Verificarlo stochastic rounding and variable precision : exploring accuracy and reproducibility.

Pablo de Oliveira Castro <pablo.oliveira@uvsq.fr>

CEEC Webinar 2025-03-04

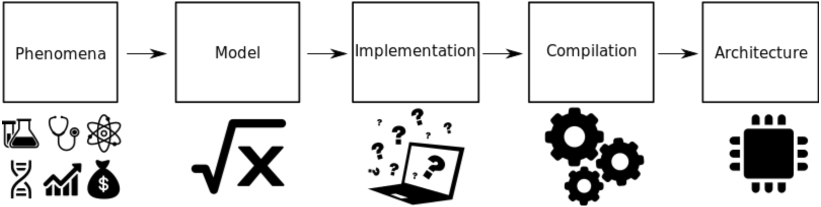
LI-PaRAD, UVSQ, Université Paris-Saclay

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

Context: Floating-Point issues

Building numerically robust numerical simulations is a complex task



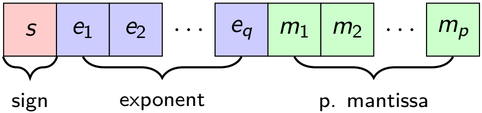
Floating-Point (FP) challenges

- Model or Discretization error (approximation, conditioning)
- IEEE-754 (representation, absorption, cancellation)
- Order of operations matters (vectorization, compiler, parallelisation)
- Reducing precision saves energy and time-to-solution

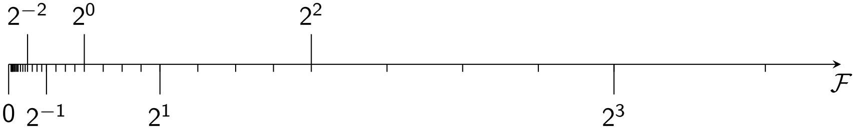
Floating-Point IEEE-754 representation

IEEE-754 defines a standardized FP representation

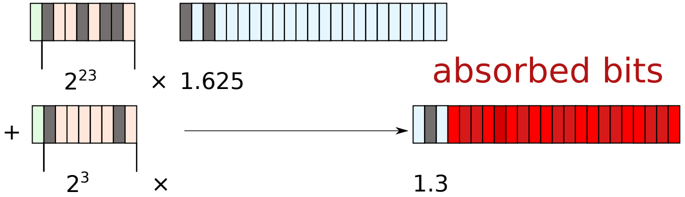
$$f = s \times 2^e \times m$$



- binary64: 1 bit sign, 11 bits exponent, 52 bits pseudo-mantissa
- binary32: 1 bit sign, 8 bits exponent, 23 bits pseudo-mantissa



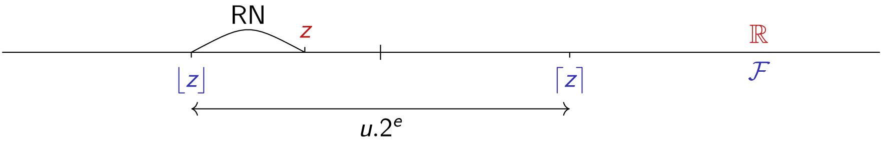
Floating-point arithmetic errors



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

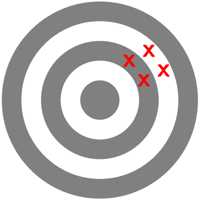
$$\widehat{z} = (x \circ y)(1 + \delta) \quad \text{with } |\delta| \leq u/2$$

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

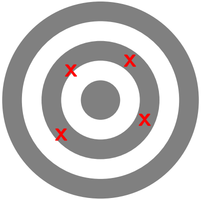


- IEEE-754 rounding is deterministic

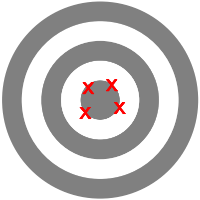
Metrology ISO-5725



precision



trueness



accuracy

- We do not always have a reference value
  - multiple solutions are admissible
  - unknown : new simulation, intermediate result

- Checking precision and reproducibility do not require a reference
  - Part 1: Monte Carlo Arithmetic / Stochastic Rounding
  - Part 2: Verificarlo + VPREC

Outline

Monte Carlo Arithmetic [Stott Parker, 1999]

- Each FP operation may introduce a δ error

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

- Monte Carlo Arithmetic makes δ a random variable

$$\begin{aligned} \widehat{z_1} &= (a + b)(1 + \delta_1) \\ \widehat{z_2} &= (c + d)(1 + \delta_2) \\ \widehat{z} = \widehat{z_3} &= (z_1 \times z_2)(1 + \delta_3) \end{aligned}$$

- The forward error $\Psi = \frac{\hat{z} - z}{z}$ is analyzed probabilistically
  - Stochastic process function of the $\delta_1, \ldots, \delta_k$.

