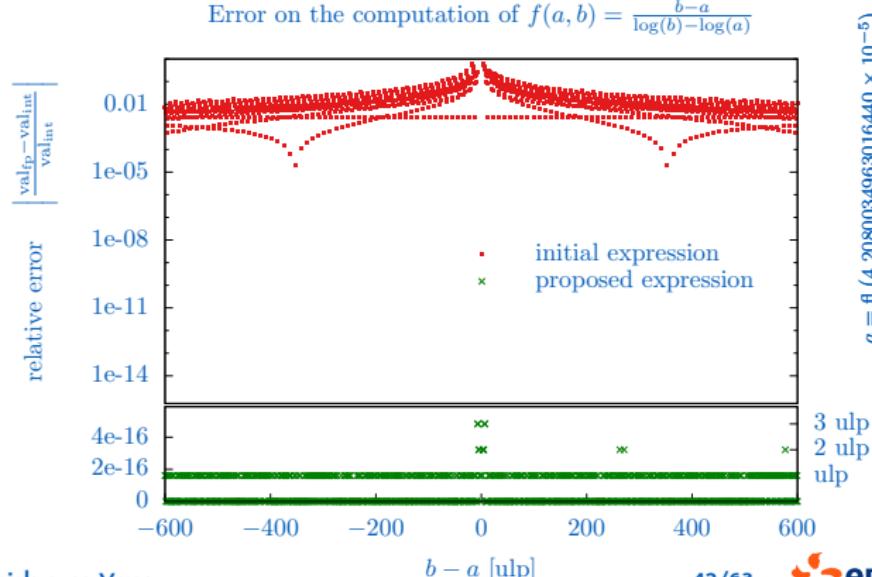
$$f(a, b) = \begin{cases} a & \text{si } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{sinon} \end{cases}$$

réécriture
manuelle

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

Étude empirique

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



Correction

Formule de code_aster

$$f(a, b) = \begin{cases} a & \text{si } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{sinon} \end{cases}$$

réécriture
manuelle

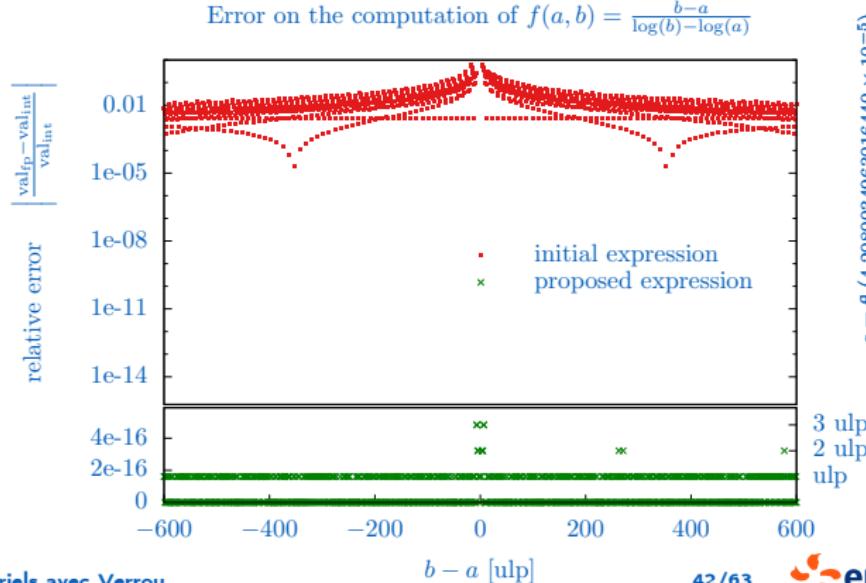
$$f(a, b) = \begin{cases} a & \text{si } a = b \\ a \frac{\frac{b}{a}-1}{\log(\frac{b}{a})} & \text{sinon} \end{cases}$$

Étude empirique

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

Preuve

- ▶ erreur bornée par 10 ulps



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 ? (→ 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;
```

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)
    continue
30   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
```

Localisation par bisection

Pouvait-on éviter de tester CompSum?

