Verificarlo: Floating-point Computing Verification and Optimization

Eric Petit, Intel DCG-E&G

Pablo Oliveira, Yohan Chatelain, UVSQ, ECR Lab.

David Defour, UPVD, ECR Lab.

Et al.

IXPUG CERN, Geneva, Swiss, 27th September 2019

verificarlo

Motivation

- Gap between real numbers and machine representation is amazingly complex
  - IEEE754 Floating point is an highly engineered solution (hw and sw) that has empirically proved to be a good trade-off
- HPC dev are conveniently focusing on using double precision
- Processor arithmetic is entering a new burst of evolution
  - New application such as ML can use more efficient representations (e.g. BF16)
  - Power efficiency requirement
  - Indeterminism is the new rule (parallelism, runtime events, system. . .)

Issue with reproducibility

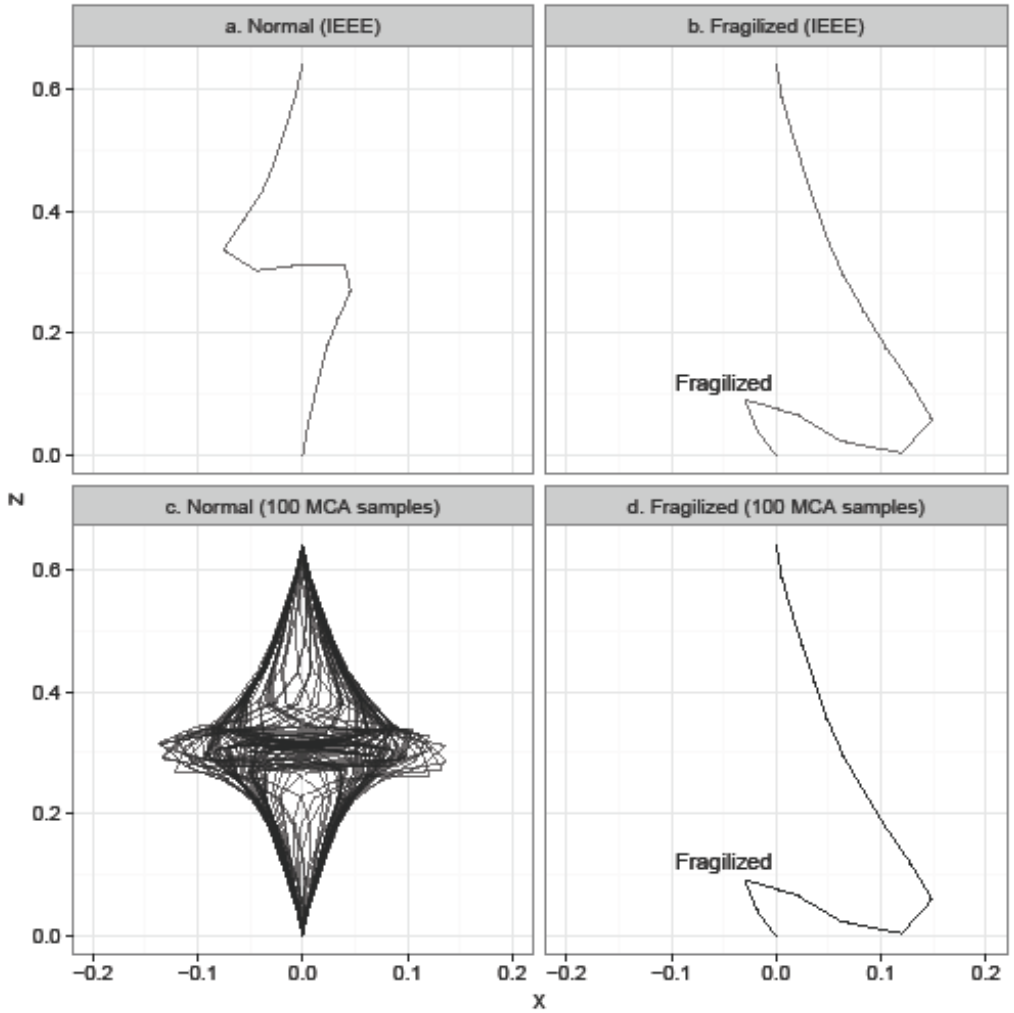
- Same program, same data, can generate different numerical results:
  - Performance improvement allows larger, more complex, higher resolution simulations.
  - Changing architecture, parallelization, heterogeneity, compiler, optimizations level and language

- **How to assess correctness of a result?**
- **How to validate an implementation (hw or sw)?**
- **How to produce code resilient to indeterminism?**
- **How to find the most efficient format for a given application?**

Does different results means wrong results?

- Expected behavior?
  - Physics?
  - Model error?
  - FP error?

- Ensuring the numerical reproducibility is not always a good idea/requirement!



