Verificarlo: Checking Floating-Point Accuracy Through Monte Carlo Arithmetic

Study on Europlexus

Pablo de Oliveira Castro[1], Olivier Jamond[2], E. Petit[3], C. Denis[4], Y. Chatelain[1]

[1]UVSQ - [2] CEA - [3]Intel - [4]ENS Paris-Saclay

2017-06-28 – Forum Teratec

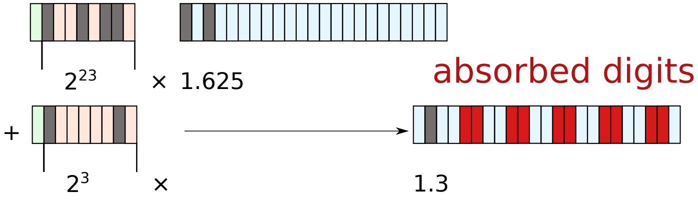
Motivation

Changing architecture, parallelisation, heterogeneity, compiler, optimizations level and language can result in different numerical results.

- Verificarlo is a numerical debugger for the IEEE-754 floating point model
  - Estimate significant digits of a computation
  - Compromise between performance, precision and reproducibility
  - Open Source GPLv3 at `github.com/verificarlo/verificarlo`

Floating point computation: some adverse effects

- A floating point computation approximates the exact computation
- Loss of arithmetical properties (for example the floating point summation is not associative)
  - Absorption, a part of the significant digits cannot be represented in the result format
  - Cancellation, relative error when subtracting variables with very close values



Example by W. Kahan, condition number 2.497e8

$$\begin{pmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{pmatrix} x = \begin{pmatrix} 0.1440 \\ 0.8642 \end{pmatrix}, x_{exact} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$$

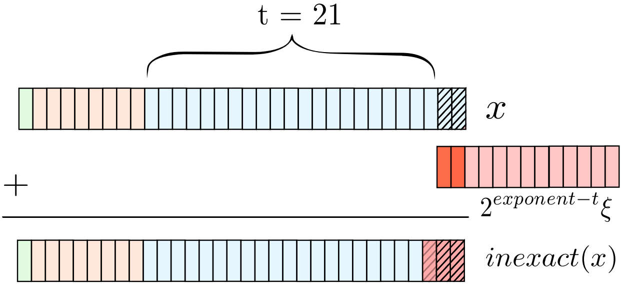
- Results obtained using the LAPACK routines

$$x_{single} = \begin{pmatrix} 1.33317912 \\ -1.00000000 \end{pmatrix} x_{double} = \begin{pmatrix} 2.00000000240030218 \\ -2.00000000359962060 \end{pmatrix}$$

- How to automatically estimate s, the number of significant digits ?
  - Without computing the exact theoretical answer
  - On a whole full scale scientific code
  - Without modifying the application code
  - And taking into account compiler optimization and special instructions

Monte Carlo Arithmetic, [S. Parker, 1999]

$$inexact(x) = x + 2^{exponent-t}\xi \quad, \xi \in [-\frac{1}{2}, \frac{1}{2}], \text{t virtual precision}$$



FP operations $\circ$ are replaced by:

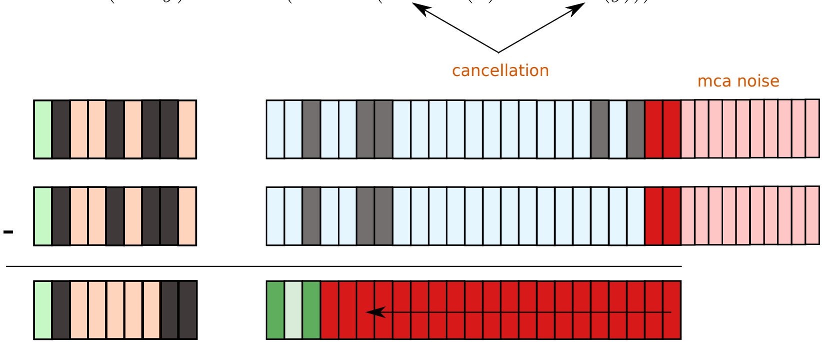
$$mca(x \circ y) = round(inexact(inexact(x) \circ inexact(y)))$$

absorption

cancellation

Monte Carlo Arithmetic: models FP Error Propagation

$$mca(x \circ y) = round(inexact(inexact(x) \circ inexact(y)))$$



Across multiple MCA executions: error digits will change
significant digits will stay stable

