Verificarlo, Verrou, Interflop: Floating point computing verification on new architecture and large scale systems

Éric Petit[1],
Pablo Oliveira[2], Yohan Chatelain [2], Devan Sohier [2],
François Févotte[3], Bruno Lathuilière[3],
David Defour [4]
et al.

[1]Intel DCG E&G [2]UVSQ Li-Parad; [3]EDF R&D Pericles; and [4] UPVD

MDS workshop June 28th, 2019

Table of contents

Section 1

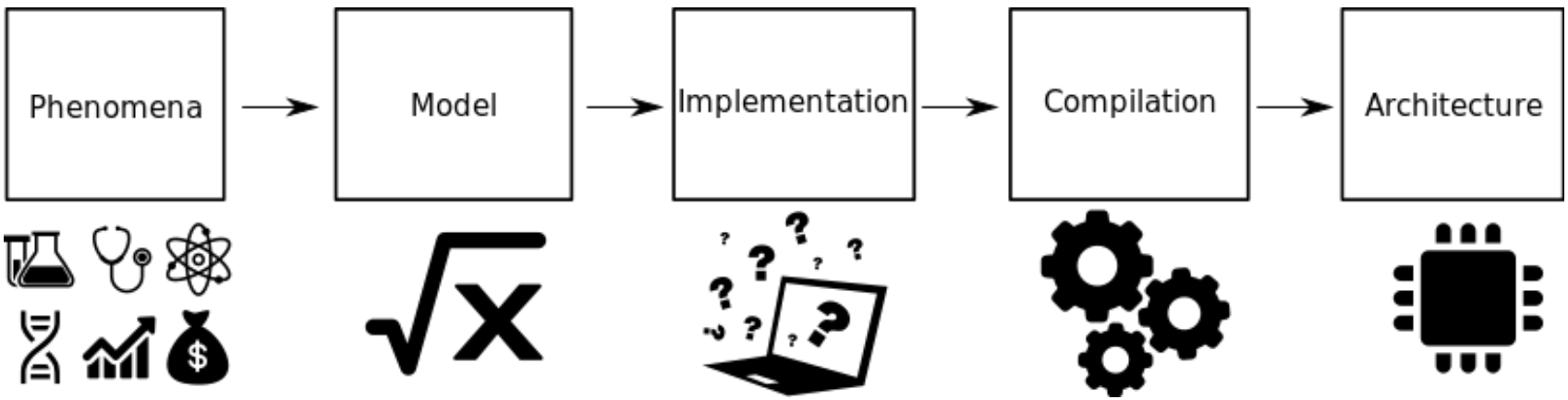
Asynchronous Stochastic Arithmetic

The computer scientist perspective

- 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 event, system...)

The application developer perspective

Building fast and robust applications for HPC is a complex task !



- At each step, numerical bugs can be introduced

Objective 1: Track and analyze numerical bugs
Objective 2: Optimize FP usage

Issue with implementing FP codes on a moving base

Performance improvement allows larger, more complex, higher resolution simulations. Changing architecture, parallelization, heterogeneity, compiler, optimizations level and language

generates different numerical results.

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

Verifying correctness?

Does different results means wrong results?

Ensuring the numerical reproducibility is not always a requirement!

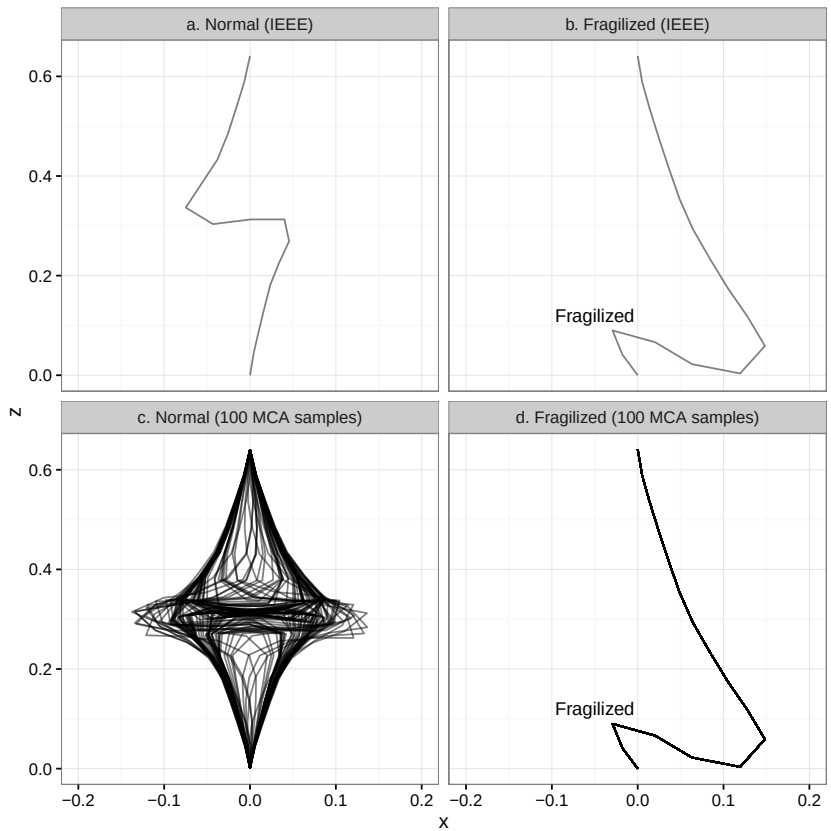


Figure: Buckling simulation of a 1D beam with EPX

Code verification for FP errors?

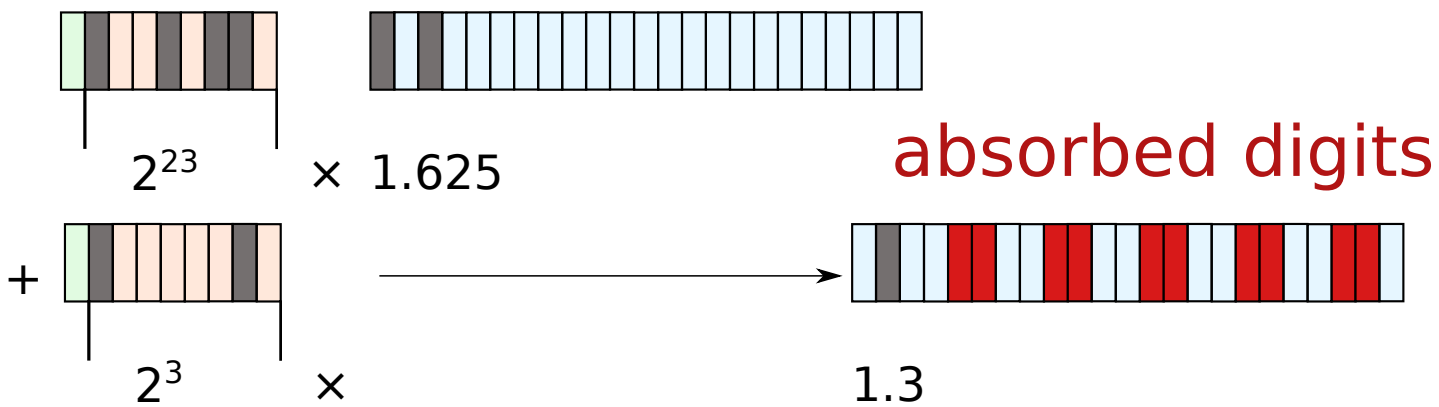
- Is statistical FP implementation debugging, optimization and verification enough for you?
  - Unitary test coverage?
  - On my use case, I am 95% confident that with a probability p:
    - (The model predict that) My nuclear reactor will not melt.
    - My search engine is giving accurate results.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

Floating point computation: the IEEE-754 standard

- What Every Computer Scientist Should Know About Floating-Point Arithmetic, *David Goldberg*, 1991 ACM issue of Computing Surveys
- Floating point (FP) numbers approximate real numbers with a finite precision
  - Discrete and finite set of values
  - In base 2
- Different representation and encoding in memory defined in IEEE 754
- Trade-off between range and precision
  - Single ($1 + 8 + 23 = 32$ bits), Double ($1 + 11 + 52 = 64$ bits)...
- And four rounding modes :
  - nearest, toward $+\infty$, toward $-\infty$, toward zero

Floating point computation: some adverse effects

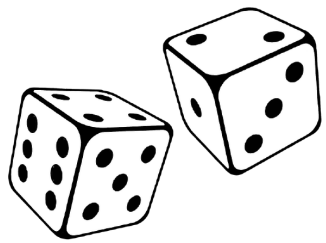
- A floating point computation approximates the real computation
  - Representation errors
    - 3.14159265359300
  - Loss of arithmetical properties (for example the floating point summation is not associative)
  - Absorption, a part of the significant digits cannot be represented in the result format.
    - 3.14159+0.00141421=3.14300{421}
  - Cancellation, relative error when subtracting variables with very close values
    - 3.14300-3.14159=0.00141



The most common sources of FP arithmetic bugs, indeterminism or imprecision in HPC

- 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...

Modeling error with stochastic Arithmetic



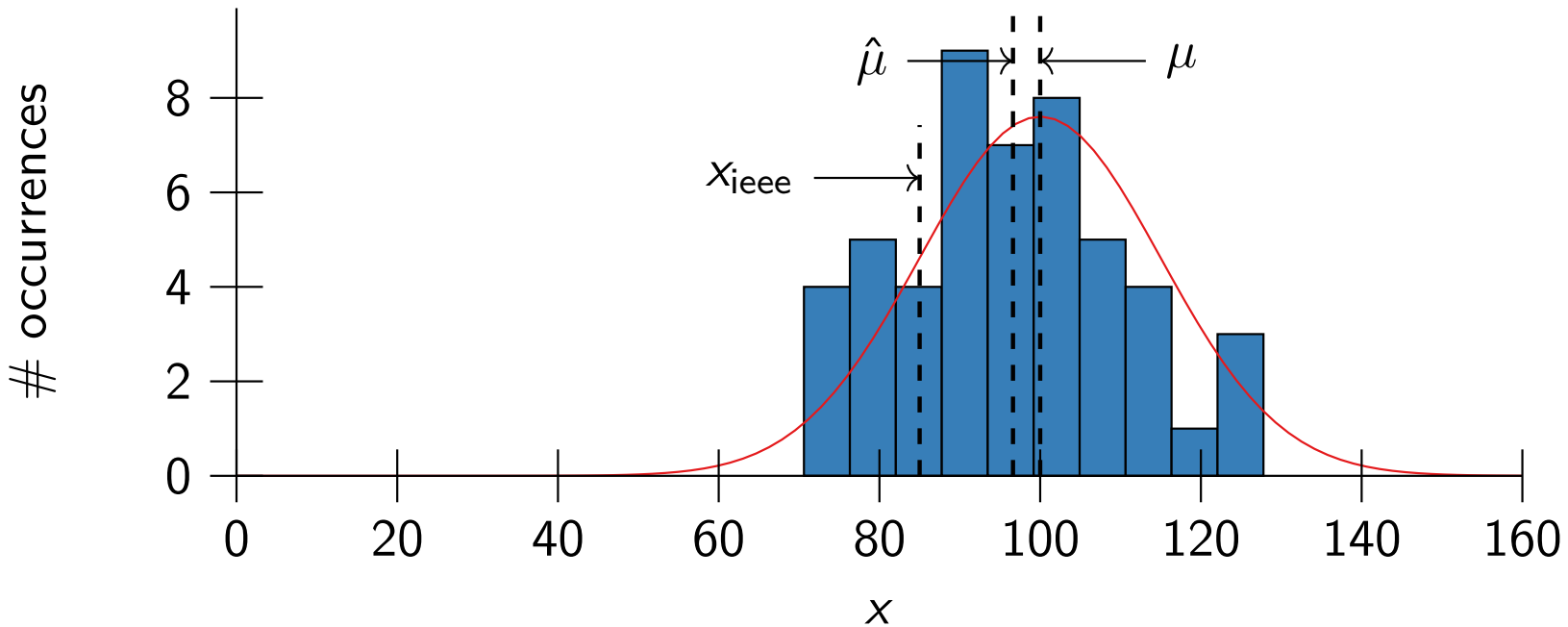
- Each FP operation may introduce a δ error

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

- When chaining multiple operations, errors can accumulate and snowball
- Monte Carlo Arithmetic key principle
  - Make δ a random variable
  - Use a Monte Carlo simulation to empirically estimate the FP error distribution [Stott Parker, 1999]

Some notations

- x_{IEEE} is the IEEE-754 result
- $X_1, X_2, \ldots, X_n$ are the values returned by n runs of the program using stochastic arithmetic. These are seen as realizations of a random variable X.
- $\hat{\mu}$ and $\hat{\sigma}$ are the empirical average and standard deviation.
- μ and σ are the mean and std. deviation of the random variable X.



How to measure the significance of a result ? (2/2)

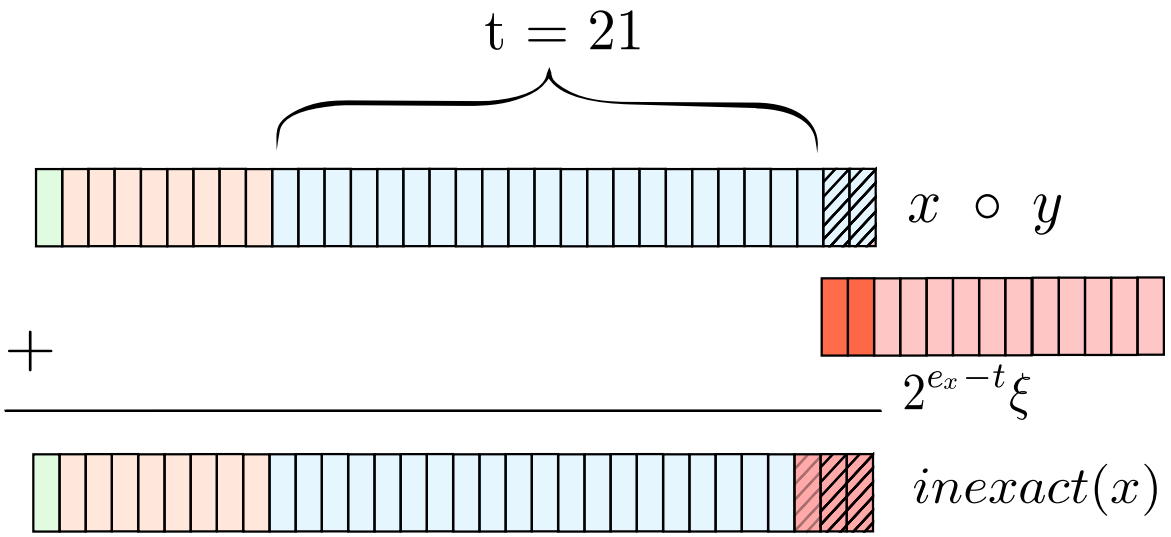
- Stott Parker defines the number of significant digits as $\hat{s} = -\log \frac{\hat{\sigma}}{|\hat{\mu}|}$
  - $\hat{\sigma}$ is the empirical standard deviation of X
  - $\hat{\mu}$ is the empirical average of X
- This term is the magnitude of the signal to noise ratio.
- Intuitively it maps to the number of common digits among the MCA samples.
- Later we will introduce a rigorous probabilistic definition of the number of significant digits. In this probabilistic framework, the digit computed by Stott Parker formula has 68% chances of being significant with confidence 95% for a centered normal distribution.

