Verificarlo: Numerical debugger and optimizer

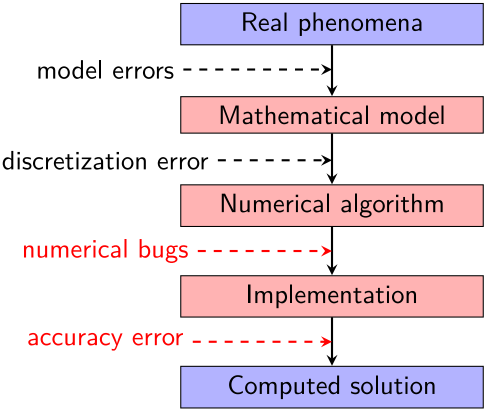
Pablo de Oliveira Castro, pablo.oliveira@uvsq.fr

VI-HPS TW43, CALMIP, January 2024

LI-PaRAD, UVSQ, Université Paris-Saclay

Numerical HPC codes

Building numerically robust simulations is a complex task



- Avoid numerical bugs
- Order of operations matters (vectorization, compiler, parallelisation)
- Explore trade-offs between precision and performance

Outline

IEEE-754 Floating-Point arithmetic

Finding numerical bugs with MCA

Optimizing precision

Perspectives

Floating-Point IEEE-754 representation

IEEE-754 defines a standardized FP representation

$$f = s \times 2^e \times m$$

$$\underbrace{\boxed{s}}_{\text{sign}}\ \underbrace{\boxed{e_1}\boxed{e_2}\cdots\boxed{e_q}}_{\text{exponent}}\ \underbrace{\boxed{m_1}\boxed{m_2}\cdots\boxed{m_p}}_{\text{p. mantissa}}$$

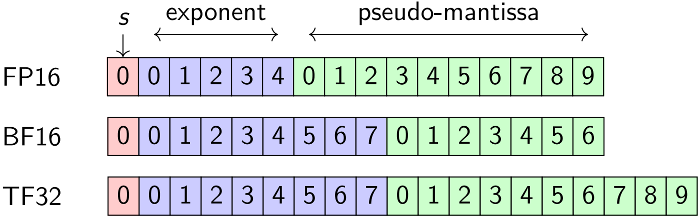
$$\begin{aligned}(1.1001 \times 2^0)_2 =& (1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4})_{10} \\ =& (1 + 0.5 + 0.125)_{10} = 1.625_{10}\end{aligned}$$

Context: Reducing precision

IEEE-754

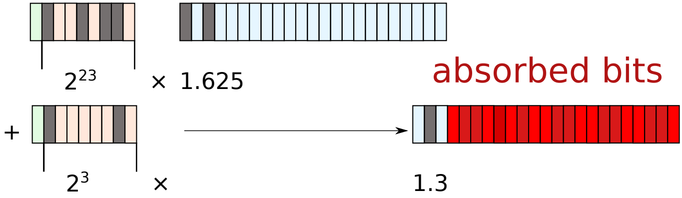
- FP64 (double): 1 (sign) / 11 (exp.) / 52 (p.mantissa)
- FP32 (float): 1 (sign) / 8 (exp.) / 23 (p.mantissa)

- Shift from FP64-CPUs towards high throughput smaller datatypes
  - Strong trend driven by AI workloads
  - Neural network using lower-precision TF32, FP16 and BF16
  - Gain in throughput performance and energy efficiency



Floating-point: numerical errors

- Multiplication and division (away from 0) are *usually* safe
- Summation can produce large errors
  - Absorption, a part of the significant digits cannot be represented in the result format
  - Cancellation, high-relative error when subtracting variables with close values
- Floating-point summation is not associative



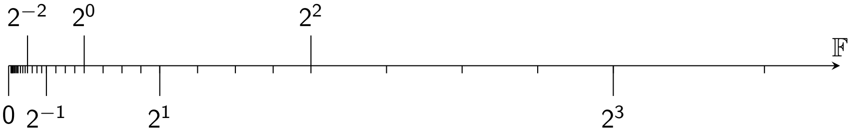
Floating-point arithmetic standard error model

- $x = \pm 2^e \times m$

IEEE-754 implementation guarantees for $\circ \in \{+, -, *, /\}$ that

$$\widehat{z} = fl(x \circ y) = (x \circ y)(1 + \delta) \quad \text{with } |\delta| \leq u \text{ unit roundoff}$$

