Verrou : l'arithmétique stochastique sans recompilation

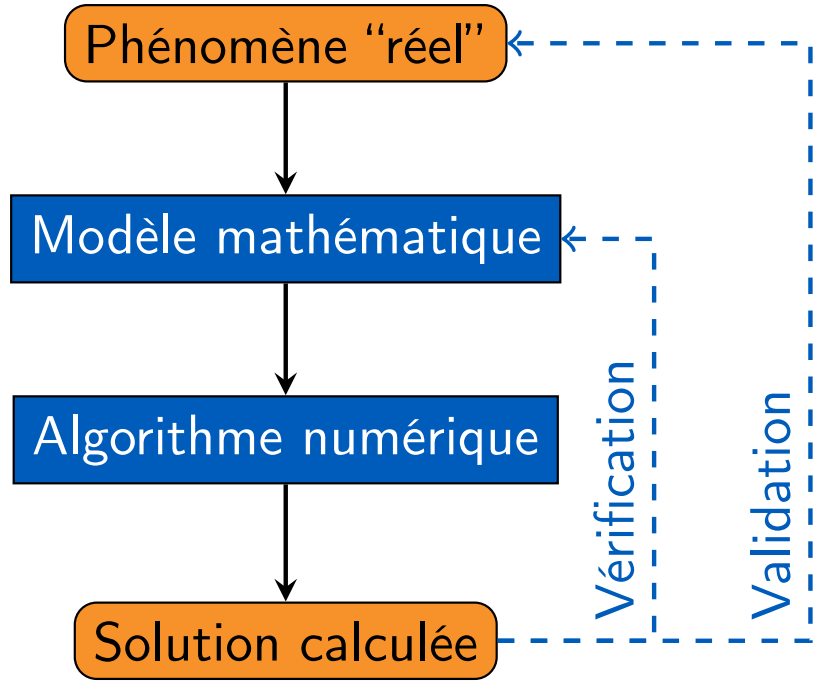
Demandez le Programme !
17 octobre 2017

François Févotte
Bruno Lathuilière

EDF R&D
PERICLES / I23

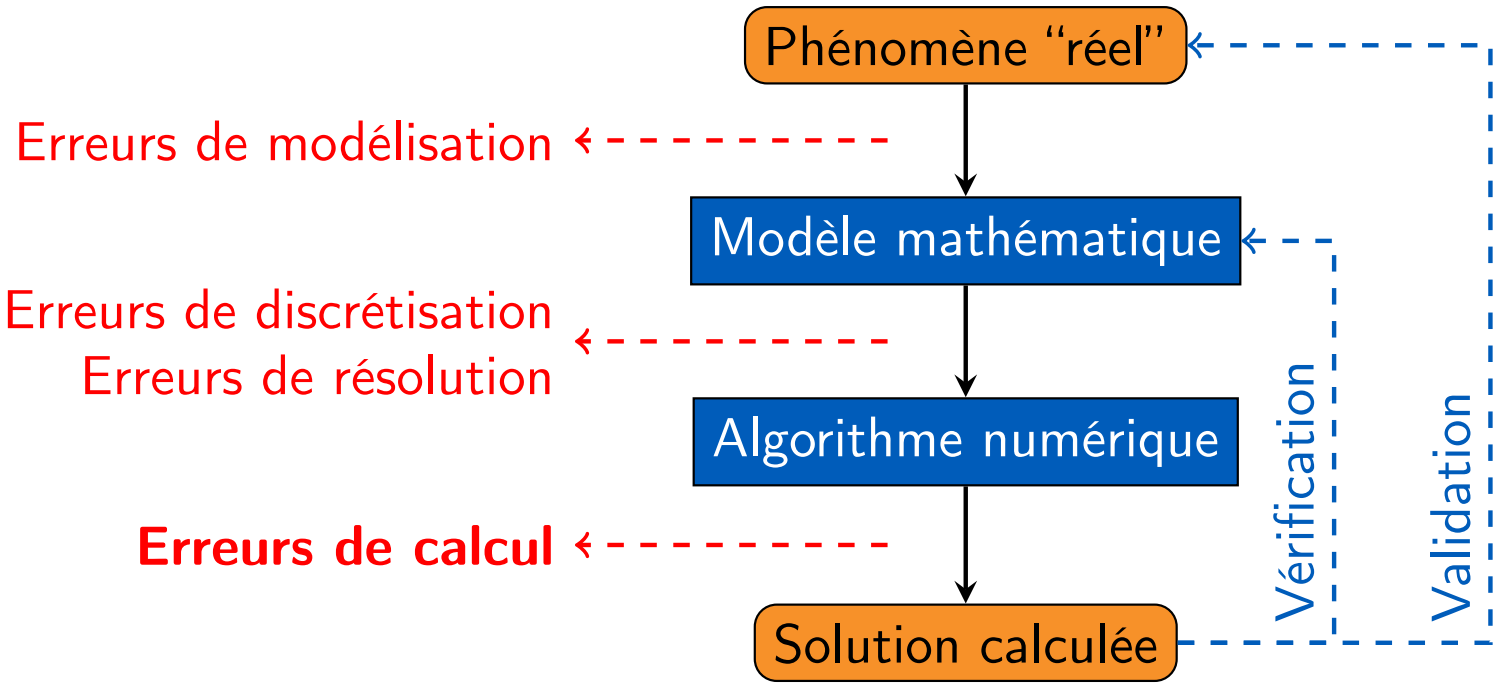
Contexte & enjeux

Vérification & Validation



Contexte & enjeux

Vérification & Validation



Quantification des erreurs : enjeux

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

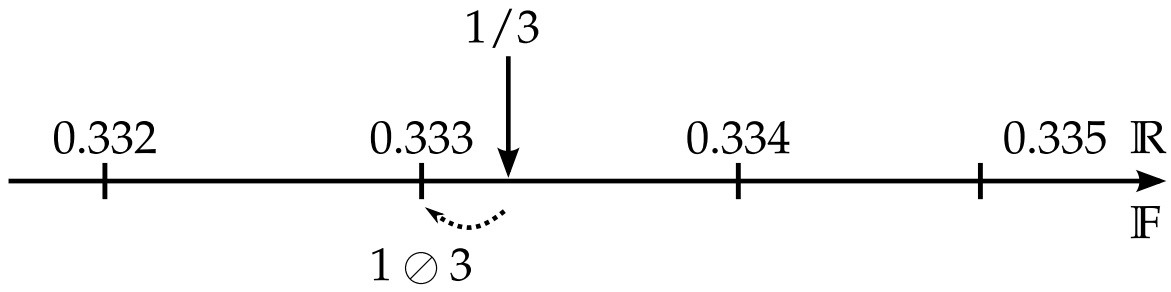
Plan

1. Arithmétique flottante
2. Méthode CESTAC & Verrou
3. L'outil Verrou : Démo
4. Application : code_aster
5. Conclusions – perspectives

Arithmétique flottante

De quoi s'agit-il ?

- 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 $(a \oplus b) \oplus c \neq a \oplus (b \oplus c)$

Arithmétique flottante

Perte de précision

- Absorption

$$\begin{array}{rr} & 3.14159 \\ + & 0.00141421 \\ \hline & 3.14300421 \end{array}$$

- Annulation

$$\begin{array}{rr} & 3.14300 \\ - & 3.14159 \\ \hline & 0.00141xxx \end{array}$$

Arithmétique flottante

Erreurs de calcul

- Quelques algorithmes à risque:
  - **accumulation de beaucoup de calculs dans un résultat** $\rightarrow$ absorption
    - somme, produit scalaire...
  - **soustractions de nombres proches** $\rightarrow$ annulation
    - calcul d'erreur / écart par rapport à une référence
  - cumul de plusieurs situations précédentes
    - **norme d'un vecteur d'écarts** : $\varepsilon = \sum_i \left| v_i - v_i^{ref} \right|$
    - variance, écart-type : $\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[v_i - \frac{1}{N} \sum_{j=1}^{N} v_j \right]^2}$

Arithmétique flottante

Erreurs de calcul

- Quelques algorithmes à risque:
  - **accumulation de beaucoup de calculs dans un résultat** $\rightarrow$ absorption
    - somme, produit scalaire...
  - **soustractions de nombres proches** $\rightarrow$ annulation
    - calcul d'erreur / écart par rapport à une référence
  - cumul de plusieurs situations précédentes
    - **norme d'un vecteur d'écarts** : $\varepsilon = \sum_i \left| v_i - v_i^{ref} \right|$
    - 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 ?**

Arithmétique flottante

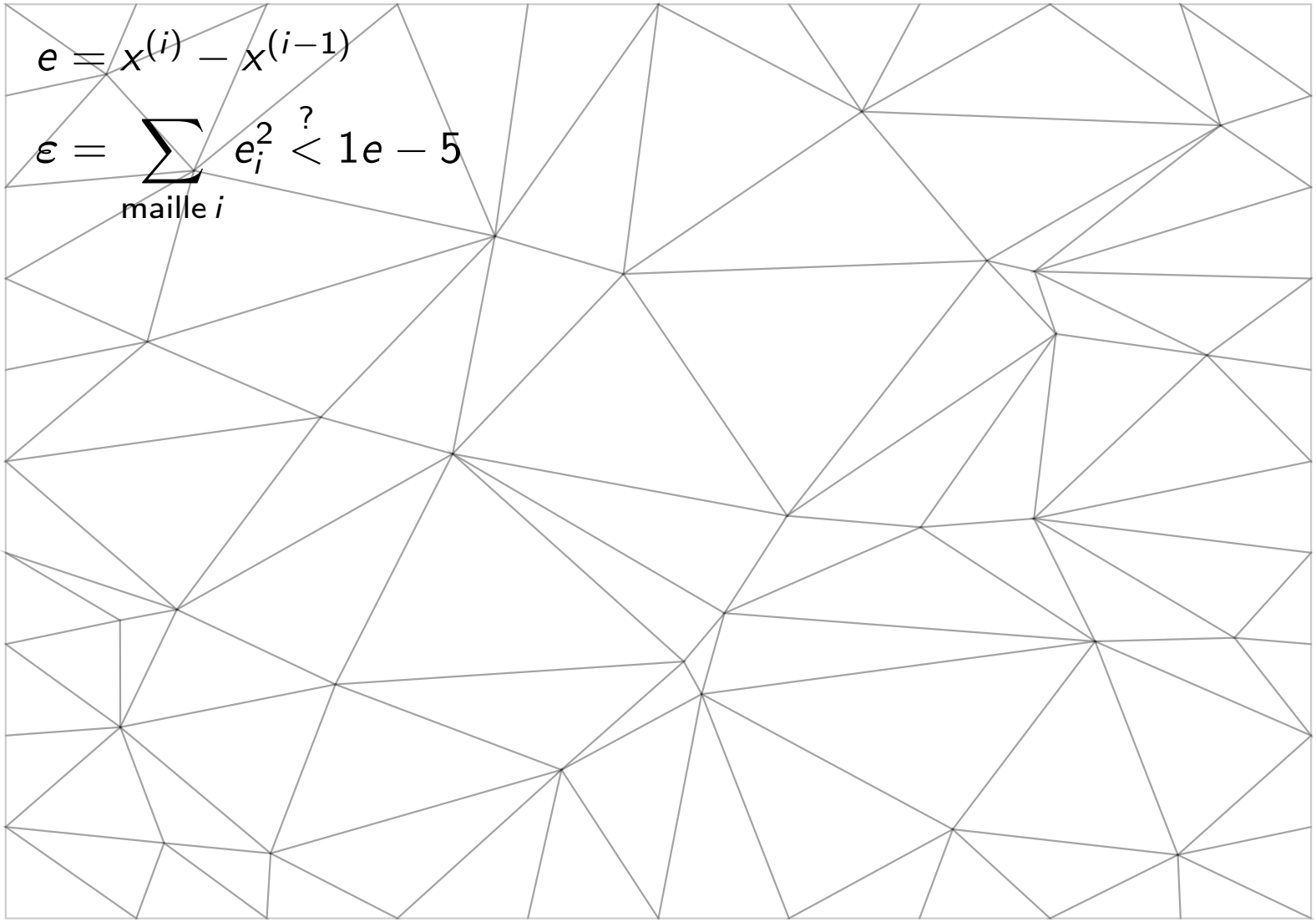
Quelles conséquences ?