Option -source=ddmax.inc

mrmmvr.F90 116 mrmmvr_

mrmmvr.F90 117 mrmmvr_

mrmmvr.F90 119 mrmmvr_

Comparaison de la sensibilité aux additions et aux multiplications

Verrou option	$-\log_{10} \left(\frac{\hat{\sigma}}{\hat{\mu}} \right)$
défaut (toutes les instructions)	5.95
-vr-instr=add	6.17
-vr-instr=mul	6.26

Conclusions

Estimation de la fidélité après correction

sdnl112a

Version	nearest	Statut			# chiffres en commun $C(\text{rnd}_1, \text{rnd}_2, \text{rnd}_3)$
		rnd_1	rnd_2	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]

Perte de précision

Diagnostic et localisation

Introduction d'erreurs

▶ Representation

$\pi = 3.14159\textcolor{red}{265}...$

Diagnostic : NON

▶ Absorption

$$\begin{array}{r} 3.14159 \\ + 0.00141421 \\ \hline 3.14300\textcolor{red}{421} \end{array}$$

Diagnostic : OUI

Localisation : Delta-Debugging

Perte de précision

Diagnostic et localisation

Introduction d'erreurs

▶ Representation

$$\pi = 3.14159\textcolor{red}{265}\dots$$

Diagnostic : **NON**

▶ Absorption

$$\begin{array}{r} 3.14159 \\ + 0.00141421 \\ \hline 3.14300\textcolor{red}{421} \end{array}$$

Diagnostic : **OUI**

Localisation : Delta-Debugging

Amplification d'erreurs

▶ Annulation

$$\begin{array}{r} 3.14300 \\ - 3.14159 \\ \hline 0.00141\textcolor{red}{000} \end{array}$$

Diagnostic : **OUI**

Localisation : désactivée

▶ Branchements instables

Diagnostic : **OUI**

Localisation : couverture



Localisation & correction

Corrections

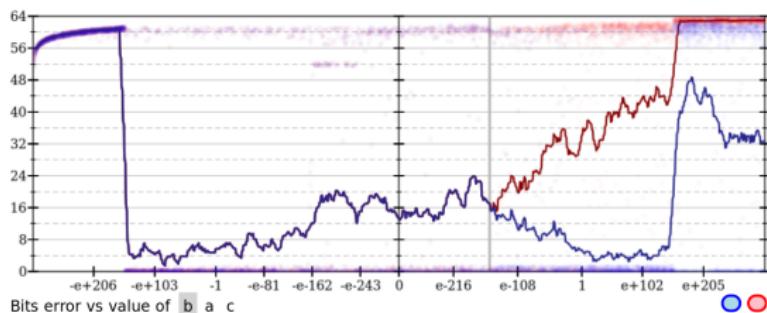
1. Contexte & enjeux
2. Diagnostic
3. Localisation & correction
 - Tour d'horizon
 - Bisection
 - Couverture de code
 - Corrections
4. Conclusions – perspectives

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 $ax^2 + bx + c$

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



$$x^+ = \begin{cases} \frac{-b + \sqrt{b^2 - 4ac}}{2a} & \text{si } b \leq 0 \\ \frac{2c}{-b - \sqrt{b^2 - 4ac}} & \text{sinon} \end{cases}$$

$(-b + \sqrt{b^2 - 4ac})/(2a)$

Correction

Algorithmes classiques

► Pour toute une classe d'algorithmes classiques, il existe des variantes "compensées" :

- ▶ somme des éléments d'un vecteur
- ▶ produit scalaire
- ▶ évaluation polynomiale
- ▶ ...