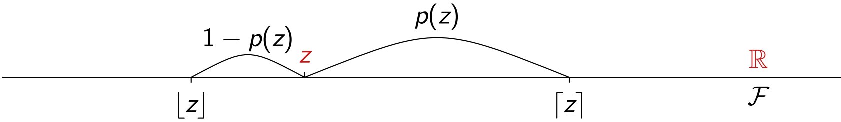
- How to choose the δ_k distribution?

Stochastic Rounding (SR) $\to$ unbiased

- Upward rounding $\lceil z \rceil$ and downward rounding $\lfloor z \rfloor$:

$$\widehat{z} = z(1+\delta) \text{ with } |\delta| \le u$$

$$\widehat{z} = \begin{cases} \lceil z \rceil & \text{with probability } p(z), \\ \lfloor z \rfloor & \text{with probability } 1 - p(z). \end{cases}$$



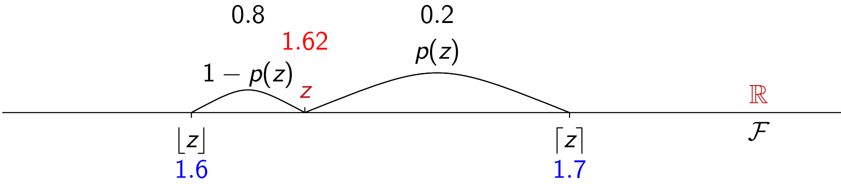
- $p(z) = \frac{z - \lfloor z \rfloor}{\lceil z \rceil - \lfloor z \rfloor}$ and $E(\hat{z}) = p(z)\lceil z \rceil + (1 - p(z))\lfloor z \rfloor = z$.

Stochastic Rounding (SR) $\rightarrow$ unbiased

- Upward rounding $\lceil z \rceil$ and downward rounding $\lfloor z \rfloor$:

$$\widehat{z} = z(1+\delta) \text{ with } |\delta| \le u$$

$$\widehat{z} = \begin{cases} \lceil z \rceil & \text{with probability } p(z), \\ \lfloor z \rfloor & \text{with probability } 1 - p(z). \end{cases}$$



- $p(z) = \frac{z - \lfloor z \rfloor}{\lceil z \rceil - \lfloor z \rfloor}$ and $E(\hat{z}) = p(z)\lceil z \rceil + (1 - p(z))\lfloor z \rfloor = z$.
- $1.7 \times 0.2 + 1.6 \times 0.8 = 1.62$.

SR errors are mean independent

- In SR, for $x_1, x_2, x_3 \in \mathcal{F}$ and $\circ_1, \circ_2 \in \{+, -, *, /\}$,

$$z = x_1 \circ_1 x_2 \circ_2 x_3 \quad \Longrightarrow \quad \hat{z} = ((x_1 \circ_1 x_2)(1+\delta_1) \circ_2 x_3)\,(1+\delta_2),$$

- $E(\delta_1) = E(\delta_2) = 0$.

Lemma (Connolly et al.)

For $\delta_1, \delta_2, \ldots,$ obtained from an SR computation in that order, the δ_k are mean independant random variables,

$$E(\delta_k/\delta_1, \ldots, \delta_{k-1}) = E(\delta_k) = 0$$

- Independence $\Longrightarrow$ **Mean independence** $\Longrightarrow$ uncorrelatedness.

Bounds for sum-product DAGs

For z resulting of a multi-linear sum-product computation graph with n SR operations,

- $\Psi = \frac{\widehat{z} - z}{z}$ is a martingale (generalisation of a random walk)
- $E(\Psi) = 0$
- $|\Psi|$ is bounded by $\mathcal{O}(\sqrt{n}u)$ at fixed probability where n is the number of operations

Error Analysis of sum-product algorithms under stochastic rounding de Oliveira Castro, El-Arar, Petit, Sohier, arXiv 2024.

- The paper gives tighter bounds depending on the operations combinations.