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.



FP operations $\circ$ are replaced by:

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

absorption and rounding errors

Absorption example

```
float a = 13631488.0f; // 1.625 * 2^23
float b = 10.4f; // 1.3 * 2^3
printf("%0.7f\n", a+b);
```

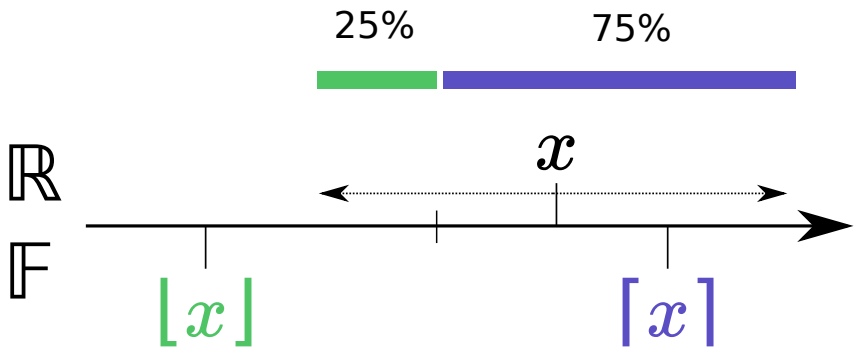
- Exact result = 13631498.4
- IEEE-754 result = 13631498 (always)
- MCA Random Rounding noise with $t = 24$

Sample	Result
1	13631498
2	13631499
3	13631498

- Exact digits are common across MCA samples
- Error digits change across MCA samples

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 $]-\frac{\epsilon}{2}, \frac{\epsilon}{2}[$.
- MCA result is either the downwards or upwards roundoff, with a probability proportional to $\frac{\lceil x \rceil - x}{\epsilon}$.
- For exact values no error is introduced in this mode.



- **This mode is equivalent to Verrou's *average* random rounding.**

Summation example: t=53 (random rounding at the ulp)

- 0.1 is not representable in $\mathbb{F}$. The closest value is 0.10000000000000000555...

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

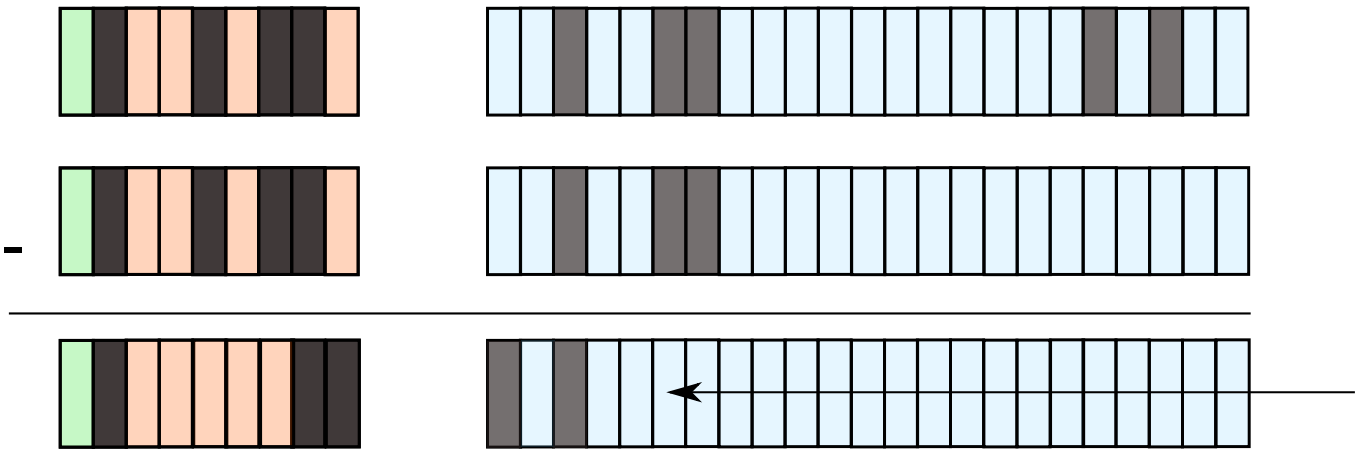
Sample	MCA RR t=53
1	1000.0000000001186891
2	1000.0000000001174385
3	1000.0000000001175522

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

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

Catastrophic Cancellation

- A cancellation happens when we subtract two close values:
  - $9633812.0 - 9633792.0 = 20.000000$



- Random Rounding models this operation as exact: the right-digits are always the same.

Sample	MCA RR t=53
1	20.0000000
2	20.0000000
3	20.0000000

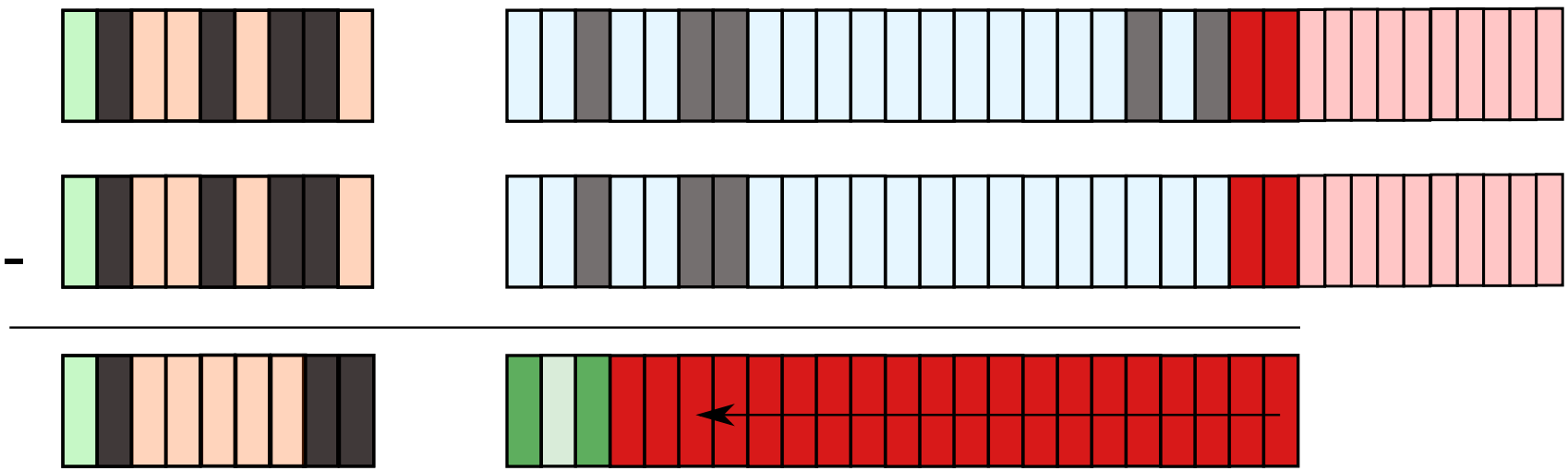
- How to model the fact that the right-digits came out of thin-air and information may have been lost ?

Monte Carlo Arithmetic: Precision Bound Mode

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

cancellation

mca noise



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

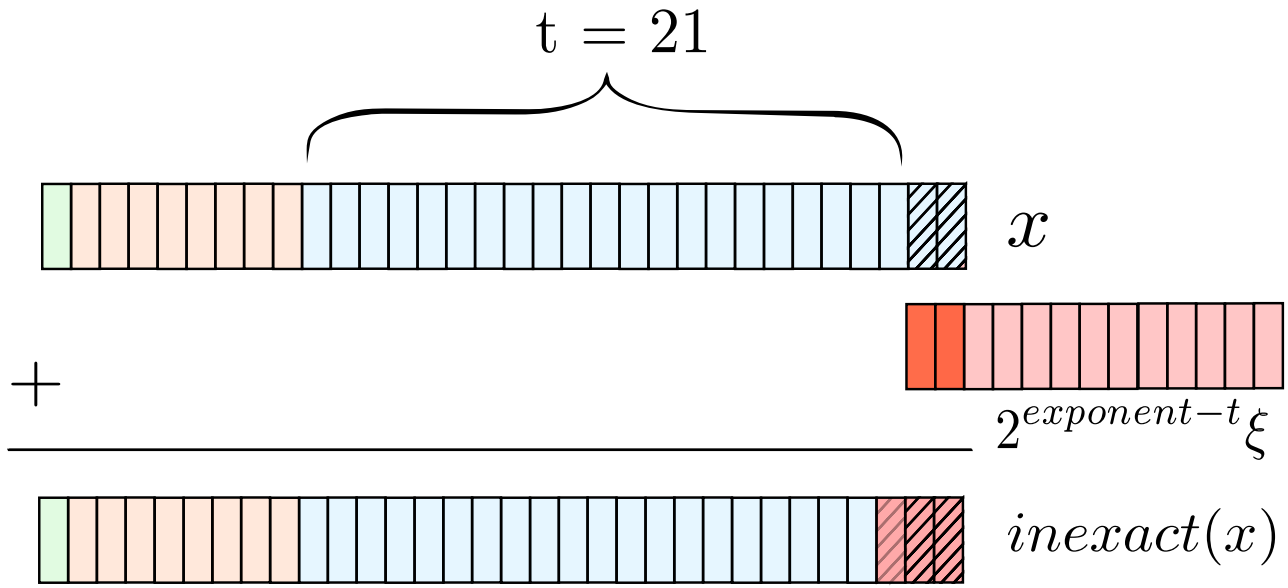
Precision Bound Mode Example

```
float a = 9.633812E6 ;
float b = 9.633792E6  ;
printf("%0.7f\n", a - b);
```

Sample	MCA PB t=24	MCA RR t=24
1	19.9846210	20.0000000
2	20.0592098	20.0000000
3	19.8788948	20.0000000

Monte Carlo Arithmetic: Full MCA Mode

- Full MCA Mode combines both Random Rounding and Precision Bound.



FP operations $\circ$ are replaced by:

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

absorption

cancellation

Example: Kahan 2x2 ill conditioned System

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

$$\begin{pmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{pmatrix} x = \begin{pmatrix} 0.1440 \\ 0.8642 \end{pmatrix} \tag{1}$$

$$x_{\text{real}} = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \tag{2}$$

$$x_{single} = \begin{pmatrix} 1.33317912 \\ -1.00000000 \end{pmatrix} x_{double} = \begin{pmatrix} 2.0000000240030218 \\ -2.0000000359962060 \end{pmatrix}$$

- The IEEE-754 result has 8 significant decimal digits.
- $x_{\text{IEEE}}[0]$ has 28.8 significant bits.

Example: Kahan 2x2 ill conditioned System

- Computations using IEEE-754 FP numbers (C=2.497e8)

Precision	**Result**	s
SP	$x_1 = 1.33317912$	0
	$x_2 = -1.00000000$	0
DP	$x_1 = 2.0000000240030218$	9
	$x_2 = 2.0000000359962060$	9

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

Precision	$\hat{\mu}$	$\hat{\sigma}$	$\hat{s}$
MCA SP	$\hat{\mu}_1 = 1.02463705$	$\hat{\sigma}_1 = 6.4...$	0.0
	$\hat{\mu}_2 = 6.46717332$	$\hat{\sigma}_2 = 9.6...$	0.0
MCA DP	$\hat{\mu}_1 = 1.9999999992$	$\hat{\sigma}_1 = 8.4... \times 10^{-9}$	8.3
	$\hat{\mu}_2 = -1.9999999988$	$\hat{\sigma}_2 = 1.2... \times 10^{-8}$	8.2

- For this example, Verificarlo automatically instrumented LAPACK and BLAS libraries without any modification of their source code
- But how confident are we that it is a good estimate? Could we have used a smaller number of samples and still get a reliable estimation of the results quality?

Existing tools

	Method	Name	Implementation
Debugging	StochasticArithmetic	CADNA [9]	CESTAC/DSA (library)
		VERROU [6]	CESTAC (Valgrind)
		Verificarlo [4]	MCA (LLVM)
	ExtendedPrecision	HPC Craft [10]	Exponent comparison (DynInst)
		FpDebug [2]	MPFR (Valgrind)
		Herbgrind [14]	MPFR (Valgrind)

	Name	Method	Mixed-Precision	Any-Precision
Optimization	Verificarlo [3, 4]	Vprec:MCA (Heuristic, temporal)	✓	✓
	HPC Craft [10]	Bitmask (ref value)	✓	✓
	Promise [8]	CESTAC/DSA ($\Delta - debug$)	✓	✗
	Precimonious [13]	EP ($\Delta - debug$)	✓	✗
	Herbie [12]	EP (Rewriting)	✗	✗

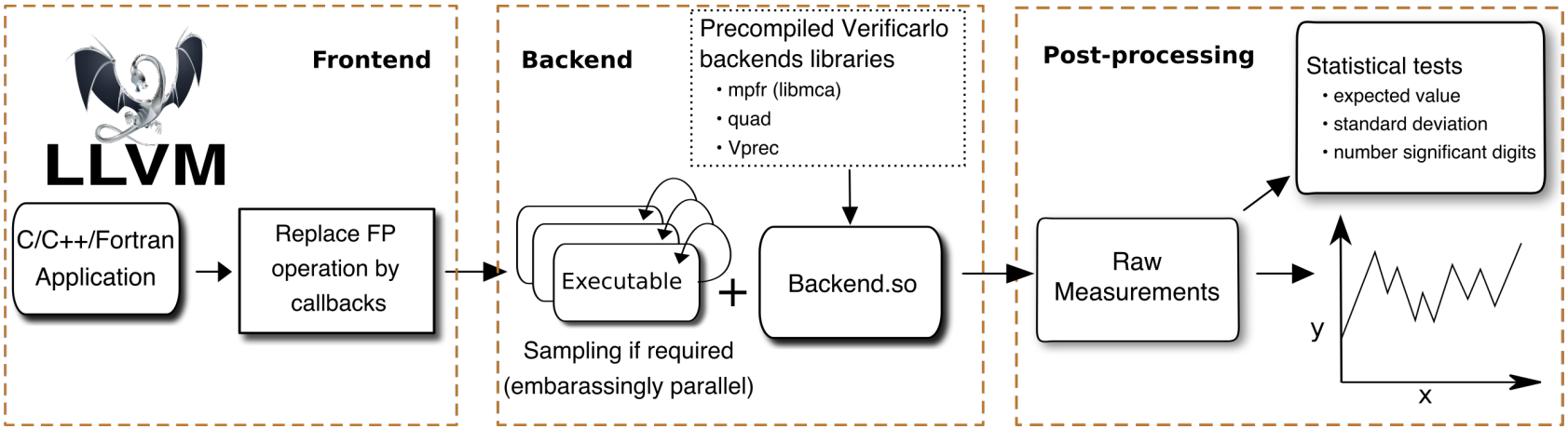
Section 2

Verificarlo: Debugging and optimizing floating point usage in numerical simulation

