Confidence Intervals for Stochastic Arithmetic

Devan Sohier, Pablo de Oliveira Castro, François Févotte, Bruno Lathuilière, Eric Petit, Olivier Jamond

Confidence Intervals for Stochastic Arithmetic

Devan Sohier*
University of Versailles – Li-PaRAD

Pablo de Oliveira Castro†
University of Versailles – Li-PaRAD

François Févotte‡
EDF R&D – PERICLES

Bruno Lathuilière§
EDF R&D – PERICLES

Eric Petit¶
Intel Corp.

Olivier Jamond‖
CEA

July 2, 2018

Abstract

Quantifying errors and losses due to the use of Floating-Point (FP) calculations in industrial scientific computing codes is an important part of the Verification, Validation and Uncertainty Quantification (VVUQ) process. Stochastic Arithmetic is one way to model and estimate FP losses of accuracy, which scales well to large, industrial codes. It exists in different flavors, such as CESTAC or MCA, implemented in various tools such as CADNA, Verificarlo or Verrou. These methodologies and tools are based on the idea that FP losses of accuracy can be modeled via randomness. Therefore, they share the same need to perform a statistical analysis of programs results in order to estimate the significance of the results.

In this paper, we propose a framework to perform a solid statistical analysis of Stochastic Arithmetic. This framework unifies all existing definitions of the number of significant digits (CESTAC and MCA), and also proposes a new quantity of interest: the number of digits contributing to the accuracy of the results. Sound confidence intervals are provided for all estimators, both in the case of normally distributed results, and in the general case. The use of this framework is demonstrated by two case studies of large, industrial codes: Europlexus and code_aster.

*devan.sohier@uvsq.fr
†pablo.oliveira@uvsq.fr
‡francois.fevotte@edf.fr
§bruno.lathuiliere@edf.fr
¶eric.petit@intel.com
‖olivier.jamond@cea.fr

1 Introduction

Modern computers use the IEEE-754 standard for implementing floating point (FP) operations. Each FP operand is represented with a limited precision. Single precision numbers have 23 bits in the mantissa and double precision numbers have 52 bits in the mantissa. This limited precision may cause numerical errors [Hig02] such as absorption or catastrophic cancellation which can result in loss of significant bits in the result.

Floating Point computations are used in many critical fields such as structure, combustion, astrophysics or finance simulations. Determining the precision of a result is an important problem. For well known algorithms, mathematical bounds of the numerical error can be derived for a given dataset [Hig02]. But for arbitrary computer programs of thousands of code lines, deriving such an analysis is intractable.

The Stochastic Arithmetic field proposes automatic methods for estimating the number of significant digits in a result. Two main methods have been proposed: CESTAC [Vig04] and Monte Carlo Arithmetic (MCA) [SP97], which differ in many subtle ways, but share the same general principles. Numerical errors are modeled by introducing random perturbations at each FP operation. This transforms the output of a given simulation code into realizations of a random variable. Performing a statistical analysis of a set of sampled outputs allows to stochastically approximate the impact of numerical errors on the code results.

In this paper, we propose a solid statistical analysis of Stochastic Arithmetic that does not necessarily rely on the normality assumption and provides strong confidence intervals. Unlike traditional analysis which only considers the number of reliable significant digits, this paper introduces a new quantity of interest: the number of digits contributing to the accuracy of the final result.

Section 2 reviews the stochastic arithmetic methods. Section 3 formulates the problem rigorously and defines several interesting scopes of study. Then we provide in section 4 a statistical analysis for normal distributions and in section 5 for general distributions. Section 6 validates our statistical framework on two industrial scientific computing codes: Europlexus and code_aster. Finally, section 7 discusses some of the remaining limitations of stochastic arithmetic methods, which should be addressed in future work. A few concluding notes are gathered in section 8.

2 Background on Stochastic Arithmetic Methods

2.1 Modeling accuracy loss using randomness

When a program is run on an IEEE-754-compliant processor, the result of each floating-point operation $x \circ y$ is replaced by a rounded value: round$(x \circ y)$. For

example, the default rounding mode for IEEE-754 binary formats is given by:

$$\text{round}(x) = \lfloor x \rceil,$$

where $\lfloor . \rceil$ denotes rounding to nearest, ties to even, for the considered precision (`binary32` or `binary64`).

When $x \circ y \neq \lfloor x \circ y \rceil$, *i.e.* when $x \circ y$ is not in the set $\mathbb{F}$ of representable FP values for the considered precision, rounding causes a loss of accuracy. Stochastic arithmetic methods model this loss of accuracy using randomness.

2.1.1 CESTAC

The CESTAC method models round-off errors by replacing rounding operations with randomly rounded ones [VLP74, CV88]. The result of each FP operation $x \circ y$ is substituted with random_round$(x \circ y)$, where random_round is a function which randomly rounds FP values upwards or downwards equiprobably:

$$\text{random\_round}(x) = \begin{cases} x & \text{if } x \in \mathbb{F} \\ \zeta \lfloor x \rfloor + (1 - \zeta) \lceil x \rceil & \text{otherwise,} \end{cases}$$

where $\lfloor . \rfloor$ and $\lceil . \rceil$ respectively represent the downward and upward rounding operations for the considered precision, and ζ is a random variable such that $P[\zeta = 0] = P[\zeta = 1] = \frac{1}{2}$. CESTAC proposes different variants by changing the probability function P distribution.

2.1.2 Monte Carlo Arithmetic

MCA can simulate the effect of different FP precisions by operating at a virtual precision t. To model errors on a FP value x at virtual precision t, MCA uses the noise function

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

where $e_x = \lfloor \log_2 |x| \rfloor + 1$ is the order of magnitude of x and ξ is a uniformly distributed random variable in the range $\left(-\frac{1}{2}, \frac{1}{2}\right)$. During the MCA run of a given program, the result of each FP operation is replaced by a perturbed computation modeling the losses of accuracy [SP97, DdOCP16, FL15]. It allows to model the computing at virtual precision t. Three possible expressions can be substituted to $x \circ y$, defining variants of MCA:

1. "Random rouding" only introduces perturbation on the output: round(inexact$(x \circ y)$).
2. "Inbound" only introduces perturbation on the input: round(inexact$(x) \circ$ inexact(y)).
3. "Full MCA" introduces perturbation on operand(s) and the result: round(inexact(inexact$(x) \circ$ inexact(y))).

In any case, using stochastic arithmetic, the result of each FP operation is replaced with a random variable modeling the losses of accuracy resulting from the use of finite-precision FP computations. Since the result of each FP operation in the program is in turn used as input for the following FP operations, it is natural to assume that the outputs of the whole program in stochastic arithmetic are random variables.

Stochastic Arithmetic methods run the program multiple times in order to produce a set of output results (*i.e.* a set of realizations or samples of the random variable modeling the program output). The samples are then statistically analyzed in order to asses the quality of the result.

2.2 Estimating the result quality: significant digits

Let us denote by x the quantity computed by a deterministic numerical program. Different values can be defined for this result:

- x_{real} is the value of x that would be computed with an infinitely precise, real arithmetic;
- x_{IEEE} is the value which is computed by the program, when run on a computer that uses standard IEEE arithmetic;
- $X_1, X_2, \ldots, X_n$ are the values returned by n runs of the program using stochastic arithmetic. These are seen as n realizations of the same random variable X.

Figure 1 illustrates some of the quantities of interest which can be useful to analyze the quality of the results given by the program. The real density of random variable X is unknown, but some of its characteristics can be estimated using n sample values $(X_1, \ldots, X_n)$. In particular:

- the expected value $\mu = E[X]$ can be estimated by the empirical average value of X_i, $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$;
- the standard deviation $\sigma = \sqrt{E[(X-\mu)^2]}$ can be estimated by the empirical standard deviation, $\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu})^2}$.

To estimate the numerical quality of the result, we would like to compute the number of *significant* bits. In the following we review the definitions of *significance* used in CESTAC and MCA. Then, we introduce the definition that will be used in this paper.