verificarlo

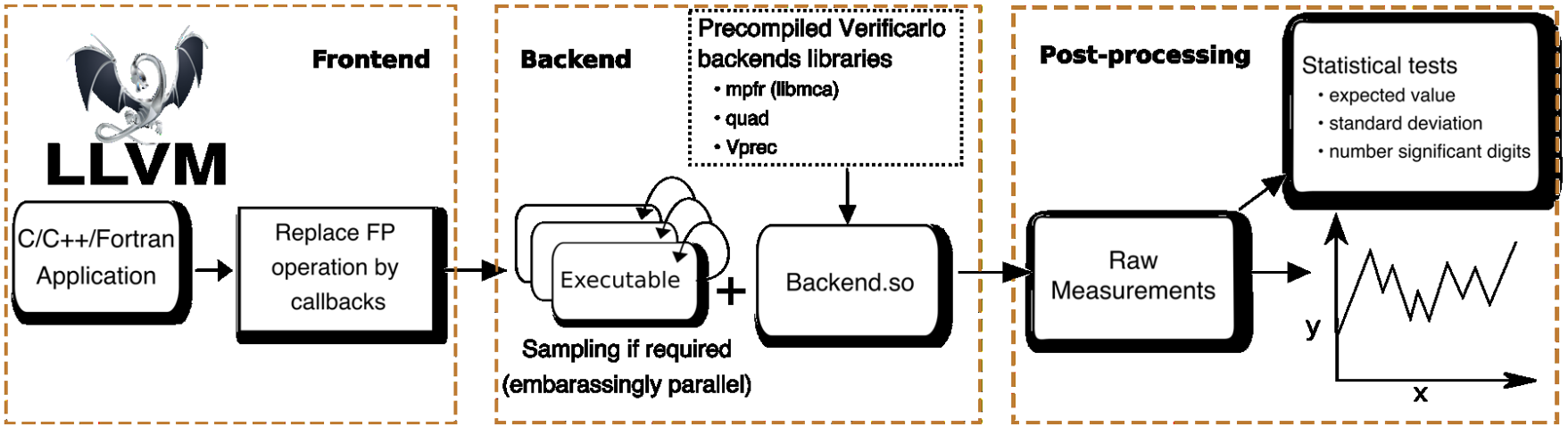
Statistical debugging and verification downside

- Is statistical FP implementation debugging, optimization and verification enough for you?
  - Unitary test coverage? (generalization issue)
  - For my use case, I am 95% confident that with a probability *p*:
    - (The model predicts that) My nuclear reactor will not melt.
    - My search engine is giving accurate results.
  - FP arithmetic statistical verification tools verify the implementation of your model with a probability p on a given set of experiment with a given confidence
- Debugging on the other hand is always a good idea ;)

Verificarlo [Veri15, VeriT17, Inter18, Vprec19]

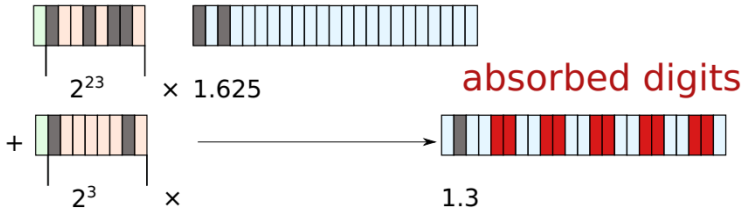
https://github.com/verificarlo/verificarlo

- LLVM compiler pass to replace all floating point operations by call-back to custom arithmetic backend
  - Verifying/debugging: MonteCarlo Arithmetic allows statistical analysis of rounding errors and tracer extension follows FP characteristics over time with contextual info (which variable, callsite, iteration…) **=> Today's focus**
  - New Variable precision backend and variable precision runtime

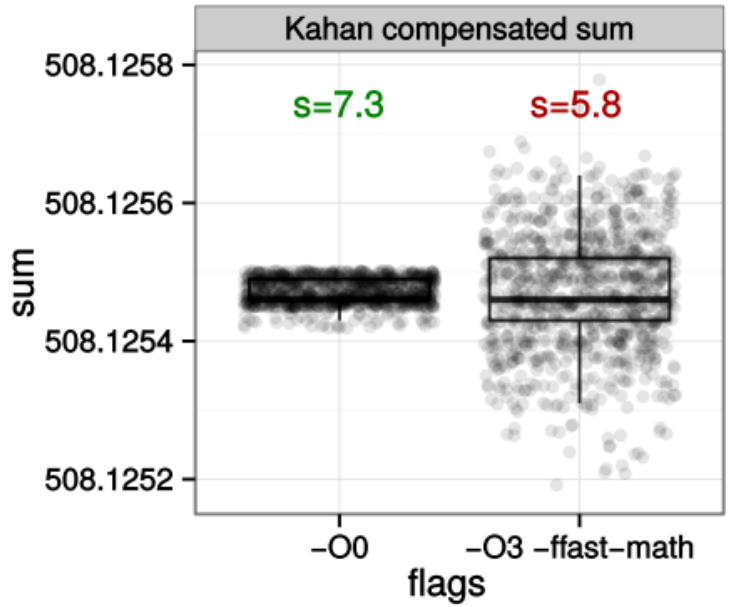


Verificarlo for **debugging and validation** of FP computation

- The instrumentation occurs **just before code generation**
- Verificarlo analyzes the code **after optimization**



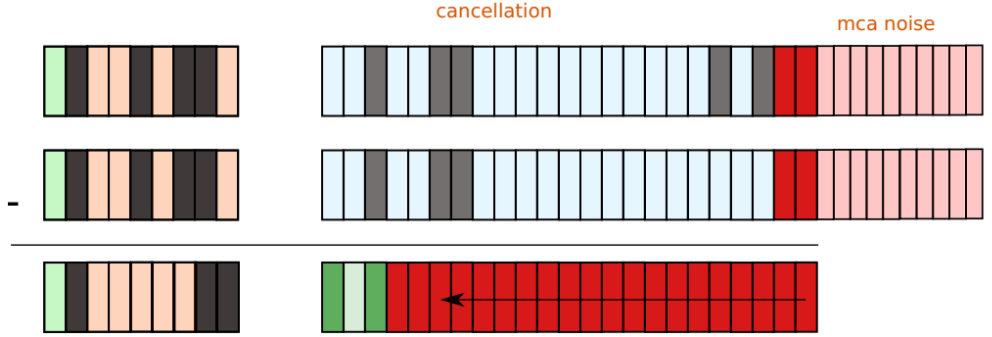

```
for (int i=1;i<n;i++) {
  y = f[i] - c;
  t = sum + y;
  c = (t - sum) - y;
  sum = t;
 }
return sum;
```