Conséquences possibles :

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

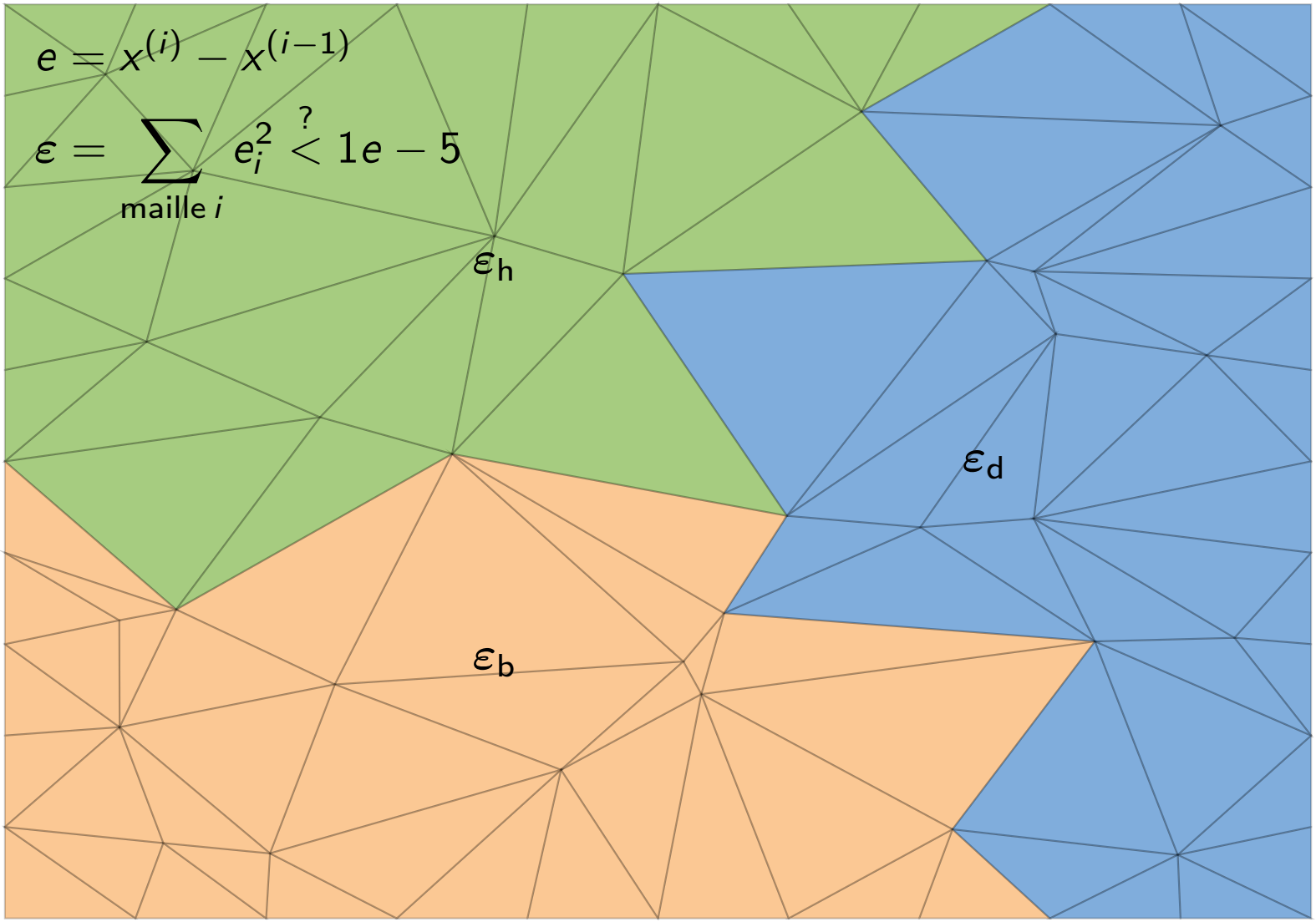
Arithmétique flottante

Calcul bloqué : ex. Telemac2D



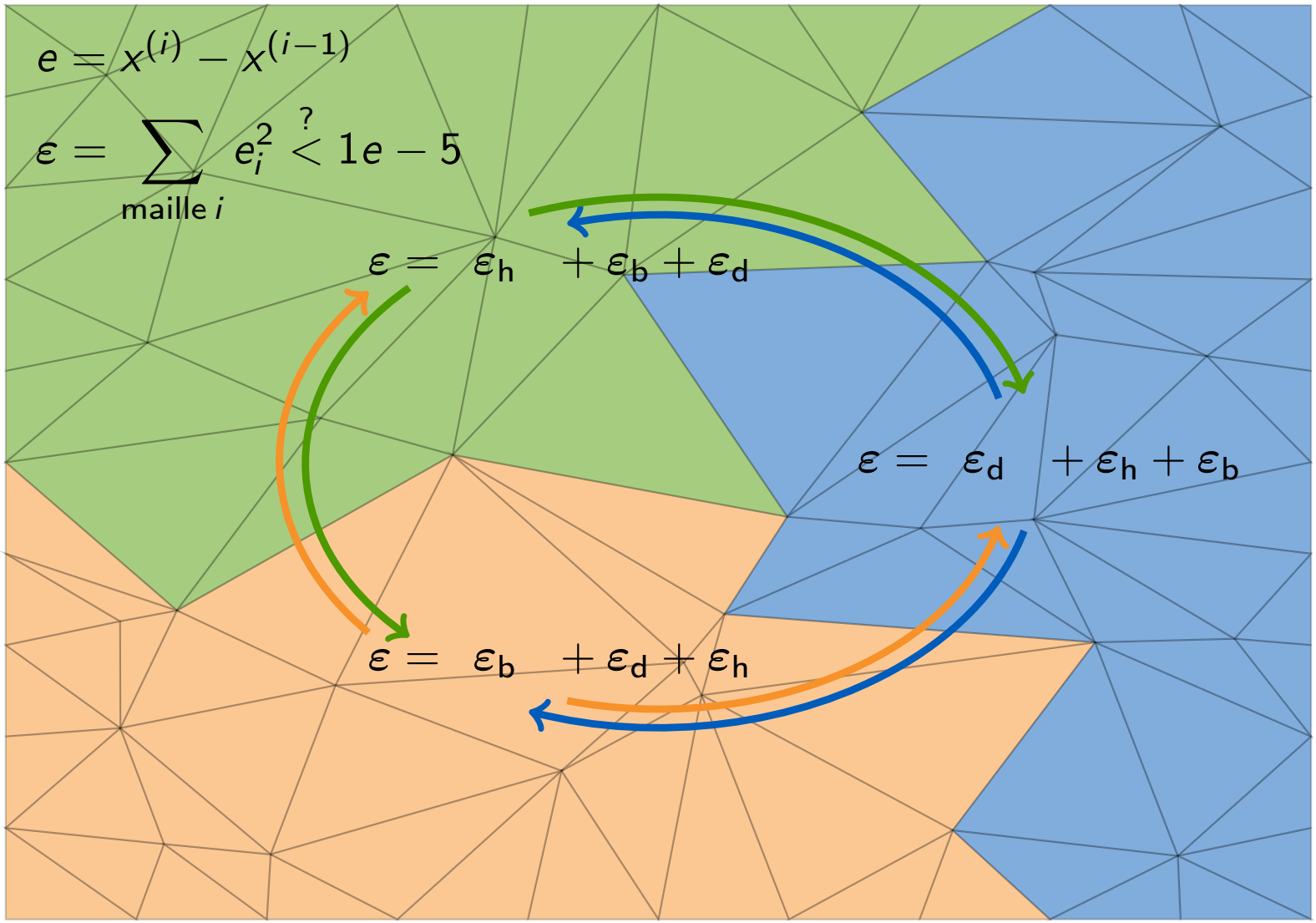
Arithmétique flottante

Calcul bloqué : ex. Telemac2D



Arithmétique flottante

Calcul bloqué : ex. Telemac2D



Arithmétique flottante

Quelles conséquences ?

Conséquences possibles :

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

Arithmétique flottante

Performances : ex. optimisation production hydraulique

- Objectif : maximiser le revenu ⚠ **Annulation possible**

$$\rho = \max_{p,v} \underbrace{\left[\sum_{t \in T} \lambda_t \, p_t \right.}_{\text{puissance produite}} \underbrace{\left. + \sum_{r \in R} \omega_r \, \left(v_r^{\text{fin}} - v_r^{\text{ini}} \right) \right]}_{\text{eau en réserve}},$$

Arithmétique flottante

Performances : ex. optimisation production hydraulique

- Objectif : maximiser le revenu

$$\rho = \max_{p,v} \left[\sum_{t \in T} \lambda_t \, p_t + \sum_{r \in R} \omega_r \, v_r^{\text{fin}} \right] - \sum_{r \in R} \omega_r \, v_r^{\text{ini}},$$

- Gain en performance : supérieurs à $\times 50$ dans de nombreux cas

Arithmétique flottante

Quelles solutions ?