2.2.1 CESTAC definition of significant bits

In CESTAC, the average $\hat{\mu}$ of the small set of samples (usually three) is taken as the computed result and μ is assumed to be the mathematical result x_{real}. CESTAC defines the number of significant digits [Vig04] as the number of common digits between x_{real} and $\hat{\mu}$. This analysis is based on two hypotheses:

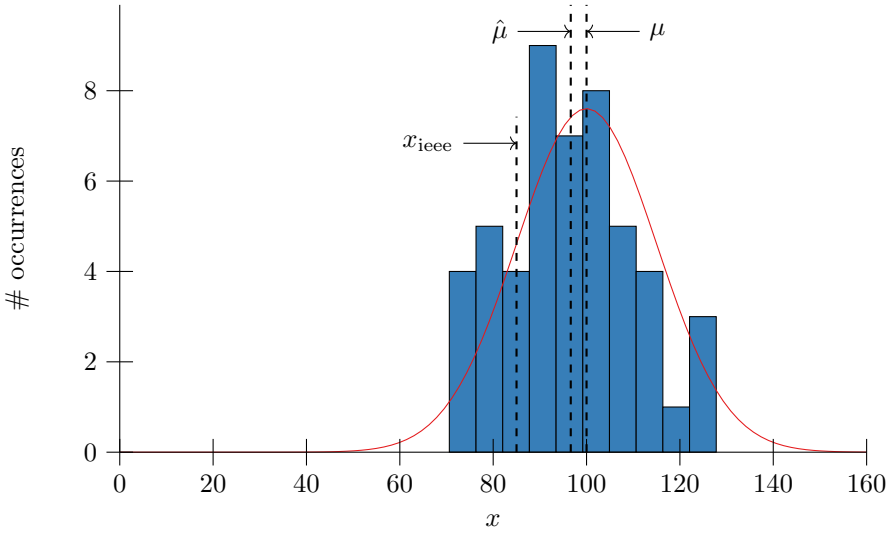

Figure 1: Schematic view of the various quantities of interest when evaluating the numerical quality of a result using stochastic arithmetic

1. the distribution X is normal, and
2. the distribution X is centered on the real result $\mu \approx x_{\text{real}}$.

Since X is assumed normal, one can derive the following Student t-distribution interval with confidence $(1 - \alpha)$:

$$\mu \in \left[\hat{\mu} \pm \frac{\tau_n \, \hat{\sigma}}{\sqrt{n}} \right],$$

where τ_n is a constant given by the Student distribution with $n - 1$ degrees of freedom and parameter $1 - \frac{\alpha}{2}$.

The maximum error between μ and $\hat{\mu}$ is bounded by this interval for a normal distribution, it follows [Li13] that the number of significant bits as defined above is

$$s_{\text{CESTAC}} = -\log_2 \left| \frac{x_{\text{real}} - \hat{\mu}}{x_{\text{real}}} \right| \approx -\log_2 \left| \frac{\mu - \hat{\mu}}{\hat{\mu}} \right| \geq \underbrace{\log_2 \left(\frac{\tau_n \, \hat{\sigma}}{\sqrt{n} \, |\hat{\mu}|} \right)}_{\hat{s}_{\text{CESTAC}}}. \tag{1}$$

In this paper, we do not follow this approach. First, the two hypotheses, while reasonable in many cases, do not always hold [Cha88, Kah96]: Stott Parker shows that the normality assumption of X is not always true [SP97, p. 49] and it is not necessarily centered on the real result. The robustness of CESTAC with respect to violations of these hypotheses is discussed in [CV88].

Second, and more important, the CESTAC definition of the number of significant digits may not necessarily be the most useful for the practitioner. Oftentimes, the objective of the numerical verification process consists in evaluating the precision of the actual IEEE computer arithmetic. CESTAC does not evaluate the number of significant digits of the IEEE result but rather of the average of the CESTAC samples. But in practice, x_{IEEE} does not match $\hat{\mu}$.

Last, with this definition, a problem clearly appears when considering the asymptotic behavior of the bound: $\hat{s}_{\text{CESTAC}} \xrightarrow[n\to+\infty]{} +\infty$. Increasing the number of samples arbitrarily increases the number of significant digits computed by CESTAC. On the one hand, this is expected because, according to the definition proposed, any computation is actually infinitely precise when $n \to \infty$ since the strong law of large numbers states that the empirical average is in this case almost surely the expected value. On the other hand, however, this asymptotic case also questions the pertinence of the CESTAC metric for the evaluation of the quality of the results produced by IEEE-754 computations. CESTAC is usually applied to three samples [CV88], the validity of the special case where $n = 3$ is discussed in section 4.3.

2.2.2 MCA definition of significant bits

In his study of MCA, Stott Parker defines the estimated number of significant digits as $s_{\text{MCA}} = -\log_2 \left|\frac{\sigma}{\mu}\right|$, which can be estimated by

$$\hat{s}_{\text{MCA}} = -\log_2 \left|\frac{\hat{\sigma}}{\hat{\mu}}\right|. \tag{2}$$

Stott Parker lays this definition on the habits of biology and physics regarding the precision of a measurement. It is a form of signal to noise ratio: if most random samples share the same first digits, these digits can be considered significant. On the contrary, digits varying randomly among sampled results are considered noise. Another way of giving meaning to this estimation is to consider x_{IEEE} as one possible realization of the random variable X. As such, its distance to μ is characterized by σ.

The problem with the MCA definition of significant bits is that it is empirical: the actual meaning of "significance" is not clearly laid, as well as the consequences one can draw from it. The estimator $\hat{s}_{\text{MCA}}$ can be computed in any situation but, since the number of samples n is finite, $\hat{s}_{\text{MCA}}$ is only an approximation of the exact value s_{MCA}. And no confidence interval is provided in order to help choose an appropriate number of samples. As shall be seen in section 4.3, this estimation is nevertheless a good basis when the underlying phenomenon is normal.

Our definition of significance generalizes Stott Parker's. It requires a refer-

ence which can be either a scalar value[1] or another random variable[2]. Informally, in this paper, the significant digits are the digits in common between the samples of X and the reference (up to a rounding effect on the last significant digit). Section 3 formalizes this definition in a probabilistic framework and provides sound confidence intervals.

2.3 Software tools presentation

The experimental validation of the presented confidence intervals on synthetic and industrial use cases has been conducted thanks to the Verificarlo and Verrou tools which are presented in the next subsections.

2.3.1 Verificarlo

Verificarlo [DdOCP16, Ver18a] is an open-source tool based on the LLVM compiler framework replacing at compilation each floating point operation by custom operators. After compilation, the program can be linked against various backends [CdOCP$^+$18], including MCA to explore random rounding and virtual precision impact on an application accuracy.

Doing the interposition at compiler level allows to take into account the compiler optimization effect on the generated FP operation flow. Furthermore, it allows to reduce the cost of this interposition by optimizing its integration with the original code.

2.3.2 Verrou

Verrou [FL16, Ver18b] is an open-source floating point diagnostics tool. It is based on Valgrind [NS07] to transparently intercept floating point operations at runtime and replace them by their random rounding counterpart. The interposition at runtime allows to address large and complex code based applications with no intervention of the end-user.

Verrou also provides two methods allowing to locate the origin of precision losses in the sources of the analyzed computing code. The first one is based on the code coverage comparison between two samples. Discrepancies in the code coverage are good indicators of potential branch instabilities. The second localization method leverages the delta-debugging algorithm [Zel09] to perform a binary search to find a maximal scope for which MCA perturbations do not produce errors or large changes in results. The remaining symbols (or lines if the binary is compiled with debug mode) are good candidates for correction.

[1] Good choices for the scalar reference are x_{real}, x_{IEEE} or μ_X depending on the aims of the study

[2] Using a second random variable as reference allows comparing two versions of a program or two algorithms variants, more details are given in section 3.

2.4 Synthetic example: Ill-conditioned linear system

To illustrate these methods in the following we use a simple synthetic example proposed by Kahan [Kah66]: solving an ill-conditioned linear system,

$$\begin{pmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{pmatrix} x = \begin{pmatrix} 0.1440 \\ 0.8642 \end{pmatrix} \tag{3}$$

The exact and IEEE `binary64` solutions of equation (3) are:

$$x_{\text{real}} = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \qquad x_{\text{IEEE}} = \begin{pmatrix} 1.9999999958366637 \\ -1.9999999972244424 \end{pmatrix} \tag{4}$$

To keep the example simple, the floating-point solution x_{IEEE} has been obtained by solving the system with the naive C implementation of Cramer's formula in double precision, as shown in listing 1.