Monte Carlo Arithmetic: Estimating output error

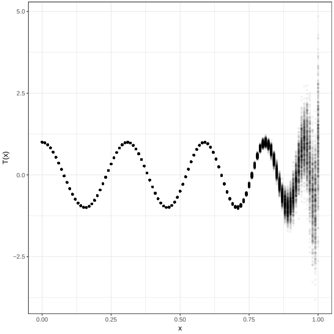
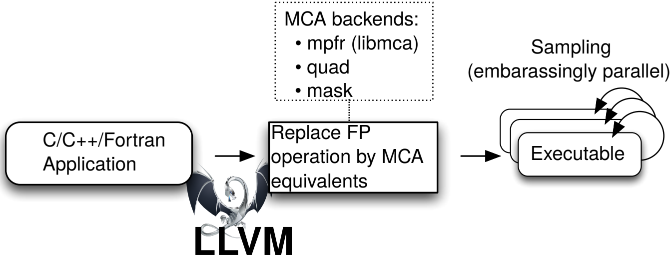


Figure: Tchebychev Polynomial, catastrophic cancellation near 1 [Wilkinson, 1994]

- Estimate the number of significant digits by using N Monte Carlo samples $\tilde{s} = -\log_{10}\left(\frac{\tilde{\sigma}}{\tilde{\mu}}\right)$
- $\tilde{\mu}$: empirical mean and $\tilde{\sigma}$: empirical standard deviation

Verificarlo



- Transparently transforms code to Monte Carlo Arithmetic
- Operates on optimized code: evaluates floating point errors introduced by compiler optimizations

```
-   %37 = fadd fast <4 x float> %wide.load.2, %31
+   %37 = call <4 x float> @_4x_mca_floatadd(<4 x float> %wide.load.2,
                                              <4 x float> %31)
```

Verificarlo Overhead

version	samples	total time (s)	$\frac{time}{sample}$ (s)
original program	1	.056	.056
Verificarlo MASK	128	12.42	.097
Verificarlo MPFR	128	834.57	6.52
Verificarlo QUAD	128	198.58	1.55
Verificarlo MPFR 16 thds.	128	54.39	.42
Verificarlo QUAD 16 thds.	128	12.54	.098

Table: Verificarlo overhead on a compensated sum algorithm (double) on a 16-core 2-socket Xeon E5@2.70GHz.

- Monte Carlo Arithmetic requires additional precision which is costly
- No size fits all
  - MASK backend is cheap (x2 per iteration) but imprecise
  - QUAD backend implements exact MCA model but costly (x27 per iteration)
  - MPFR used only for validation
- Embarrassingly parallel across executions

Example by W. Kahan, condition number 2.497e8

- Recall: computations using IEEE-754 FP numbers

Precision	Result	s
SP	$x(1) = 1.33317912$ $x(2) = -1.00000000$	0 0
DP	$x(1) = 2.00000000240030218$ $x(2) = 2.00000000359962060$	9 9

- Computation performed with Verificarlo($N = 1000$ samples)

Precision	$\hat{\mu}$	$\hat{\sigma}$	$\hat{s}$
Verificarlo SP	$\hat{\mu}(x(1)) = 1.02463705$ $\hat{\mu}(x(2)) = 6.46717332$	$\hat{\sigma}(x(1)) = 6.4...$ $\hat{\sigma}(x(2)) = 9.6...$	0.0 0.0
Verificarlo DP	$\hat{\mu}(x(1)) = 1.9999999992$ $\hat{\mu}(x(2)) = -1.9999999988$	$\hat{\sigma}(x(1)) = 8.4... \times 10^{-9}$ $\hat{\sigma}(x(2)) = 1.2... \times 10^{-8}$	8.3 8.2

  - For this example, verificarlo automatically instrumented LAPACK and BLAS libraries without any modification of their source code

Study on Europlexus (EPX)

www-epx.cea.fr

- Fast transient dynamic simulation
  - soundness of mechanical confinement barriers in nuclear reactors
  - soundness of public structures in case of explosions
- 1 million lines of Fortran 77 and Fortran 90
- Instrumented out of the box with Verificarlo without any change to the source code

Europlexus: Checking FP reproducibility

- Port to new architectures with more vectorization or parallelism
- Select reproducible regression checks for code modernization