- De nombreuses solutions existent :
  - changements de formulation, pré-conditionnement. . .
  - algorithmes (rejouables $\rightarrow$ reproductibles $\rightarrow$ compensés $\rightarrow$ exacts)
  - précision supérieure (`float` $\rightarrow$ `double` $\rightarrow$ `quad` $\rightarrow$ MPFR $\rightarrow$ intervalles)
  - tolérances dans les comparaisons

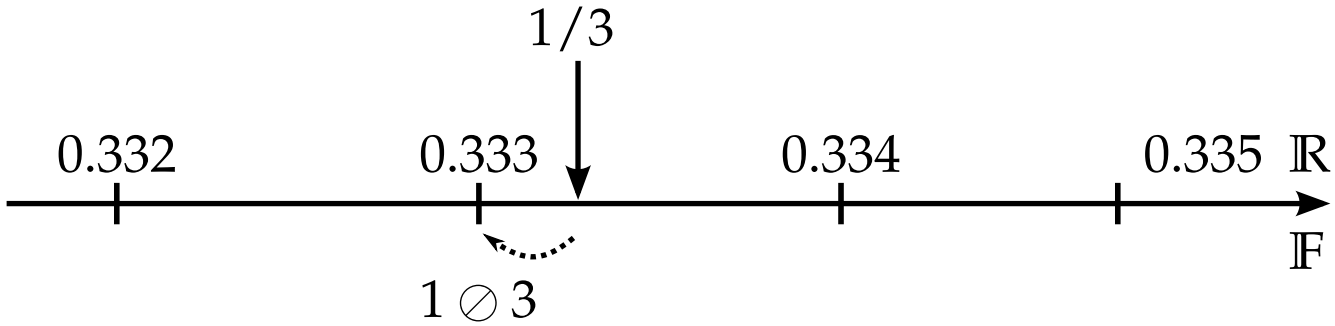
- Besoin = détection des problèmes
  - existence
  - ampleur
  - localisation

Méthode CESTAC & Verrou

Méthode CESTAC

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

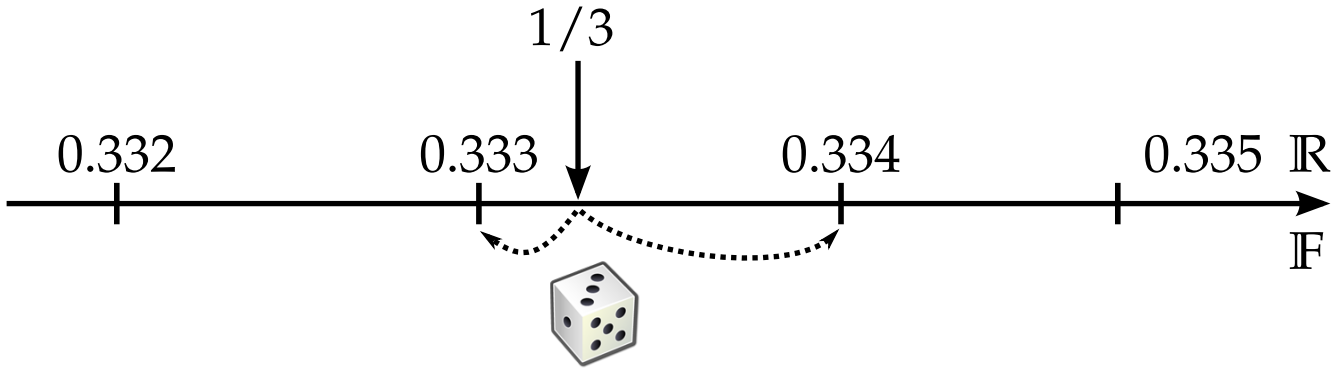
Arrondi au plus proche (défaut IEEE-754)



Méthode CESTAC

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

Arrondi aléatoire (CESTAC)



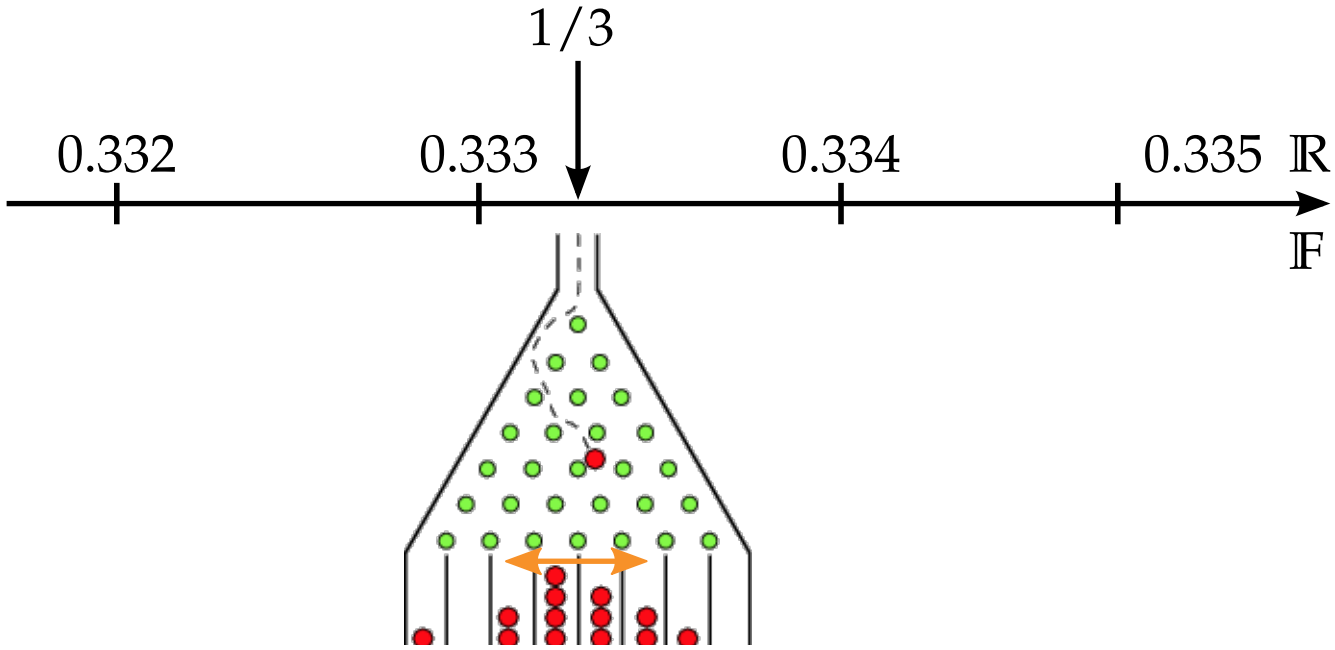
[1] J. Vignes, "A stochastic arithmetic for reliable scientific computation," *Mathematics and Computers in Simulation*, vol. 35, no. 3, 1993.

[2] J.-L. Lamotte, J.-M. Chesneaux and F. Jézéquel, "CADNA_C: A version of CADNA for use with C or C++ programs", *Computer Physics Communications*, vol. 181, no. 11, 2010.

Méthode CESTAC

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

Arrondi aléatoire (CESTAC)

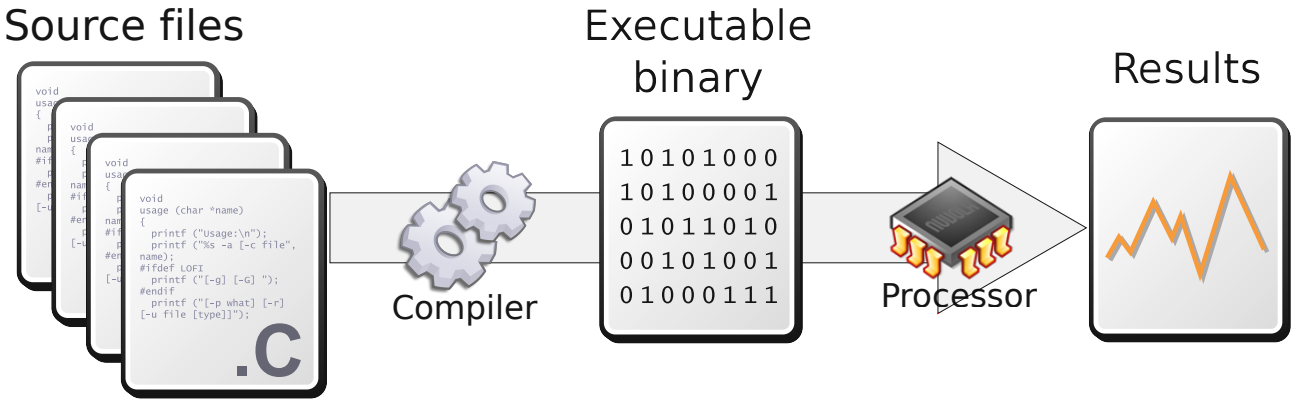


Instruction	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**0.33**4
$b = a \times 3$	0.999	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.00**