Listing 1: Solving 2x2 system $a.x = b$ with Cramer's rule

```
void solve(const double a[4], const double b[2], double x[2]) {
  double  det = a[0] * a[3] - a[2] * a[1];
  double det0 = b[0] * a[3] - b[1] * a[1];
  double det1 = a[0] * b[1] - a[2] * b[0];
  x[0] = det0/det;
  x[1] = det1/det;
}
```

The condition number of the above system is approximately 2.5×10^8, therefore we expect to lose at least $\log_2(2.5 \times 10^8) \approx 28$ bits of accuracy or, equivalently, 8 decimal digits. By comparing the IEEE and exact values, we see that indeed the last 8 decimal digits differ. The number of common bits between x_{real} and x_{IEEE} is given by

$$s_{\text{IEEE}} = -\log_2 \left| \frac{x_{\text{real}} - x_{\text{IEEE}}}{x_{\text{real}}} \right| \approx \begin{pmatrix} 28.8 \\ 29.4 \end{pmatrix}.$$

Now let us use MCA to estimate the number of significant digits. We compile the above program with Verificarlo [DdOCP16] which transparently replaces every FP operation by its noisy MCA counterpart. Here a virtual precision of 52 is used to simulate roundoff errors in double precision. Then, we run the produced binary $n = 10\,000$ times and observe the resulting output distribution X.

Both $X[0]$ and $X[1]$ are normal with high Shapiro-Wilk test p-values 73 % and 74 % respectively.[3] Figure 2 shows the distribution and quantile-quantile (QQ) plots for $X[0]$, for which the empirical average and standard deviation are given by

$$\hat{\mu} \approx 1.99999999909,$$
$$\hat{\sigma} \approx 5.3427 \times 10^{-9}.$$

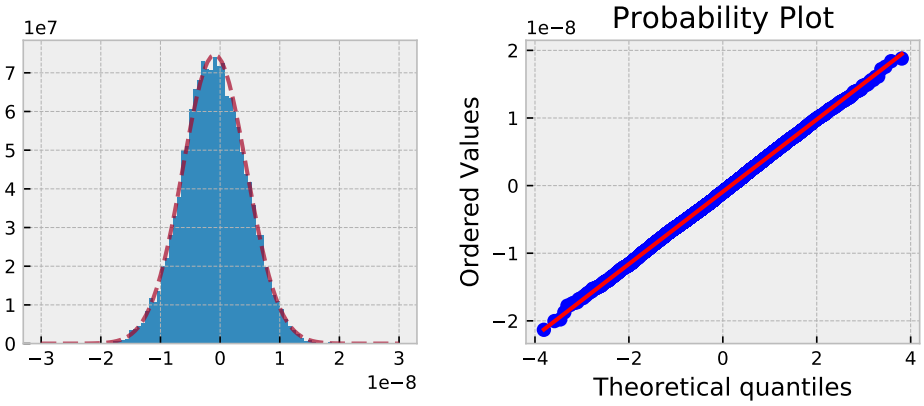

Figure 2: Normality of Cramer $X[0]$ sample

Using Stott Parker's formula (2) to compute $\hat{s}_{\text{MCA}}$ for $X[0]$, we get a figure close to the expected value 28.8:

$$\hat{s}_{\text{MCA}} = -\log_2 \left| \frac{\hat{\sigma}}{\hat{\mu}} \right| \approx 28.5. \tag{5}$$

But how confident are we that $\hat{s}_{\text{MCA}}$ is a good estimate of s_{MCA}? Could we have used a smaller number of samples and still get a reliable estimation of the results quality?

On the other hand, using these $n = 10\,000$ samples to compute the CESTAC lower bound defined in equation (1) with confidence 95 % gives

$$\hat{s}^{(10\,000)}_{\text{CESTAC}} = -\log_2 \left(\frac{\tau_n \, \hat{\sigma}}{\sqrt{n} \, |\hat{\mu}|} \right) \approx 34.2, \tag{6}$$

which is a clear overestimation of the quality of the IEEE result, but also of the CESTAC result, since the number of bits in common between the real result and the sample average is given by

$$s^{(10\,000)}_{\text{CESTAC}} = -\log_2 \left| \frac{x_{\text{real}}[0] - \hat{\mu}}{x_{\text{real}}[0]} \right| \approx 31.0.$$

Such a large n exhibits the bias between μ and $x_{\text{real}}[0]$, invalidating the CESTAC hypotheses. In practice, CESTAC implementations such as CADNA use $n = 3$, a choice of which the validity is discussed in section 4.3. For the Cramer benchmark, computing $\hat{s}_{\text{CESTAC}}$ for only 3 samples of X[0] yields a conservative estimate:

$$\hat{s}^{(3)}_{\text{CESTAC}} \approx 27.5 \leq s^{(3)}_{\text{CESTAC}} \approx 28.5.$$

[3]Interestingly $X[0]$ fails the Anderson-Darling test, 27 % p-value, due to some anomalies on the tail.

In the following, we present a novel probabilistic formulation to get a confidence interval for the number of significant bits with and without assumption of normality.

3 Probabilistic accuracy of a computation

We consider one output of a program performing FP operations as a random variable X. The output is a random variable either because the program is inherently nondeterministic or because we are artificially introducing numerical errors through MCA, CESTAC, or another stochastic arithmetic model. We want to study how the probabilistic properties of the computation impact its accuracy. The real distribution of X is unknown but we can approximate it with n samples, $X_1 \dots X_n$.

The accuracy of a result must be defined against a reference value. If the program is deterministic when executed in IEEE arithmetic, the IEEE result is one straightforward choice for the reference value. If the program is nondeterministic, one can also choose as reference, the empirical average of X. Finally a third option consists in computing the accuracy against a second random variable Y, which allows computing the accuracy between runs of the same program or allows finding the accuracy between two different programs, such as when comparing two different versions or implementations of an algorithm. We will write the reference value y when it is a constant and Y when it is another random variable.

Four types of studies can be led, depending on whether we are interested in absolute of relative error, and whether we have a reference value. For each study we can model the errors as a random variable Z defined as follows

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

We have reduced the four types of problems to study the probabilistic properties of Z whose error distribution represents the error of a computation in a broad sense. With no error, the expected result of Z is 0.

To define the significance of a digit we use Stott Parker's $\frac{1}{2}$ulsp algorithm [SP97, p. 20]. The significant bit is at the rightmost position at which the digits differ by less than one half unit in the last place. That is to say, two values x and y have s significant digits iff,

$$
\begin{aligned}
|x - y| &\leq \frac{1}{2} \times 2^{e_y - s} = 2^{-s+(e_y-1)} && \text{(absolute error)} \\
|x/y - 1| &\leq \frac{1}{2} \times 2^{1-s} = 2^{-s} && \text{(relative error)} \qquad (7)
\end{aligned}
$$

Without loss of generality, to unify the definition for the relative and absolute cases, in the following sections we assume $e_y = 1$. When working with abso-

lute errors, one should therefore shift the number of digits by, $(e_y - 1)$, the normalizing term[4].

The first quantity of interest is the **probability that the result is significant up to a given bit**. By generalizing equation 7 to random variables, we define the probability of the k-th digit being significant as $\mathbb{P}(|Z| \leq 2^{-k})$. For a given computation, the number of significant digits with probability p can be defined as the largest number s such that

$$\mathbb{P}\left(|Z| \leq 2^{-s}\right) \geq p. \tag{8}$$

The second quantity we will consider is the **probability that a given bit contributes to the precision of the result**: even if a bit on its left is already wrong, a bit can either improve the result precision, or deteriorate it. Because the expected result of Z is 0, a bit will improve the accuracy if it is 0 and deteriorate it if it is 1. Now, the k-th bit of Z is 0 if and only if there exists an integer i such that,

$$\begin{aligned} & \left\lfloor 2^k \left|Z\right| \right\rfloor = 2i \\ \Leftrightarrow \quad & 2i \leq 2^k \left|Z\right| < 2i + 1 \\ \Leftrightarrow \quad & 2^{-k}(2i) \leq |Z| < 2^{-k}(2i+1). \end{aligned} \tag{9}$$

In the following, we study these two quantities, **significant** and **contributing** bits, under the normality assumption (section 4) and in the general case (section 5).