How MCA works? (in a nutshell)

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

cancellation



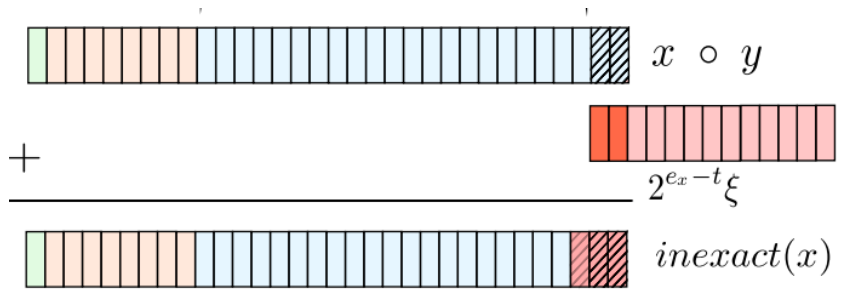
Across multiple MCA executions: error digits will change
significant digits will stay stable

- Each FP operation may introduce a δ error

$$z = fl[x + y] = (x + y)(1 + \delta)$$

- When chaining multiple operations, errors can accumulate and snowball
- Monte Carlo Arithmetic key principle
  - Make δ a random variable

Monte Carlo simulation to empirically estimate the FP error
ution [Stott Parker, 1997]



FP operations $\circ$ are replaced by:

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

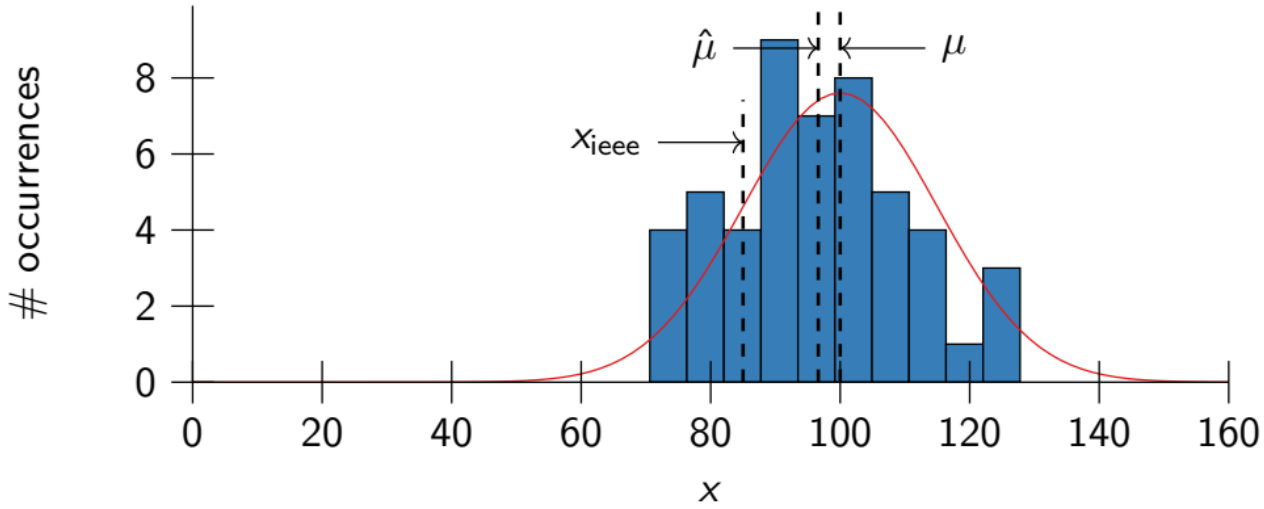
absorption and rounding errors

verificarlo

MCA results analysis

$$s = -\log\left(\frac{\hat{\sigma}}{\hat{\mu}}\right)$$

$\hat{\mu}$ is the empirical mean of the samples
$\hat{\sigma}$ is the empirical standard deviation
log_{10} translate to significant digit
log_2 translate to significant bits



- Intuitively noise over signal ratio
- This formula has 68% probability with 95% confidence for a normal distribution
- Deeper analysis and formula for normal and general distribution can be found in [inter18]

The need for tracing capabilities

- Existing tools explore the spatial dimension of numerical computations
  - which variable, operations or functions are imprecise
- However functions in a programs have different numerical requirements over execution time when the context varies
  - Call site (e.g. dot product called in many places with various size and conditioning)
  - Iteration (e.g. iterative solver)
  - Input data (e.g. polynomial evaluation)