Introducing Verificarlo

- LLVM compiler pass to replace all floating point operation by call-back to custom arithmetic backend
  - MonteCarlo Arithmetic that allows statistical analysis of rounding errors and tracer extension to follow FP characteristic over time with context info (which variable, callsite, iteration. . .)
  - Variable precision backend and variable precision runtime
  - Open Source GPLv3 at `github.com/verificarlo/verificarlo`

Verificarlo workflow



Verificarlo: Illustration of key principles

- Instrumentation occurs **just before code generation**
- Enables analyzing what is really executed after optimizations

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

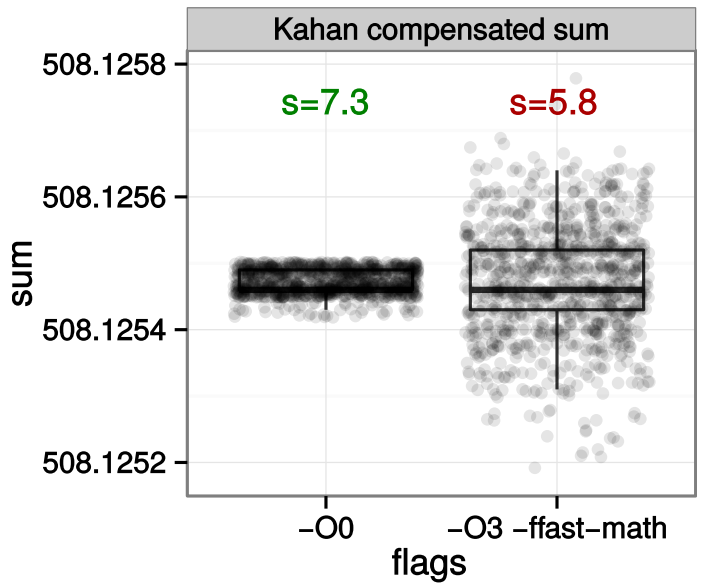


Figure: Analysis of the effect of compiler flags on a Kahan compensated sum algorithm (Random Rounding with p=53)

Kahan Sum

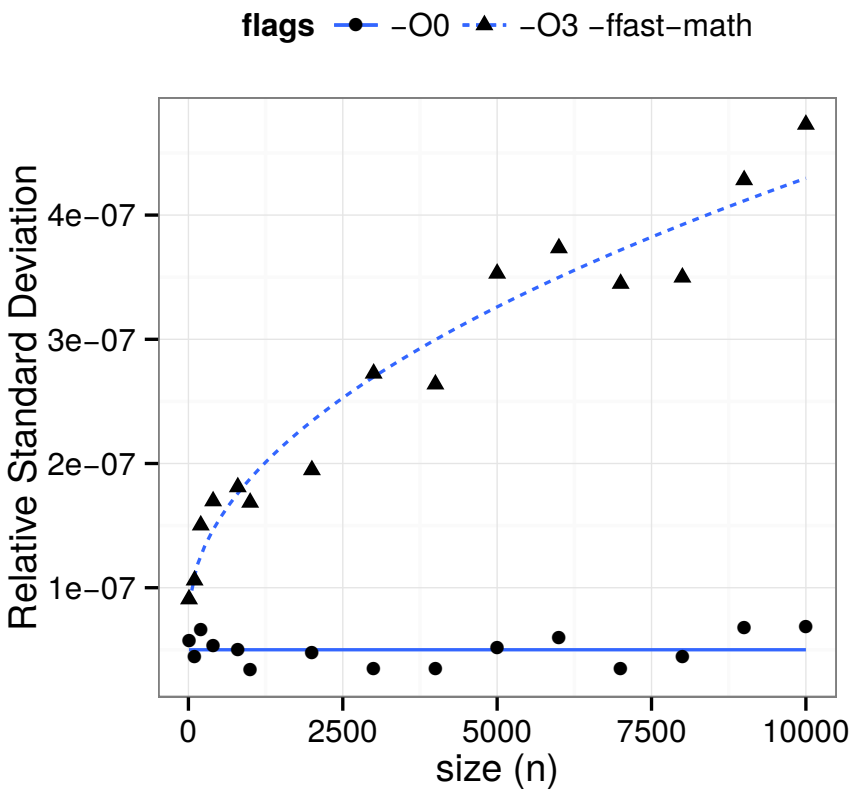


Figure: Relative Standard Deviation $\frac{\sigma}{\mu}$ of Kahan's compensated sum computed on 1000 verificarlo samples. The compensated `-O0` version has a constant error while the `-O3 -fast-math` error increases as $O(\epsilon\sqrt{n})$.

Veritracer motivation

Existing tools explore the spatial dimension of numerical computations:

- which variable, operation or function is imprecise

But function in a programs have different numerical requirements over execution time when the context vary

- 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)

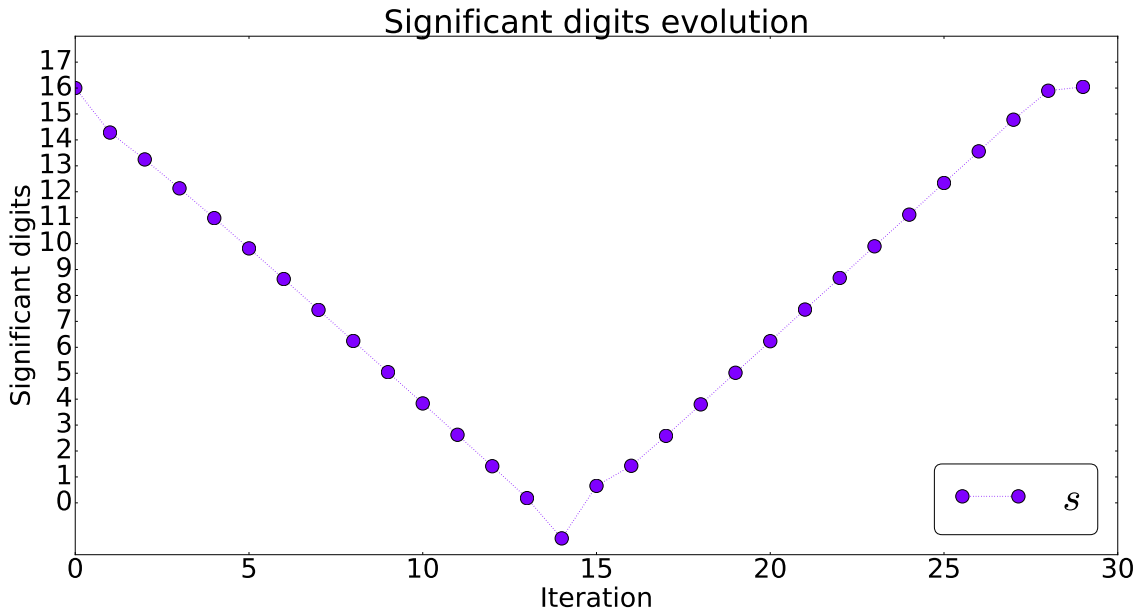
Veritracer: Illustration of key principles

Computing the numerical limit of the following Muller's sequence:

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

- The accurate result is 6
- Whatever the finite precision, a computer will answer 100
  $\Rightarrow$ That is an example of being 'precisely wrong' !

Muller's sequence



- At $n = 30$, $s = 16$
  $\Rightarrow$ Fully precise !
- for $n = 14$, $s < 0 \Rightarrow u_{14}$ has no correct decimal digits.
- Most of FP analysis tools will conclude on a fully precise result.
- Veritracer allow to trace the precision and highlight the problem

Results on ABINIT

- ABINIT [7] Calculates observable properties of materials (optical, mechanical, vibrational)
- Works on any chemical composition (molecules, nanostructures, solids)

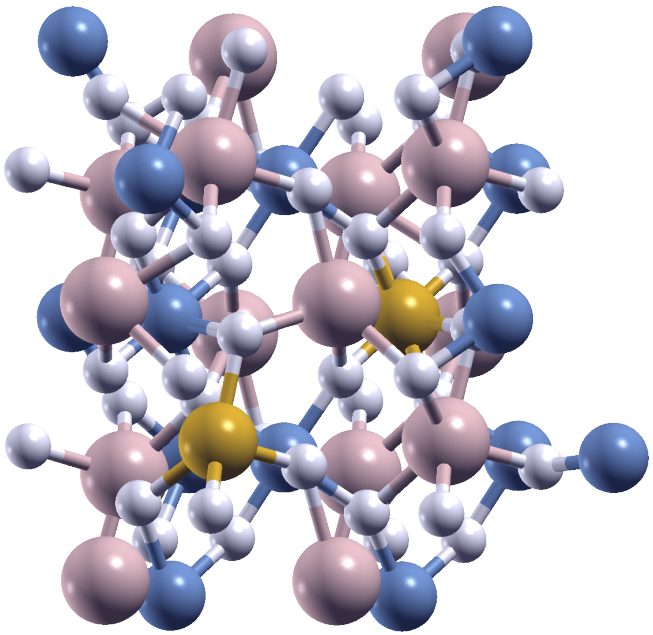
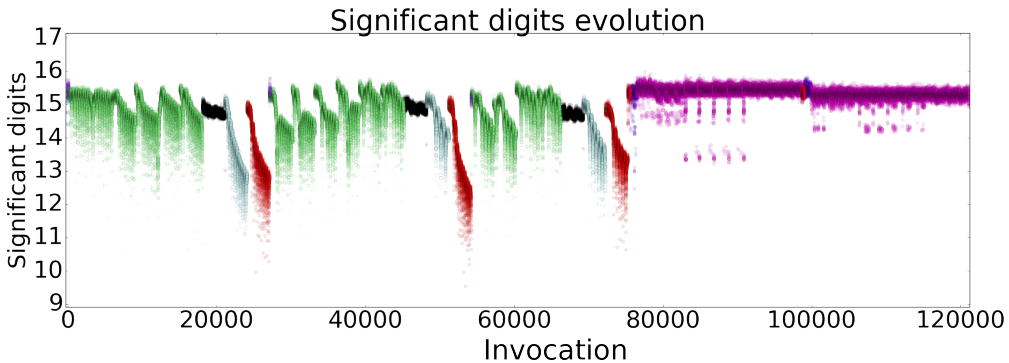
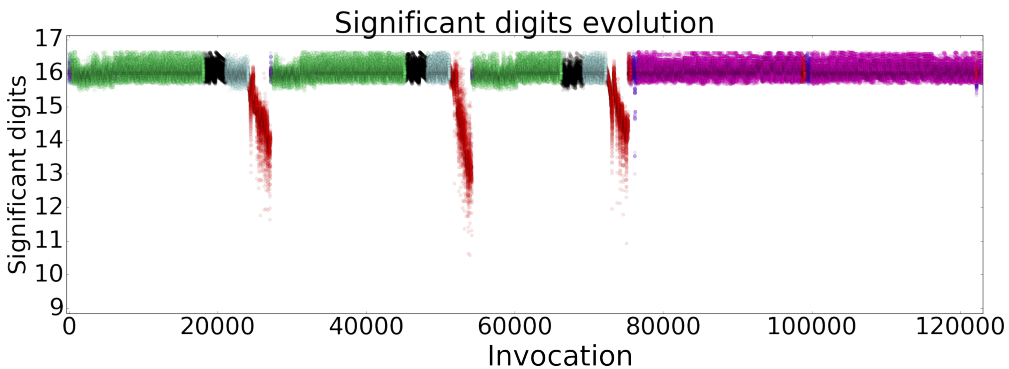


Figure: Sound velocity calculation in an earth mantle component ($MgSiO_3$ perovskyte with Al impurities) [1]

Compensated version of Simp_gen

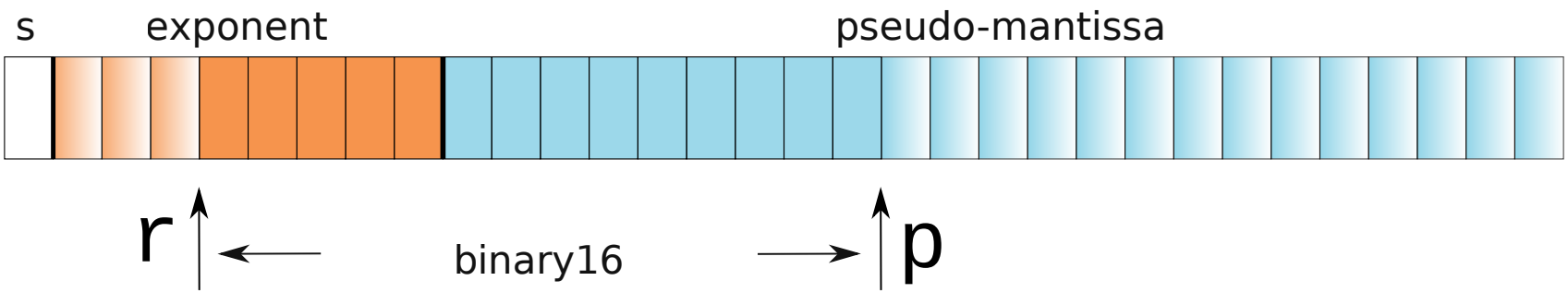




- Computes an integral by Simpsons' rule over a generalized 1D-grid
- Can be seen a dot product
- Replaces by a compensated version Dot2 [11]
- The precision of 30/31 CSPs is improved
- 1 CSP has a low precision due to reentrance of the error