Méthode CESTAC

CADNA : Analyse dynamique des sources

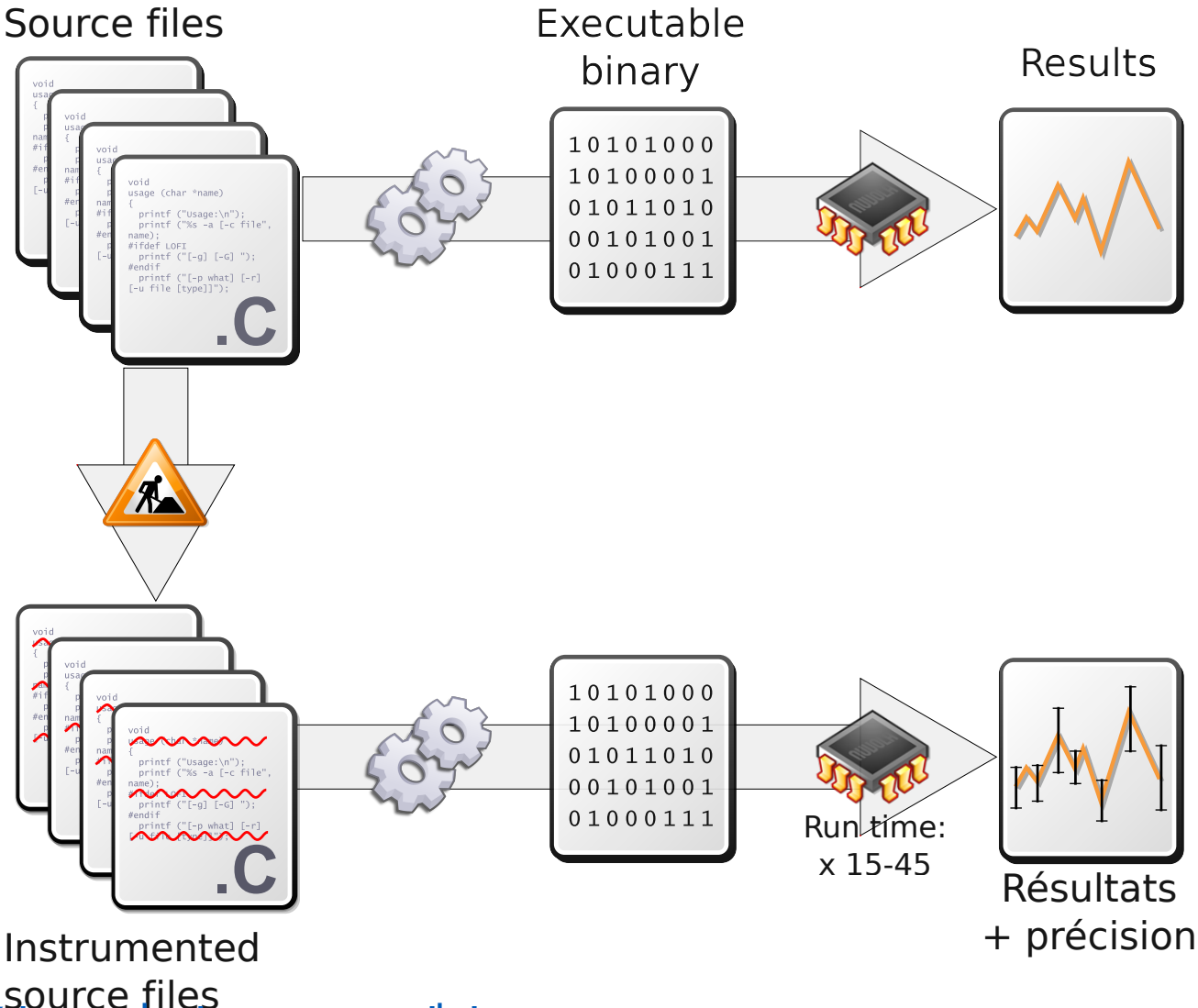
```
$ athena2d casTest
```



Méthode CESTAC

CADNA : Analyse dynamique des sources

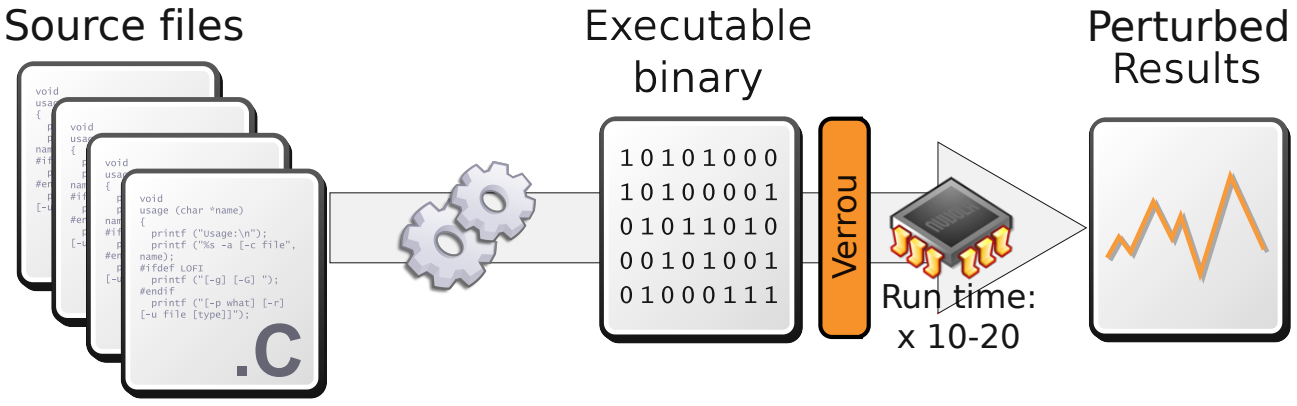
$ athena2d-cadna `casTest`



Méthode CESTAC

Verrou : Analyse dynamique du binaire

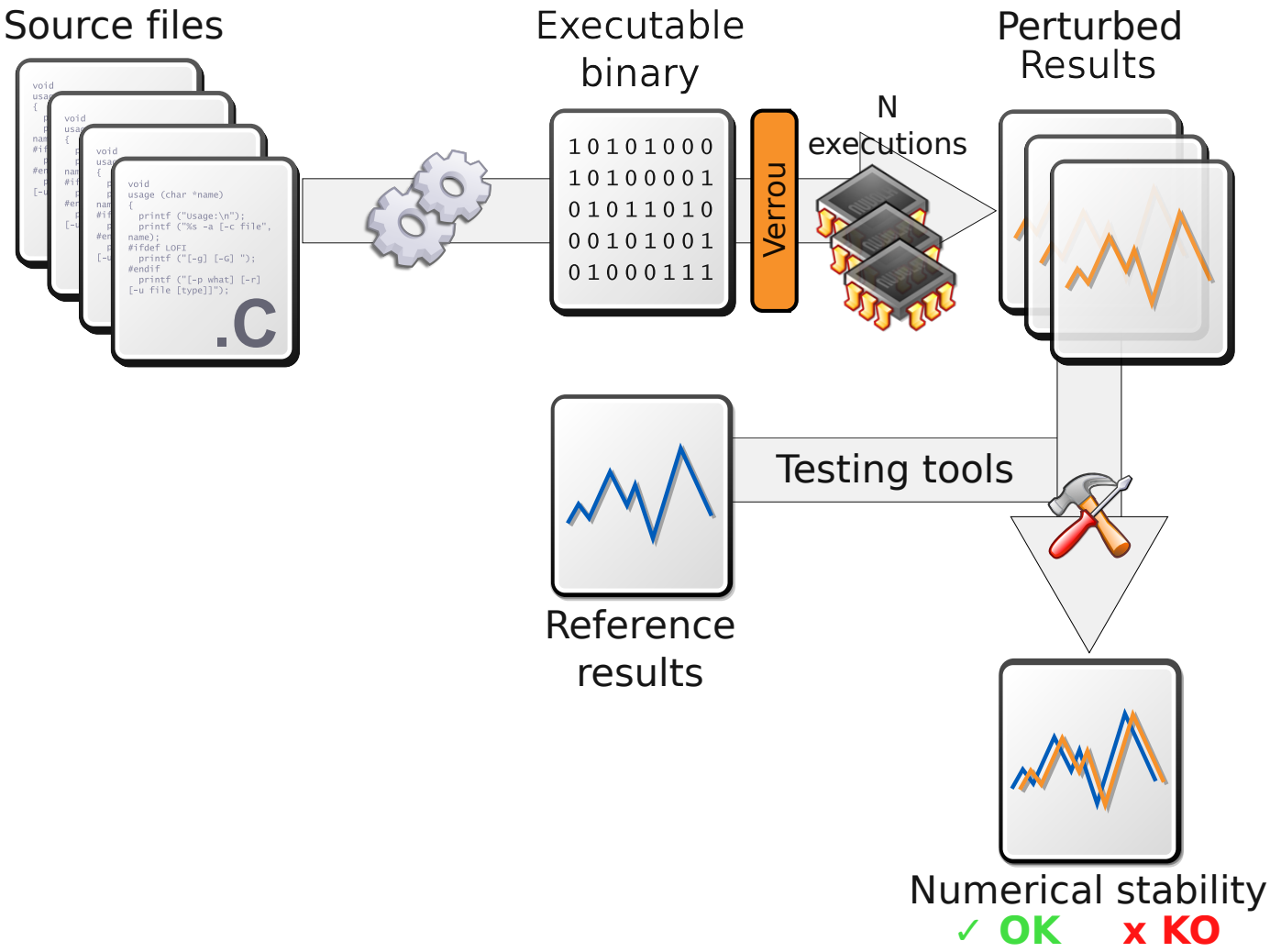
```
$ valgrind --tool=verrou --rounding-mode=random athena2d casTest
```



Méthode CESTAC

Verrou : Analyse dynamique du binaire

```
$ valgrind --tool=verrou --rounding-mode=random athena2d casTest
```



L'outil Verrou : Démo

L'outil Verrou

Démonstration : premiers pas avec Python

- lancement : comme `valgrind`
- perturbation des modes d'arrondi
- ... seulement s'il y a un arrondi
- problème avec la `libm`
- listes d'exclusion (deux méthodes)

L'outil Verrou

Démonstration : un calcul d'intégrales

- Objectif : calculer une intégrale du type :

$$I = \int_a^b f(x)\,\mathrm{d}x.$$

- Méthode implémentée : n rectangles de largeur $h = \frac{b-a}{n}$ et de centres x_i

$$I_n = \sum_{i=1}^{n} f(x_i)\ h.$$

Cas-test : $f = \cos$, $[a, b] = \left[0, \frac{\pi}{2}\right]$

$$I = \int_0^{\frac{\pi}{2}} \cos(x)\,\mathrm{d}x = 1 \qquad I_n \xrightarrow[n\to\infty]{?} I$$

L'outil Verrou

Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

L'outil Verrou

Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

L'outil Verrou

Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

L'outil Verrou

Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

L'outil Verrou

Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

L'outil Verrou

Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

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

Application : code_aster

Application : mécanique

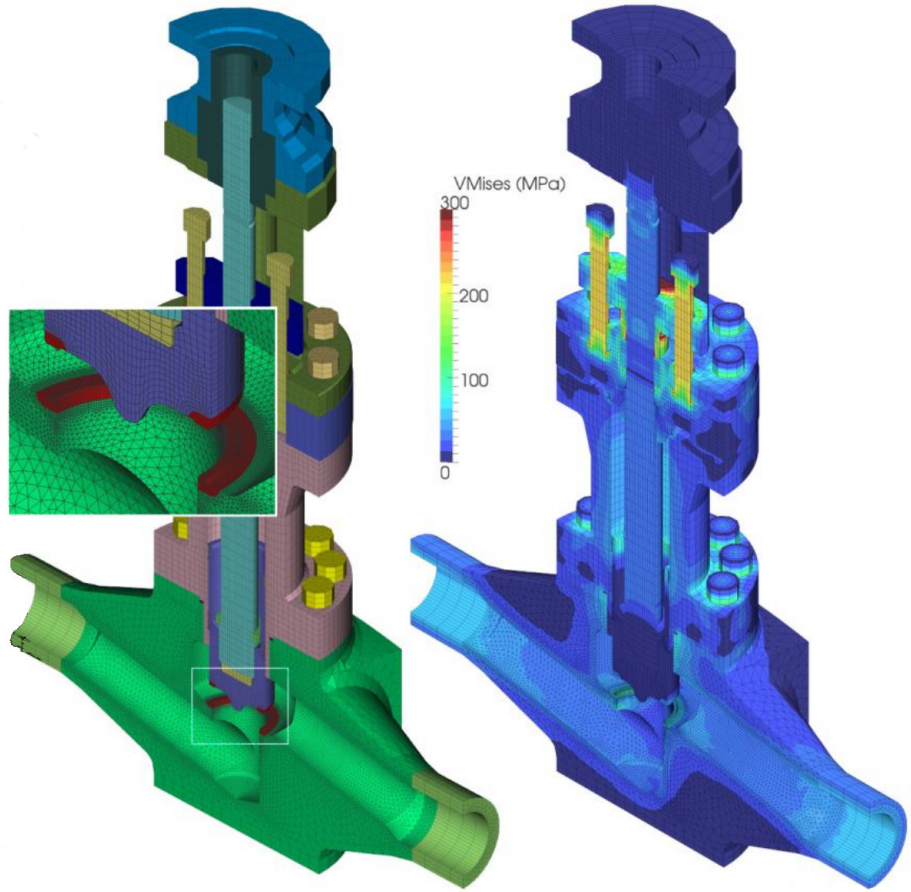
Code_Aster

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 : mécanique

Preliminary work

Selection of 72 test-cases

- some (believed to be) stable
- some (known to be) unstable

"Meta-verification": verification of the verification process itself

- Verrou could have bugs
- Problem of "Heisenbugs": they (dis)appear when instrumenting
  - introspection
  - problematic assembly instructions (*e.g* x87 instruction set)
  - specific algorithms setting the rounding mode

Application : mécanique

Preliminary work

Selection of 72 test-cases

- some (believed to be) stable
- some (known to be) unstable

"Meta-verification": verification of the verification process itself

- Verrou **had a bug**, could have others
- Problem of "Heisenbugs": they (dis)appear when instrumenting
  - introspection
  - problematic assembly instructions (*e.g* x87 instruction set)
  - specific algorithms setting the rounding mode

Analysis of numerical instabilities

Using Verrou and Random Rounding

Test case	nearest
ssls108i	OK
ssls108j	OK
ssls108k	OK
ssls108l	OK
sdnl112a	OK
ssnp130a	OK
ssnp130b	OK
ssnp130c	OK
ssnp130d	OK

Analysis of numerical instabilities

Using Verrou and Random Rounding

Test case	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

Analysis of numerical instabilities

Using Verrou and Random Rounding

Test case	Status 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
`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 (72 test cases)

20 minutes each

Analysis of numerical instabilities

Using Verrou and Random Rounding

Test case	Status nearest	rnd_1	rnd_2	rnd_3	# common decimal digits $C(rnd_1, rnd_2, rnd_3)$
`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 test cases)

20 minutes each

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

Application : mécanique

Localisation de branchements instables par couverture de code

```
$ 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
```