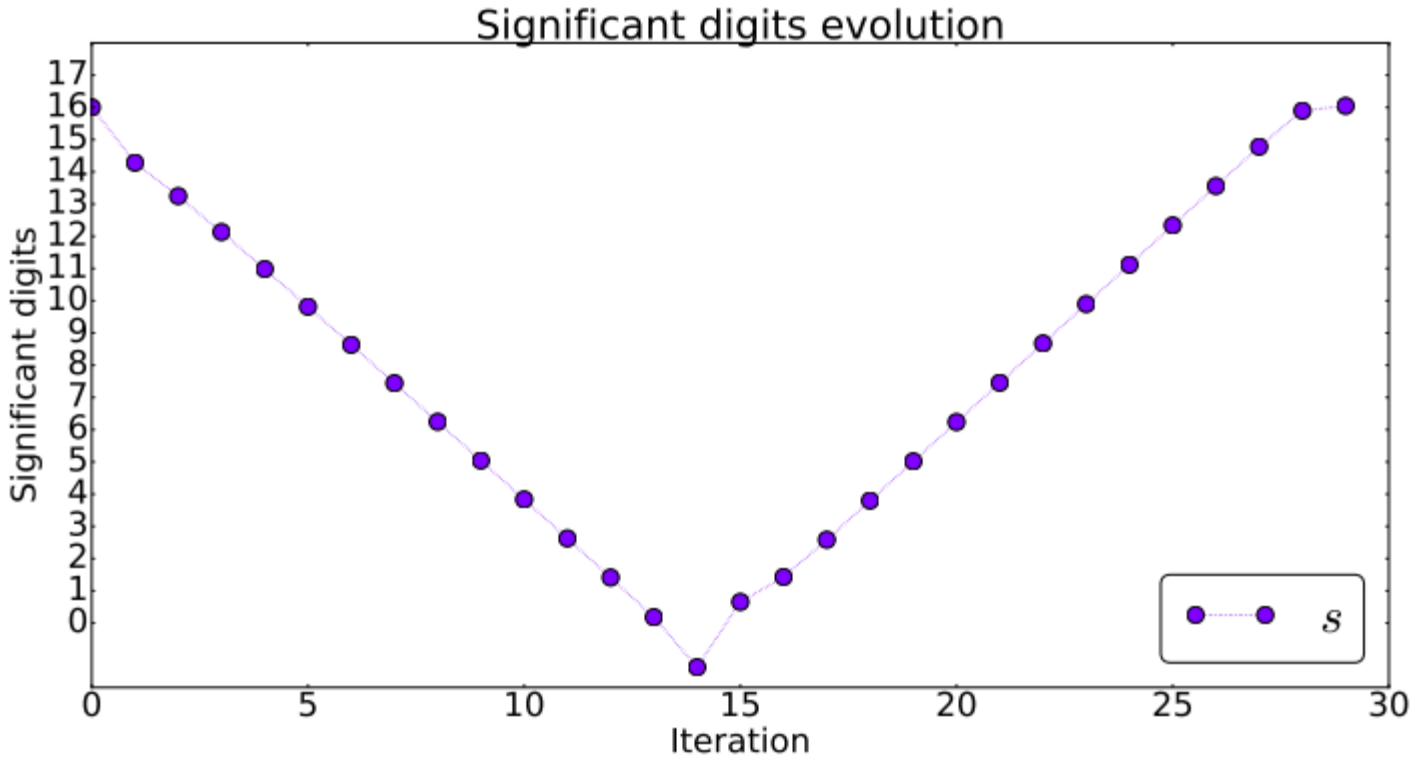
Motivating example

JM Muller's sequence:

$$u_{n+1} = 111 - \frac{1130}{u_n} + \frac{3000}{u_n u_{n-1}}$$
$$with \quad u_0 = 1, u_1 = -4$$

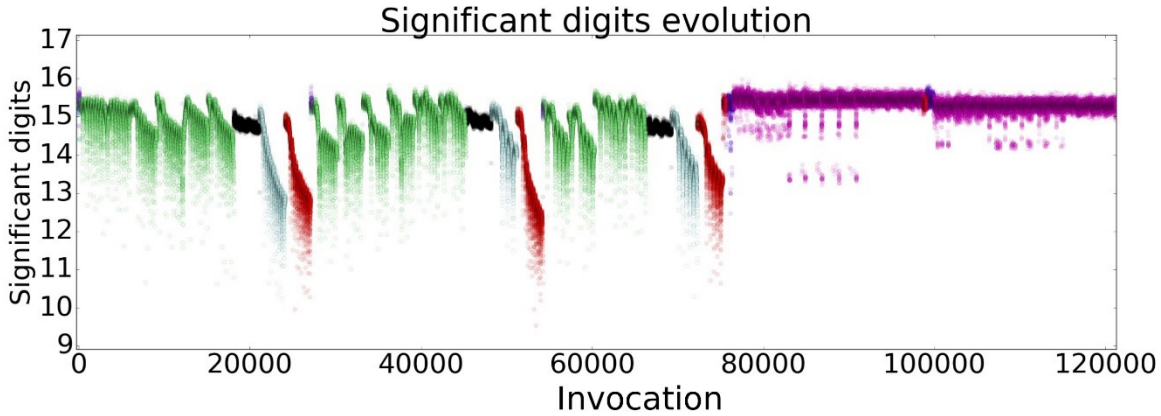
- Converge to 6 (accurately)
- Any finite precision machine will converge to 100 (precisely)
- **This is an example of being precisely wrong and reproducible!**

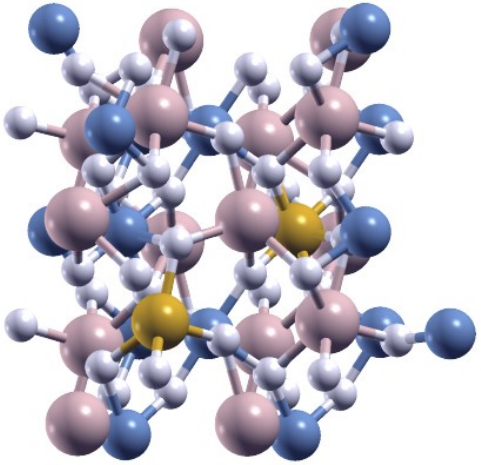
Muller suite analysis with Veritracer



verificarlo

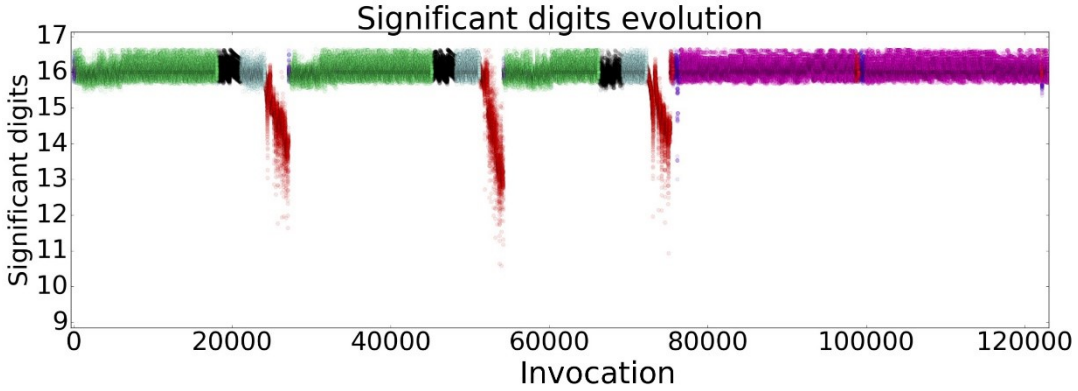
Veritracer on Abinit [Veri17]