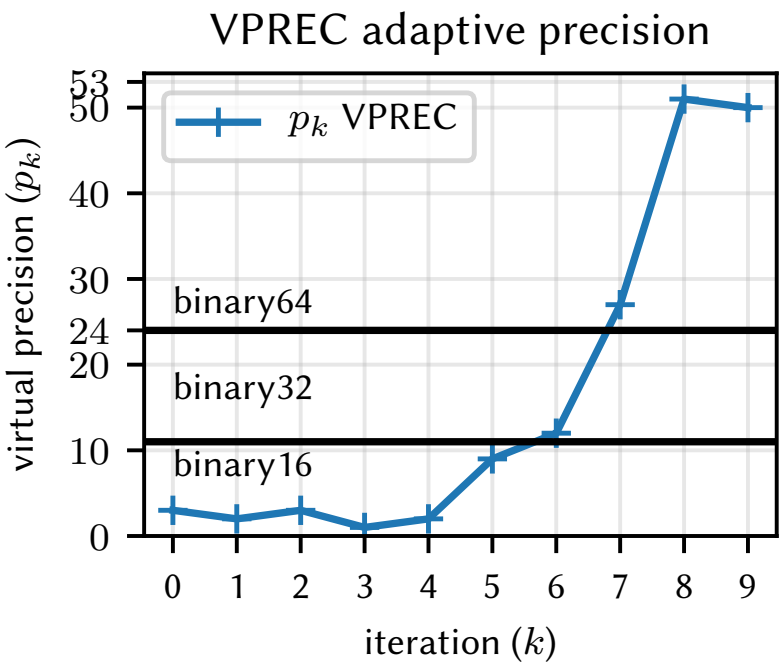
Vprec backend

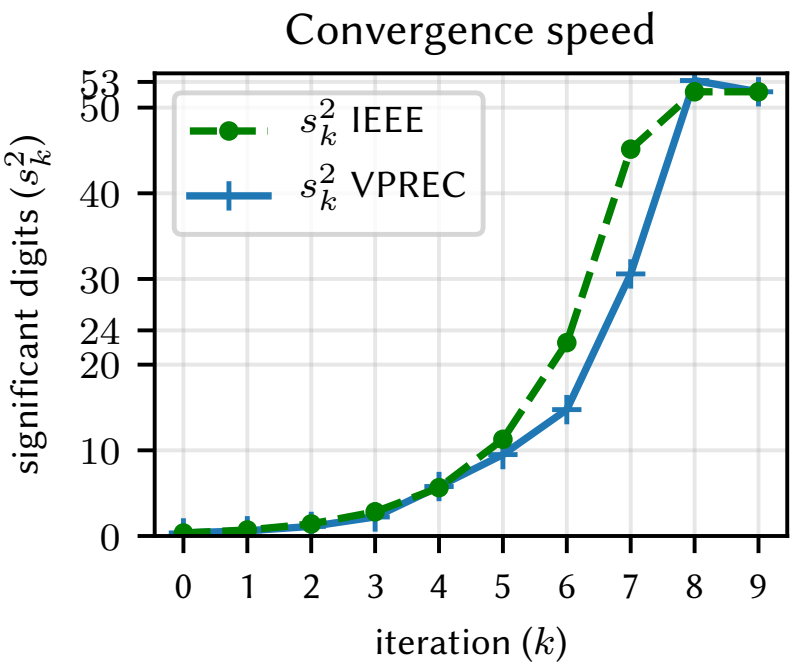
- Emulate any range and precision fitting in original type
  - Handle denormals, special values
  - Implement correct rounding to nearest
- Propose a heuristic based algorithm to explore lower precision implementation of an algorithm over time
  - Complementary to other spatial exploration like delta debug (verrou, precimonious) and automatic differentiation (Adapt)
  - Putting all together in the interflop initiative



Vprec: Illustration of key principles

- Newton Raphson iteration to compute *sqrt()*
- Self-correcting iterations
- Quadratic convergence expected (min)
- Precision required for accurate computation can be gradually increased
- However the convergence profile vary with the input or alternative algorithm (e.g. Goldsmith use p=53 on all iterations)



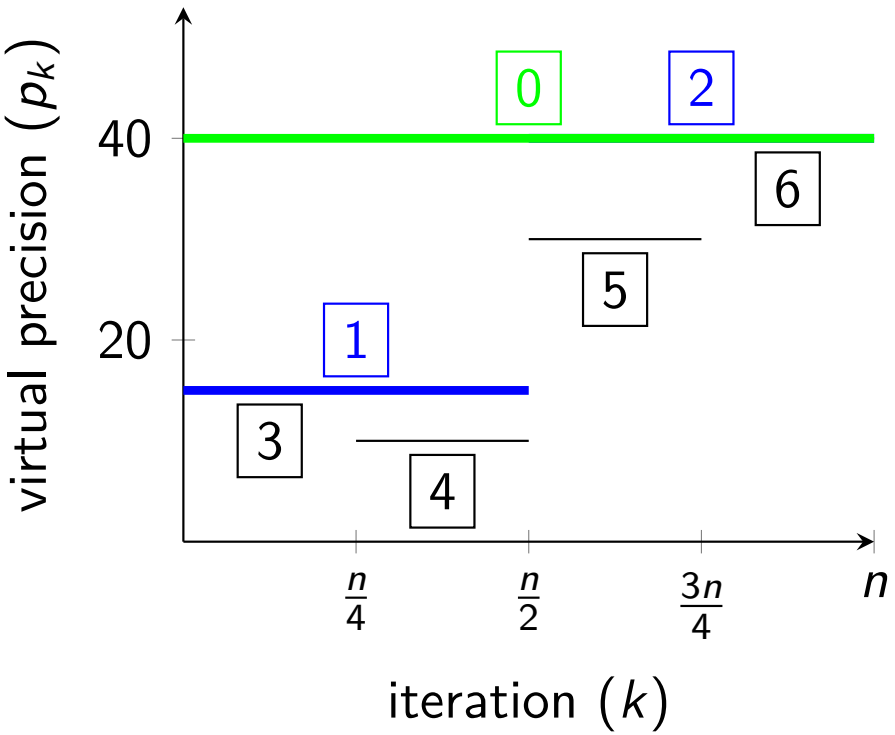


Heuristic exploration of optimization space

Binary tree search heuristic folowing three principles:

1. Configurations where the precision changes slowly over time are preferable to configurations which quickly oscillate
2. Precision lowering should be distributed among all iterations
3. Early iterations are generally more robust to error

↪ Width or depth first tree traversal



Results on Yales2 (1/4)

- Yales2is a solver for two-phase combustion from primary atomization to pollutant prediction
- CFD is using DPCG solver on unstructured meshes up to billions of elements
- Allows Direct Numerical Simulation of laboratory and industrial configurations
- Coria-CNRS and Safran, Solvay, GDF-Suez, `https://www.coria-cfd.fr`

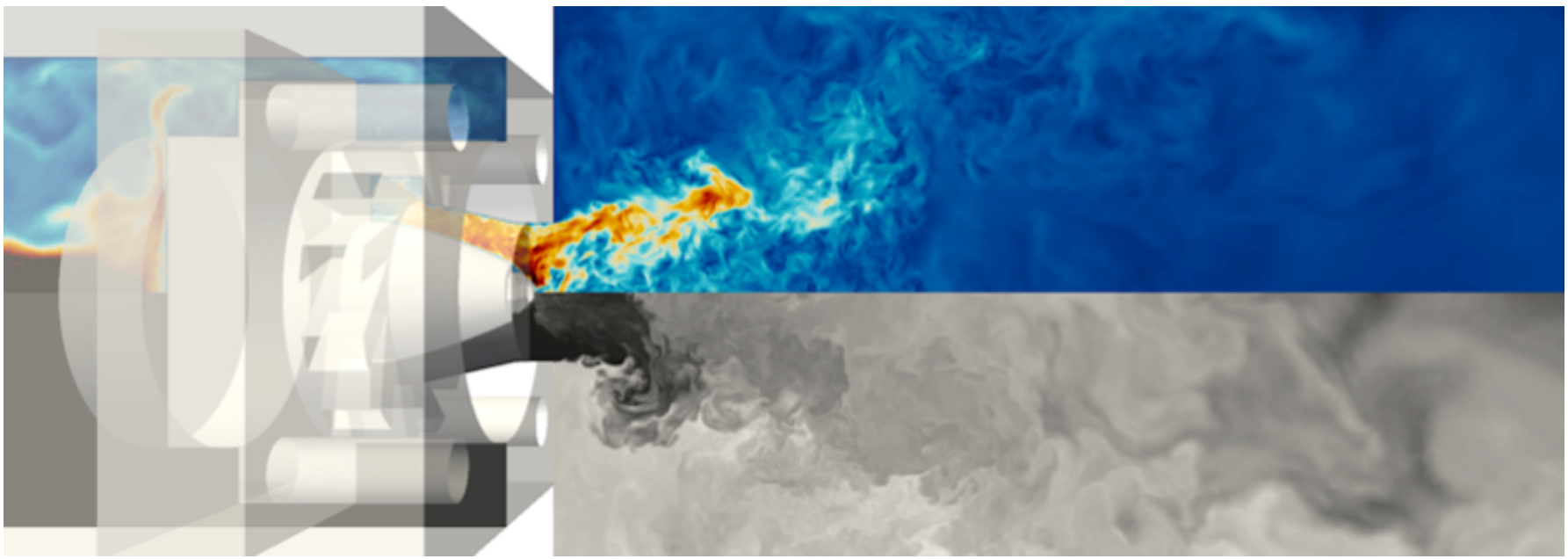


Figure: Preccinsta burner

Results on Yales2 (2/4)

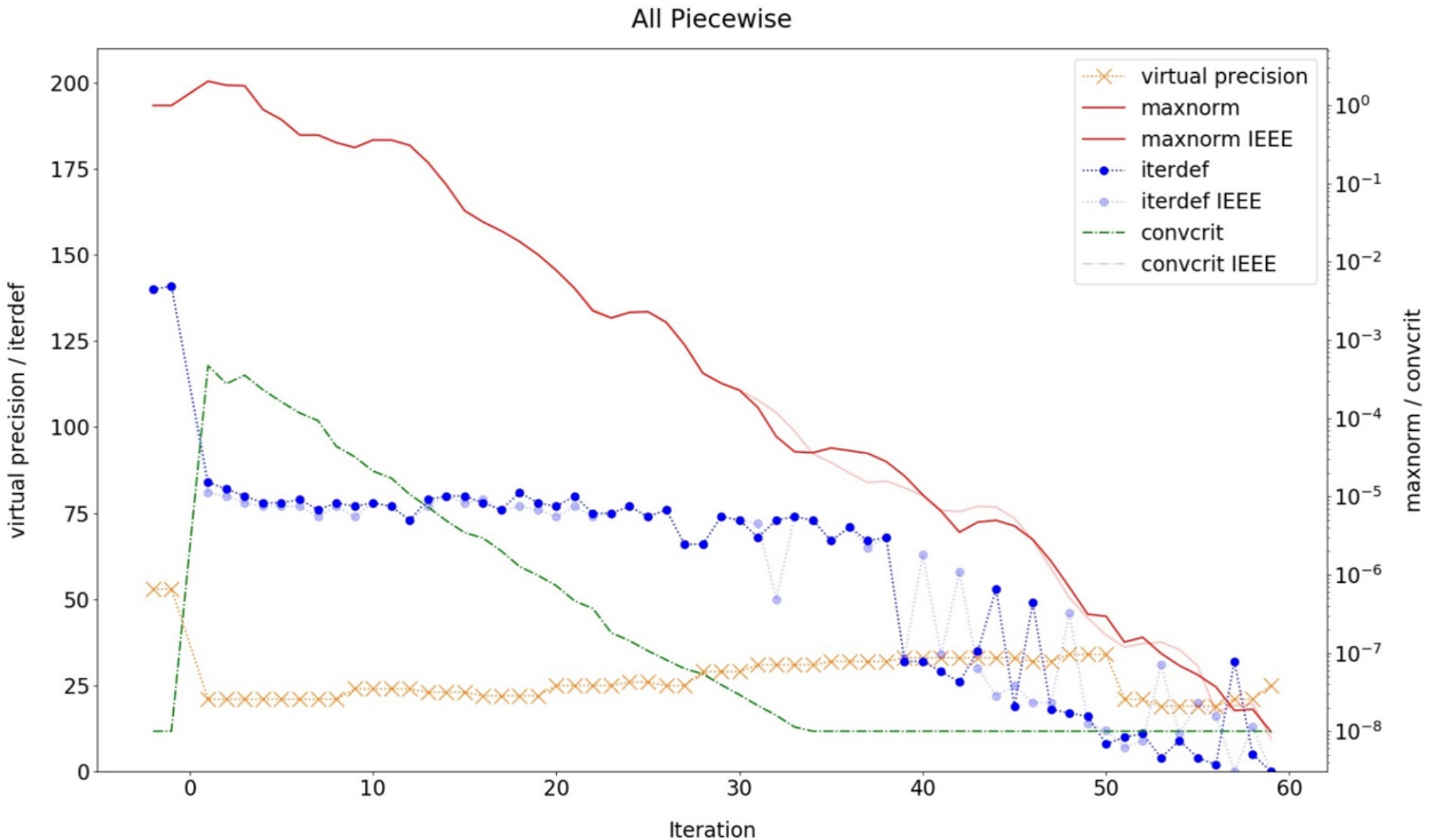


Figure: Reducing precision on the full DPCG solver

Results on Yales2 (3/4)

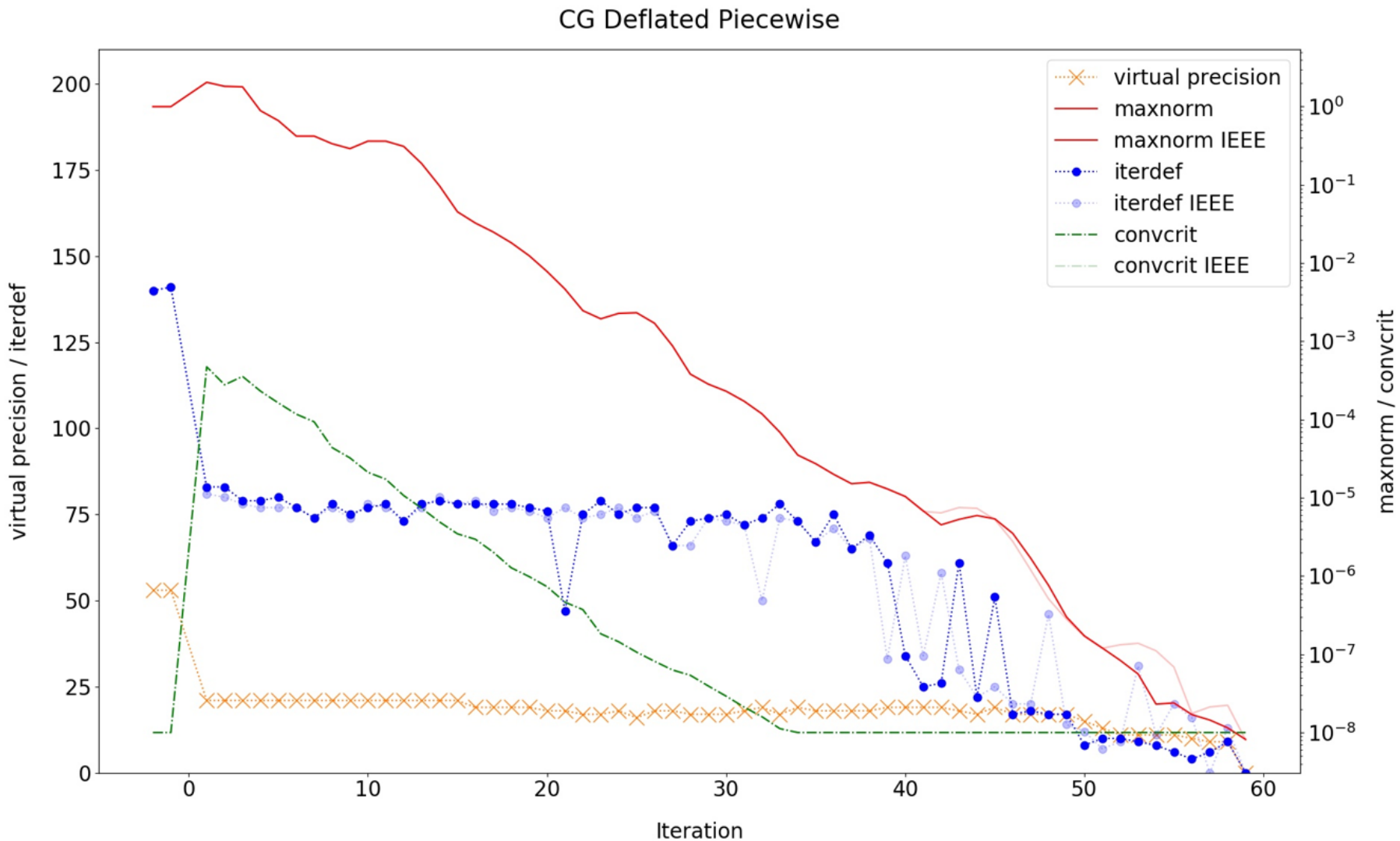


Figure: Reducing precision on the Deflated part of the DPCG solver

Results on Yales2 (4/4)

- 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 to 67% communication volume gain
  - (nearly) Linear with energy gain
- Perf gain from -2% to 27%...
  - Expected: perf dominated by comm latency
  - With commonly used mesh size and core count, around 10% SpeedUp

Future work with Interflop

- Consortium: UVSQ, Intel, EDF, Lip6, CEA, ANEO, UPVD
- Objectives is to mutualize and formalize a common interface to build synergies in floating point analysis tools and provides:
  - new frontend/backend,
  - new hybrid formal/stochastic approaches,
  - common background theory,
  - new methodologies to apply the tools on REAL USE-CASES
- Prototype Verrou $\Longleftrightarrow$ Verificarlo fully functional, yet to be finalize and released.

