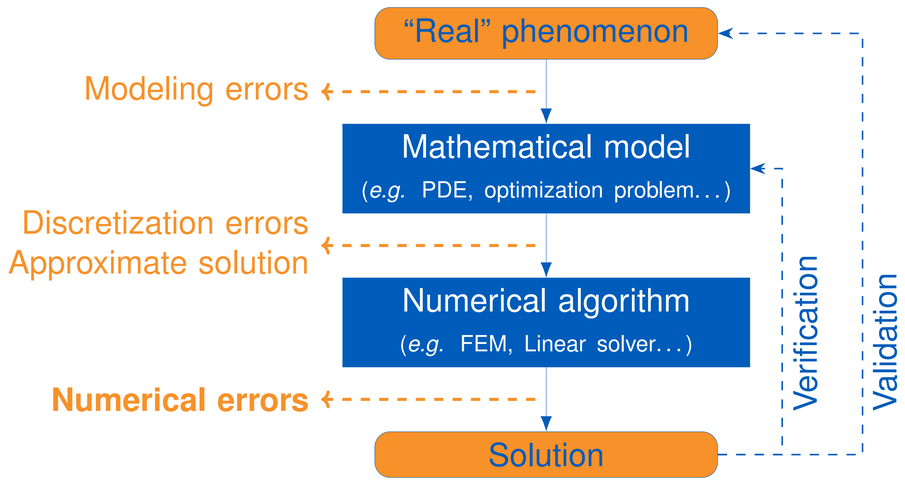
VERROU: NUMERICAL VERIFICATION OF SCIENTIFIC COMPUTING CODES

FRANÇOIS FÉVOTTE & BRUNO LATHUILIÈRE

EDF R&D, DÉPARTEMENT PERICLES
7 BD GASPARD MONGE, 91120 PALAISEAU, FRANCE

NUMERICAL VERIFICATION



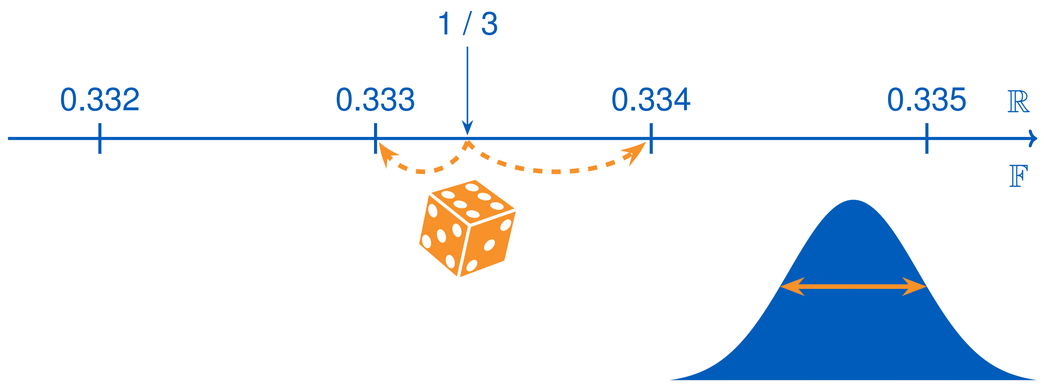
Numerical errors are due to the difference beween the ideal manipulation of real numbers, and what actually gets computed by the CPU, which typically uses IEEE-754 Floating-Point Arithmetic:

Finite Precision: results are rounded, which may cause a decrease in the resulting accuracy;

Loss of Associativity: the order of operations counts, which may cause a loss of reproducibility.

RANDOM ROUNDING ARITHMETIC (RRA)

Among the different techniques which can be used to evaluate numerical instabilities and round-off errors, the wide family of methods revolving around Monte-Carlo Arithmetic (MCA) seems to be one of the most promising in industrial contexts.



VERROU is based on a "Random Rounding" variant of MCA: all arithmetic operations are **randomly rounded up or down**, instead of always rounding to the nearest floating-point.

Global results of the computation are thus **turned into random variables**, which are affected by the cumulative effect of all randomly rounded intermediate results. Studying the variance of these results gives indications as to whether the computation was numerically stable.