Sound velocity calculation in an earth mantle component
($MgSiO_3$ perovskyte with Al impurities)

- ABINIT [7] Calculates observable properties of materials (optical, mechanical, vibrational)
- We isolate and study the numerical stability of a problematic function (3% overhead to trace it)
- It has been fixed for most of the callsite using compensated dot-product [Ogita05]



Common sources of precision loss in numerical simulations

- Large summation
  - Dot product, integral computation, global values reduction (global energy...)
- Gradient computation of near values
  - Small variations in large quantities, gradient with neighbor (e.g. stencil, CFD), residual
- Small contributions overtime
  - Explicit methods, last iterations of a linear solver
- Duplication of mathematically equivalent computation on parallel actors
- Or a combination of the above
  - L2 norm of a residual, standard-deviation...

VPREC for **mixed precision exploration [Vprec19]**

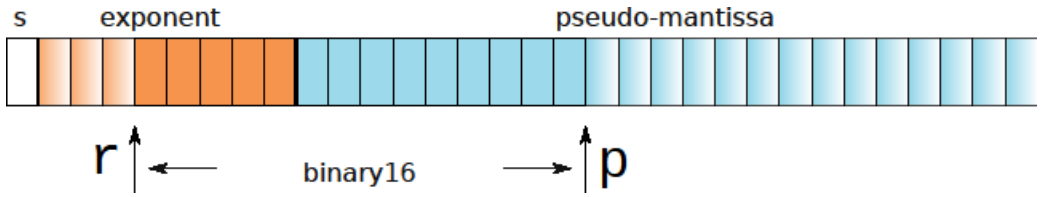
Emulate any range and precision fitting in original type

- Handle denormals, special values
- Implement correct rounding to nearest

Propose a heuristic based algorithm to explore lower precision implementation of an algorithm **over time**

- Complementary to other spatial exploration like delta debug and automatic differentiation

focus on iterative solver

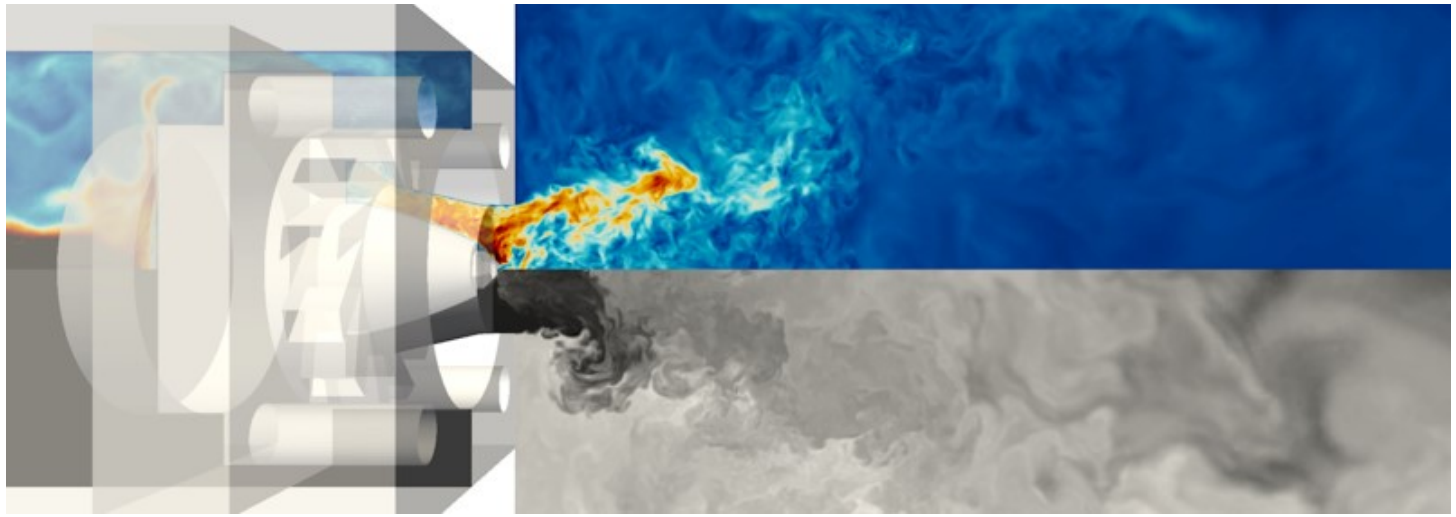


verificarlo

Yales2 application

Coria-CNRS and Safran, Solvay, GDF-Suez
https://www.coria-cfd.fr

- Solver for two-phase combustion from primary atomization to pollutant prediction
- Unstructured meshes up to billions of elements
- Direct Numerical Simulation of laboratory and industrial configurations
- Deflated Preconditioned Conjugate Gradient (DPCG)