$(1 + \delta)$ captures the relative error of an IEEE-754 operation



Common sources of FP bugs in simulation

- Large summations: dot products, integral computations, reductions, averages or deviations
- Accumulation of errors over time: explicit methods
- Gradient computation of near values: small variations in large quantities, residual
- Non reproducibility
  - Parallel computation: changes the order of operations and the decomposition
  - Aggressive loop vectorization
  - Unstable branches

Some tricks to attenuate numerical bugs

- Rewrite faulty expression to remove cancellations or absorptions
- Replace faulty expression with a well-behaved approximation
- Use good scaling factors / format to avoid over/underflows
- Increase accuracy (mixed-precision, compensated algorithms)

Objectives

Numerical Simulation

- Find numerical bugs
- FP portability across SW and HW
- Optimize and harness smaller FP formats

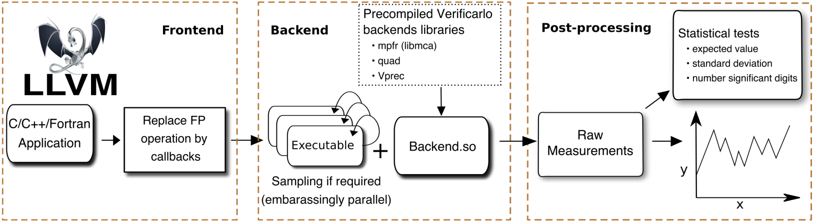
What methods and tools can help us in the process?

- **Static analysis**: formal assisted proof or interval arithmetic
  - Coquelicot, Fluctuat, Gappa, FPTaylor ...
  - Strong guarantees but costly and intractable on complex applications
- **Dynamic analysis**:
  - Instability detection (Verificarlo, Verrou, Cadna, FPDEBUG, Herbgrind)
  - Automatic expression rewriting (Herbie)
  - Mixed-precision exploration (Verificarlo, Precimonious, Promise, CRAFT, fpPrecisionTuning)

Verificarlo

✓erificarlo github.com/verificarlo/verificarlo

- Based on the LLVM compiler
- Active open source project with 15 contributors
- Backends: debugging (MCA, Cancellation) + mixed-precision (Vprec)
- MCA overhead from $\times 6$ (binary32) to $\times 160$ (binary64).



Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic.
Denis, de Oliveira Castro, Petit. IEEE Symposium on Computer Arithmetic 2016

Outline

Monte Carlo Arithmetic [Stott Parker, 1999]

- Each FP operation may introduce a δ error

$$\widehat{z} = fl(x \circ y) = (x \circ y)(1 + \delta)$$

- Monte Carlo Arithmetic key principle
  - Make δ a random variable (stochastic rounding)
  - Monte Carlo sampling

- The values returned by n runs of the program using stochastic arithmetic are seen as realizations of a random variable X.
- $\hat{\mu}$ and $\hat{\sigma}$ are the empirical average and standard deviation.

Why Monte Carlo Arithmetic?

- Compare computation against an exact reference is easier.
- Sometimes,
  - hard to get an exact reference value (intermediate computations)
  - different results are not necessarily wrong

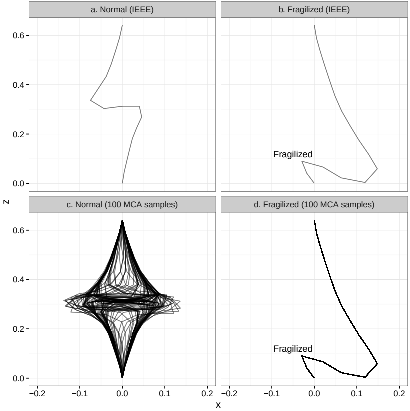


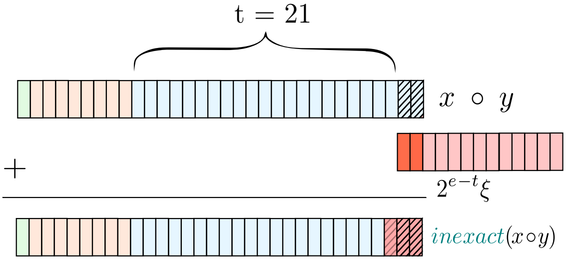
Figure: Buckling of a 1D beam with Europlexus collaboration with O. Jamond

Monte Carlo Arithmetic: Random Rounding

- MCA simulates error with

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

  - $e_x = \lfloor \log_2 |x| \rfloor + 1$ is the order of magnitude of x;
  - ξ is an uniform random variable in $(-\frac{1}{2}, \frac{1}{2})$;
  - t is the virtual precision, selects the magnitude of the simulated error.