Application : mécanique

Correction de la formule

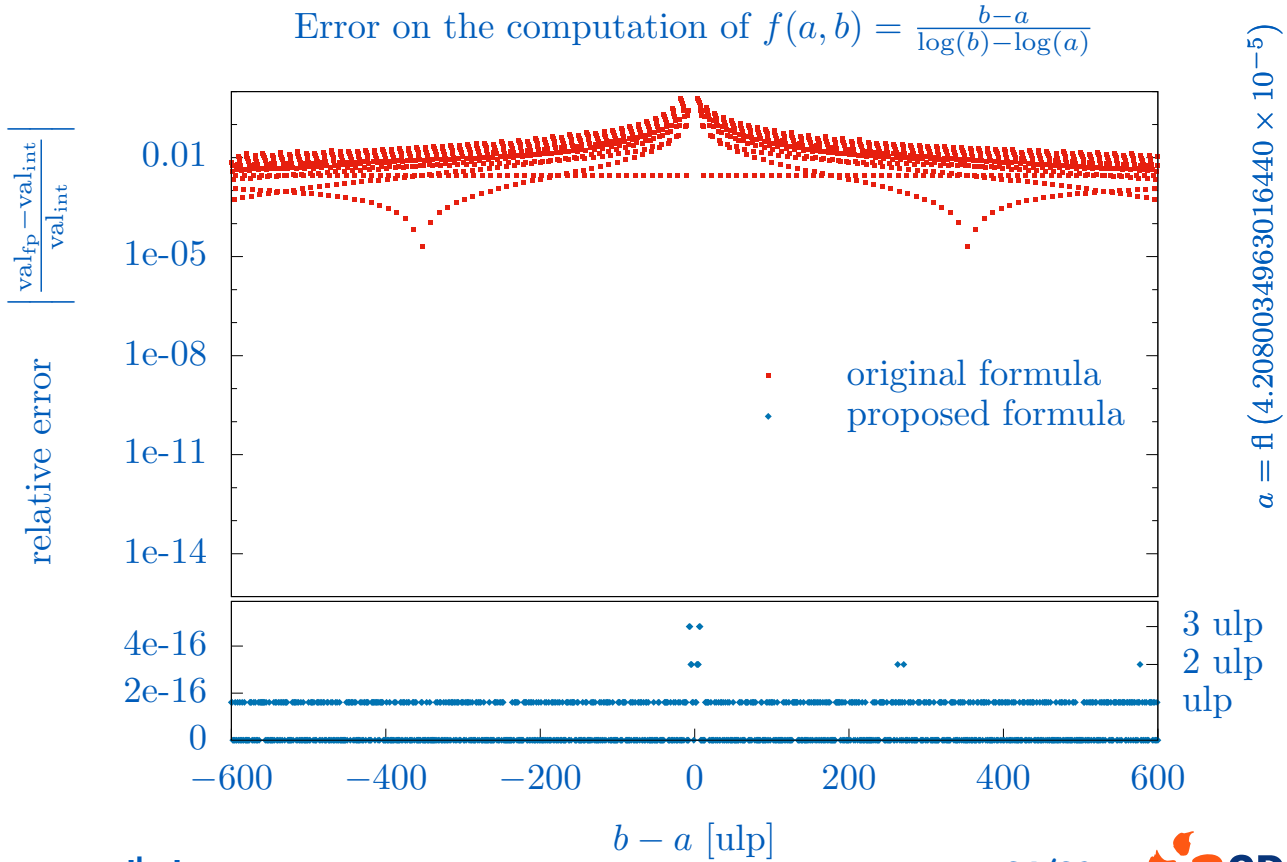
$$f(a,b) = \begin{cases} a & \text{si } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{sinon} \end{cases} \longrightarrow f(a,b) = \begin{cases} a & \text{si } a = b \\ a\,\frac{\frac{b}{a}-1}{\log(\frac{b}{a})} & \text{sinon} \end{cases}$$

Étude empirique

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

Preuve

- erreur bornée par 10 ulps



Application : mécanique

Delta-Debugging on code_aster (test-case `sdnl112a`)

- 2:20' run time =

(would Herbgrind be competitive?)

- 86 configurations
- 15 random rounding runs to validate
- 10s. per RR run (*vs* 4s. native)

```
      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
```

Application : mécanique

Delta-Debugging on code_aster (test-case `sdnl112a`)

- 2:20' run time =

(would Herbgrind be competitive?)

- 86 configurations
- 15 random rounding runs to validate
- 10s. per RR run (*vs* 4s. native)

```
      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=zmdi(ilig-1)+1
              xsol(jcol,jvec)=xsol(jcol,jvec) + zr(jvalms-1+k1) * 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
```

- Correction: compensated algorithms [Ogita, Rump, Oishi. 2005]
  - Sum2 is not enough
  - Dot2

Conclusions

Accuracy quantification after correction

`sdnl112a`

Version	Status nearest	rnd_1	rnd_2	rnd_3	# common digits $C(\text{rnd}_1, \text{rnd}_2, \text{rnd}_3)$
Before correction	OK	KO	KO	KO	6 6 6 * 3 0
After correction	OK	OK	OK	OK	10 10 9 * 6 0

Conclusions – Perspectives

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.

Comparaison des outils

CADNA

VERROU

VERIFICARLO

	CADNA	VERROU	VERIFICARLO
Méthode	CESTAC sync.	CESTAC async.	MCA async.
# échantillons	3	$N \geq 3$	$N \geq 3$
Instrumentation	sources	binaire	compilation
Portabilité	partout	Linux x86	partout
Langages	C++, F	tous	LLVM
Localisation	fine	moyenne	*(en cours)*
Performance	$\times 15$	$\times 10\, N$	$\times 100\, N$ $\rightarrow$ $\times 5\, N$

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 ?

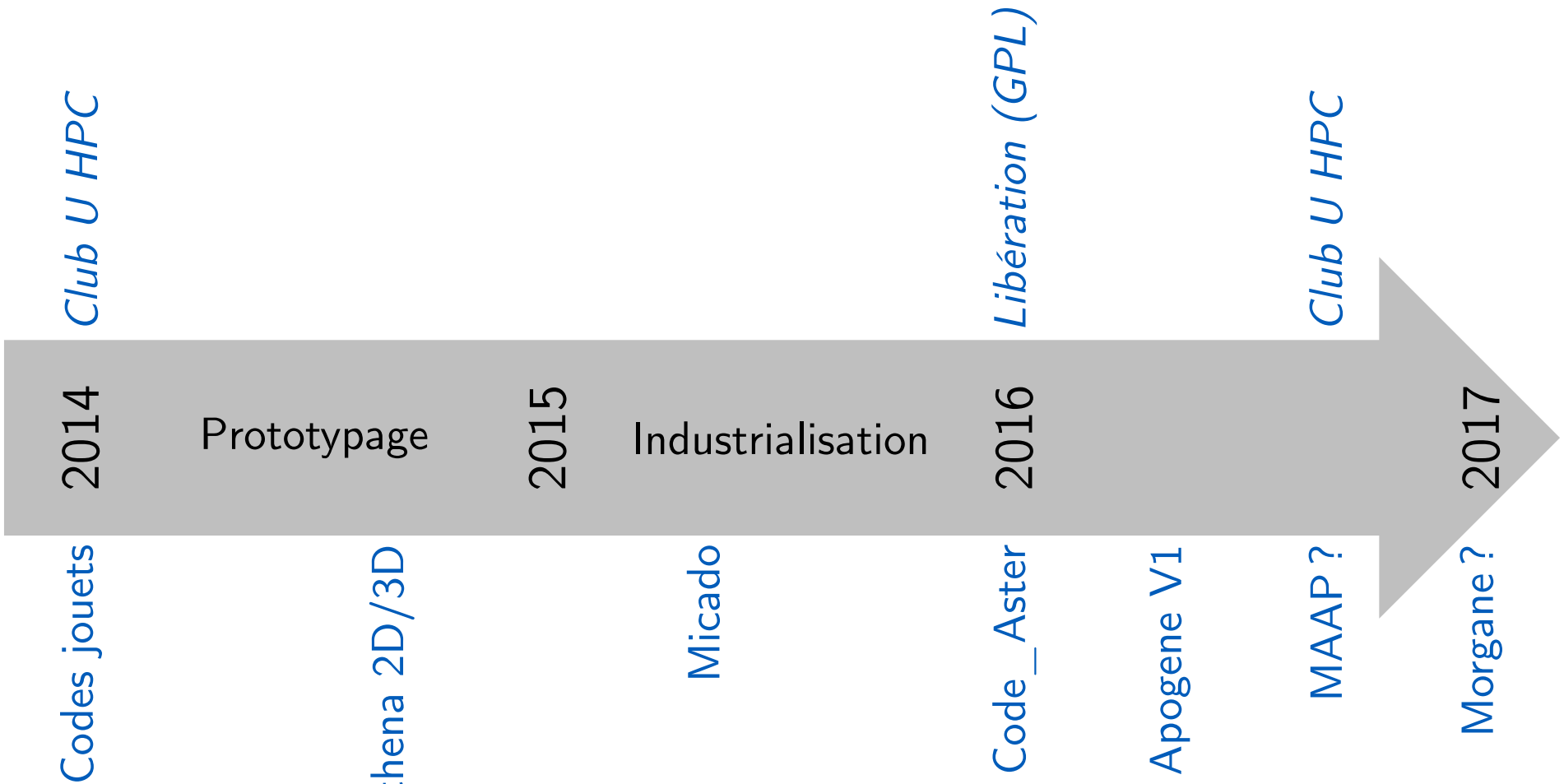
Récupérez `verrou` sur github :
`http://github.com/edf-hpc/verrou`

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

Annexes

1. L'outil Verrou
2. Traitement de la division
3. Tests & branchements
4. Synchrone vs Asynchrone
5. Athena
6. Apogene

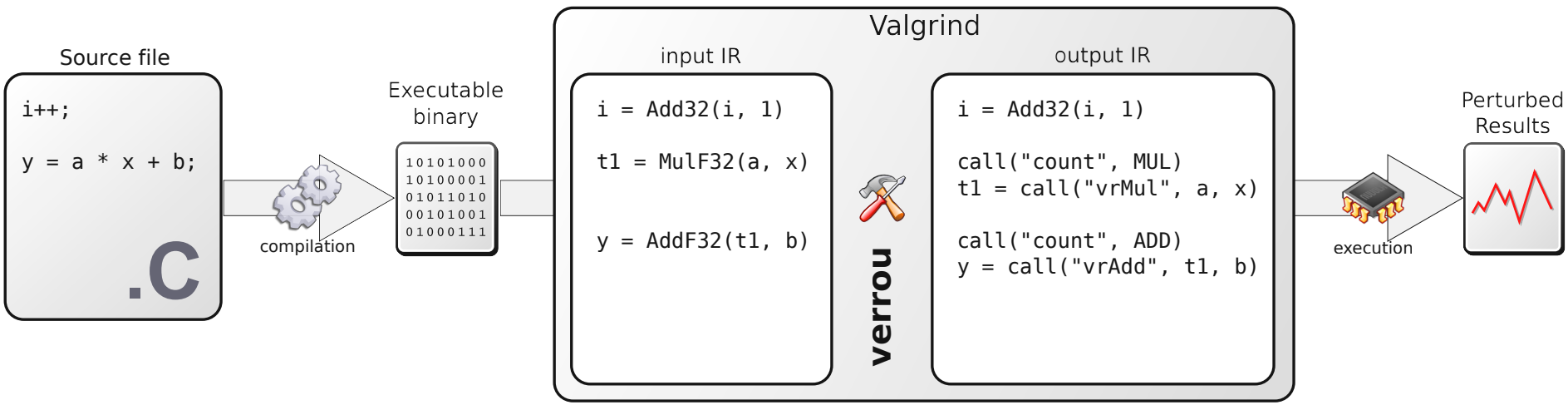
Bref historique



L'outil Verrou

Analyse dynamique du binaire avec Valgrind

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



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
  - Détecter des erreurs
  - Remplacer les opérations