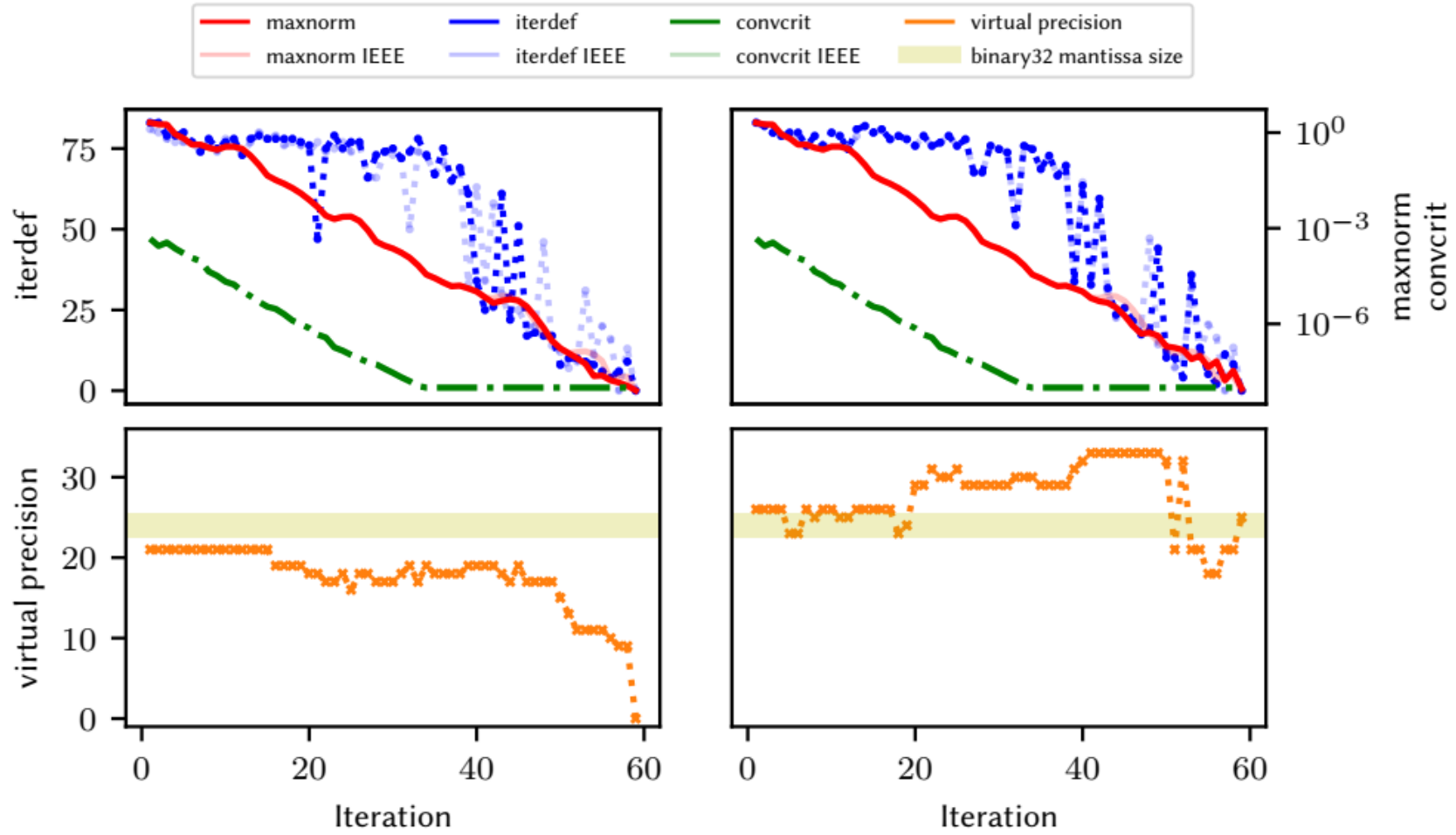


Figure: Adaptive precision searching on YALES2's DPCG with the deflated part (*left*) and the entire code (*right*). On both plots, we can see that our reduced precision solution follows the reference IEEE convergence profile.

5

Conclusion on mixed precision experiment in Yales2

- Exploration mixed single/double configuration
  - Evaluation on 1.75, 40 and 870 Million element mesh
  - from 28 to 560 cores on CRIANN cluster (Atos, Intel OPA, Intel CPUs)
- 28% communication volume gain on average
  - ~Linear with energy gain [Anzt18]
- Performance gain from -2% to 27%...
  - Expected: perf dominated by communication latency
  - With commonly used mesh size and core count, around 10% SU
- Numerical stability of the reduced precision implementation is discussed in [Vprec19]

verificarlo

Future work

- Mixed precision usage
  - Leverage BF16 (and FP32…) various workloads
  - Explore other methodologies to assist mixed precision usage
- Debugging, verification
  - Leverage delta-debugging
  - Mix UQ and stochastic arithmetic
- Interflop: community driven development
  - Mutualize and formalize a common interface to build synergies in floating point analysis tools
  - Build hybrid methodologies and new analysis
  - Prototype Verou <-> Verificarlo fully functional