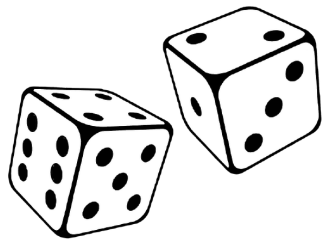
Section 3

Verrou : deboguage numerique des codes de calcul industriels

Section 4

Confidence interval for stochastic arithmetic

Statistical Analysis of Stochastic Arithmetic: Motivation



- Stochastic Arithmetic
  - Numerical errors modeled by introducing random perturbations.
  - Estimate significance of result by collecting many samples.
- Motivation for statistical analysis
  - How many stochastic samples should be run?
  - What is the probability of over-estimating the number of significant digits?
  - Can we give a sound confidence interval for the number of significant digits?

Contributions

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

Preprint: *Confidence Intervals for Stochastic Arithmetic*, D. Sohier, P. de Oliveira Castro, F. Févotte, B. Lathuilière, E. Petit, O. Jamond. 2018.

Choosing a reference value

- We require a reference value against which accuracy is measured.
- Examples of common reference values,
  - x_{real}, if the exact solution is known.
  - x_{IEEE}, when the program is deterministic.
  - $\hat{\mu}$, if the program is non-deterministic.
  - Y, a random variable, to compare two implementations of an algorithm or measuring significance between runs of the same program.

Modeling the error

- Four kind of scenarios are studied in our paper.
- 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.

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

- With no error, the expected result of Z is 0.

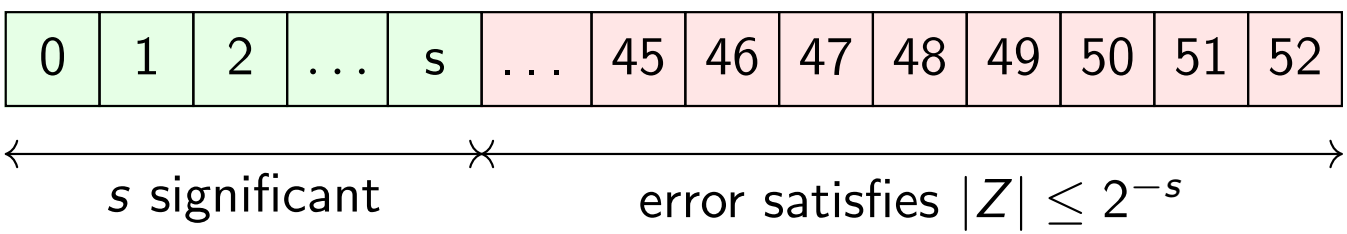
Significant bits

- Stott Parker defines the significant digits between x and y as the largest s that satisfies $|x/y - 1| \leq 2^{-s}$.
- Or put more simply, the error is less than 2^{-s}.
- We naturally extend this definition to Z the random variable modeling the stochastic error.

Significant bits

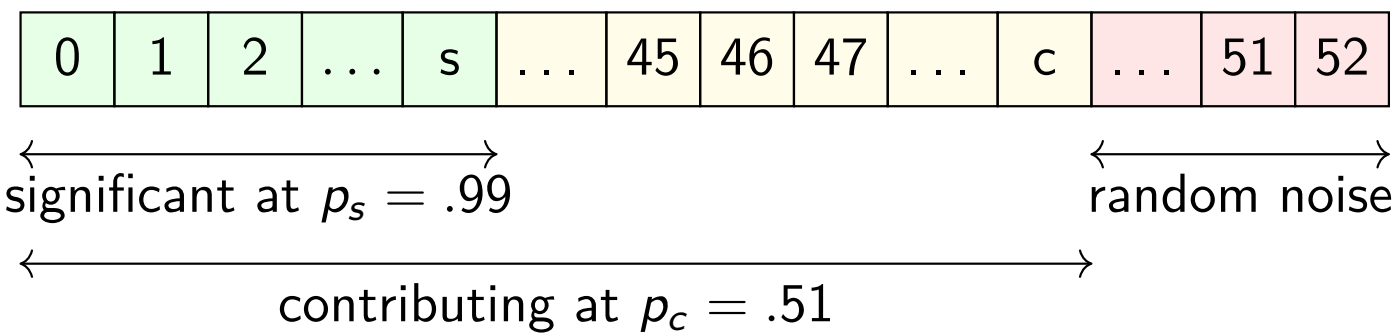
The number of significant digits with probability p_s can be defined as the largest number s such that

$$\mathbb{P}\left(|Z| \leq 2^{-s}\right) \geq p_s. \tag{3}$$



Contributing bits

- Bits after s still can encode useful information about the result.
  - Even if bits on its left are wrong, they can improve the accuracy...
  - ...if they are correct on average ($p_c > 51\%$).
  - Keeping these bits improves the rounded result on average.
- A bit k after s contributes to the result with probability p_c *iff* the k-th bit of Z is 0 (no error in this bit) with probability p_c.



Normality of the Kahan 2x2 System

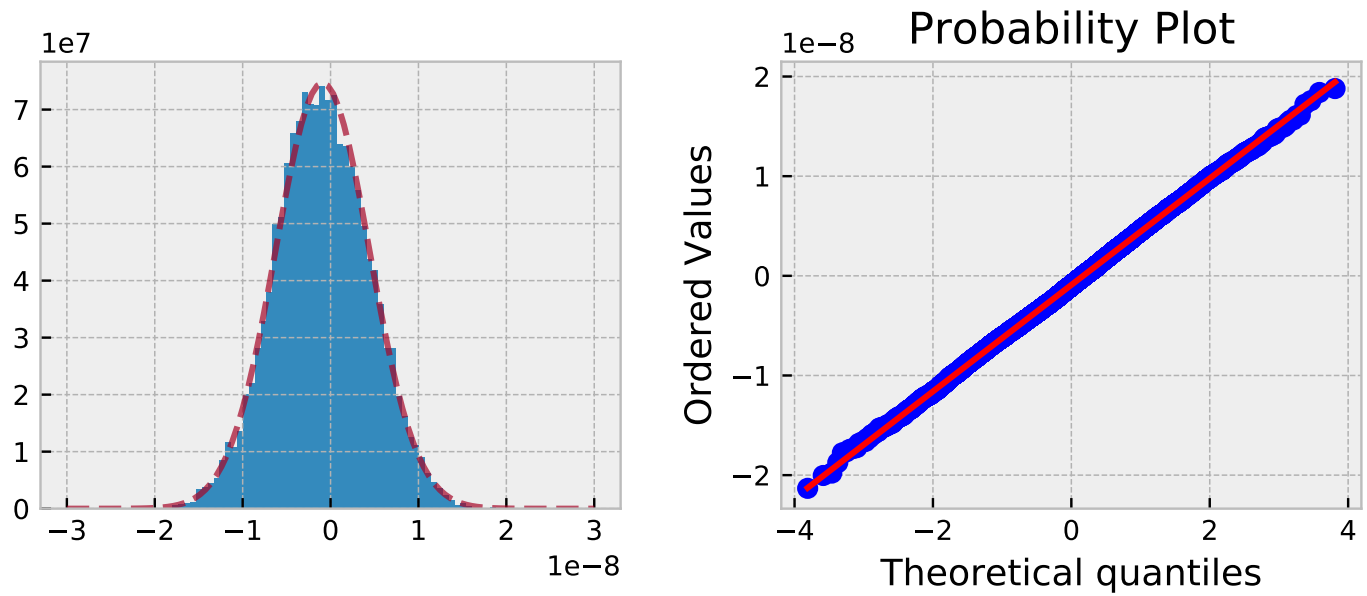
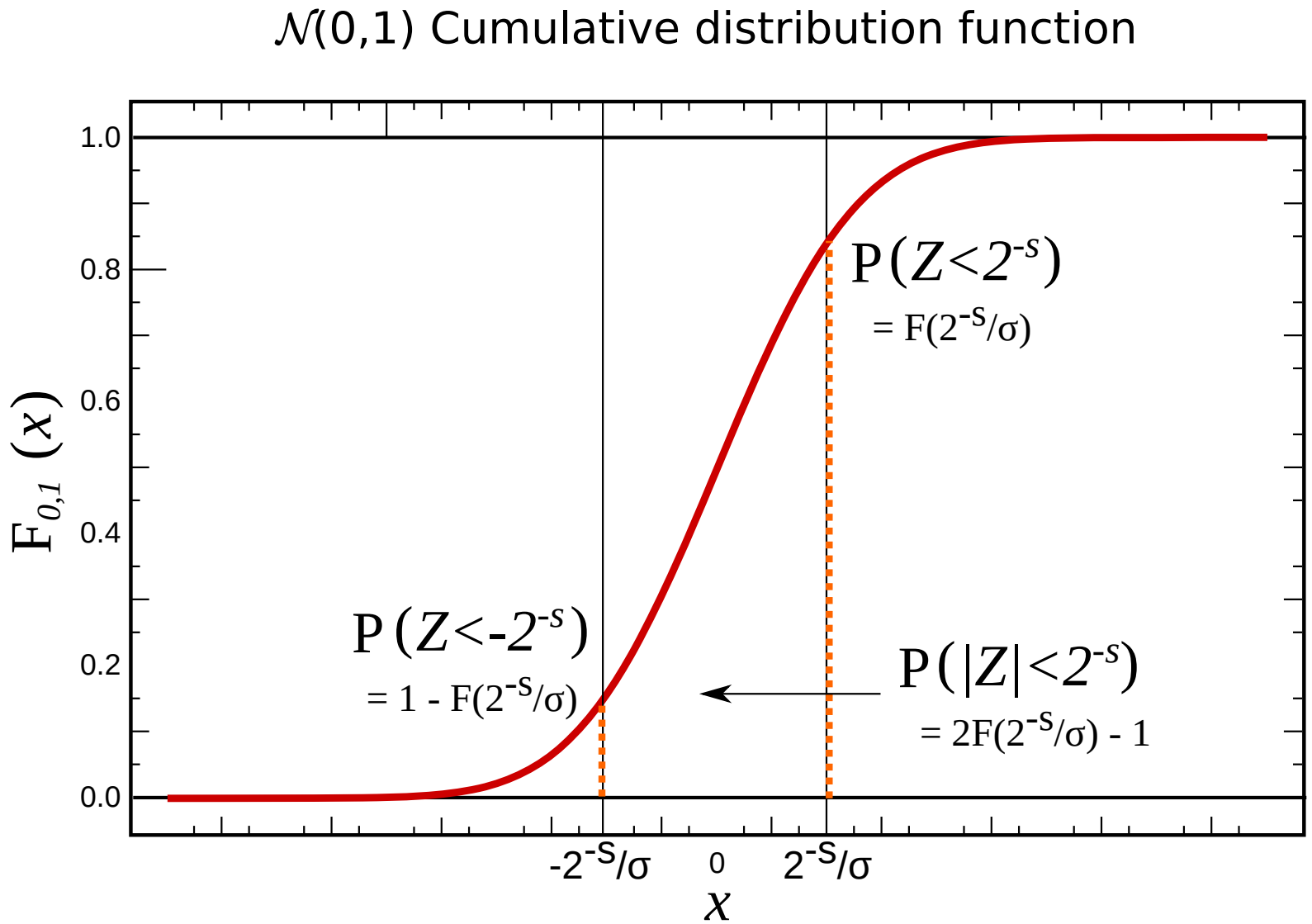


Figure: Normality of 10000 samples of $X[0]$ with $t = 52$ and FULL MCA

- We take as reference the empirical mean $\hat{\mu}$.

Centered Normal Hypothesis: Significant bits



Centered Normal Hypothesis: Significant bits

Theorem

For a normal centered error distribution $Z \sim \mathcal{N}(0, \sigma)$, the s-th bit is significant with probability

$$p_s = 2F\left(\frac{2^{-s}}{\sigma}\right) - 1,$$

with F the cumulative function of the normal distribution with mean 0 and variance 1.

- By inverting this formula, we can provide a formula for the number of significant digits that only depends on σ and p_s,

$$s = -\log_2(\sigma) - \log_2\left(F^{-1}\left(\frac{p_s + 1}{2}\right)\right).$$

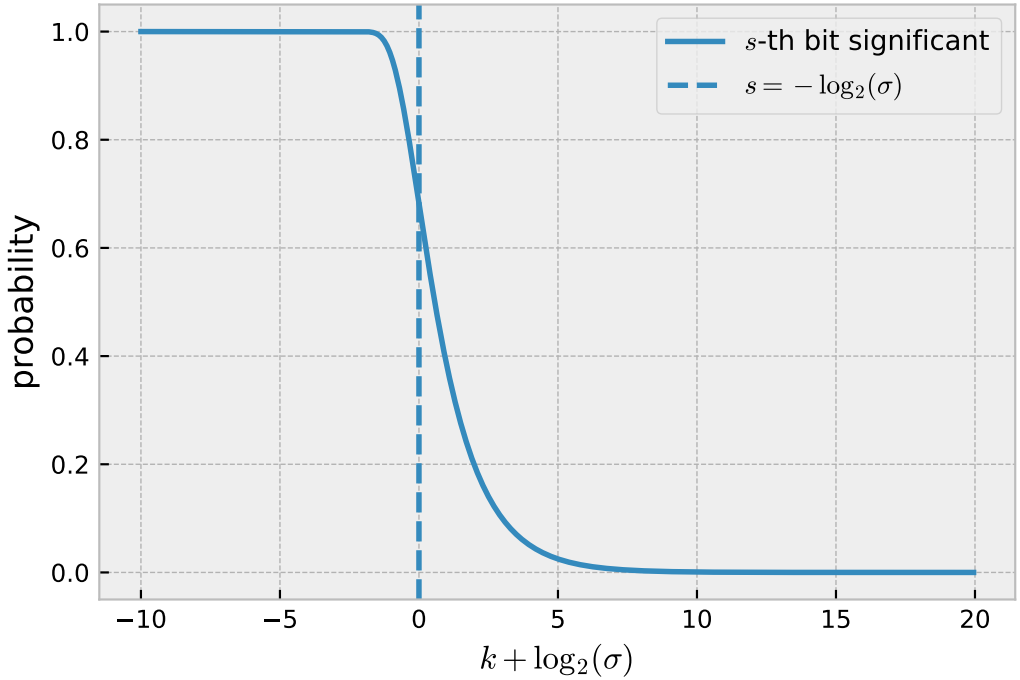


Figure: Profile of the significant bit curve $p_s = 2F\left(\frac{2^{-s}}{\sigma}\right) - 1$

- If we take the empirical average as reference value, we fall back into Stott Parker definition of significant bits assuming a large number of samples $-\log_2(\sigma) = -\log_2(\frac{\sigma_X}{|\hat{\mu}|})$
- The digit of Stott Parker's formula has 68 % chances of being significant. (1-sigma rule)
- If we substract 1.37 bits from Stott Parker's formula, the resulting bit has 99 % chances of being significant.