Bounds for multi-linear algorithms

SR sum-product analysis gives error bounds for multi-linear algorithms:

- Dot product $\mathcal{O}(\sqrt{n}.u)$
- Horner's polynomial evaluation $\mathcal{O}(\sqrt{n}.u)$
- Pairwise summation $\mathcal{O}(\sqrt{\log_2 n}.u)$
- Karatsuba multiplication $\mathcal{O}(\sqrt{\log_2 n}.u)$

What about non-linear algorithms or complex numerical software with thousands of lines?

$\rightarrow$ Monte Carlo Simulation

Example: Linear 2x2 System

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

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

$$x_{\text{real}} = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \qquad x_{\mathrm{IEEE}} = \begin{pmatrix} 1.99999999 58366637 \\ -1.99999999 72244424 \end{pmatrix}$$

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

MCA 2x2 System: Stott Parker's significant bits

$1.9999999850477848e+00$

$1.9999999957687429e+00$

$2.0000000024646973e+00$

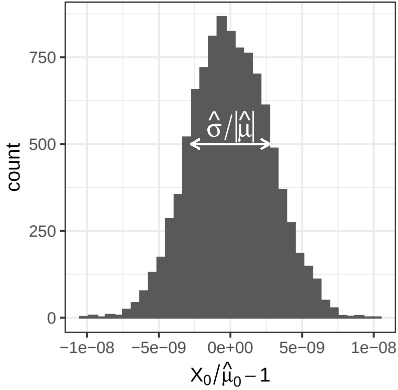


Figure: Error distribution for 10000 samples FULL MCA ($t = 53$)

- 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.
- We provide confidence intervals depending on number of samples[a]

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

SR to detect rounding bias in IEEE-754

- Round-to-nearest is prone to absorptions and becomes biased in large summations.
- SR unbiasedness avoids (and detects) stagnation.

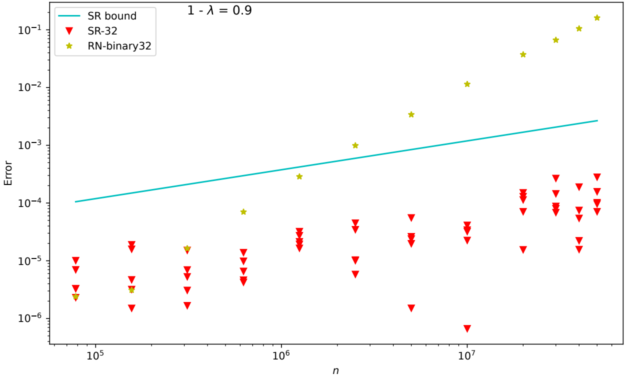


Figure: Dot product of two vectors of n elements, SR vs. RN errors

Stochastic Rounding Variance and Probabilistic Bounds: a new approach. El Arar, Sohier, de Oliveira Castro, Petit. SIAM JSC, 2022.