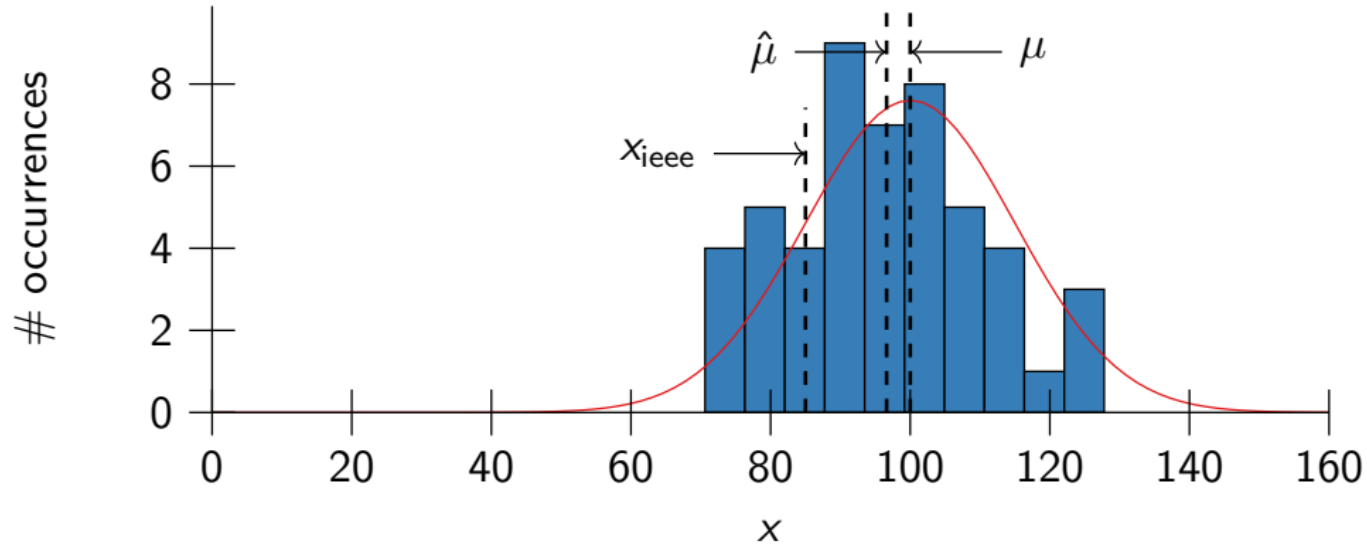
Verificarlo

- [Veri15] **Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic**. Christophe Denis, Pablo de Oliveira Castro, and Eric Petit. In *23nd IEEE Symposium on Computer Arithmetic, ARITH 2016, Silicon Valley, CA, USA, July 10-13, 2016*, pages 55--62, 2016.
- [Veri17] **VeriTracer: Context-enriched tracer for floating-point arithmetic analysis**. Yohan Chatelain, Pablo de Oliveira Castro, Eric Petit, David Defour, Jordan Bieder, and Marc Torrent. In *25th IEEE Symposium on Computer Arithmetic, ARITH 2018, Amherst, MA, USA. June 25th-27th, 2018*, pages 65--72. IEEE, 2018.
- [Inter18] **Confidence Intervals for Stochastic Arithmetic**. Devan Sohier, Pablo de Oliveira Castro, François Févotte, Bruno Lathuilière, Eric Petit, and Olivier Jamond. preprint, July 2018
- [Vprec19] **Automatic exploration of reduced floating-point representations in iterative methods**. Yohan Chatelain, Eric Petit, Pablo de Oliveira Castro, Ghislain Lartigue, and David Defour. In *Euro-Par 2019 Parallel Processing - 25th International Conference*, Lecture Notes in Computer Science. Springer, 2019
- [Parker97] **Monte Carlo Arithmetic: exploiting randomness in floating-point arithmetic** Douglas Stott Parker, *Technical Report CSD-970002*, UCLA Computer Science Dept., 1997
- [Anzt18] **Adaptive Precision in Block-Jacobi Preconditioning for Iterative Sparse Linear System Solvers**, H. Anzt, J. Dongarra, G. Flegar, N. Higham, E. Quintana-Orti, Concurrency and Computation: Practice and Experience, http://dx.doi.org/10.1002/cpe.4460, January, 2018.

Some theory for **verification** using stochastic arithmetic [Inter18]

We propose a unified theory that allows to retrieve all formula and confidence bounds for state of the art stochastic arithmetic method within two hypothesis:

1. Probability for significance and contribution for **Normal Centered Distributions**.
2. Probability for significance and contribution for **General Distributions**.



Probabilistic reformulation of the error

- We consider four kind of scenarios

	reference x	reference Y
absolute precision	$Z = X - x$	$Z = X - Y$
relative precision	$Z = X/x - 1$	$Z = X/Y - 1$

- In each case the error is modeled by a random variable Z.
- For simplicity, in the following we consider the relative precision with scalar reference.
- With no error, the expected result of Z is 0.

Confidence interval for stochastic arithmetic CNH [Inter18]

$$s \geqslant \underbrace{-\log_2(\hat{\sigma}) - \underbrace{\left[\frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right) + \log_2\left(F^{-1}\left(\frac{p+1}{2}\right)\right)\right]}_{\delta_{\mathrm{CNH}}}}_{\hat{s}_{\mathrm{CNH}}}$$

With 95% confidence:

- For $n = 30$ samples and $p = 99\%$ $s \geq -log_2\hat{\sigma} - 1.792$
- For $n = 15$ samples and $p = 99\%$ $s \geq -log_2\hat{\sigma} - 2.023$

($log_2\hat{\sigma}$ is Stott Parker's formula when the reference is $\hat{\mu}$)

Some(many)times CNH for the error distribution is not met!

