Numerical quality: an industrial case study on code_aster

Numerical Software Verification
22/07/2017

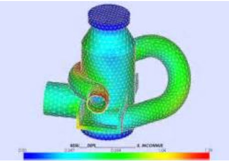
François Févotte*
Bruno Lathuilière

EDF R&D
PERICLES / I23
(Analysis and Numerical Modeling)

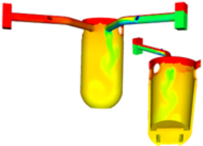
EDF

Industrial context – Numerical Simulation

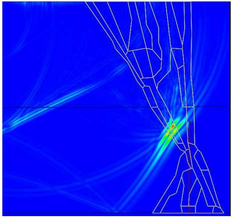
In-house development of Scientific Computing Codes



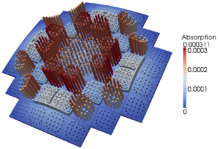
Structures



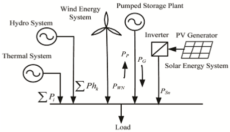
Fluid dynamics



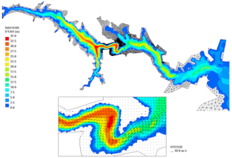
Wave propagation



Neutronics



Power Systems



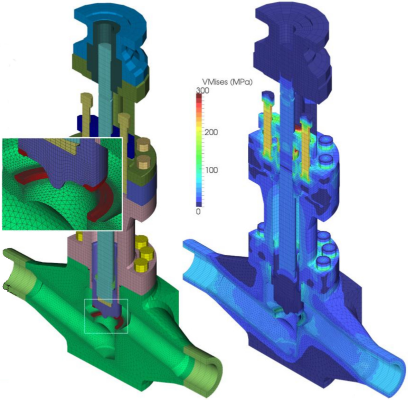
Free surface hydraulics

Industrial context – Numerical Simulation

Code_aster

Mechanics

- Seismic
- Acoustic
- Thermo-mechanics



Code_Aster

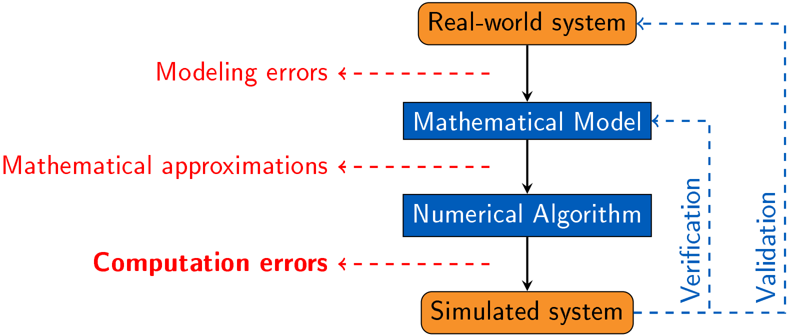
- 1.2M code lines
- Fortran 90, C, Python
- thousands of test cases
- Large number of dependencies :
  - Linear solvers (MUMPS. . .)
  - Mesh generator and partitioning tools (Metis, Scotch. . .)

Goals

- understand the non-reproducibility between test computers

Industrial context – Numerical Simulation

Verification & Validation

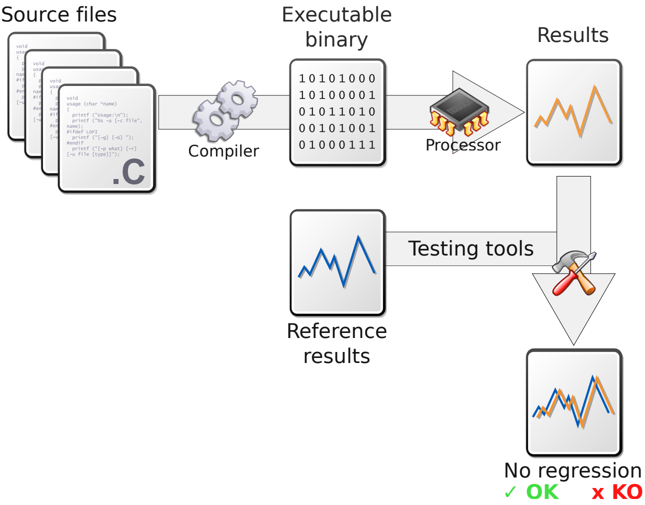


Quantifying numerical errors: at stake

- quality of produced results
- efficient use of resources (development & computing time)