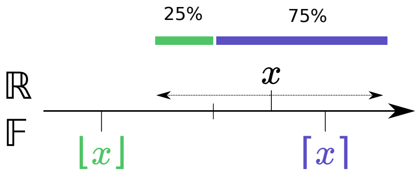


$$mca(x \circ y) = round(inexact(x \circ y))$$

absorption and rounding errors

MCA Random Rounding at the ulp

- t=24 for `float` and t=53 for `double` is a special case: the virtual precision corresponds to a one ulp ϵ error.
- The random error introduced is in $\left(-\frac{\epsilon}{2}, \frac{\epsilon}{2}\right)$.
- MCA result is either the downwards or upwards roundoff, with a probability proportional to the fractional part.



- MCA naturally preserves exact operations.

Summation example: t=53

- 0.1 is not representable in $\mathbb{F}$. The closest value is 0.1000000000000000555...

```
double a = 0;
for (int i=0; i < 10000; i++)
  a += 0.1;
```

```
double a = 0;
for (int i=0; i < 10000; i++)
  a += 0.25;
```

Sample	MCA RR t=53
1	1000.0000000001186891
2	1000.0000000001174385
3	1000.0000000001175522

Sample	MCA RR t=53
1	2500.0
2	2500.0
3	2500.0

Example: Linear 2x2 System

- Ill-conditioned linear system (condition number 2.5×10^8).
- We solve it with the Cramer's formula.

$$\begin{pmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{pmatrix} x = \begin{pmatrix} 0.1440 \\ 0.8642 \end{pmatrix}$$

$$x_{\text{real}} = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \qquad x_{\mathrm{IEEE}} = \begin{pmatrix} 1.999999958366637 \\ -1.999999972244424 \end{pmatrix}$$

- The IEEE-754 binary64 result has 8 significant decimal digits or 28.8 significant bits.

MCA 2x2 System: Stott Parker's significant bits

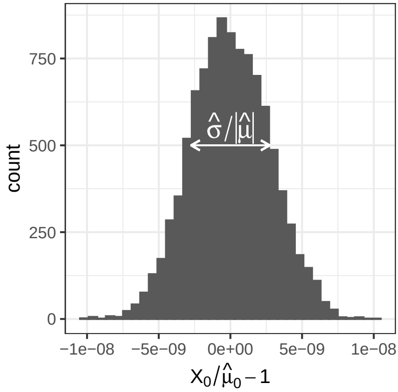


Figure: Error distribution for 10000 samples FULL MCA ($t = 53$)

$1.999999850477848e+00$

$1.9999999957687429e+00$

$2.000000024646973e+00$

- Stott Parker defines the number of significant bits as

$$s_{\text{PARKER}} = -\log_2 \frac{\hat{\sigma}}{|\hat{\mu}|} \approx 28.5.$$

$$(s_{\text{IEEE}} \approx 28.8)$$

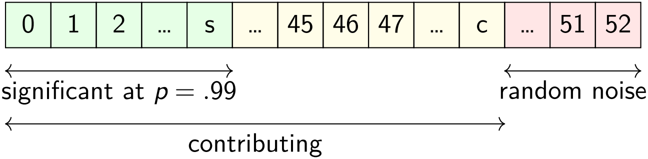
- Magnitude of the signal to noise ratio.
- But how confident are we that it is a good estimate?

Probabilistic definition of Significant bits

Significant bits

The number of significant bits with probability p can be defined as the largest number s such that

$$\mathbb{P}\left(|Z| \leq 2^{-s}\right) \geq p \quad \text{where } Z = X/X_{ref} - 1$$



Confidence Intervals for Stochastic Arithmetic. Sohier, de Oliveira Castro, Févotte, Lathuilière, Petit, Jamond. ACM Transactions Mathematical Software 2022.