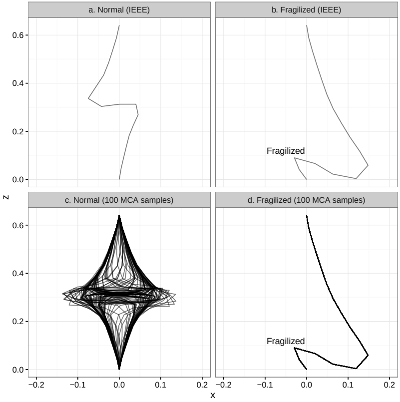
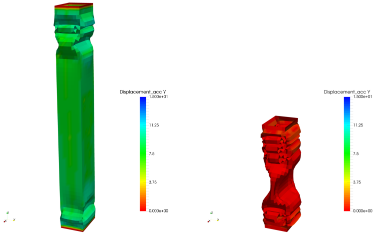
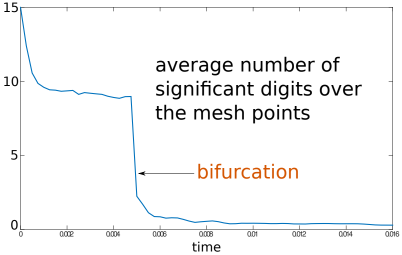


Figure: Buckling simulation of a 1D beam

VTK plugin for Europlexus

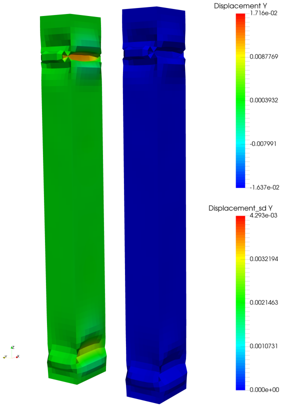
- Europlexus dynamic buckling of a rectangular section beam
- Generic and automatic Verificarlo VTK post-process plugin

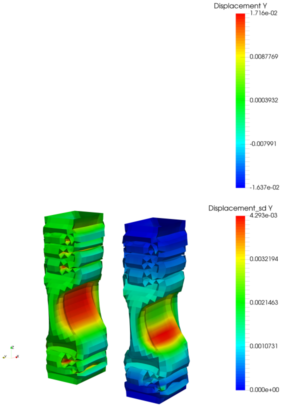




VTK plugin for Europlexus

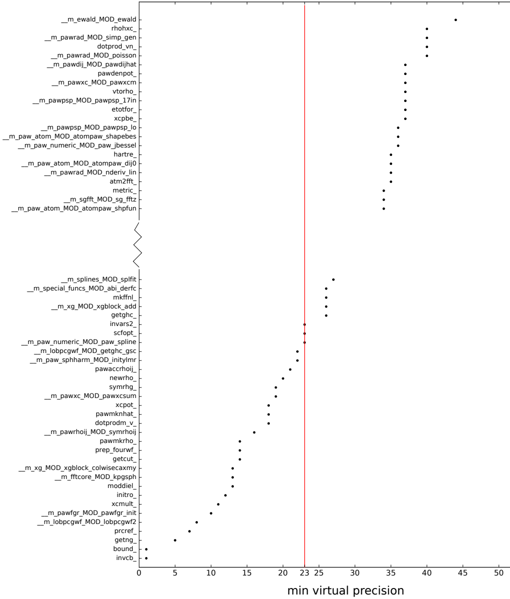
- VTK plugin includes multiple views allowing a refinement of the analysis
- Using stdev and mean allows to better localize the loss of precision on the beam





Verificarlo for Mixed Precision optimization

- Study on ABINIT: DFT abinitio simulation
- Perovskite simulation with non-local effects
- Find the minimal virtual precision for each function that guarantees accurate results (here machine precision)
  - pinpoint fragile routines
  - detect routines that can be converted to single precision
- results from Y.Chatelain, PhD student in collaboration with M. Torrent and J. Bieder



Concluding remarks and future work

- The assessment of the numerical accuracy of scientific codes becomes crucial
  - When porting a scientific code on another programming language or on different computing resources
  - To find the best compromise between performance and precision
- Transparent instrumentation eases large code bases analysis.
- Verificarlo is a young project but tested on
  - Code ASTER and Telemac (EDF)
  - Europlexus, Abinit (CEA)
  - Yales2 (CORIA)
- Future work: Interflop (`github.com/interflop/interflop`)
  - Open framework for numerical analysis tools
  - Collaboration between EDF, Intel, and UVSQ
  - Enables sharing Verrou and Verificarlo back-ends (MCA / Random Rounding) and front-ends (Valgrind / LLVM instrumentation)
  - Unite efforts on post-processing and analysis tools
  - If interested, join us!