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

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

tels que

$$r = [x \circ y]$$

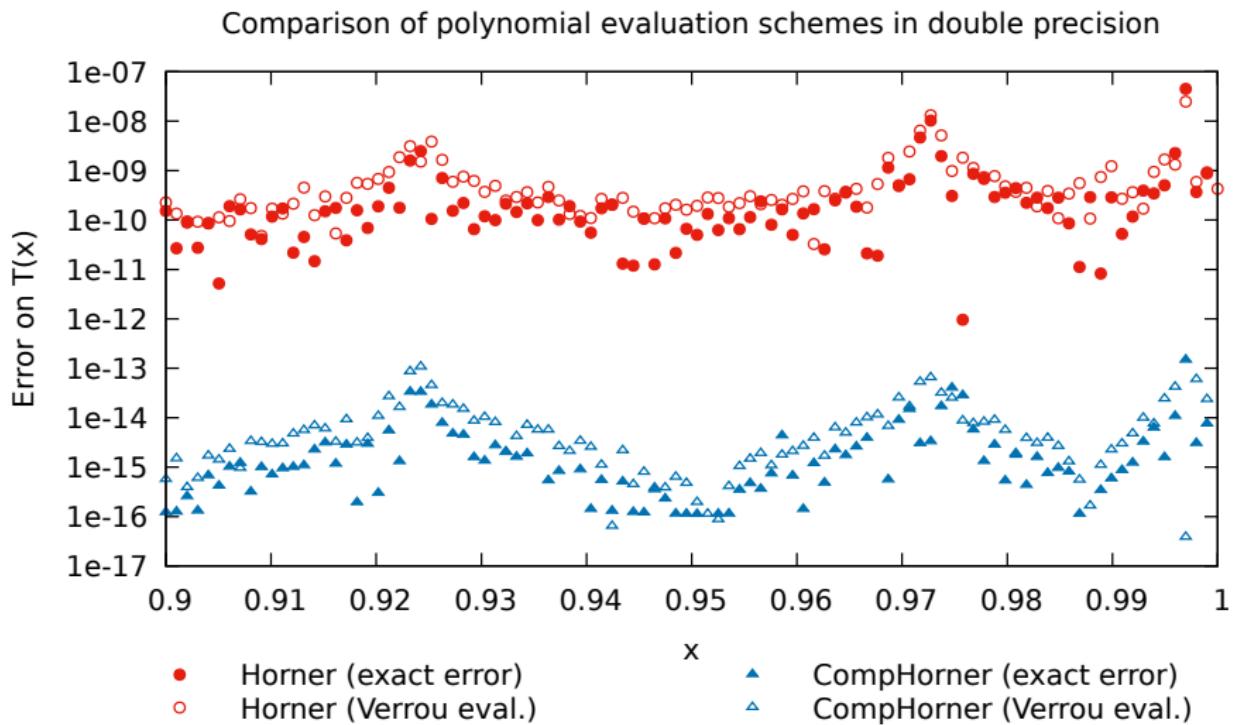
$$r + \delta = x \circ y$$

ex : LibEFT implémente

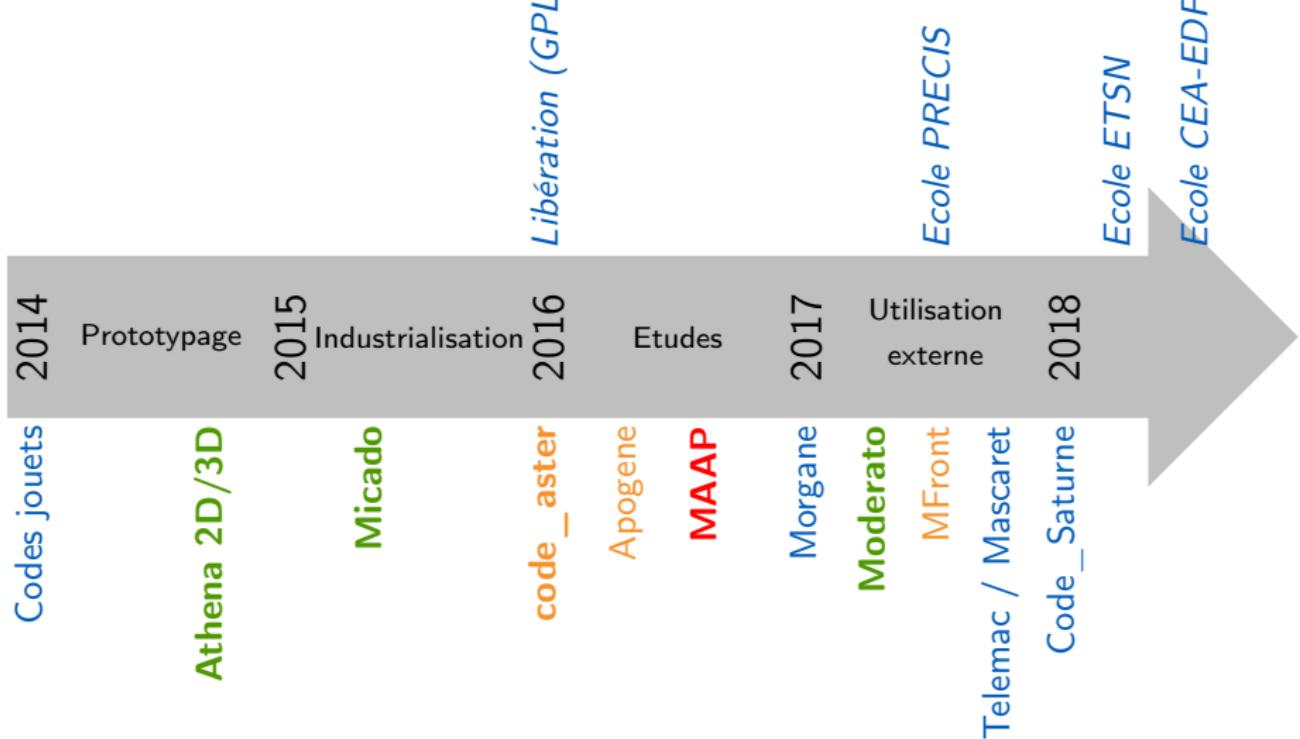
- des transformations exactes
- pour +, - et \times
- en simple et double précision

Correction

Algorithmes compensés, exemple : évaluation polynomiale



Bref historique de Verrou



Actualité Verrou

Nouvelle version 2.0 : 18/06/2018

- ▶ Generation of Valgrind errors for NaN values
- ▶ Instrumentation of all FP binary instructions, as obtained by any combination of:
 - ▶ an operation: ADD / SUB / MUL / DIV
 - ▶ a vector variant: LLO / SSE / AVX2
 - ▶ a precision: single / double
- ▶ Instrumentation of cast instructions (double -> float)
- ▶ Preparation for the common interflop backend interface
- ▶ Bug fix : compatibility with Intel compiler and openblas
- ▶ Verification improvment
- ▶ Various bug fixes
- ▶ ...

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.

Perspectives

- ▶ intégration Interflop ;
- ▶ gérer toutes les instructions
 - ▶ instructions scalaires x87 ;
- ▶ conforter les fonctionnalités de Delta-Debugging ;
- ▶ conforter les fonctionnalités de localisation de branchements instables ;
- ▶ gérer “proprement” la libmath.

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"

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 International 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.



Annexes

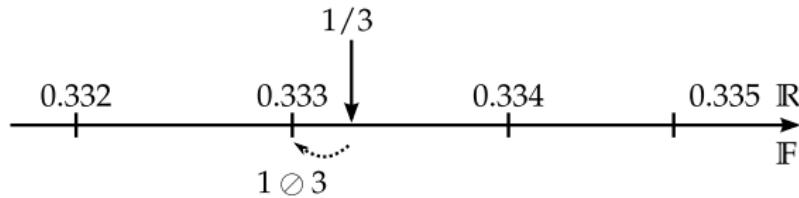
- ① Arithmétique flottante
- ② L'outil Verrou
- ③ Traitement de la division