CNH: Taking into account the estimation bias

$$s = -\log_2(\sigma) - \log_2\left(F^{-1}\left(\frac{p_s+1}{2}\right)\right).$$

- Why is this formula independent of the number of samples n ?
- σ is unknown; we can only estimate it from $\hat{\sigma}$
- For normal distributions, the following confidence interval with confidence $1-\alpha$ based on the χ^2 distribution with $(n-1)$ degrees of freedom is sound [**?**]:

$$\frac{(n-1)\hat{\sigma}^2}{\chi^2_{\alpha/2}} \leq \sigma^2 \leq \frac{(n-1)\hat{\sigma}^2}{\chi^2_{1-\alpha/2}}. \tag{4}$$

- In the following we choose a confidence of $1-\alpha = 95\%$.

CNH: Significant bits lower bound

- By combination, we produce a sound lower bound on the significant bits,

$$s \geqslant \underbrace{-\log_2(\hat{\sigma}) - \underbrace{\left[\frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right) + \log_2\left(F^{-1}\left(\frac{p+1}{2}\right)\right)\right]}_{\delta_{\text{CNH}}}}_{\hat{s}_{\text{CNH}}} \tag{5}$$

- 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 $\hat{\mu}$)

CNH: Contributing bits

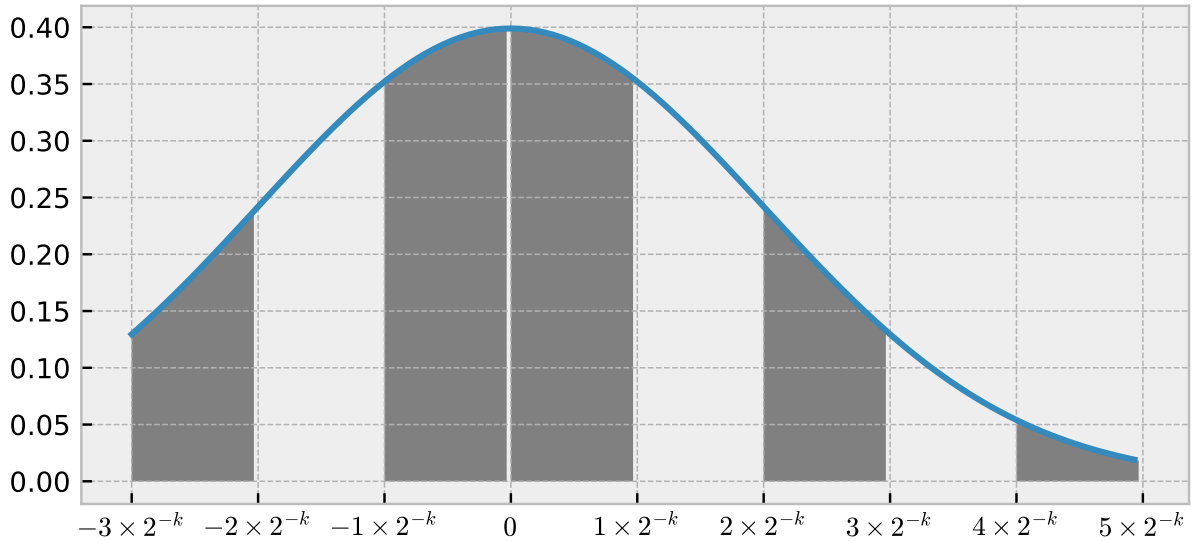


Figure: Normal curve; the gray zones correspond to the area where the k-th bit contributes to make the result closer to 0 (whatever the preceding digits).

$$\begin{aligned} & \exists i, \left\lfloor 2^k |Z| \right\rfloor = 2i \\ \Leftrightarrow \quad & 2i \leq 2^k |Z| < 2i + 1 \\ \Leftrightarrow \quad & 2^{-k}(2i) \leq |Z| < 2^{-k}(2i + 1). \end{aligned} \tag{6}$$

CNH: Contributing bits

Theorem

For a normal centered error distribution $Z \sim \mathcal{N}(0, \sigma)$, when $\frac{2^{-c}}{\sigma}$ is small, the c-th bit contributes to the result accuracy with probability

$$p_c \sim \frac{2^{-c}}{2\sigma\sqrt{2\pi}} + \frac{1}{2}.$$

If we wish to keep only bits improving the result with a probability greater than p, then we will keep c contributing bits, with

$$c = -\log_2(\sigma) - \log_2(p_c - \frac{1}{2}) - \log_2(2\sqrt{2\pi}). \tag{7}$$

Again, this formula only depends on σ and the probability p_c. As previously, a sound lower or upper bound can be computed with the Chi-2 confidence interval of σ.

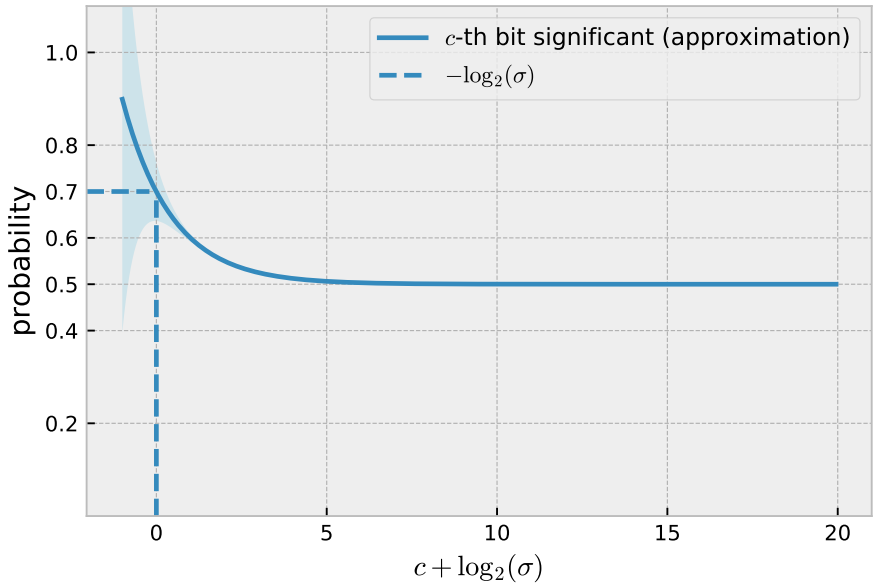


Figure: Profile of the contribution bit curve: The shaded area represents the bound on the error. The approximation is very tight for probabilities less than 70%.

Results: Significant bits

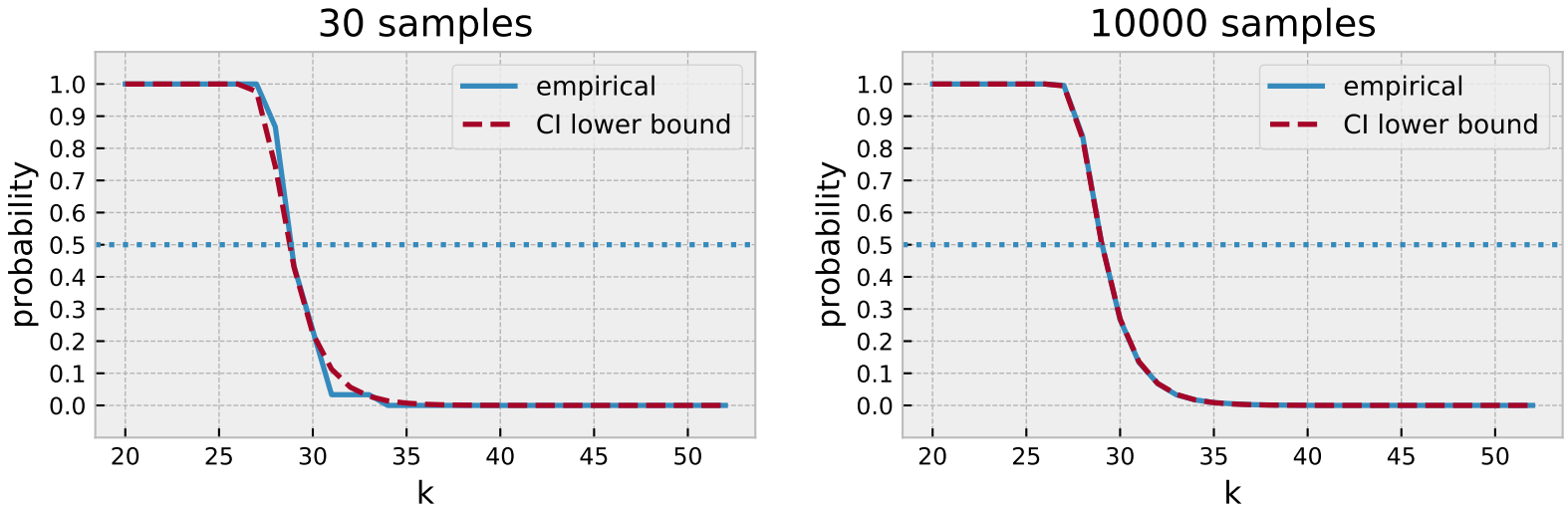


Figure: Significant bits for Cramer $x[0]$ variable computed under the normal hypothesis using 30 and 10000 samples. The Confidence Interval (CI) lower bound is computed by using the probability of theorem 1 and bounding σ with a 95% Chi-2 confidence interval.

Results: Contributing bits

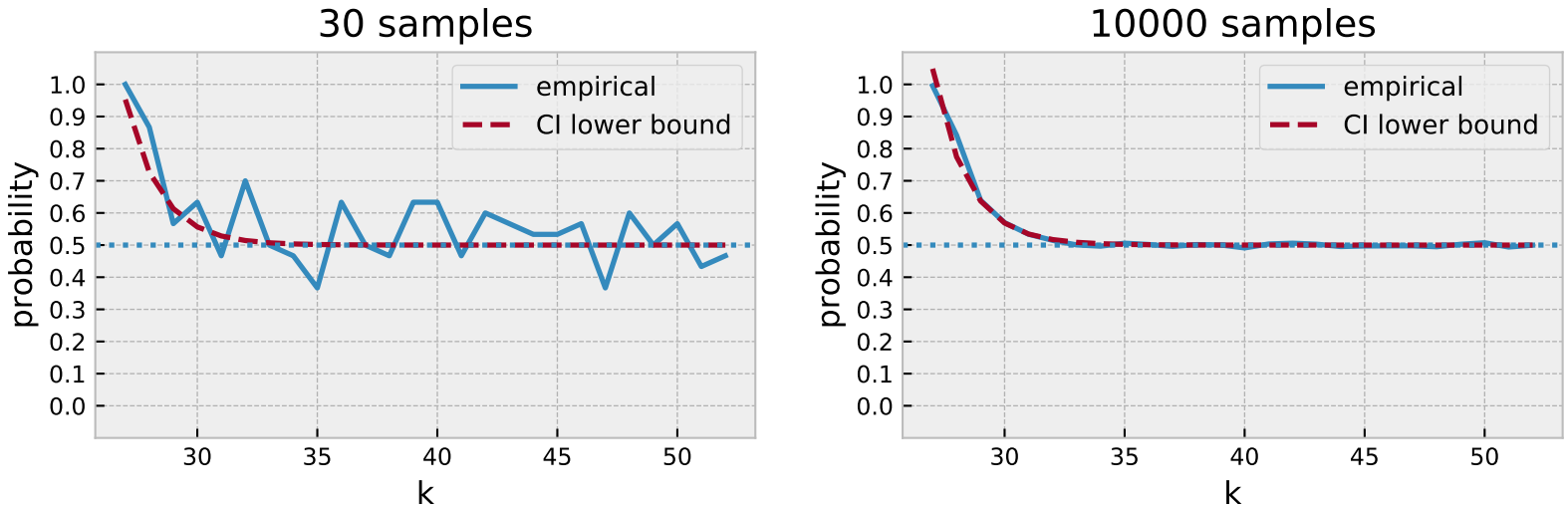
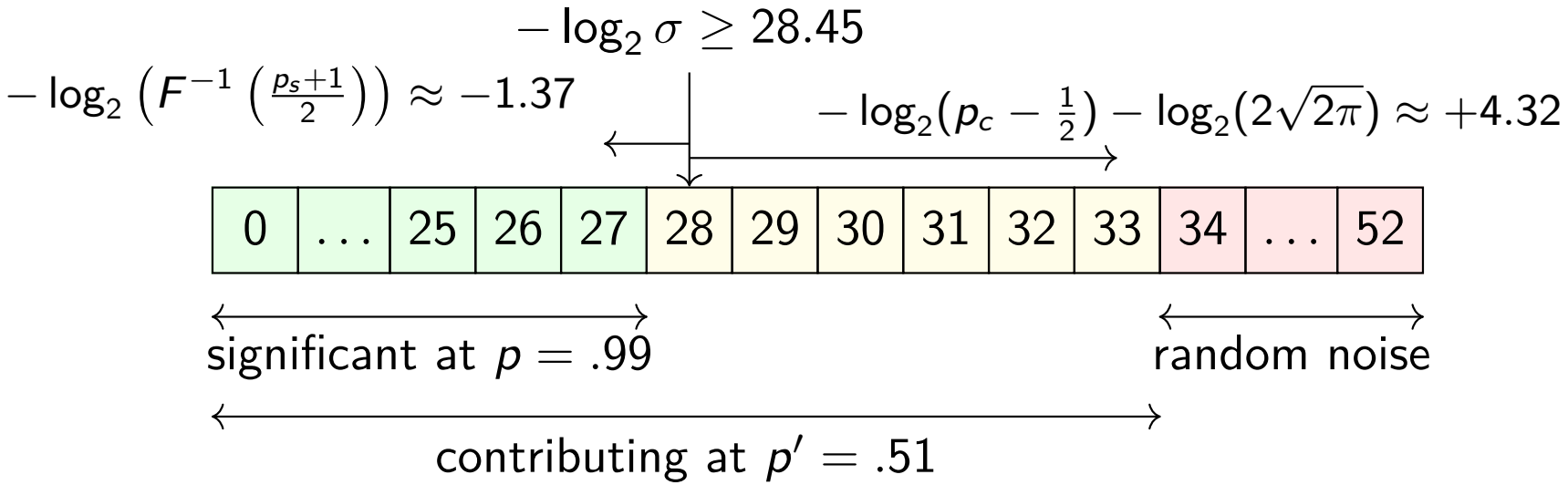