Valgrind ne gère que le mode d'arrondi `NEAREST`

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

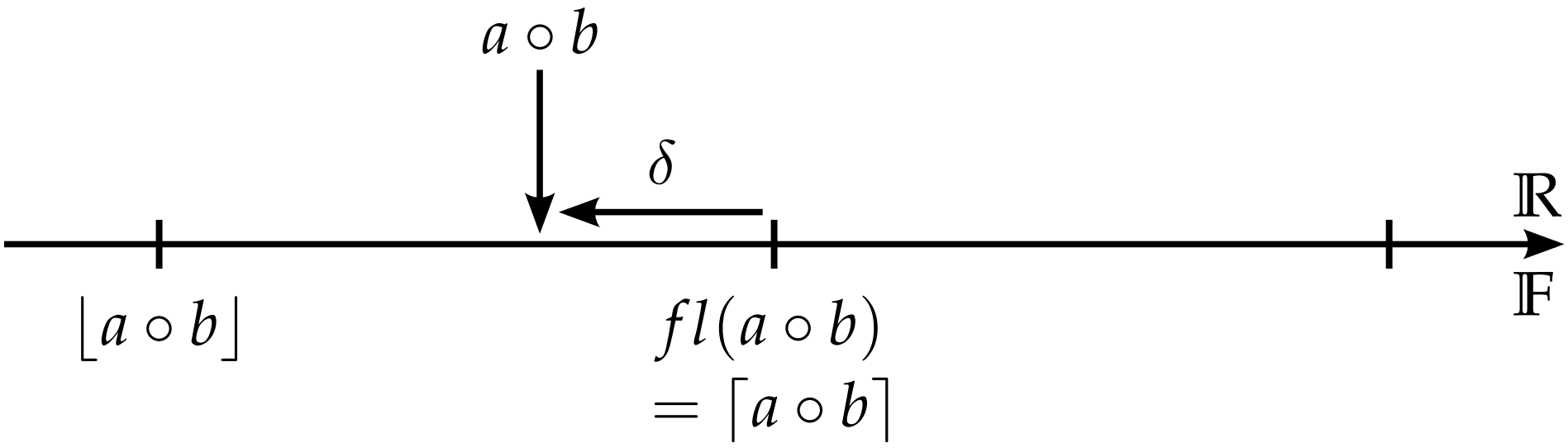
```
$ valgrind --tool=verrou --rounding-mode=random PROGRAM [ARGS...]

==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 : yes
...
==4683==  ----------------------------------------------------------------
==4683==
==4683==
==4683==  ----------------------------------------------------------------
==4683==                                                          (100%)
==4683==                                                  5468    (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==  ----------------------------------------------------------------
...
```

Sorties normales du programme

+ **Warnings pour les instructions "dangereuses" (ex : x87)**

L'outil Verrou

Simulation des modes d'arrondi

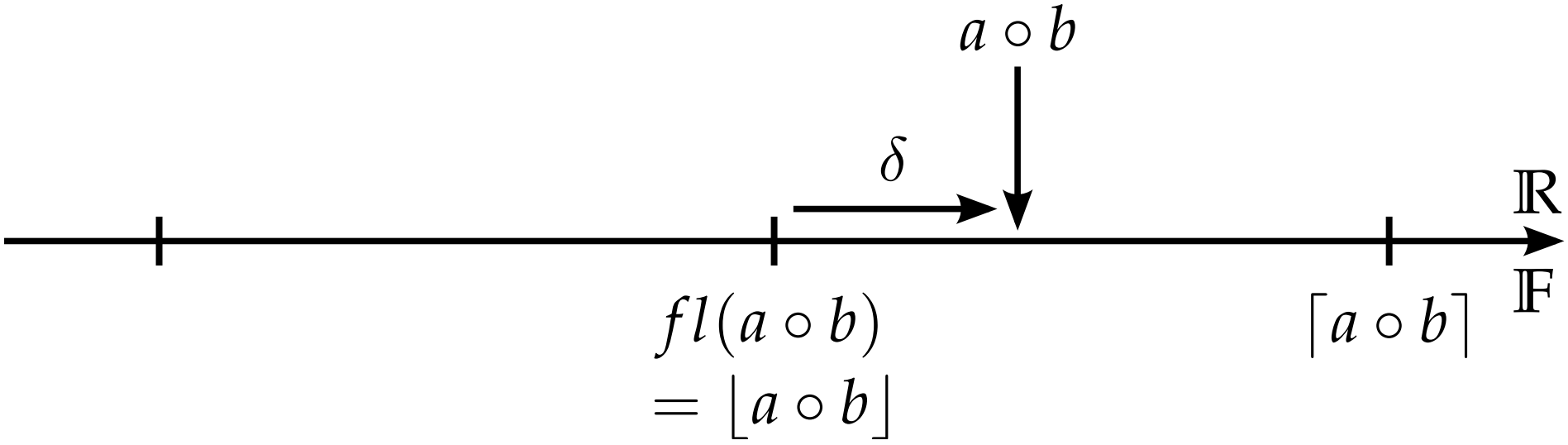


- Transformation sans erreur
  (la division est un peu plus compliquée):
  - $a \circ b = \sigma + \delta$,
  - $\sigma = fl(a \circ b)$

- 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

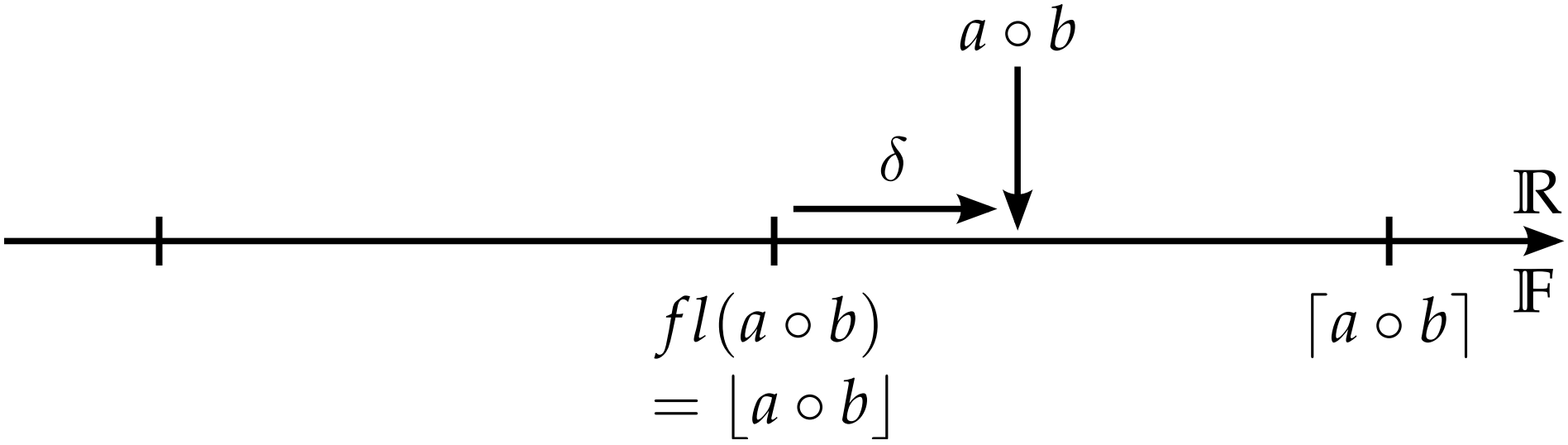
Simulation des modes d'arrondi



- Transformation sans erreur
  (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

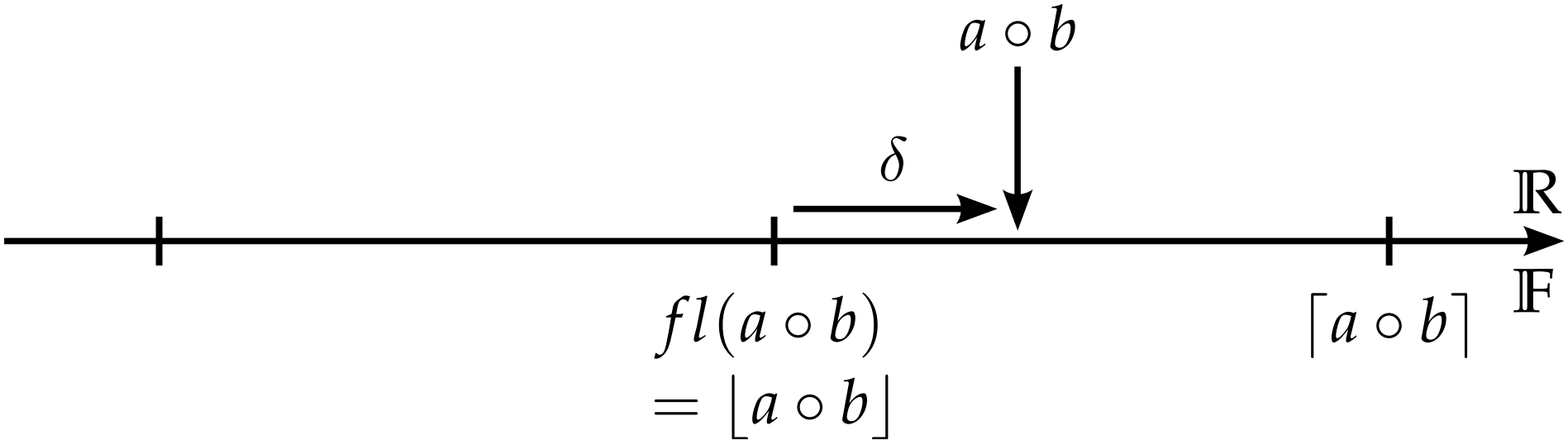
Simulation des modes d'arrondi



- `random` : arrondi équiprobable (CESTAC)
  - $p(\lceil a \circ b \rceil) = 0.5$
  - $p(\lfloor a \circ b \rfloor) = 0.5$

L'outil Verrou

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

Sections (non-)instrumentées

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

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

L'outil Verrou

Sections (non-)instrumentées

```
$ 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
```

Transformation approchée pour la division

- Ce qu'on cherche :

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

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

- Algorithme proposé :

Input : a, b
Output : $\tilde{r}$ tel que
$a/b \simeq \mathrm{fl}(a/b) + \tilde{r}$.

