Verificarlo: Checking Floating-Point Accuracy Through Monte Carlo Arithmetic

Christophe Denis, Pablo de Oliveira Castro, and Eric Petit
eric.petit@uvsq.fr

Li-Parad, University of Versailles and CMLA, ENS Cachan - UPS
Computing with Floating Point Arithmetic

Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

Verificarlo: an LLVM Tool for automatic MCA analysis

Usage example and industrial applications

Concluding remarks and future work
1. Computing with Floating Point Arithmetic

2. Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

3. Verificarlo: an LLVM Tool for automatic MCA analysis

4. Usage example and industrial applications

5. Concluding remarks and future work
Non reproducibility in HPC simulations

- Same input data:
  - Simulation 1
  - Simulation 2

- Architecture:
  - 4 x86 CPUs
  - 8 Xeon Phi

- Compiler:
  - gfortran
  - icc

- Compiler flags:
  - -O0
  - -O3 –fast-math -mmic

- Language:
  - Fortran 90
  - C++

Numerical results will probably be different

This is not a bug!
Reproducibility versus precision

- Ensuring the numerical reproducibility is not always a requirement!
  - Most the HPC users want to be conservative
  - However does different results means wrong results?

- Precision analysis is required
  - For a given algorithm, precision bounds accuracy
  - Estimate the significant digits of a computation
  - Find the best compromise between performance, precision and reproducibility
Floating point computation: the IEEE-754 standard

- Floating point (FP) numbers approximate real numbers with a finite precision
  - Discrete and finite set of values
- Different representation and encoding in memory defined in IEEE 754
- Trade-off between range and precision
  - Single (32 bits), Double (64 bits)...
- And four rounding modes:
  - nearest, toward $+\infty$, toward $-\infty$, toward zero
Floating point computation: some adverse effects

- A floating point computation \( fl(a \circ b) \) is a model of an exact computing \( a \circ b \)
  - \( fl(a \circ b) = (a \circ b)(1 + \epsilon) \)
- Representation error of real numbers
  - e.g. \( 0.1_d \) is not exactly representable
  - \( fl(0.1) \neq 0.1 \)
- Loss of arithmetical properties (for example the floating point summation is not associative)
  - In single precision, \( 10^{32} - 10^{32} + 0.01 = 0.01 \neq 10^{32} + 0.01 - 10^{32} = 0.0 \)
An example provided by W. Kahan (UCLA)

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

(1)

Results obtained using the LAPACK routines

\[
\begin{pmatrix}
x_{\text{single}}
\end{pmatrix} =
\begin{pmatrix}
1.33317912 \\
-1.00000000
\end{pmatrix},
\begin{pmatrix}
x_{\text{double}}
\end{pmatrix} =
\begin{pmatrix}
2.0000000000240030218 \\
-2.000000000359962060
\end{pmatrix}
\]

How to automatically estimate \( s \), the number of significant digits?

- On a whole full scale scientific code
- Without modifying the application code
- And taking into account compiler optimization and special instructions
Computing with Floating Point Arithmetic

Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

Verificarlo: an LLVM Tool for automatic MCA analysis

Usage example and industrial applications

Concluding remarks and future work
Estimating the numerical precision by using Monte Carlo Arithmetic (MCA) [PARKER97]

- Stochastically simulate rounding and catastrophic cancellation errors
- Introduce a uniformly-distributed error at a virtual precision $t$

$$\text{inexact}(x) = x + 2^{e_x - t} \xi$$

- $e_x$ exponent of $x$, $\xi$ uniform random variable in $[-\frac{1}{2}, \frac{1}{2}]$
- Each floating point operation is transformed in a MCA operation:

$$x \circ y \rightarrow \text{round}(\text{inexact}(\text{inexact}(x) \circ \text{inexact}(y)))$$

- Distribution of the errors is estimated using $N$ Monte Carlo samplings $x$
  - Costly in time, but not in memory and embarrassingly parallel
- $\hat{s}(x)$: estimation of $s$ computed as follows:

$$\hat{s}(x) = -\log_{10} \frac{\hat{\sigma}(x)}{\hat{\mu}(x)}$$

- $\hat{\mu}$: empirical mean value; $\hat{\sigma}$: empirical standard deviation
Computing with Floating Point Arithmetic

Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

Verificarlo: an LLVM Tool for automatic MCA analysis

Usage example and industrial applications

Concluding remarks and future work
Verificarlo: an Automatic LLVM Tool for FP Accuracy Checking using MCA