4 Accuracy under the Centered Normality Hypothesis

In this section we consider that Z is a random variable with normal distribution $\mathcal{N}(0, \sigma)$. In practice, we only know an empirical standard deviation $\hat{\sigma}$, measured over n samples. Because Z is normal, the following confidence interval with confidence $1 - \alpha$ based on the χ^2 distribution with $(n-1)$ degrees of freedom is sound [Sap11, p. 282]:

$$\frac{(n-1)\hat{\sigma}^2}{\chi^2_{\alpha/2}} \leq \sigma^2 \leq \frac{(n-1)\hat{\sigma}^2}{\chi^2_{1-\alpha/2}}. \tag{10}$$

It is important to note that σ is the standard deviation of Z and not of X. For example, when taking a second random variable Y as reference. If X and Y both follow a distribution $\mathcal{N}(\mu, \sigma')$, $Z = X - Y$ follows $\mathcal{N}(0, \sqrt{2}\sigma')$.

[4] When Y is a random variable, we choose $e_Y = \lfloor \log_2 |E[Y]| \rfloor + 1$.

4.1 Significant bits

Theorem 1. *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.

Proof. The probability that the k-th bit is significant is $\mathbb{P}\left[|Z| \leq 2^{-k}\right] = \mathbb{P}\left[Z \leq 2^{-k}\right] - \mathbb{P}\left[Z \leq -2^{-k}\right]$. Now $\mathbb{P}\left[Z \leq -2^{-k}\right] = 1 - \mathbb{P}\left[Z \leq 2^{-k}\right]$ by symmetry of the normal distribution, so that $\mathbb{P}\left[|Z| \leq 2^{-k}\right] = 2\mathbb{P}\left[Z \leq 2^{-k}\right] - 1$. Therefore,

$$\mathbb{P}\left[|Z| \leq 2^{-k}\right] = 2\mathbb{P}\left[\frac{Z}{\sigma} \leq \frac{2^{-k}}{\sigma}\right] - 1 = 2F\left(\frac{2^{-k}}{\sigma}\right) - 1.$$

□

The number of significant digits with probability p is s such that $2F\left(\frac{2^{-s}}{\sigma}\right) - 1 = p$, *i.e.* $F\left(\frac{2^{-s}}{\sigma}\right) = \frac{p+1}{2} \Leftrightarrow \frac{2^{-s}}{\sigma} = F^{-1}\left(\frac{p+1}{2}\right)$, so that

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

The above formula is remarkable because, whatever σ, the confidence interval to reach a given probability is constant and can be computed from a table for F^{-1}. Therefore, one just needs to subtract a fixed number of bits from $-\log_2(\sigma)$ to reach a given probability, as illustrated in figure 3.

In practice, only the sampled standard deviation $\hat{\sigma}$ can be measured, but it can be used to bound σ thanks to the the χ^2 confidence interval in equation (10). This allows computing a sound lower bound $\hat{s}_{\text{CNH}}$ on the number of significant digits in the Centered Normality Hypothesis:

$$s \geqslant \underbrace{-\log_2(\hat{\sigma}) - \underbrace{\left[\frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right) + \log_2\left(F^{-1}\left(\frac{p+1}{2}\right)\right)\right]}_{\delta_{\text{CNH}}}}_{\hat{s}_{\text{CNH}}}. \tag{11}$$

Again, this formula is interesting since $\hat{s}_{\text{CNH}}$ can be determined by just measuring the sample standard deviation $\hat{\sigma}$ and shifting $-\log_2(\hat{\sigma})$ by a value δ_{CNH}, which only depends on a few parameters: the number of samples n, the confidence $1 - \alpha$ and the probability p. Some values for this shift are tabulated in appendix A, table 4.

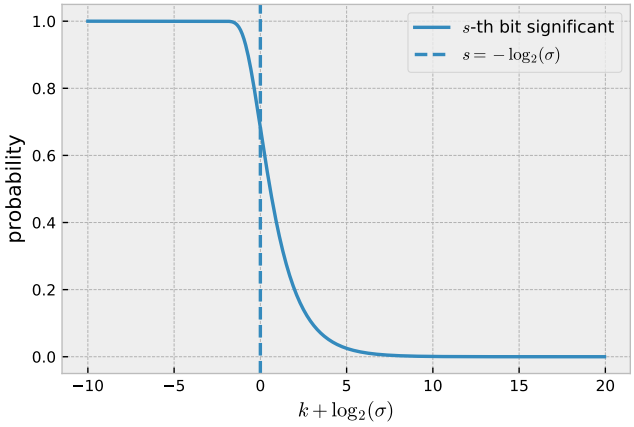

Figure 3: Profile of the significant bit curve: when the dashed line is positioned on the $-\log_2 \sigma$ abscissa, the curve corresponds to the probability that the result is significant up to a given bit.

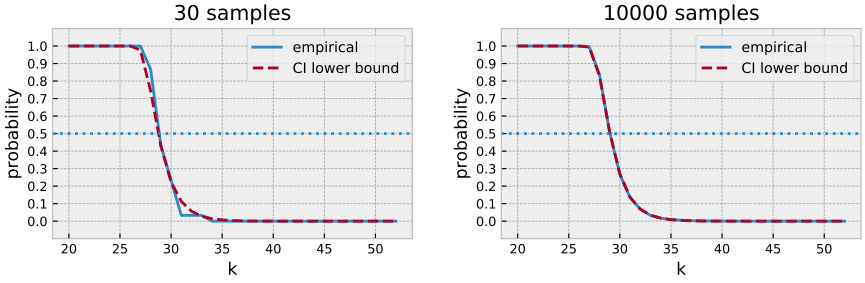

Figure 4: Significant bits for Cramer $x[0]$ variable computed under the normal hypothesis using 30 and 10000 samples. The Confidence Interval (CI) lower bound is computed by using the probability of theorem 1 and bounding σ with a 95% Chi-2 confidence interval.

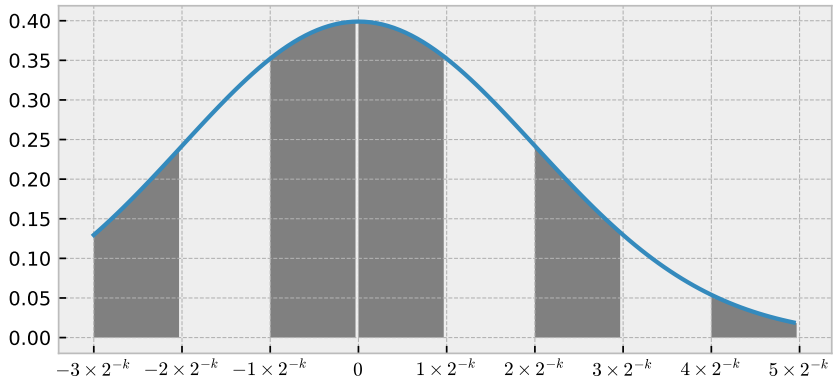

Figure 5: Normal curve; the gray zones correspond to the area where the k-th bit contributes to make the result closer to 0 (whatever the preceding digits).

Application Let us consider the $X[0]$ variable from the the ill-conditioned Cramer system from section 2.4. Normality tests show that it follows a normal distribution. Here we would like to compute the number of significant digits relative to the mean of the sample with a 99 % probability. Following section 3, we consider the relative error, $Z = \frac{X[0]}{\hat{\mu}} - 1 \to \mathcal{N}(0, \sigma)$. Here σ will be estimated from $\hat{\sigma}$ with the χ^2 95 % confidence interval presented in equation (10). Computing δ_{CNH} for $n = 10\,000$, $p = 0.99$ and $1 - \alpha = 0.95$ (or reading it in table 4), yields $\delta_{\text{CNH}} \approx 1.4$. Recalling the sampled measurements from section 2.4, we get $-\log_2(\hat{\sigma}) \approx 28.5$.

Therefore, at least $28.5 - 1.4 = 27.1$ bits are significant, with probability 99 % at a 95 % confidence level. Figure 4 shows that the proposed confidence interval closely matches the empirical probability on the $X[0]$ samples. When the number of samples increases, the confidence interval tightness increases.

4.2 Contributing Bits

In the previous section we computed the number of significant bits. Now we are interested in the number of contributing bits: even if a bit is after the last significant digit, it may still contribute partially to the accuracy if it brings the result closer to the reference value.