1 $q \leftarrow \mathrm{fl}(a/b)$
2 $(p, s) \leftarrow \mathrm{twoprod}(b, q)$
3 $t \leftarrow \mathrm{fl}(a - p)$
4 $u \leftarrow \mathrm{fl}(t - s)$
5 $\tilde{r} \leftarrow \mathrm{fl}(u/b)$

- Idée de la preuve :

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

$$\begin{aligned} p &= b\,q\,(1 + \epsilon_2) \\ &= a\,(1 + \epsilon_1)\,(1 + \epsilon_2) \end{aligned}$$

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

$$\begin{aligned} u &= (t - s)\,(1 + \epsilon_3) \\ &= \Big(a - (p + s)\Big)\,(1 + \epsilon_3) \\ &= (a - bq)\,(1 + \epsilon_3) \\ &= b\,r\,(1 + \epsilon_3) \end{aligned}$$

$$\begin{aligned} \tilde{r} &= \frac{u}{b}(1 + \epsilon_4) \\ &= r\,(1 + \epsilon_3)\,(1 + \epsilon_4)\,. \end{aligned}$$

Tests et branchements : exemple de MAAP

- Entrées:

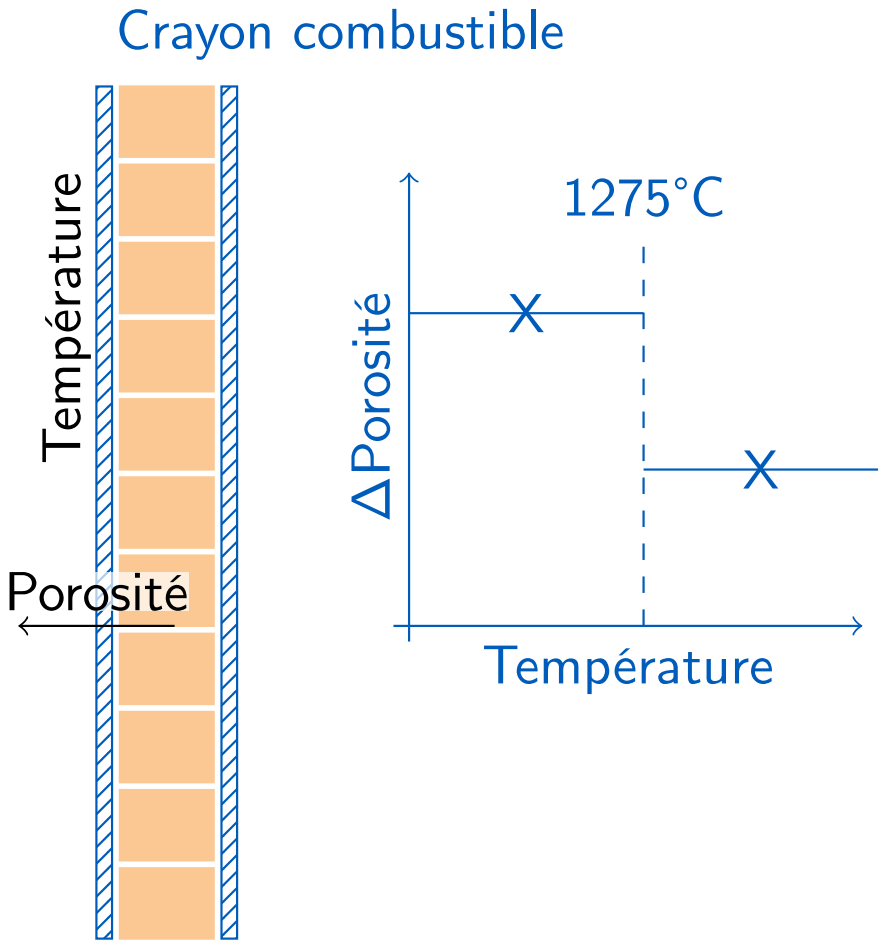
	Eval. 1	Eval. 2
Température	**1275**.2	1274.8
Porosité	**0.42382**	0.42383

- 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



Arithmétique stochastique

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 $b \geqslant 1$ then				
$\quad b = b - 1$				
else				
$\quad b = 1 - b$				
end				
$b = \sqrt{b}$				

Arithmétique stochastique

Synchrone *vs* asynchrone

Évaluation :

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

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**3.3**4e-1
$b = a \times 3$				
if $b \geqslant 1$ then				
$\quad b = b - 1$				
else				
$\quad b = 1 - b$				
end				
$b = \sqrt{b}$				

Arithmétique stochastique

Synchrone *vs* asynchrone

Évaluation :

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

Opération	Eval. 1	Eval. 2	Eval. 3	Moyenne
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**3.3**4e-1
$b = a \times 3$	$0.999_{\downarrow}$	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.0**0
if $b \geqslant 1$ then	———	True	———	**Warning**
$b = b - 1$	-1.00e-3	0.00	1.00e-2	3.00e-3
else				
$b = 1 - b$				
end				
$b = \sqrt{b}$	×	0.00	1.00e-1	×

Arithmétique stochastique

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 $b \geqslant 1$ then	False			
$\quad b = b - 1$				
else				
$\quad b = 1 - b$	1.00e-3			
end				
$b = \sqrt{b}$	$3.17\text{e-}2^{\uparrow}$			

Arithmétique stochastique

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 $b \geqslant 1$ then	False	True	True	
$\quad b = b - 1$		0.00	1.00e-2	
else				
$\quad b = 1 - b$	1.00e-3			
end				
$b = \sqrt{b}$	$3.17\text{e-}2^{\uparrow}$	0.00	1.00e-1	
print b				4.39e-2

Arithmétique stochastique

Synchrone *vs* asynchrone : exemple

- Asynchrone

	1.e-4	2.e-5	-1.e-5	
`if x < 0:`	False	False	True	
`  x = 0`	1.e-4	1.e-5	0.	
`y = sqrt( x )`	1.e-2	3.16e-3	0.	4.38e-3

Arithmétique stochastique

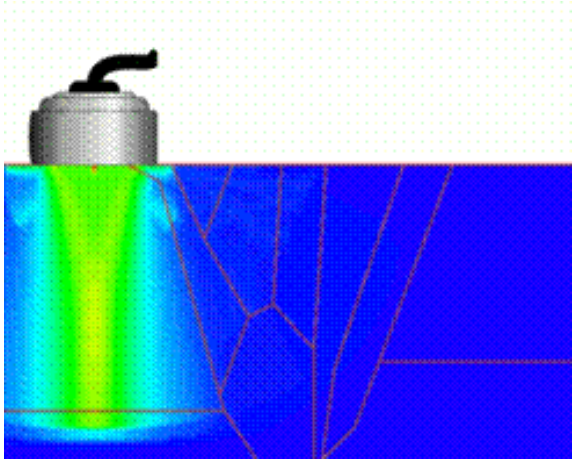
Synchrone *vs* asynchrone : exemple

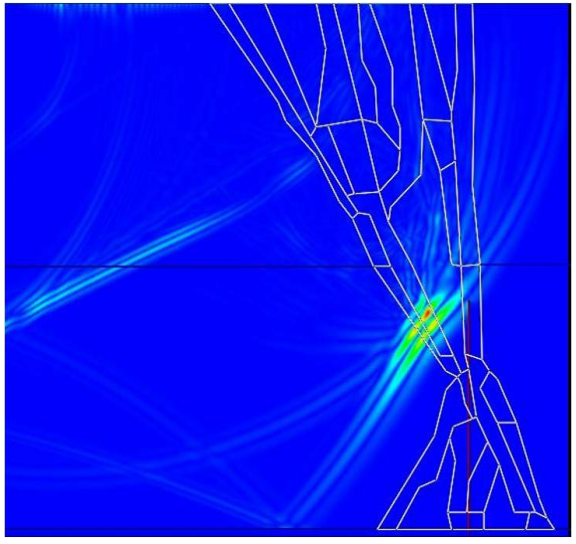
- Synchrone

	1.e-4	2.e-5	-1.e-5
`if x < 0:`	———	**False**	———
`x = 0`	1.e-4	1.e-5	**-1.e-5**
`y = sqrt( x )`	1.e-2	3.16e-3	**erreur**

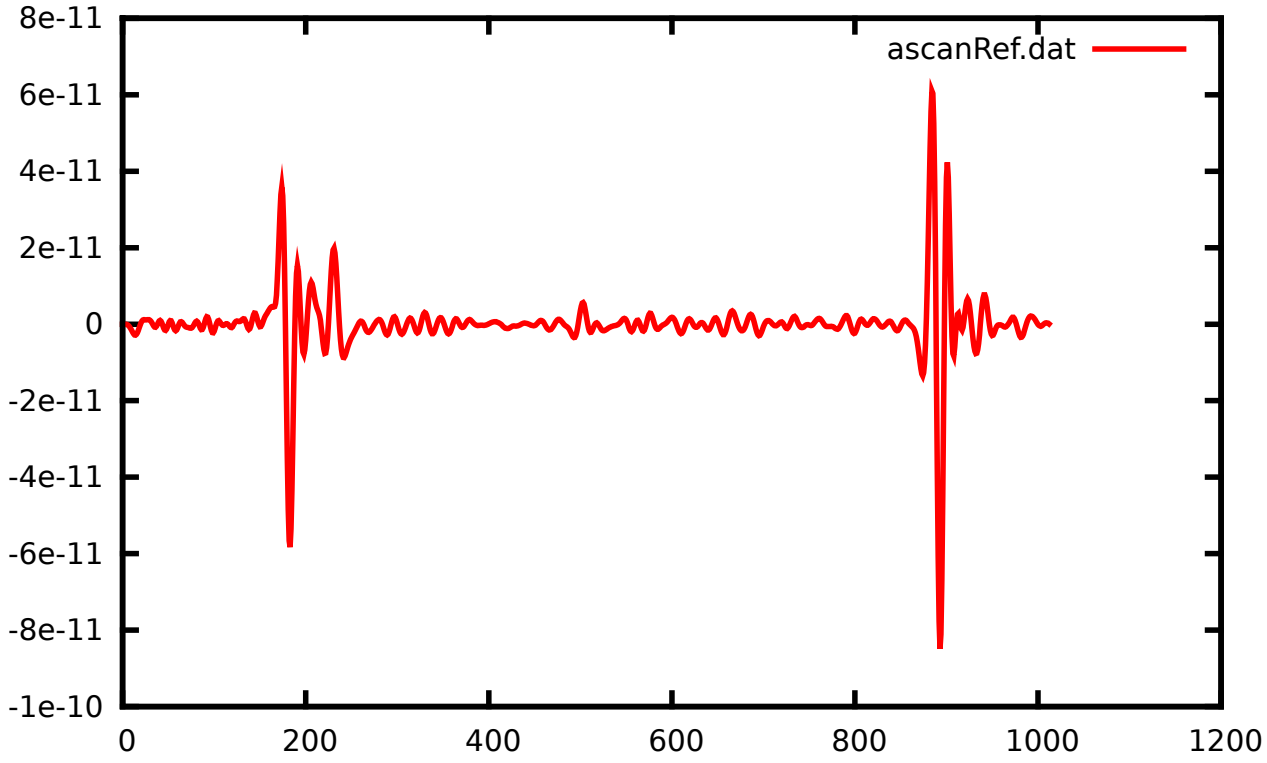
Application : CND par ultra-sons

Code Athena 2D



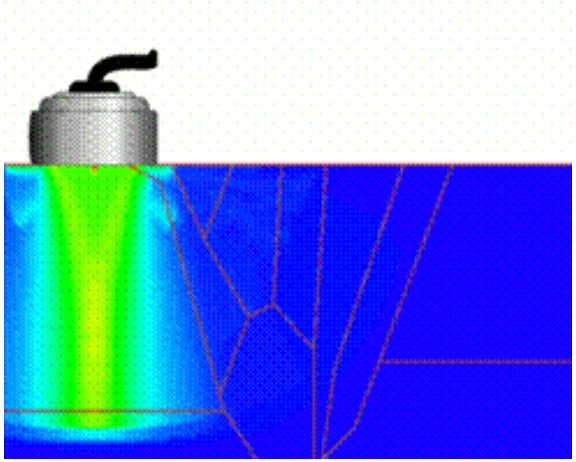


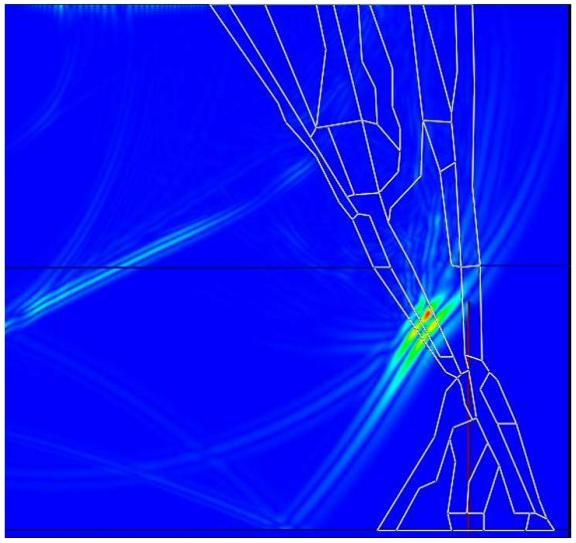
Résultat produit : “A-scan”