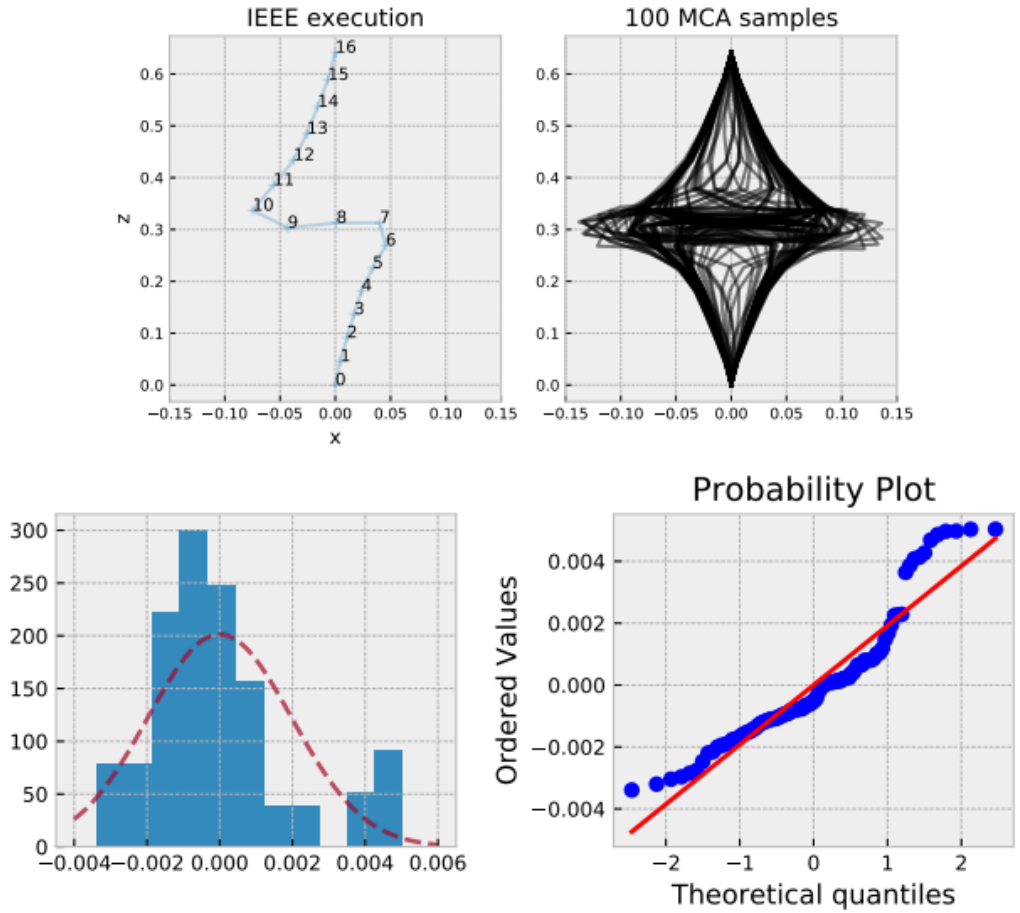
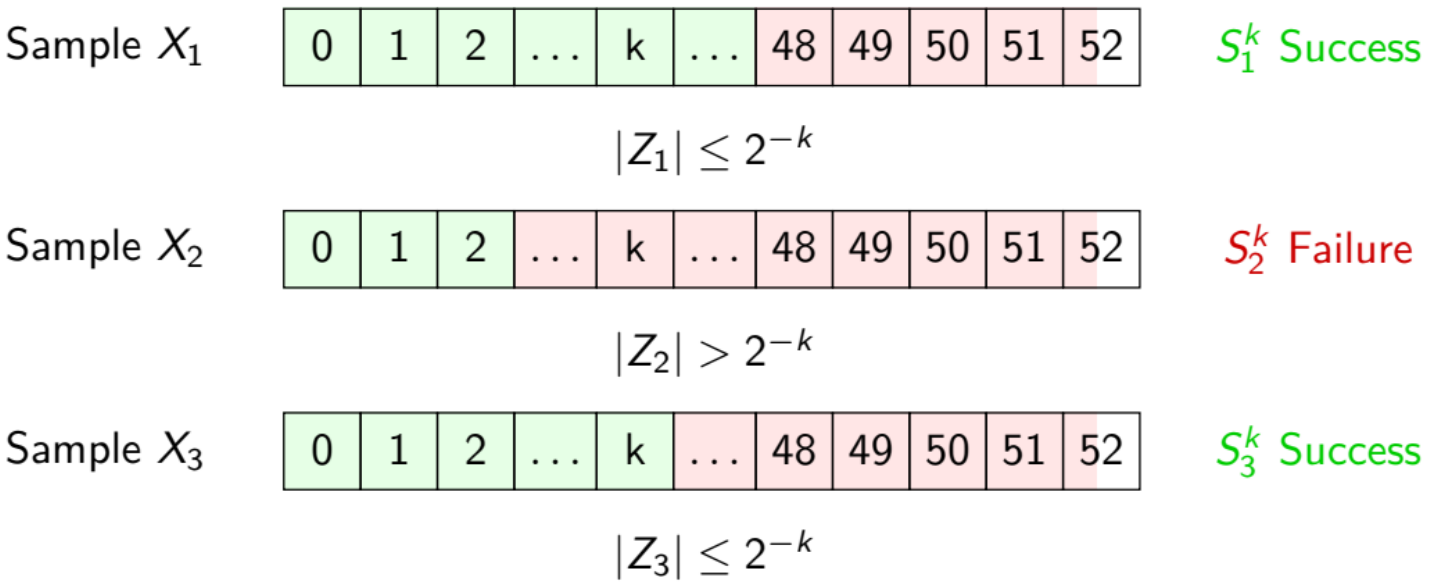


Figure: Non normality of buckling samples on z axis and node 1. Shapiro Wilk rejects the normality hypothesis.

Confidence interval for stochastic arithmetic general distribution with Bernoulli

- Let us choose a single k in the mantissa and single sample i among the n samples.
- We can define a binary test,
  - $S_i^k = |Z_i| \leq 2^{-k}$, true iff for the i-th sample the k-th first bits are significant.
- With n samples we have n Bernoulli Trials.
- The trials are realizations of Bernoulli random variables S^k

[TODO] Bernoulli lower bound formula here



Confidence intervals for stochastic arithmetic