The theorem below gives an approximation of the number of contributing bits which has the same property as theorem 1: this approximation computes the number of bits to shift from $-\log_2(\hat{\sigma})$ to obtain the contributing bits based on the same few parameters (sample size n, confidence $1-\alpha$ and probability p), and the shift being independent of $\hat{\sigma}$.

Theorem 2. *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

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

Proof. As shown in equation 9, the k-th bit of Z contributes if and only if there exists an integer i such that $2^{-k}(2i) \leq |Z| < 2^{-k}(2i+1)$. For a normal centered Z distribution, this inequality correspond to the gray stripes in figure 5. Let us write the integral of one stripe as,

$$u_{(k,2i)} = P\left[2^{-k}(2i) \leq |Z| < 2^{-k}(2i+1)\right] = \int_{2^{-k}(2i)}^{2^{-k}(2i+1)} f(x)\,\mathrm{d}x,$$

where $f(x) = \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$ is the probability distribution function of $\mathcal{N}(0,\sigma)$. The probability of contribution for the k-th bit, p_k, is therefore

$$\begin{aligned}
& p_k = \sum_{i\in\mathbb{Z}} u_{(k,2i)} = 1 - \sum_{i\in\mathbb{Z}} u_{(k,2i+1)} \\
\Leftrightarrow \quad & 2p_k = 1 + \sum_{i\in\mathbb{Z}} u_{(k,2i)} - u_{(k,2i+1)} \\
\Leftrightarrow \quad & p_k = \frac{1}{2} + \sum_{i\geq 0} u_{(k,2i)} - u_{(k,2i+1)} = \frac{1}{2} + \sum_{i\geq 0} (-1)^i u_{(k,i)} \qquad \text{(by symmetry of } f\text{)}.
\end{aligned}$$

Now, according to the *trapezoidal rule*, there exists $\xi_{(k,i)}$ in the interval $I_k^i = [2^{-k}(i), 2^{-k}(i+1)]$ such that

$$u_{(k,i)} = \int_{2^{-k}i}^{2^{-k}(i+1)} f(x)\,\mathrm{d}x = 2^{-k}\left(\frac{f\left(2^{-k}i\right) + f\left(2^{-k}(i+1)\right)}{2}\right) + \frac{\left(2^{-k}\right)^3}{12} f''(\xi_{(k,i)}).$$

Introducing $v_{(k,i)} = 2^{-k}\left(\frac{f\left(2^{-k}i\right)+f\left(2^{-k}(i+1)\right)}{2}\right)$ and $w_{(k,i)} = \frac{\left(2^{-k}\right)^3}{12} f''(\xi_{(k,i)})$, we have $u_{(k,i)} = v_{(k,i)} + w_{(k,i)}$, and

$$p_k = \frac{1}{2} + \sum_{i\geq 0} (-1)^i v_{(k,i)} + \sum_{i\geq 0} (-1)^i w_{(k,i)}.$$

Now we compute the alternate series $\sum_{i\geq 0}(-1)^i v_{(k,i)}$. Since the series is alternate and its terms tend to 0, all term cancellations are sound. All trapezoidal terms cancel, except the first half-term:

$$\begin{aligned}\sum_{i\geq 0}(-1)^i v_{(k,i)} &= \sum_{i\geq 0}(-1)^i 2^{-k}\left(\frac{f\left(2^{-k}i\right)+f\left(2^{-k}(i+1)\right)}{2}\right)\\ &= 2^{-k-1}\sum_{i\geq 0}(-1)^i\left(f\left(2^{-k}i\right)+f\left(2^{-k}(i+1)\right)\right)\\ &= 2^{-k-1}f(0) = \frac{2^{-k}}{2\sigma\sqrt{2\pi}}.\end{aligned}$$

We can also sum the error terms $w_{(k,i)}$. We have:

$$\frac{\left(2^{-k}\right)^3}{12}\min_{\xi\in I_k^i} f''(\xi) \leq w_{(k,i)} \leq \frac{\left(2^{-k}\right)^3}{12}\max_{\xi\in I_k^i} f''(\xi). \tag{12}$$

The second derivative $f''(\xi) = \frac{(\xi^2/\sigma^4 - 1/\sigma^2)}{\sigma\sqrt{2\pi}}e^{-\frac{\xi^2}{2\sigma^2}}$ is increasing between 0 and $\sqrt{3}\sigma$ and decreasing from $\sqrt{3}\sigma$ onward. We distinguish four possible cases, based on the monotony of f (if f'' is increasing on I_k^i, $\min_{\xi\in I_k^i} f''(\xi) = f''(2^{-k}i)$, $\max_{\xi\in I_k^i} f''(\xi) = f''(2^{-k}(i+1))$; if it is decreasing, $\min_{\xi\in I_k^i} f''(\xi) = f''(2^{-k}(i+1))$, $\max_{\xi\in I_k^i} f''(\xi) = f''(2^{-k}i)$; and if $\sqrt{3}\sigma \in I_k^i$, $\max_{\xi\in I_k^i} f''(\xi) = f''(\sqrt{3}\sigma)$):

- **Case 1:** when $2^{-k}(2i+2) \leq \sqrt{3}\sigma$, then

$$\begin{aligned}\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i)) \leq v_{(k,2i)} \leq \frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+1))\\ -\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+2)) \leq -v_{(k,2i+1)} \leq -\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+1))\end{aligned}$$

- **Case 2:** when $2^{-k}(2i) \leq \sqrt{3}\sigma \leq 2^{-k}(2i+1)$ then

$$\begin{aligned}\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i)) \leq v_{(k,2i)} \leq \frac{\left(2^{-k}\right)^3}{12}f''(\sqrt{3}\sigma)\\ -\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+1)) \leq -v_{(k,2i+1)} \leq -\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+2))\end{aligned}$$

- **Case 3:** when $2^{-k}(2i+1) \leq \sqrt{3}\sigma \leq 2^{-k}(2i+2)$ then

$$\begin{aligned}\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i)) \leq v_{(k,2i)} \leq \frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+1))\\ -\frac{\left(2^{-k}\right)^3}{12}f''(\sqrt{3}\sigma) \leq -v_{(k,2i+1)} \leq -\frac{\left(2^{-k}\right)^3}{12}f''(2^{-k}(2i+1))\end{aligned}$$

- **Case 4:** when $2^{-k}(2i) \geq \sqrt{3}\sigma$, then

$$\frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i+1)) \leq v_{(k,2i)} \leq \frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i))$$
$$-\frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i+1)) \leq -v_{(k,2i+1)} \leq -\frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i+2))$$

The bounds on these error terms also cancel two by two, except the first one, and the two terms around $\sqrt{3}\sigma$ (we consider cases with $2^{-k} < \sqrt{3}\sigma \Leftrightarrow \frac{2^{-k}}{\sigma} < \sqrt{3}$, so that the maximum of f'' is not in the first stripe). At worst, $-\frac{\left(2^{-k}\right)^3}{12} f''(\sqrt{3}\sigma)$ and $-\frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i+1))$ with $2i+1 = \lceil \sqrt{3}\sigma \rceil$ remain on the right side, and $\frac{\left(2^{-k}\right)^3}{12} f''(\sqrt{3}\sigma)$ and $\frac{\left(2^{-k}\right)^3}{12} f''(2^{-k}(2i))$ with $2i = \lceil \sqrt{3}\sigma \rceil$ on the left (these worst cases cannot all occur at once).