CNH: Significant bits lower bound

- Given a centered normal error distribution (CNH) and $X_{ref} = \hat{\mu}$ we show

$$s \geqslant \underbrace{-\log_2\left(\frac{\hat{\sigma}}{|\hat{\mu}|}\right)}_{s_{\text{PARKER}}} - \left[\underbrace{\frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right)}_{\chi^2 \text{ confidence interval on } \hat{\sigma}} + \underbrace{\log_2\left(F^{-1}\left(\frac{p+1}{2}\right)\right)}_{\text{depends only on } p}\right] \tag{1}$$

- F is the cumulative distribution function of $\mathcal{N}(0,1)$.
- For $n \to \infty$ samples and $p = 0.68$ $s \geq -log_2\hat{\sigma}/|\hat{\mu}|$ (Parker)
- For $n = 30$ samples and $p = 0.99$ $s \geq -log_2\hat{\sigma}/|\hat{\mu}| - 1.792$
- For $n = 15$ samples and $p = 0.99$ $s \geq -log_2\hat{\sigma}/|\hat{\mu}| - 2.023$

A Bernoulli estimator provides a probabilistic lower-bound s for general distributions.

Compiler optimizations are instrumented

- Instrumentation occurs **just before code generation**
- Enables analyzing precision loss due to compiler optimizations

```
for (int i=1;i<n;i++) {
  y = f[i] - c;
  t = sum + y;
  c = (t - sum) - y;
  sum = t;
 }
return sum;
```

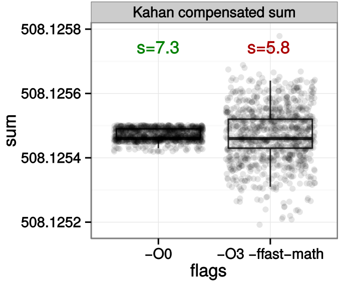


Figure: Analysis of the effect of compiler flags on a Kahan compensated sum algorithm (binary32)

Overhead

		verificarlo backends		
	original	IEEE	MCA quad	MCA integer
Kahan binary32	1.34s	2.36s ($\times$1.7)	6.28s ($\times$4.7)	7.76s ($\times$5.8)
Kahan binary64	1.34s	2.34s ($\times$1.7)	105s ($\times$78)	64s ($\times$48)
NAS CG A	0.80s	6.41s ($\times$8)	173s ($\times$216)	128s ($\times$160)

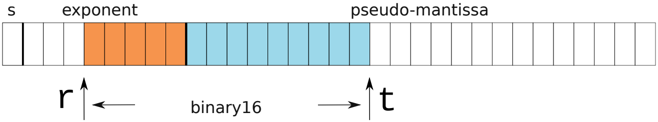
Table: Execution time (and slowdown) for a Kahan sum of 100 millions elements and for the NAS CG A using different Verificarlo backends.

Outline

VPREC for mixed precision

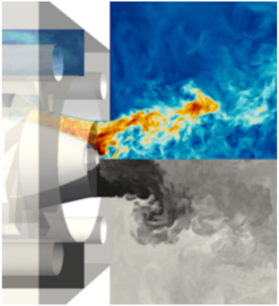
- Estimate numerical effect of bfloat16, tensorflow32, fp24 on standard IEEE-754 hardware (before paying the porting cost)

- VPREC emulates any range and precision fitting in original type
  - Uses native types for storage and intermediate computations
  - Handle overflows, underflows, denormals, NaN, $\pm\infty$
  - Rounding to nearest (faithful)
  - Fast: $\times$ 2.6 to $\times$ 16.8 overhead



YALES2 application

- Computational Fluid Dynamics solver from Coria-CNRS



- Deflated Preconditioned Conjugate Gradient
- CG iterations alternate between a:
  - Deflated coarse grid
  - Fine grid

VPREC: Find minimal precision over iterations that preserves convergence (dichotomic exploration)

Automatic exploration of reduced floating-point representations in iterative methods. Chatelain, Petit, de Oliveira Castro, Lartigue, Defour. Euro-Par 2019

Mixed-precision on Yales2

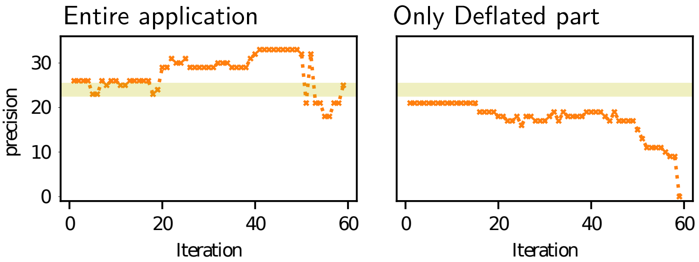


Figure: Minimal precision that preserves convergence.