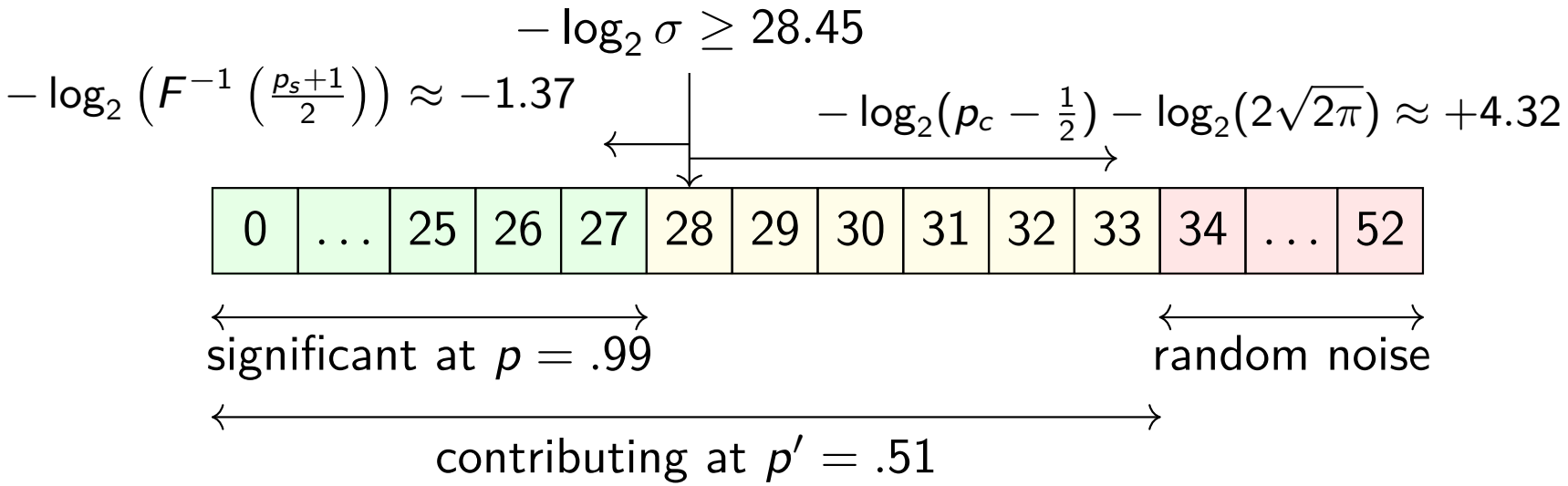
Figure: Contributing bits for Cramer $x[0]$ variable computed under the normal hypothesis using 30 and 10000 samples.

Summary: Significant and Contributing bits in the CNH (1/2)



1. We estimate a lower bound for
   $-log\sigma \geq 28.45 \approx -\log_2 \hat{\sigma} - \frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right)$
2. We apply a shift left (computed with $p_s = 99\%$) to get a safe significant bits lower-bound.
3. We apply a shift right (computed with $p_c = 51\%$) to get a safe contributing bits lower-bound.

Summary: Significant and Contributing bits in the CNH (2/2)



- Contributing bits help decide how many digits to print or store during a check-point restart.
- Only keeping contributing bits can help reducing storage and database sizes!

General Distributions

- What if the distribution is not centered normal?

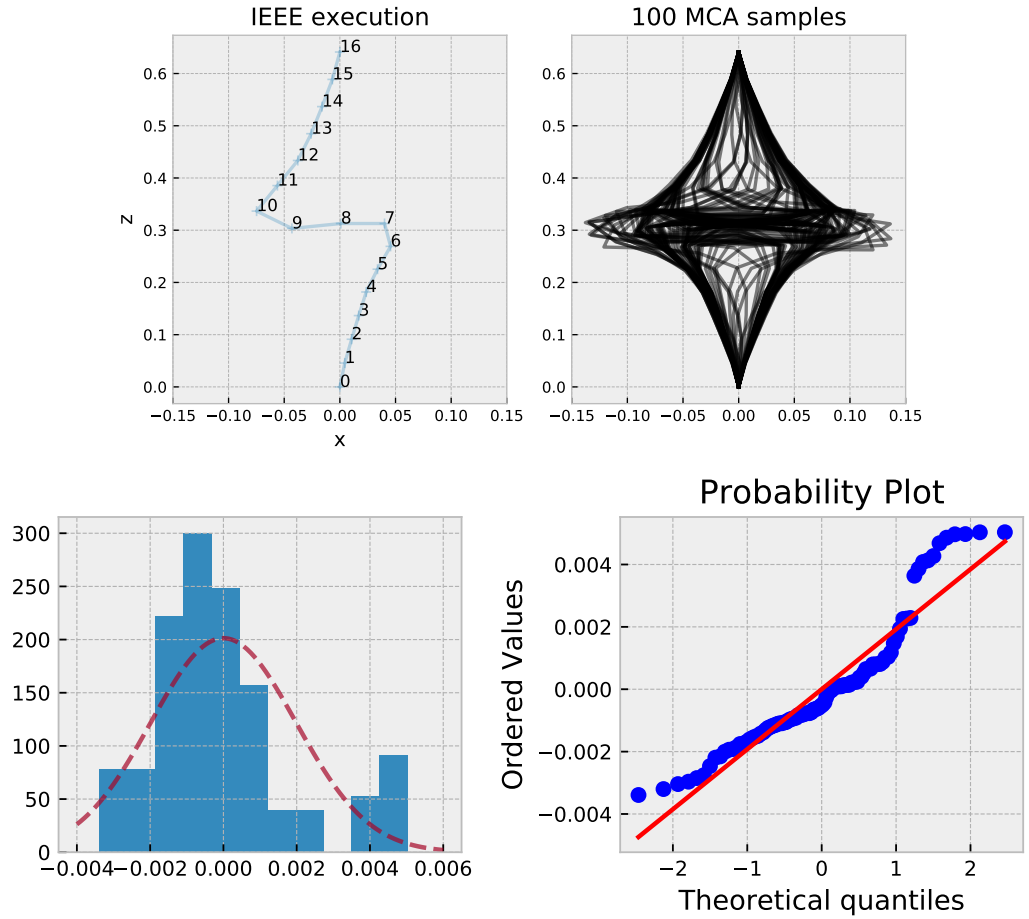


Figure: Non normality of buckling samples on z axis and node 1. Shapiro Wilk rejects the normality hypothesis.

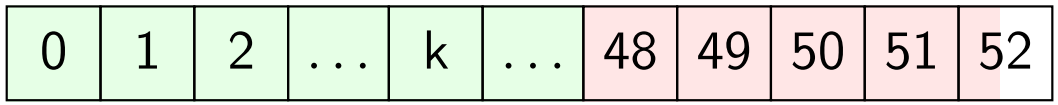
Model by Bernoulli Trials (1/2)

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

Model by Bernoulli Trials (2/2)

- We choose a given k.

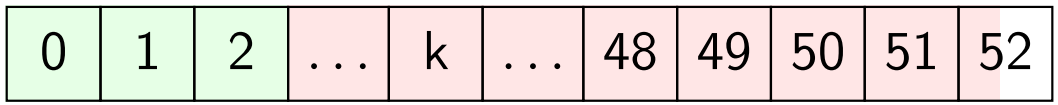
Sample X_1



S_1^k Success

$$|Z_1| \leq 2^{-k}$$

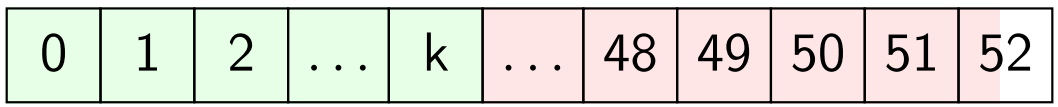
Sample X_2



S_2^k Failure

$$|Z_2| > 2^{-k}$$

Sample X_3



S_3^k Success

$$|Z_3| \leq 2^{-k}$$

- Out of three samples: 2 success and 1 failure; $n_s = 2$.
- Can we estimate the Bernoulli distribution of S^k ?

Bernoulli Estimator

- [**?**] gives the following lower-bound for the success probability of a Bernoulli distribution at 95% confidence,

$$\frac{n_s + 2}{n + 4} - 1.65\sqrt{\frac{(n_s + 2)(n - n_s + 2)}{(n + 4)^3}}$$

- By counting for S_i^k the number of successes n_s (where the first k digits are significant) we can derive a safe lower-bound probability.

Probability vs. Confidence

- We want to estimate the probability p_s of the s-th bit being significant.
- Suppose $p_s = \frac{1}{3}$ is the true parameter of the Bernoulli distribution.
- We do m different samplings of $n = 10$ values:
  - 1st sampling: $n_s = 3 \rightarrow p_s \in [.15, .57]$
  - 2rd sampling: $n_s = 8 \rightarrow p_s \notin [.52, .91]$
  - 3nd sampling: $n_s = 2 \rightarrow p_s \in [.08, .48]$
  - ...
- The confidence $1 - \alpha$ measures the proportion of samplings that produce an interval containing p_s.
- Increasing the number of samples n reduces the probability of a biased interval and therefore increases the confidence.

Example of Bernoulli Estimator on Kahan's system

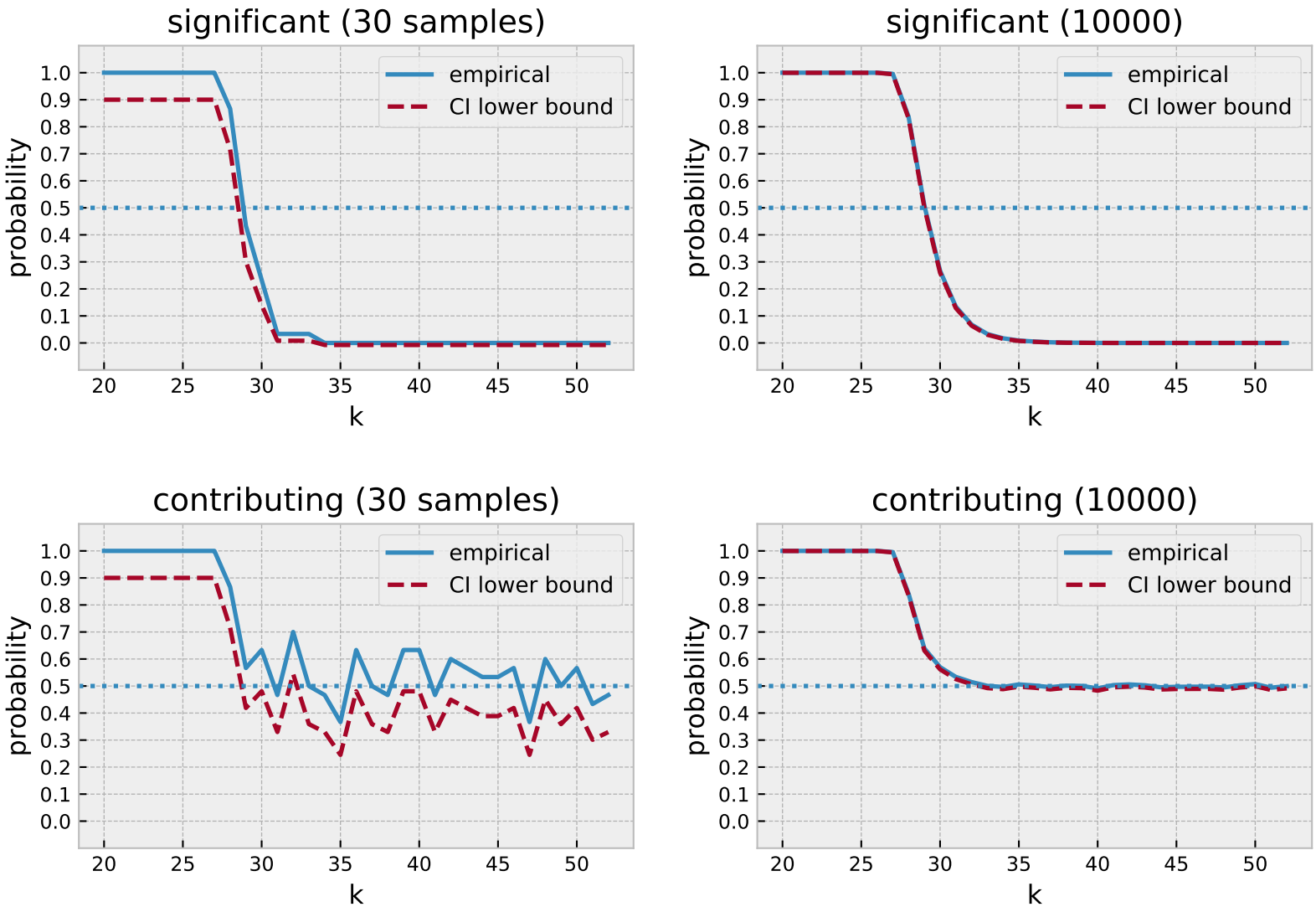


Figure: Significance and contribution per bit for variable $X[0]$ of the Cramer's system with 30 and 10000 samples.

Special Case: No failures

- Let us consider the largest k so that S_i^k is true for all i. In other words, k is significant in all the collected samples.
- In that case, [5] shows that $\mathbb{P}(S^k) > p$ with confidence $1 - \alpha$ if we have

$$n = n_s \geq \left\lceil \frac{\ln(\alpha)}{\ln(p)} \right\rceil$$

- This formula gives us a simple criterion for choosing a minimal number of samples depending on the required confidence level.
  1. Choose a probability and confidence level that are acceptable for your experiment: eg. $p = 90\%$ and $1 - \alpha = 95\%$
  2. Compute and collect the required number of samples, here $n = 29$.
  3. Find the largest k that is significant for all samples; that k is significant with $p = 90\%$ at confidence level 95%.

How many samples are required?

Confidence level $1-\alpha$	Probability p								
	0.66	0.75	0.8	0.85	0.9	0.95	0.99	0.995	0.999
0.66	3	4	5	7	11	22	108	216	1079
0.75	4	5	7	9	14	28	138	277	1386
0.8	4	6	8	10	16	32	161	322	1609
0.85	5	7	9	12	19	37	189	379	1897
0.9	6	9	11	15	22	45	230	460	2302
0.95	8	11	14	19	29	59	299	598	2995
0.99	12	17	21	29	44	90	459	919	4603
0.995	13	19	24	33	51	104	528	1058	5296
0.999	17	25	31	43	66	135	688	1379	6905

Table: Number of samples necessary to obtain a given confidence interval with probability p, according to the Bernoulli estimator (*i.e.* without any assumption on the probability law).