All in all, we are left with (using that f'' reaches its maximum at $\sqrt{3}\sigma$):

$$\frac{\left(2^{-k}\right)^3}{12} f''(0) - 2\frac{\left(2^{-k}\right)^3}{12} f''(\sqrt{3}\sigma) \leq \sum_{i\geq 0} (-1)^i w_{(k,i)} \leq 2\frac{\left(2^{-k}\right)^3}{12} f''(\sqrt{3}\sigma)$$
$$-\frac{\left(2^{-k}\right)^3}{12\sigma^3\sqrt{2\pi}} - 2\frac{\left(2^{-k}\right)^3}{12}\frac{2}{\sigma^3\sqrt{2\pi}} e^{-3/2} \leq \sum_{i\geq 0} (-1)^i w_{(k,i)} \leq 2\frac{\left(2^{-k}\right)^3}{12}\frac{2}{\sigma^3\sqrt{2\pi}} e^{-3/2}$$

Finally:

$$\frac{1}{2} + \frac{2^{-k}}{2\sigma\sqrt{2\pi}} - \frac{\left(2^{-k}\right)^3 (4e^{-3/2}+1)}{12\sigma^3\sqrt{2\pi}} \leq p_k \leq \frac{1}{2} + \frac{2^{-k}}{2\sigma\sqrt{2\pi}} + \frac{\left(2^{-k}\right)^3 (4e^{-3/2})}{12\sigma^3\sqrt{2\pi}} \tag{13}$$

It follows that, when $\frac{2^{-k}}{\sigma}$ is small, bit k contributes to the result accuracy with probability

$$p_k \sim \frac{2^{-k}}{2\sigma\sqrt{2\pi}} + \frac{1}{2}. \tag{14}$$

□

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

$$c = -\log_2(\sigma) - \log_2(p - \frac{1}{2}) - \log_2(2\sqrt{2\pi}). \tag{15}$$

As above, this formula can be further refined by replacing σ with $\hat{\sigma}$ and adding a term taking into account the confidence level.

Figure 6 plots the approximation of equation 14. The approximation of equation 14 is tight for $k > -\log_2 \sigma$: in this case, the absolute error of the

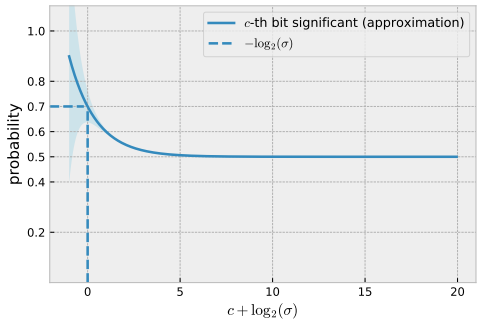

Figure 6: Profile of the contribution bit curve: when the dashed line is positioned on the $-\log_2 \sigma$ abscissa, the curve corresponds to the approximation 14 of the probability that the bit contributes to the result accuracy. The shaded area represents the bound on the error given by equation 13.

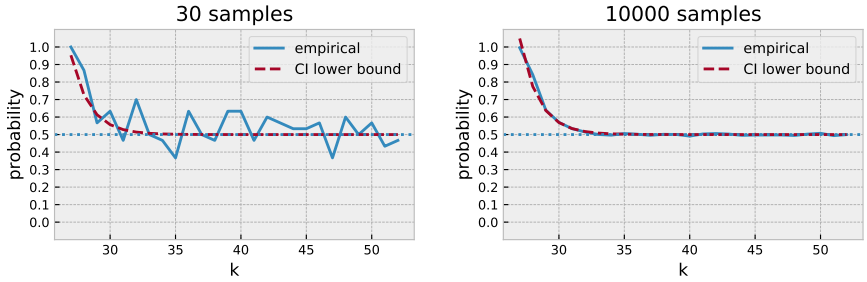

Figure 7: Contributing bits for Cramer $x[0]$ variable computed under the normal hypothesis using 30 and 10000 samples with the approximation of equation 14.

approximation formula is less than 2 %. The probability of contribution at $k = -\log_2 \sigma$ is 0.7. Therefore, equation 15 can be safely used for probabilities less than 0.7. In this paper, we want to find the limit after which bits are random noise. This limit corresponds to a probability of 0.5 and the approximation is tight for $p < 0.7$.

Application Figure 7 shows that the approximation proposed in this section tightly estimates the empirical samples in Cramer $x[0]$ example.

If we consider a 51 % threshold for the contribution of the bits we wish to keep, then we should keep $c = -\log_2(\sigma) - \log_2(p - \frac{1}{2}) - \log_2(2\sqrt{2\pi}) = -\log_2(\sigma) + 4.318108$. As in section 4.1, we estimate $-\log_2(\sigma)$ with a 95 % Chi-2 confidence interval, and compute $c = 32.8$.

This means that with probability 51% the first 32 bits of the mantissa will

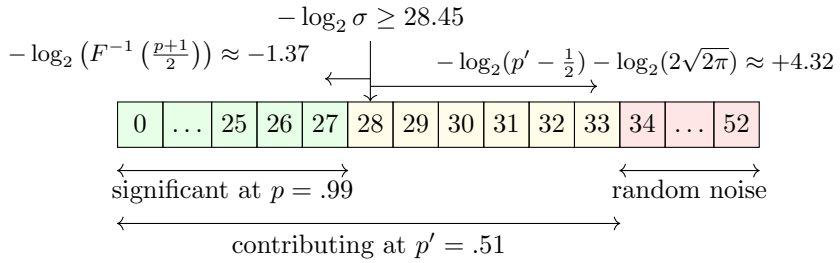

Figure 8: Summary of results on the $X[0]$ Cramer example. $28.45 \approx -\log_2 \hat{\sigma} - \frac{1}{2}\log_2\left(\frac{n-1}{\chi^2_{1-\alpha/2}}\right)$

round the result towards the correct reference value. After the 34th bit the chances of rounding correctly or incorrectly are even: the noise after the 34th bit is random and does not depend on the computation. Bits 34 onwards can be discarded.

4.3 Summary of results for a Normal Centered Distribution

Under the normality hypothesis, the quantity $-\log_2 \frac{\sigma_X}{|\mu_X|}$ introduced by Stott Parker is pivotal, but needs to be refined. In our framework, Stott Parker's definition maps to $Z = \frac{X}{|\mu_X|} - 1$, which computes the relative error to the mean. In this case Stott Parker's formula computes the position of the bit until which the result has 68 % chance of being significant, and that contributes to the result precision with a probability of 70 %.

However, from this bit, for each desired probability level, there is a simple way to compute a quantity by which to move back to be sure that the result is significant; and another easy to compute quantity by which to move forward to guarantee that all bits contributing more than a fixed level are kept. Figure 8 demonstrates this on the Cramer's example.

First a lower bound, 28.45, on $-\log_2 \sigma$ is computed with the Chi-2 95 % confidence interval. This is the number of significant bits computed with Stott Parker's definition, it is exact with 68 % probability. To compute a lower bound on bits that are significant with probability 99 %, we simply subtract 1.37 from this number. By adding 4.32 to this number we get the number of bits that contribute or round towards the reference with a probability of 51 %. The remaining bits in the mantissa are considered random noise and do not depend on the computation.

With $k = -\log_2 \sigma$, the result should be given: for an absolute error with $\lceil k + (e_y - 1) + 4.318108 \rceil$ bits with the annotation $\pm 2^{\lfloor k + (e_y - 1) - 1.365037 \rfloor}$ at 99 %; for a relative error with $\lceil k + 4.318108 \rceil$ and $\pm 2^{\lfloor k - 1.365037 \rfloor} \times y$ at 99 % for a relative error. In this notation, only digits that are likely to round correctly

the final result with a probability greater than 1 % are written; the error at probability 99 % is written. In decimal, this notation takes up to two additional digits ($4.318108 \times \log_{10} 2 \approx 1.29$ digits) that are probably wrong, but still have a chance to contribute to the result precision. As an example, using this notation to display the IEEE-754 result of Cramer's $X[0]$ yields

$$1.999999996 \pm 1.4\text{e-}08.$$

These 10 digits contain all the valuable information in the result, and are the only ones that it would make sense to save, for example in a checkpoint-restart scheme.

Interestingly enough, the CESTAC definition of significance can be reinterpreted in this statistical framework. Equation 1 defines the CESTAC number of significant bits as,

$$\begin{aligned} \hat{s}_{\text{CESTAC}} &= \log_2 \left(\frac{\tau_n \, \hat{\sigma}_X}{\sqrt{n} \, |\hat{\mu}_X|} \right) \\ &= \log_2 \hat{\sigma} - \log_2 \frac{\tau_n}{\sqrt{n}} \quad \left(\text{taking } Z = \frac{X}{|\mu_X|} - 1\right) \end{aligned}$$

We showed previously that the formula tends to infinity when increasing the number of samples n. Yet CADNA [LCJ10], the most popular library implementing CESTAC, fixes $n = 3$. In this case, with a confidence of 95 %,

$$\hat{s}_{\text{CADNA}} = \log_2 \hat{\sigma} - 1.3128.$$

Reinterpreting this formula with theorem 1 and figure 3, we see that the significant digits reported by CADNA with three samples are correct in the normal centered case with probability 98.7% under the strong assumption that $\log_2 \sigma$ is well estimated by a sample of size 3. With only three samples σ estimation could be biased. Reinterpreting the -1.3128 shift as the δ_{CNH} term, which accounts for the estimation bias, gives us a probability of 30.8% (with 95% confidence) for the CADNA estimator being correct.