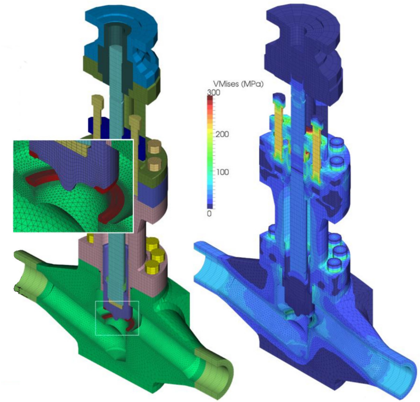
Industrial context – Numerical Simulation

V&V process: non-regression testing

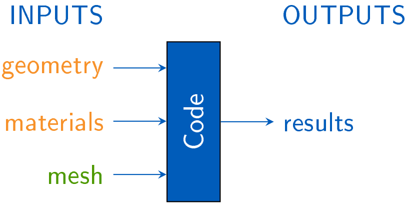


Industrial context – Numerical Simulation

V&V process: ad-hoc numerical instability detection methods



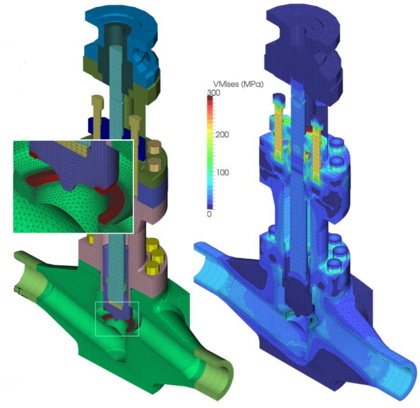
Physical input: affects the result
Simulation parameter: should be neutral



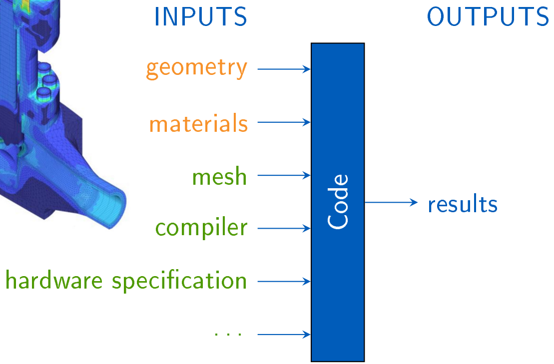
- Idea: measure the sensitivity of the results w.r.t "neutral" parameters
  - easy to do
  - *ad hoc*, no localization

Industrial context – Numerical Simulation

V&V process: ad-hoc numerical instability detection methods



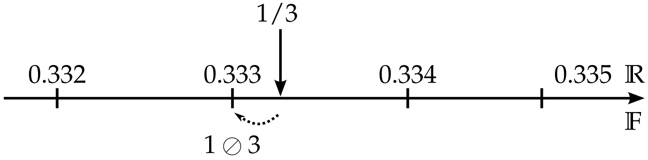
Physical input: affects the result
Simulation parameter: should be neutral