Outline

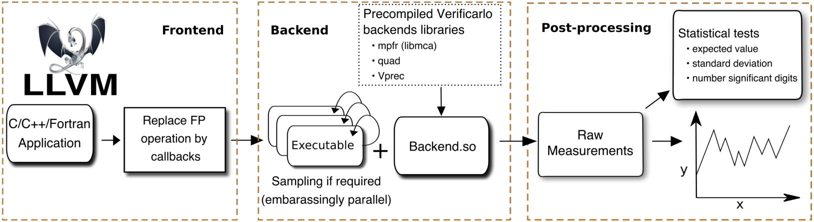
Stochastic Rounding

Verificarlo

Verificarlo

`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

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;
```

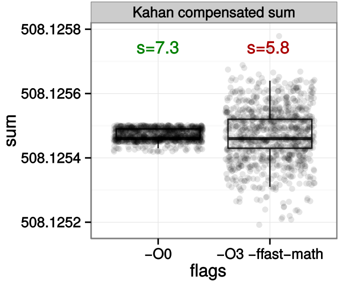


Figure: Analysis of the effect of compiler flags on a Kahan compensated sum algorithm (binary32)

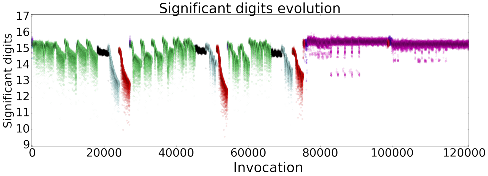
Overhead

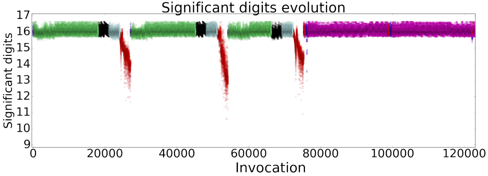
		verificarlo backends		
	original	IEEE	MCA quad	MCA integer
Kahan binary32	1.34s	2.36s ($\times$1.7)	6.28s ($\times$4.7)	7.76s ($\times$5.8)
Kahan binary64	1.34s	2.34s ($\times$1.7)	105s ($\times$78)	64s ($\times$48)
NAS CG A	0.80s	6.41s ($\times$8)	173s ($\times$216)	128s ($\times$160)

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

Example: Loss of signifiance in ABINIT

- ABINIT, collaboration with CEA (Chatelain, Torrent, Bieder)
- Calculates observable properties of materials (optical, mechanical, vibrational)



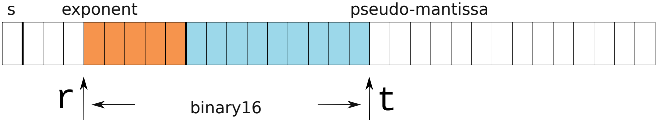


Fixing Simp_gen

- Run: total-energy for $BaTiO_3$. Trace of Simpson's integral.
- Replaced by a compensated version Dot2 (Ogita et al.)
- Colors capture the different call-site paths
- 1 CSP has still precision loss due to reentrance of the error

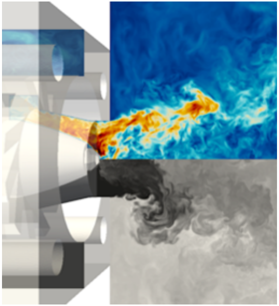
VPREC for mixed precision

- Estimate numerical effect of fp32, 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

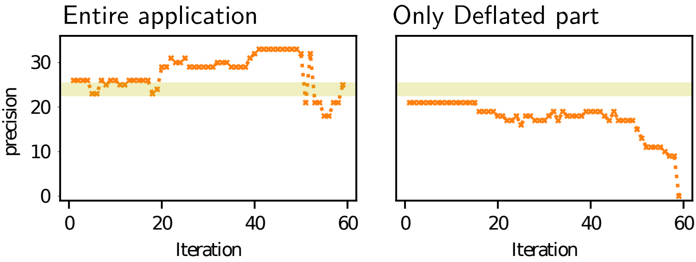


Figure: Minimal precision that preserves convergence.

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

Combining VPREC + SR

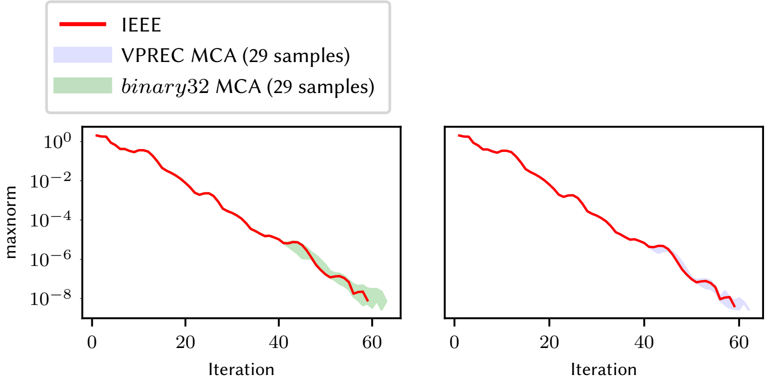


Figure: Resiliency of VPREC and `binary32` configurations. In red the IEEE maxnorm convergence for reference. Blue envelop shows the 29 MCA samples for the previously found VPREC configuration. Green envelop shows the 29 MCA samples for the `binary32` configuration. All samples converge, showing the resiliency of both configurations.

Conclusion

- Verificarlo, an LLVM based tool, transparently instruments large codes with VPREC or SR rounding.
  - SR *in average* analysis is a powerful tool to analyze the reproducibility of a numerical program.
  - VPREC emulates the effect of mixed-precision on standard hardware.

- Used on many large codes: ABINIT, Dipy, EPX, Yales2, QMCkl, etc.
- Limitations: costly overhead and data-dependent analysis.

- Collaboration with Y. Chen and R. Iakymchuk on Nekbone and Neko.

Thanks !

Acknowledgments: Y. Chatelain, E. El-Arar, E. Petit, D. Sohier, ...