5 Accuracy in the General Case

The hypothesis that the distribution Z is normal is not always true. We propose statistical tools to study the significance of bits as well as their contribution to the result accuracy that do not rely on any assumption regarding the distribution of the results.

To address the problem in the general case we reframe it in the context of Bernoulli estimation, which is interesting because:

- it does not rely on any assumptions on the distribution of Z;
- it provides a strong confidence interval for determining the number of significant digits when using stochastic arithmetic methods;

- thanks to a more conservative bound, it allows to estimate *a priori* in all cases for a given probability and confidence a safe number of sample to draw from the Monte Carlo experiment.

5.1 Background on Bernoulli estimation

In the next section, we restate the problem of estimating the number of significant bits as a series of estimations of Bernoulli parameters. We present here some basic results on such estimations.

Consider a sequence of independent identically distributed Bernoulli experiments with an unknown parameter p and outcomes (p_i). Each value of the parameter p gives a model of this experiment, and, among them, we will only keep an interval of model parameters under which the probability of the given observation is greater than α. The set of possible values for p will then be called a confidence interval of level $1-\alpha$ for p [5]: if the actual value of p is not in the confidence interval computed from the outcome, it means that the observed outcome was an "accident" the probability of which is less than α.

A case of particular interest in our study is the one when all experiments succeed. Then, the probability of this outcome is p^n under the model that the Bernoulli parameter value is p. We then reject models (i.e., values of p) such that $p^n < \alpha \Leftrightarrow n\ln(p) < \ln(\alpha)$. Now, $\ln(p) \leq p-1$ and $\ln(p) \sim p-1$ is a first order approximation when p is close to 1. Thus, taking $p < 1 + \frac{\ln(\alpha)}{n}$ leads to a probability of the observation less than α, and one can reject these values of p. In particular, taking $1-\alpha = 95\ \%$ (and so $\ln(\alpha) = \ln(.05) = -2.996$), we keep values of p greater than $1-\frac{3}{n}$, and $\left[1-\frac{3}{n}, 1\right]$ is a 95 % confidence interval. This result is known in clinical trial's literature as the *Rule of Three* [ELKT95]. Vice versa, in an experiment with no negative outcome, one can conclude with confidence $1-\alpha$ that the probability of a positive outcome is greater than p after $\left\lceil \frac{\ln(\alpha)}{\ln(p)} \right\rceil$ positive trials.

The general case can be dealt by using the Central Limit Theorem, which shows that for a number n of experiments large enough (with respect to to $\widehat{p} = \frac{1}{n}\sum p_i$), $\sqrt{n}(\widehat{p}-p)/(\widehat{p}(1-\widehat{p}))$ is close to a Gaussian random variable with law $\mathcal{N}(0,1)$. This approximation is generally considered good enough when $n\widehat{p} \geq 5$ and $n(1-\widehat{p}) \geq 5$, *i.e.* when at least 5 trials are successes and 5 are failures. Then, with F the cumulative function of $\mathcal{N}(0,1)$,

$$\left[\widehat{p} - \sqrt{\widehat{p}(1-\widehat{p})/n}F^{-1}(1-\alpha/2), \widehat{p} + \sqrt{\widehat{p}(1-\widehat{p})/n}F^{-1}(1-\alpha/2)\right]$$

is an $1-\alpha$ confidence interval for p. If we focus on a lower bound on the parameter p, we can also use $\left[\widehat{p} - \sqrt{\widehat{p}(1-\widehat{p})/n}F^{-1}(1-\alpha), 1\right]$ as a confidence interval of level $1-\alpha$. For $1-\alpha = 95\ \%$, $F^{-1}(1-\alpha/2) \approx 1.96$ and $F^{-1}(1-\alpha) \approx 1.65$. This approximation is known to be unfit in many cases, and can be

[5] $1-\alpha$ is not the "probability" that the real value of the parameter is in the interval, as this parameter is not a random variable

improved by considering $\tilde{p} = \frac{1}{n+4}(\sum p_i + 2)$ rather than $\hat{p}$ as shown by Brown et al. [BCD01].

Thus, from n independent experiments, of which n_s have been a success, we can affirm with confidence 95 % that the probability of success is greater than $\frac{n_s+2}{n+4} - 1.65\sqrt{\frac{(n_s+2)(n-n_s+2)}{(n+4)^3}}$. We can note that when $n_s = n$, this confidence interval is valid, but much more conservative than the one obtained above, that can thus be preferred in this particular case.

5.2 Statistical formulation as Bernoulli trials

Now, for each of the four discussed settings, presented in section 3, we can form two series of Bernoulli trials based on collected data.

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

When the reference is a constant x, we consider n samples X_i. We form N pieces of data by computing $Z_i = X_i - x$ or $Z_i = X_i/x - 1$ respectively.

When the reference is another random variable Y, we form N pieces of data by computing $Z_i = X_i - Y_i$ or $Z_i = X_i/Y_i - 1$. In the case where $X = Y$ and we study the distance between samples of a random process, this requires $2N$ samples from X.

From these N pieces of data, we form Bernoulli trials by counting the number of success of

$$S_i^k = \text{“}|Z_i| \leq 2^{-k}\text{”}$$

for studying k-th bit significance, and

$$C_i^k = \text{“}\lfloor 2^k\,|Z_i| \rfloor \text{ is even”}$$

for studying k-th bit contribution.

From these two Bernoulli samples, the estimation can be made as above to determine the probability that the k-th bit is significant and the probability that it contributes to the result, for any k. The result can then be plotted as two probability plots, one for significance, the other for the contribution. The significance plot is non-increasing by construction, should start at 1 if at least one bit can be trusted, and tends to 0. The contribution plot should tend to $\frac{1}{2}$ in most cases, since the last digits are pure noise and are not affected by the computation.

5.3 Evaluation

The main goal of the Bernoulli formulation is to deal with non normal distributions. In this section, we evaluate the Bernoulli estimate on Cramer's $X[0]$ samples which follow a normal distribution. This is to keep a consistent example

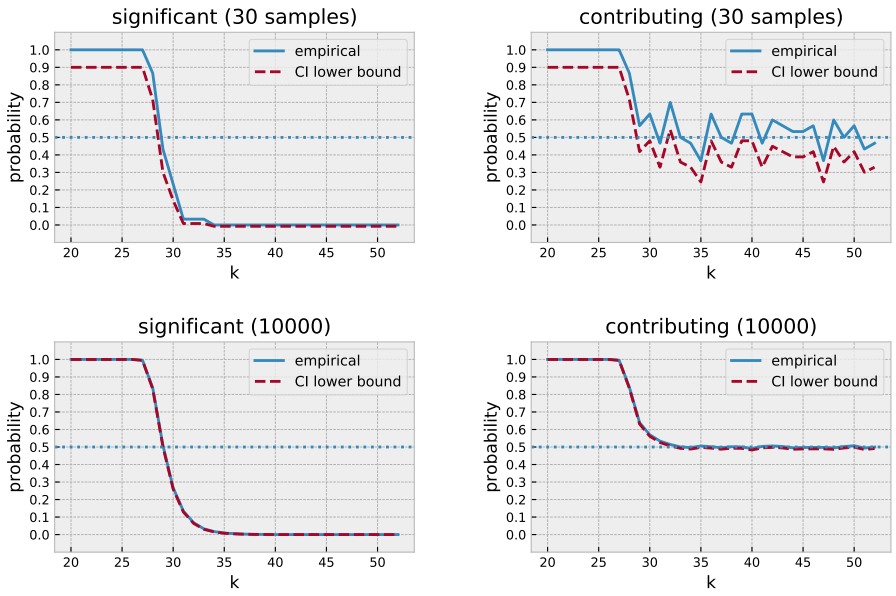

Figure 9: Significance and contribution per bit for variable $X[0]$ of the Cramer's system with 30 and 10000 samples.

across the whole paper and to compare the results with the Normal formulation estimates. Later, in section 6, we will apply the Bernoulli estimate to distributions produced by the industrial simulation codes EuroPlexus and code_aster, some of which are not normal.