Numerical Verification

The CESTAC Method: dynamic analysis with random rounding

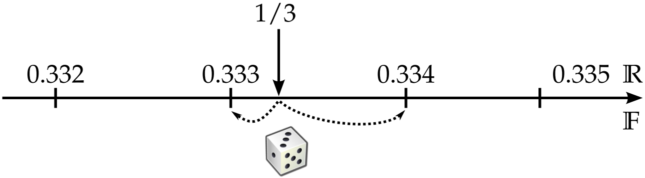
IEEE-754 nearest rounding mode



Numerical Verification

The CESTAC Method: dynamic analysis with random rounding

CESTAC random rounding mode



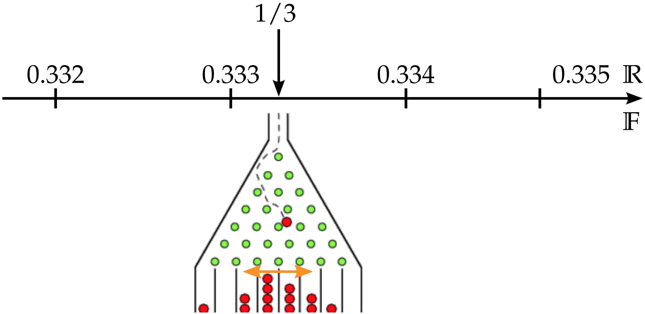
[1] J. Vignes, "A stochastic arithmetic for reliable scientific computation," *Mathematics and Computers in Simulation*, vol. 35, no. 3, 1993.

[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.

Numerical Verification

The CESTAC Method: dynamic analysis with random rounding

CESTAC random rounding mode

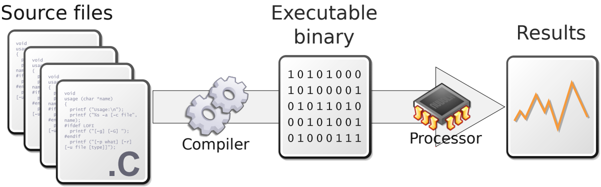


Instruction	Eval. 1	Eval. 2	Eval. 3	Average
$a = 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**0.334**
$b = a \times 3$	0.999	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.00**

Numerical Verification

CADNA: dynamic sources analysis

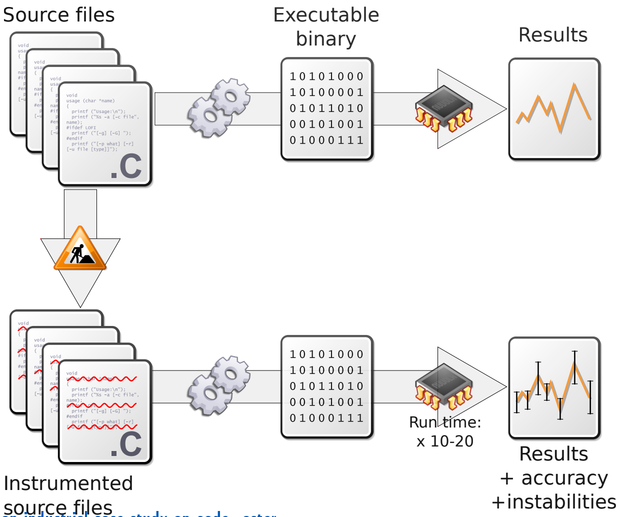
```
$ myProg in out
```



Numerical Verification

CADNA: dynamic sources analysis

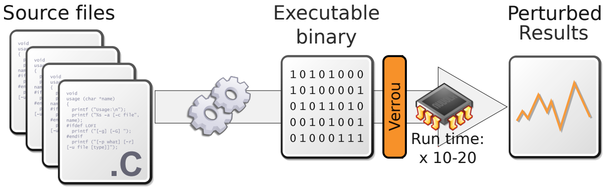
$ myProg-cadna `in out`



Numerical Verification

Verrou: dynamic binary analysis

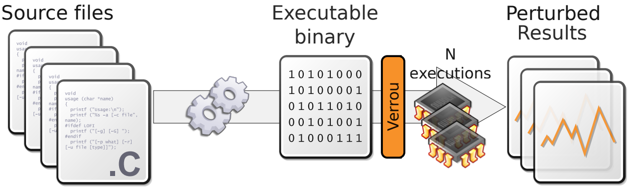
$ valgrind --tool=verrou --rounding-mode=random `myProg in out1`