EuroPlexus Buckling Analysis (1/2)

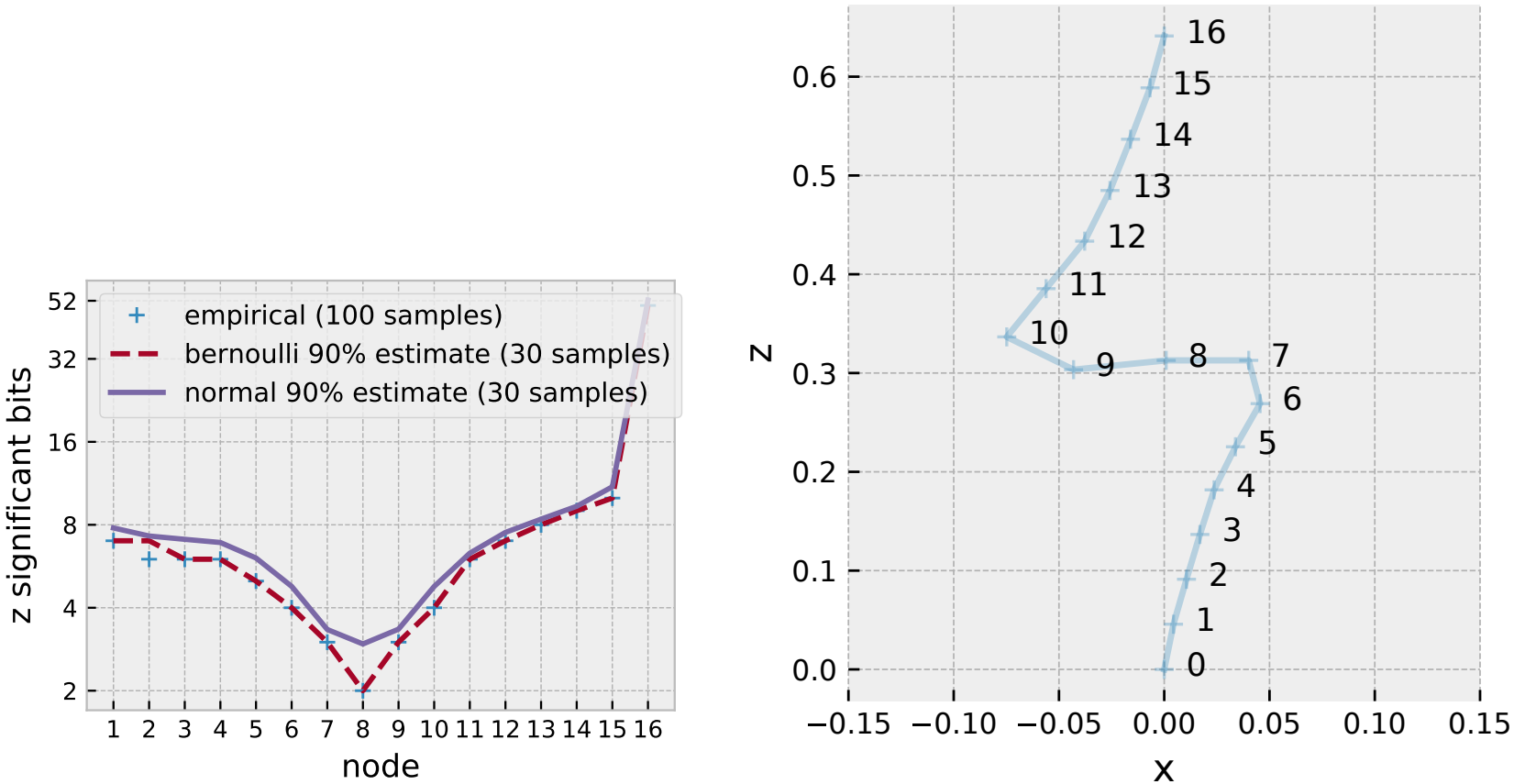


Figure: Significant bits on the z axis distribution. Bernoulli estimation captures precisely the behavior (except for node 2). Normal formula overestimates the number of digits, this is expected since the distribution is strongly non normal.

EuroPlexus Buckling Analysis (2/2)

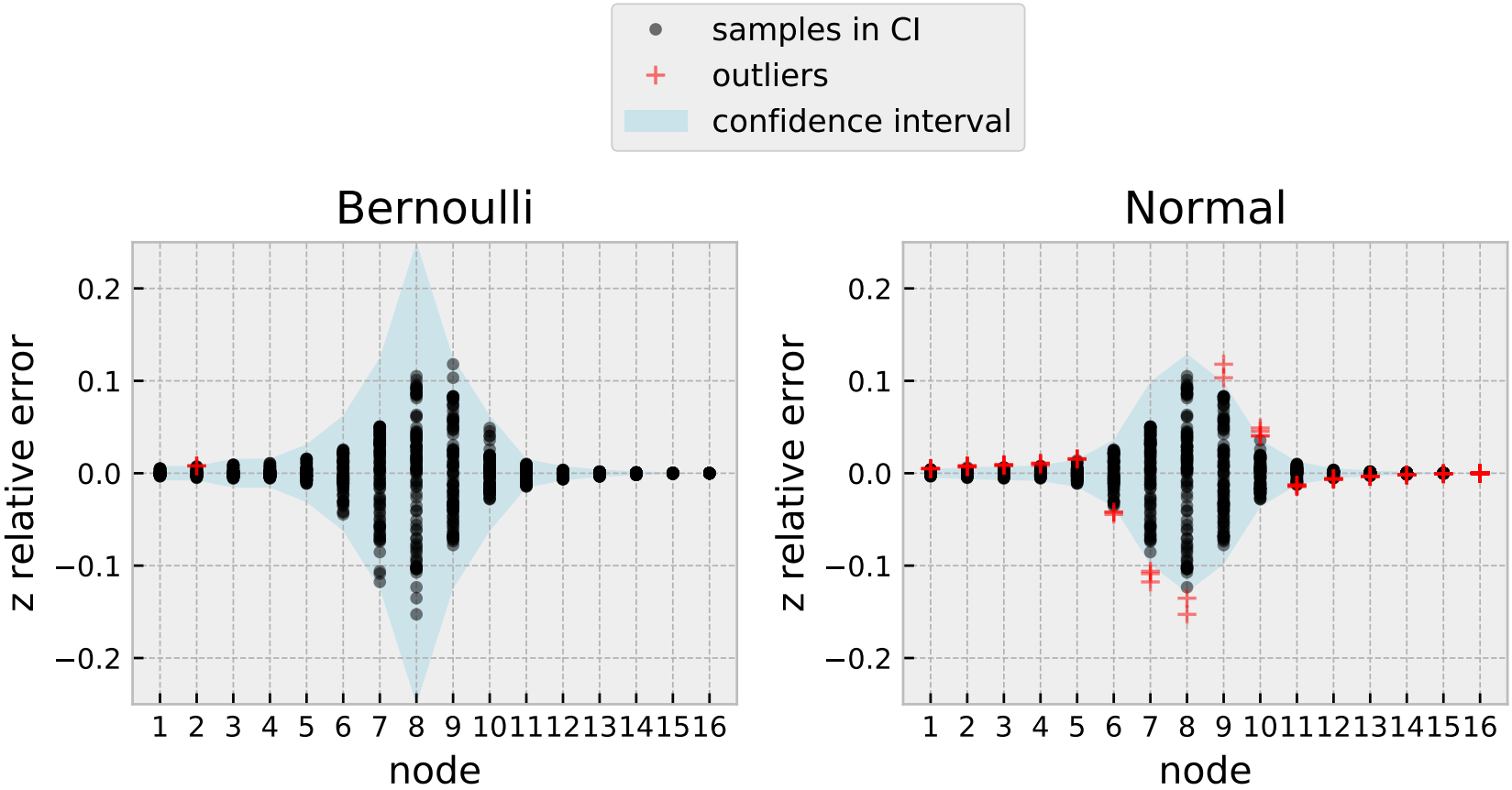


Figure: Relative error between the samples and the mean of the z-axis distribution. The blue envelope corresponds to the computed confidence interval with 30 samples. Black dots are samples that fall inside the CI. Red crosses are outliers that fall outside the CI. In the Bernoulli case, only 3 samples out of 70 fall outside of the interval; which is compatible with the 90% probability threshold.

Limits and Discussion

- These confidence intervals estimate the error of over-estimating s due to sampling errors
  - not enough samples taken or biased sampling
- These confidence intervals do not account for model errors
  - Changes in the dataset
  - Failures of MCA or CESTAC to correctly model FP errors (thread scheduling, model corner-cases, etc.)

Conclusion on Confidence Intervals for Stochastic Arithmetic

- For normal centered distributions:
  - Simple probability formulations for significance and contribution that only depend on $\hat{\sigma}$, n and $1 - \alpha$.
  - Applying a left or right shift to the pivotal $-log_2(\sigma)$ Stott Parker's estimator produces a lower-bound on the number of significant and contributing bits.
- For general distributions:
  - Model each mantissa *bit* as a separate Bernoulli distribution.
  - When only interested in the significant bits, a simple formula computes how many samples are needed to reach a given probability level.
- How can I apply these results to my studies?
  - Tables for the CNH shifts and number of required samples are available in the preprint.
  - A jupyter notebook implemenenting the formulas is also available.

Code_aster : update reference (1/3)

Implementation	$\hat{s}_{\mathrm{MCA}}$		comment
	a	*e*	
`version0`	Fail	Fail	original version
`version1`	30.89	19.73	fixes an unstable test
`version2`	30.96	19.80	compensated summation
`version3`	32.82	21.65	fully compensated dot product

Table: Summary of the numerical quality assessment of 4 versions of code_aster, using Verrou and the standard MCA estimator with 6 samples.

With `version3` the accuracy seems improved, but we need confidence intervals to update reference value. We choose $p = (1 - \alpha) = 0.995$:

- $N_{sample} = 1058$

Code_aster : update reference (2/3)

Implementation	$\hat{s}_{\mathrm{B}}^{\hat{\mu}}$	$\hat{s}_{\mathrm{B}}^{\mathrm{IEEE}}$	$\hat{s}_{\mathrm{CNH}}$ (normality test p-value)	$\hat{s}_{\mathrm{MCA}}$
`version1`	28	28	29.01 (0.10)	30.59
`version2`	29	29	29.55 (0.89)	31.13
`version3`	30	31	31.22 (0.52)	32.79

(a) quantity a

Implementation	$\hat{s}_{\mathrm{B}}^{\hat{\mu}}$	$\hat{s}_{\mathrm{B}}^{\mathrm{IEEE}}$	$\hat{s}_{\mathrm{CNH}}$ (normality test p-value)	$\hat{s}_{\mathrm{MCA}}$
`version1`	17	17	17.85 (0.10)	19.43
`version2`	18	18	18.39 (0.89)	19.97
`version3`	19	19	20.05 (0.52)	21.63

(b) quantity e

Table: Comparison of stochastic estimators for 3 version of code_aster, with 1058 samples.

Code_aster : update reference (2/3)

Implementation	$\hat{s}_{\mathrm{B}}^{\hat{\mu}}$	$\hat{s}_{\mathrm{B}}^{\mathrm{IEEE}}$	$\hat{s}_{\mathrm{CNH}}$ (normality test p-value)	$\hat{s}_{\mathrm{MCA}}$
`version1`	28.89	28.57	29.01 (0.10)	30.59
`version2`	29.33	29.35	29.55 (0.89)	31.13
`version3`	30.91	31.00	31.22 (0.52)	32.79

(a) quantity a

Implementation	$\hat{s}_{\mathrm{B}}^{\hat{\mu}}$	$\hat{s}_{\mathrm{B}}^{\mathrm{IEEE}}$	$\hat{s}_{\mathrm{CNH}}$ (normality test p-value)	$\hat{s}_{\mathrm{MCA}}$
`version1`	17.73	17.41	17.85 (0.10)	19.43
`version2`	18.16	18.19	18.39 (0.89)	19.97
`version3`	19.75	19.84	20.05 (0.52)	21.63

(b) quantity e

Table: Comparison of stochastic estimators for 3 version of code_aster, with 1058 samples.

Code_aster : update reference (3/3)

`version3` is our new reference.

`version0` analysis

- $\left| \frac{a_{ieee}^{version0} - a_{ieee}^{version3}}{a_{ieee}^{version3}} \right| = 4.29 \times 10^{-10}$
- $\left| \frac{e_{ieee}^{version0} - e_{ieee}^{version3}}{e_{ieee}^{version3}} \right| = 9.84 \times 10^{-7}$

Need to analyze other test cases related to these 2 corrections before integration

Join us!

`github.com/verificarlo/verificarlo`
`github.com/edf-hpc/verrou`
`github.com/interflop/interflop`

References I


ABINIT.
Silicon carbide, 2016.
[Online; accessed May 15, 2018].


F. Benz, A. Hildebrandt, and S. Hack.
A dynamic program analysis to find floating-point accuracy problems.
ACM SIGPLAN, 47(6):453–462, 2012.


Y. Chatelain, E. Petit, P. de Oliveira Castro, G. Lartigue, and D. Defour.
Automatic exploration of reduced floating-point representations in iterative methods.
In *Euro-Par 2019 Parallel Processing - 25th International Conference*, Lecture Notes in Computer Science. Springer, 2019.

References II


C. Denis, P. de Oliveira Castro, and E. Petit.
Verificarlo: checking floating point accuracy through monte carlo arithmetic.
In *Computer Arithmetic (ARITH), 23nd Symposium on*, pages 55–62. IEEE, 2016.


E. Eypasch, R. Lefering, C. K. Kum, and H. Troidl.
Probability of adverse events that have not yet occurred: a statistical reminder.
BMJ, 311(7005):619–620, 1995.


F. Févotte and B. Lathuilière.
VERROU: a CESTAC evaluation without recompilation.
SCAN 2016, page 47, 2016.


X. Gonze, F. Jollet, et al.
Recent developments in the ABINIT software package.
Computer Physics Communications, 205:106–131, 2016.

References III


S. Graillat, F. Jézéquel, R. Picot, F. Févotte, and B. Lathuilière.
Auto-tuning for floating-point precision with Discrete Stochastic Arithmetic.
working paper or preprint, June 2016.


F. Jézéquel and J.-M. Chesneaux.
CADNA: a library for estimating round-off error propagation.
Computer Physics Communications, 178(12), 2008.


M. O. Lam, J. K. Hollingsworth, and G. Stewart.
Dynamic floating-point cancellation detection.
Parallel Computing, 39(3):146–155, 2013.


T. Ogita, S. M. Rump, and S. Oishi.
Accurate sum and dot product.
SIAM Journal on Scientific Computing, 26(6):1955–1988, 2005.

References IV


P. Panchekha, A. Sanchez-Stern, J. R. Wilcox, and Z. Tatlock.
Automatically improving accuracy for floating point expressions.
In *ACM SIGPLAN Notices*, volume 50, pages 1–11. ACM, 2015.


C. Rubio-González, C. Nguyen, H. D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey, C. Iancu, and D. Hough.
Precimonious: Tuning assistant for floating-point precision.
In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 27. ACM, 2013.


A. Sanchez-Stern, P. Panchekha, S. Lerner, and Z. Tatlock.
Finding root causes of floating point error with herbgrind.
arXiv preprint arXiv:1705.10416, 2017.