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"

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**

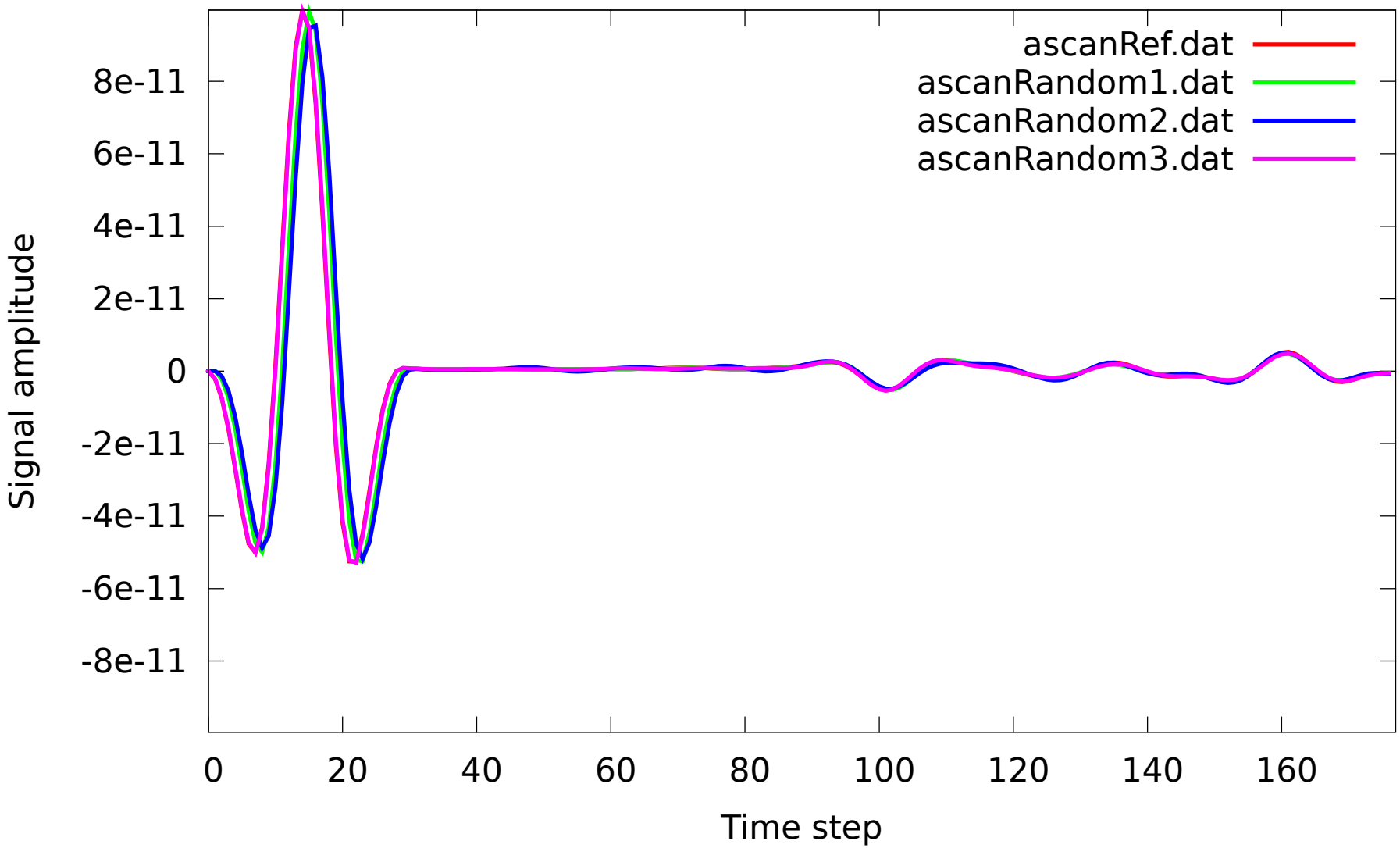
Application : CND par ultra-sons

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**

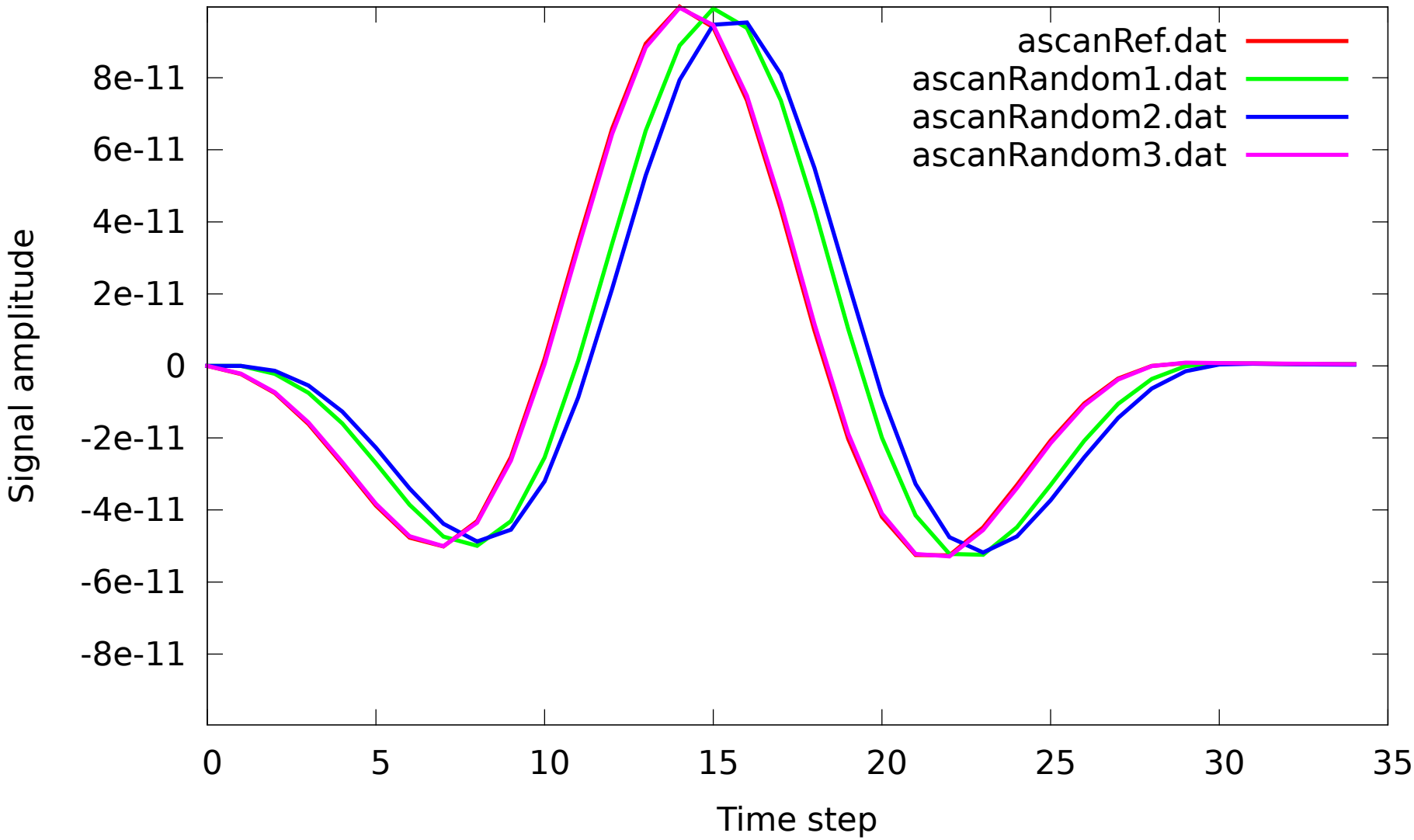
Application : CND par ultra-sons

Cas-test problématique



Application : CND par ultra-sons

Cas-test problématique



Applications

Application : planification de production

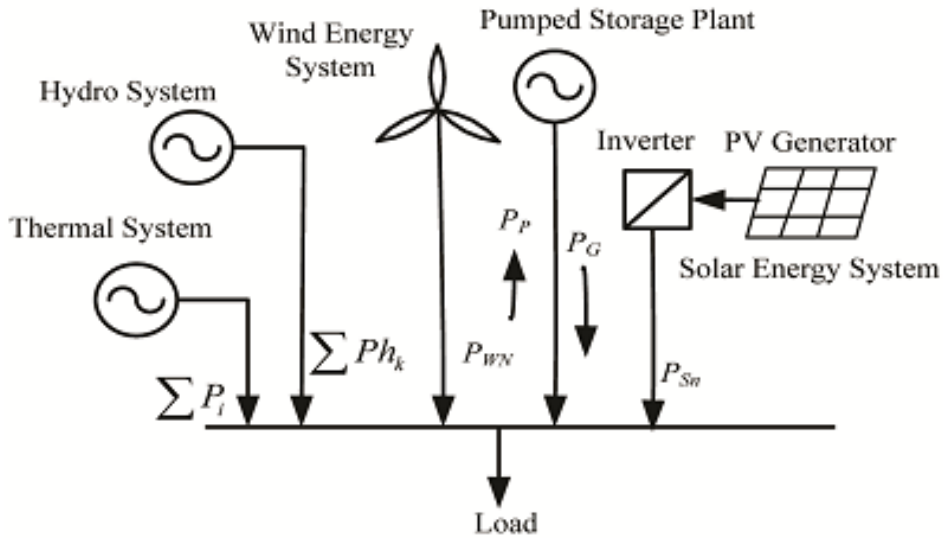
Code Apogene

Planification court terme

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

Code Apogene 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

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
# 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 [A. Zeller, 1999]

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

Application : planification de production

Verrou : Localisation d'erreurs par bisection

```
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 [A. Zeller, 1999]

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

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

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
```

- 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
:
```