Numerical Verification

Verrou: dynamic binary analysis

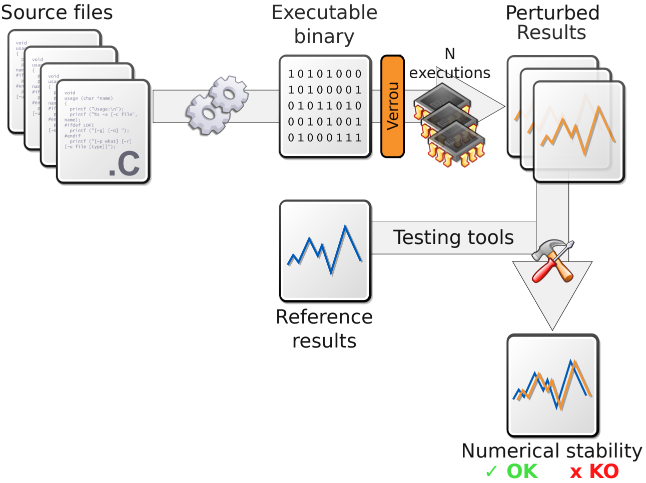
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



Numerical Verification

Verrou: dynamic binary analysis

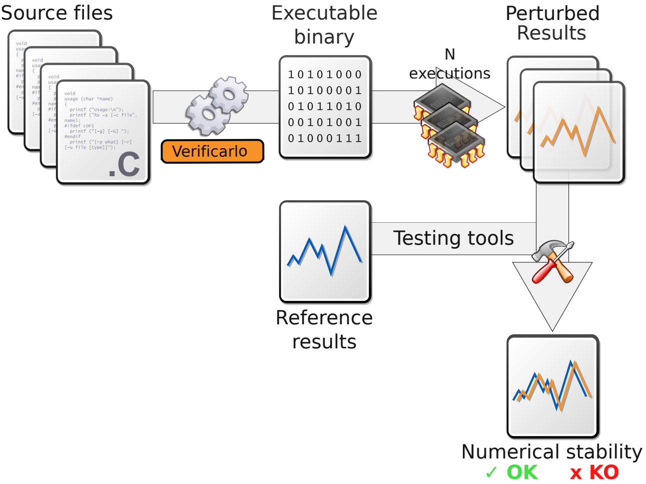
```
$ valgrind --tool=verrou --rounding-mode=random myProg in out1
$ valgrind --tool=verrou --rounding-mode=random myProg in out2
```



Numerical Verification

Verificarlo: dynamic IR analysis

$ verificarlo mySources.c -o myProg-verificarlo
$ myProg-verificarlo `in out1`



Outline

1. Detect instabilities
2. Locate (and fix) instabilities
    - unstable tests
    - round-off errors
3. Conclusions – perspectives

Detect instabilities

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

Detect instabilities

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 — new
- 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 (nearest) — 20 minutes each (rnd$_1$, rnd$_2$, rnd$_3$)

(72 test cases)

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 * 9 *

10 minutes (nearest)

20 minutes each (rnd_1, rnd_2, rnd_3)

(72 test cases)

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

Locate (and fix) instabilities

Two kinds of error origins

Trivial example: descent direction

```
f0 = f (x);
f1 = f (x + dx);

g  = f1 - f0;

if ( g > 0 ) {
  x -= dx;
} else {
  x += dx;
}
```

round-off errors and cancellation (lines f0 = f (x); f1 = f (x + dx); and g = f1 - f0;)

unstable test (line if (g > 0) {)

Localization of unstable branches

Using code coverage tools: code_aster (test-case sdnl112a)

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

"standard" coverage

```
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
```

Localization of unstable branches

Using code coverage tools: code_aster (test-case `sdnl112a`)

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

"standard" coverage

```
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
```