Example of a program execution with 3 random rounding runs:

Operation	Run 1	Run 2	Run 3	Average / *Comment*
$a \leftarrow 1/3$	$0.333_{\downarrow}$	$0.334^{\uparrow}$	$0.334^{\uparrow}$	**3.3**4e-1
$b \leftarrow a \times 3$	$0.999_{\downarrow}$	$1.00_{\downarrow}$	$1.01^{\uparrow}$	**1.0**0
if $b \geqslant 1$ then	False	True	True	*Unstable test*
$b \leftarrow b - 1$		0.00	1.00e-2	
else				
$b \leftarrow 1 - b$	1.00e-3			
end				
$b \leftarrow \sqrt{b}$	3.17e-2$^{\uparrow}$	0.00	1.00e-1	4.39e-2
print b				**4.39e-2**

EDF R&D Contacts:

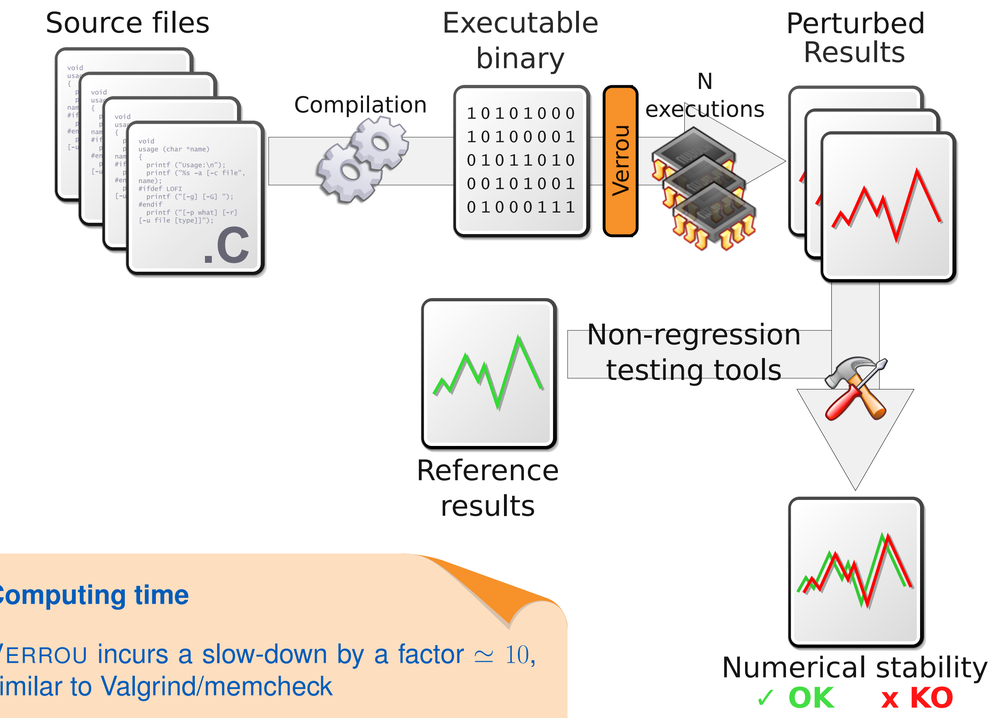
Bruno Lathuilière <bruno.lathuiliere@edf.fr>
François Févotte <francois.fevotte@edf.fr>

NUMERICAL QUALITY WITH VERROU

Verrou is based on the Valgrind platform and performs a Dynamic Binary Analysis (DBI). VERROU is therefore **as simple to use as Valgrind**: no need for any instrumentation of the sources or even recompilation. The usual command simply needs to be prefixed by an invocation of VERROU:

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

This makes it easy for Verrou to be introduced in an **industrial V&V process**: by simply ensuring that test cases are run within Verrou, their results can be perturbed using RRA. Such results can then be compared as usual to references, in order to evaluate the numerical stability of the computing code.



Computing time

VERROU incurs a slow-down by a factor $\simeq 10$, similar to Valgrind/memcheck

NUMERICAL DEBUGGING – ERROR LOCALIZATION

Large instabilities (`verrou_dd` utility):

VERROU can reduce the scope of random rounding perturbations to parts of the program: functions or source code lines[1]. This feature can be used to perform a binary search (based on the **Delta-Debugging** algorithm) to identify unstable portions of the source code, whose instrumentation produces large changes in the results.

Unstable tests (coverage test):

Composing VERROU with a **test coverage** system (such as `gcov`), one can determine which source code lines have been executed different numbers of times for different RRA runs. This identifies unstable tests.

PAST STUDIES

VERROU has been successfully used for the numerical verification of industrial Scientific Computing Codes in various fields:

- **Athena 2D** (EDF R&D): ultra-sonic non-destructive evaluations,
- **code_aster** (EDF R&D): structural mechanics and thermomechanics,
- **Apogene** (EDF R&D): production units commitment optimization,
- **MAAP** (EDF R&D): severe nuclear accidents analysis,
- **MFront** (CEA): constitutive equations in mechanics.

GET VERROU AND TRY IT

VERROU is **open source** and available on GitHub:

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

[1] if the program was compiled with the right options (like `gcc -g` for instance)