- Support MCA analysis of large code-bases without any source code modification
  - eg. LAPACK, EDF’s codes ASTER and telemac
- Instrumentation occurs after the optimization passes, just before the back-end ISA code generation
  → Verificarlo analyzes the code which is executed
  ... unlike source code instrumentation
Verificarlo: an Automatic LLVM Tool for FP Accuracy Checking using MCA

- **Using LLVM brings advantages:**
  - The instrumentation library is an independent module which can be tuned for other tools
  - LLVM supports multiple languages and multiple ISA
  - It benefits from the powerful analysis of the LLVM compiler based on code semantics
    - e.g. per function/loop analysis, access to debug info to relate the observation to the source code...

- **But also some constraints:**
  - Tied to LLVM compiler, addressing a new compiler would require to rewrite the compiler pass (but it is a short and simple piece of software)
  - Cannot handle precompiled libraries
Computing with Floating Point Arithmetic

Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

Verificarlo: an LLVM Tool for automatic MCA analysis

Usage example and industrial applications

Concluding remarks and future work
Proof of concept on the example provided by W. Kahan

- Recall: computations using IEEE-754 FP numbers

<table>
<thead>
<tr>
<th>Precision</th>
<th>Result</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>$x(1) = 1.33317912$</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$x(2) = -1.00000000$</td>
<td>0</td>
</tr>
<tr>
<td>DP</td>
<td>$x(1) = 2.000000000240030218$</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>$x(2) = 2.000000000359962060$</td>
<td>9</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Precision</th>
<th>$\hat{\mu}$</th>
<th>$\hat{\sigma}$</th>
<th>$\hat{s}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verificarlo SP</td>
<td>$\hat{\mu}(x(1)) = 1.02463705$</td>
<td>$\hat{\sigma}(x(1)) = 6.4... \times 10^{-9}$</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>$\hat{\mu}(x(2)) = 6.46717332$</td>
<td>$\hat{\sigma}(x(2)) = 9.6...$</td>
<td>0.0</td>
</tr>
<tr>
<td>Verificarlo DP</td>
<td>$\hat{\mu}(x(1)) = 1.99999999992$</td>
<td>$\hat{\sigma}(x(1)) = 8.4... \times 10^{-9}$</td>
<td>8.3</td>
</tr>
<tr>
<td></td>
<td>$\hat{\mu}(x(2)) = -1.9999999988$</td>
<td>$\hat{\sigma}(x(2)) = 1.2... \times 10^{-8}$</td>
<td>8.2</td>
</tr>
</tbody>
</table>

- For this example, verificarlo automatically instrumented LAPACK and BLAS libraries without any modification of their source codes!
Verificarlo is running on the following industrial use cases

- EDF’s TELEMAC-2D simulates free-surface flows modelisation of the free surface flows
  - More than 300k lines of source code
  - Objective: Check the precision of the computation

- EDF’s Code ASTER for structural mechanics
  - M2 Student Lin GUO
  - More than 1,500k lines of source code
  - Objective: Explain non-reproducibility in regression tests

- Lorenz Chaotic models used in weather forecasts (Oxford university)
  - Objective: Determine the best precision of FP numbers to trade performance, energy consumption and accuracy

- and others (NDA)
  - For code modernization: million of Fortran line to replace by Python, C++, etc.
  - Improving post treatment with understandable outputs to the enduser
Computing with Floating Point Arithmetic

Estimating the numerical precision by using Monte Carlo Arithmetic (MCA)

Verificarlo: an LLVM Tool for automatic MCA analysis

Usage example and industrial applications

Concluding remarks and future work
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

- The current version of Verificarlo is a fully automatic tool to estimate the numerical precision, but it still requires expertise...

Future work

- Integration of a statistical post-treatment toolbox to go beyond the standard deviation analysis
- ...and for non-experts to understand and to interpret the output of the MCA analysis
- Extend our experience on numerical verification of full-scale real-life applications
- Methodologies to pinpoint the exact operation, loop, or routine that is to blame for a precision loss
- Support external precompiled libraries
Acknowledgments

Verificarlo

Verificarlo is developed under a close scientific collaboration with Christophe Denis from ENS Cachan, Eric Petit and Pablo de Oliveira Castro from Université de Versailles Saint-Quentin-en-Yvelines

This work has been partially supported by a public grant as part of Investissement d’avenir, référence ANR-11-LABX-0056-LMH, LabEx LMH