Energy	16% gain on the deflated part
Communication	28% gain on communication volume
Time	10% speedup on CRIANN cluster (560 nodes)

Source-code localization

- Delta-Debug [Zeller 2001] in Verificarlo
  - Configurations are the sets of floating-point instructions.
  - A bug is a numerical instability.
- Find unstable instructions for rounding / cancellations.
- Find instructions that can be run in lower precision.

Step	Instructions with MCA noise	Numerically Stable
1	1 2 3 4	stable
2	. . . . 5 6 7 8	unstable
3	. . . . 5 6 . .	stable
4	. . . . . . 7 8	unstable
5	. . . . . . 7 .	unstable
Result (ddmin)	. . . . . . 7 .	

Table: Example of Delta-Debug localization. For ddmin of size 1, DD is equivalent to binary search $O(log(N))$; but DD also handles efficiently bugs that result of the combination of multiple faulty instructions.

Hands-on Tutorial !

Retrieve tutorial sources

```
$ cp -r ~oliveira/verificarlo-tutorial ~
$ cd ~/verificarlo-tutorial/
```

Activate environment (must be done from `turpanlogin`)

```
$ source env.sh
```

Allocate compute node

```
$ salloc --nodes=1 --time=2:00:00
$ ssh <node-hostname>
```

Outline

IEEE-754 Floating-Point arithmetic

Finding numerical bugs with MCA

Optimizing precision

Perspectives

Stochastic rounding can mitigate error propagation

Let us consider the inner product $y = a^\top b$ where $a, b \in \mathbb{R}^n$. We consider the forward error $Z = \frac{|\hat{y}-y|}{|y|}$.

- SR (MCA RR) errors bounds are asymptotically better

IEEE-754 in $O(n)$

$Z \leq \mathcal{K}_1 \gamma_n(u/2)^n$

SR in $O(\sqrt{n})$

- Ipsen (AH):
  $Z \leq \mathcal{K}_1 \sqrt{u\gamma_{2n}(u)}\sqrt{\ln \frac{2}{\lambda}}$
- Ours (BC):
  $Z \leq \mathcal{K}_1 \sqrt{\gamma_n(u^2)}\sqrt{\frac{1}{\lambda}}$

where $\gamma_n(u) = (1+u)^n - 1$ and K_1 is the condition number of y.

Stochastic Rounding Variance and Probabilistic Bounds: a new approach. El-Arar, Sohier, de Oliveira Castro, Petit. SIAM CSE 202.

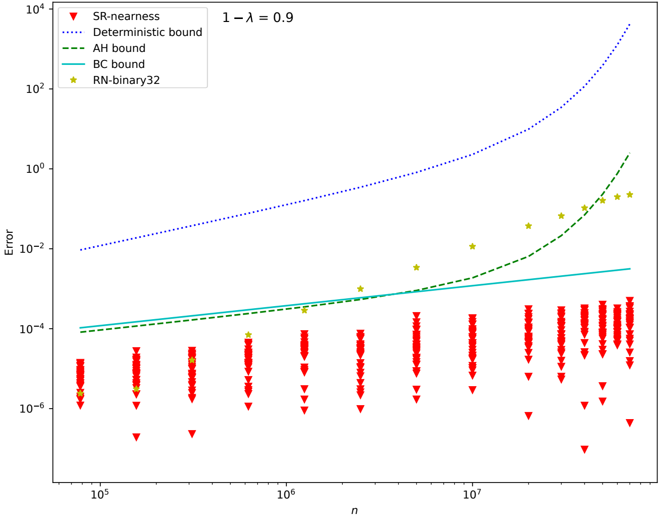


Figure: SR vs. IEEE-754 for the inner product with inputs in $(0,1)$

- SR mitigates the biased absorptions in the IEEE-754 RN summation.
- MCA is not always a good model for IEEE-754 RN. Control divergence between MCA and RN behavior in Verificarlo studies.