"Verrou" coverage

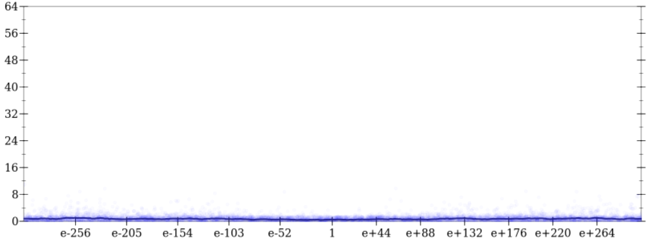
```
120:subroutine fun1(area, a1,...
  -:    implicit none
  -:    integer :: n
  -:    real(kind=8) :: area,...
120:    if (a1 .eq. a2) then
  4:        area = a1
  -:    else
116:        if (n .lt. 2) then
116:           area = (a2-a1)...
###:        else if (n .eq.2)...
###:           area = sqrt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

Localization of unstable branches

Formula correction

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

http://herbie.uwplse.org/



Bits of error vs value of b

Sampling is not enough

- Need for more rigorous alternatives (static analysis?)
- We have uncovered one counter-example → enough to test a correction

Localization of unstable branches

Formula correction

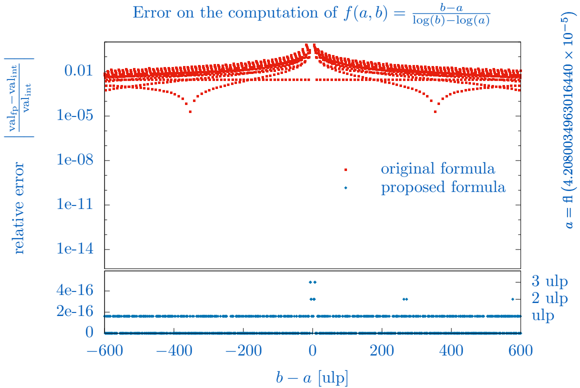
$$f(a,b) = \begin{cases} a & \text{if } a = b \\ \frac{b-a}{\log(b)-\log(a)} & \text{otherwise} \end{cases} \quad \longrightarrow \quad f(a,b) = \begin{cases} a & \text{if } a = b \\ a\,\frac{\frac{b}{a}-1}{\log(\frac{b}{a})} & \text{otherwise} \end{cases}$$

Empirical study

- outside the code (proxy app)
- around the problematic point
- reference = interval arithmetic

Proof

- error bounded by 10 ulps



EDF

Locate (and fix) instabilities

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]

$\longrightarrow$ ✓

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]

$\longrightarrow$ ✗

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]

→ ✗

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]

→ ✓

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]

→ ✗

Localization of round-off errors accumulations

Delta-debugging

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

- Delta-Debugging [A. Zeller, 1999]
- Inputs :
  - run script
  - comparison script
- Output:
  - DDmax: failure inducing functions
- Also works at the source line granularity:
  - if the code was compiled with -g

Localization of round-off errors accumulations

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
```

Localization of round-off errors accumulations

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)
```

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

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

Conclusions – perspectives

1. Detect instabilities

2. Locate (and fix) instabilities

3. Conclusions – perspectives

Conclusions

Accuracy quantification after correction

`sdnl112a`

Version	Status				# common digits
	nearest	rnd_1	rnd_2	rnd_3	$C(rnd_1, rnd_2, rnd_3)$
Before correction	OK	KO	KO	KO	6 6 6 * 3 0
After correction	OK	OK	OK	OK	10 10 9 * 6 0

Conclusions

- Problems hard to find, but easy to solve (for now!)
- Workflow for the analysis of industrial codes
- First steps are affordable and could be automatized

Perspectives

Verrou

- Interflop: common interface for Verificarlo & Verrou (soon others!)
  - share Monte-Carlo Arithmetic back-ends
  - improve performance of instrumentation front-ends
- Handle mathematics library & co

Methodology

- Apply to other fields
  - optimization (electricity production planning)
  - multi-physics (severe nuclear accidents)
- Correction steps should be further automatized
  - would Herbgrind + Herbie help?
  - static analysis?

Thank you !

Questions ?

Get verrou on github :
http://github.com/edf-hpc/verrou

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