Contexte - Vérification numérique

Arithmétique flottante



- ▶ Représentation à virgule flottante en précision limitée
 - ▶ [nos exemples] décimal, 3 chiffres significatifs (% – %): 42.0, 0.123
 - ▶ [float] binaire, 24 bits significatifs ($\simeq 10^{-7}$)
 - ▶ [double] binaire, 53 bits significatifs ($\simeq 10^{-16}$)



- ▶ Conséquences : calcul flottant \neq calcul réel
 - ▶ arrondi $a \oplus b \neq a + b$
 - ▶ perte d'associativité $(a \oplus b) \oplus c \neq a \oplus (b \oplus c)$

Contexte - Vérification numérique

Erreurs de calcul



► Quelques algorithmes à risque:

- ▶ accumulation de beaucoup de calculs dans un résultat → absorption
 - ▶ somme, produit scalaire...
- ▶ soustractions de nombres proches → annulation
 - ▶ calcul d'erreur / écart par rapport à une référence
- ▶ cumul de plusieurs situations précédentes

▶ norme d'un vecteur d'écart : $\varepsilon = \sum_i |v_i - v_i^{ref}|$

▶ variance, écart-type : $\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N \left[v_i - \frac{1}{N} \sum_{j=1}^N v_j \right]^2}$

Contexte - Vérification numérique

Erreurs de calcul



► Quelques algorithmes à risque:

- ▶ accumulation de beaucoup de calculs dans un résultat → absorption
 - ▶ somme, produit scalaire...
- ▶ soustractions de nombres proches → annulation
 - ▶ calcul d'erreur / écart par rapport à une référence
- ▶ cumul de plusieurs situations précédentes

▶ norme d'un vecteur d'écart : $\varepsilon = \sum_i |v_i - v_i^{ref}|$

▶ variance, écart-type : $\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N \left[v_i - \frac{1}{N} \sum_{j=1}^N v_j \right]^2}$

► Quis custodiet ipsos custodes?

- ▶ qui valide la référence ?

Instrumentation binaire avec valgrind



▶ Code C :

```
i++;
```

```
y = a * x + b;
```

▶ Entrée valgrind :

```
i = Add32(i, 1)
```

```
t1 = MulF32(a, x)
```

```
y = AddF32(t1, b)
```

▶ Sortie valgrind :

```
i = Add32(i, 1)
```

```
Call("count", MUL)  
t1 = MulF32(a, x)
```

```
Call("count", ADD)
```

```
Call("detectCancel", t1, b)  
y = Call("myAdd", t1, b)
```

▶ Possibilités d'instrumentation:

- ▶ Ne rien faire (recopier l'instruction telle quelle)
- ▶ Compter les instructions
- ▶ Déetecter des erreurs
- ▶ Remplacer les opérations

Valgrind ne gère que le mode d'arrondi NEAREST

Transformation approchée pour la division



► Ce qu'on cherche :

$$\frac{a}{b} = q + r,$$

avec $q = \text{fl}(a/b)$.

► Algorithme proposé :

Input : a, b

Output : \tilde{r} tel que

$$a/b \simeq \text{fl}(a/b) + \tilde{r}.$$

$$q \leftarrow \text{fl}(a/b)$$

$$(p, s) \leftarrow \text{twoprod}(b, q)$$

$$t \leftarrow \text{fl}(a - p)$$

$$u \leftarrow \text{fl}(t - s)$$

$$\tilde{r} \leftarrow \text{fl}(u/b)$$

► Idée de la preuve :

$$q = \frac{a}{b} (1 + \epsilon_1)$$

$$p = b q (1 + \epsilon_2)$$

$$= a (1 + \epsilon_1) (1 + \epsilon_2)$$

$$t = a - p \quad (\text{Lemme de Sterbenz})$$

$$u = (t - s) (1 + \epsilon_3)$$

$$= (a - (p + s)) (1 + \epsilon_3)$$

$$= (a - bq) (1 + \epsilon_3)$$

$$= br (1 + \epsilon_3)$$

$$\tilde{r} = \frac{u}{b} (1 + \epsilon_4)$$

$$= r (1 + \epsilon_3) (1 + \epsilon_4).$$