Figure 9 plots the significance and the contribution per bit probabilities for $X[0]$ using the Bernoulli estimation. The estimation closely matches the empirical results. It is interesting to compare the Bernoulli estimates with 30 samples to the Normal estimates in figures 4 and 7. The Bernoulli estimates are less tight but more conservative, this is expected since they do not build upon the normality assumption of the distribution.

If we are only interested in the number of significant digits, we can consider the Bernoulli trial with no failed outcomes since it provides an easy formula giving the required number of samples. In this case, the number of needed samples is $n = \lceil -\frac{\ln \alpha}{\ln p} \rceil$. We then determine the maximal index k for which the first k bits of all n sampled results coincide with the reference:

$$\hat{s}_{\mathrm{B}} = \max \left\{ k \in [\![1, 53]\!] \text{ such that } \forall i \in [\![1, n]\!], S_i^k \text{ is true} \right\}. \tag{16}$$

We applied this method to the $X[0]$ sample from section 2.4. Assuming a $(1 - \alpha) = 95\%$ confidence interval and a probability of $p = 99\%$ of getting s significant digits, we gather $n = 299$ samples. Among the collected samples, the 27^{th} digit is sometimes different compared with the reference solution, but the

first 26 digits coincide for all samples. Therefore we conclude with probability 99% and 95% confidence, that the first $\hat{s}_{\mathrm{B}} = 26$ binary digits are significant.

6 Experiments on industrial use-cases

6.1 Reproducibility analysis in the Europlexus Simulation Software

In this section we show how to apply our methodology to study the numerical reproducibility of the state-of-the-art Europlexus [Eur18] simulator. Europlexus is a fast transient dynamic simulation software co-developed by the french Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA), the Joint Research Center (JRC) of the European Commission, and other industrial and academic partners. While its first lines of code date back to the mid seventies, the current source code has grown to about 1 million lines of Fortran 77 and Fortran 90. Europlexus runs in parallel on distributed memory architectures through a domain decomposition strategy, and on shared memory architectures through loop parallelism.

Europlexus has two main fields of application: simulation of severe accidents in nuclear reactors to check the soundness of the mechanical confinement barriers of the radioactive matters for the CEA; and simulation of explosions in public places in order to measure their impact on the surrounding citizens and structures for the JRC.

It handles several non-linearities, geometric or material, some of which lead to a loss of unicity of the evolution problem considered. This is for example the case for some configurations with frictional contact between structures, or when the loadings cause fracture and fragmentation of the matter. Another obvious source of bifurcations of the dynamical system is the dynamic buckling.

Due to the small errors introduced by the floating point arithmetic, the introduction of parallel processing in Europlexus raises a difficulty for the developer and the users: the solutions of a given simulation may differ when changing the number of processors used for the computation. We show here how the confidence intervals proposed in this paper help the developper to design relevant non-regression tests. To this end, we study in the following a simple case which could serve as a non-regression test, and which is symptomatic of a non-reproductibility related to FP arithmetic. It involves a vertical doubly clamped column to top and bottom plates. A vertical pressure is applyed by lowering the top plate, which causes buckling of the column. The column is modeled as a set of discrete elements (here segments) connected at moving points called nodes.

The left plot in figure 10 shows the result after 300 simulation timesteps with the out-of-the-box Europlexus software using standard IEEE arithmetic. The sequential result is deterministic and does not change when run multiple times. We wish to study how the simulation is affected by small numerical errors.

We run the same simulations but this time using the Verificarlo [DdOCP16] compiler to introduce MCA randomized floating point errors with a precision

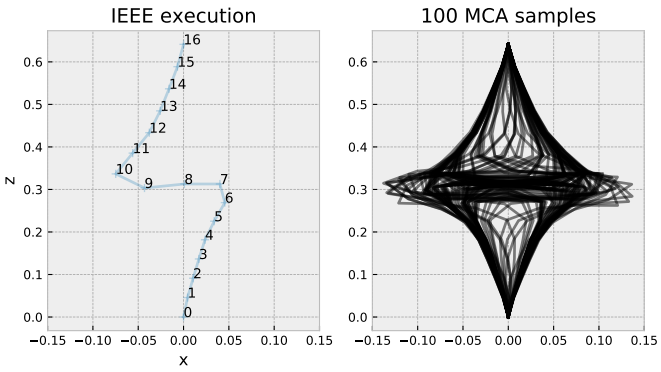

Figure 10: Europlexus buckling simulation of a doubly clamped column subject to a vertical pressure. The nodes of the column are labeled in the plot from 0 to 16. The left plot shows the deterministic and reproducible results produced by an IEEE-754 run of the simulation. In the right plot, a two-digits numerical error is simulated by collecting one hundred MCA samples with Verificarlo ($t = 50$). The buckling direction is completely dominated by the small numerical error introduced.

of $t = 50$. The cost to instrument the whole Europlexus software and its accompanying mathematical libraries was low. In particular no change to the source-code was necessary thanks to the transparent approach to instrumentation of Verificarlo, only the build system had to be configured to use the Verificarlo compiler.

The right plot in figure 10 shows the result of one hundred Verificarlo executions. The direction of the buckling is chaotic and completely dominated by the small FP errors introduced. This is not surprising as the buckling direction is physically unstable.

When parallelizing Europlexus or making changes to the code, it is important to check that there are no regression on standard benchmarks. Changing the order of the floating point operations may introduce small rounding errors. As we just saw, even small numerical errors change the buckling direction. Is it possible to use such a benchmark in non-regression tests?

The column is modeled as a set of discrete elements connected by nodes. The distribution on the x-axis is normal but whatever the node there are no significant digits among the samples. The variation between samples is strong on the x-axis, the x position clearly cannot be used as a regression test.

The distribution along the z axis is more interesting as it is non normal for all the nodes. Figure 11 shows the quantile-quantile plot for node 1 (Shapiro-Wilk rejects normality with $W = 0.9$ and $p = 1.8e-06$). Because the distribution on the z axis is non normal we should apply the Bernoulli significant bits estimator. In this study, we measure the number of significant digits considering the relative error against the sample mean, so $Z = \frac{X}{\mu_X} - 1$.

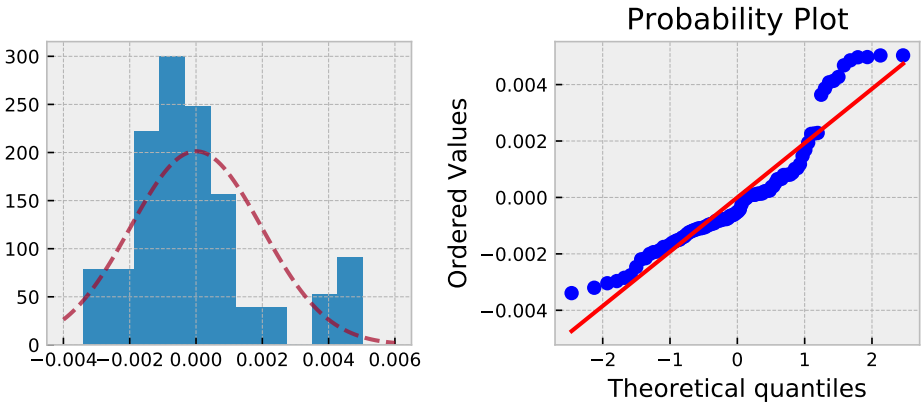

Figure 11: Non normality of buckling samples on z axis and node 1. Shapiro Wilk rejects the normality hypothesis.

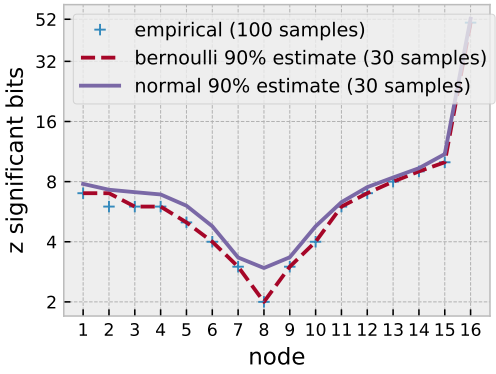

Figure 12: Significant bits on the z axis distribution. Bernoulli estimation captures precisely the behavior (except for node 2). Normal formula overestimates the number of digits, this is expected since the distribution is strongly non normal.

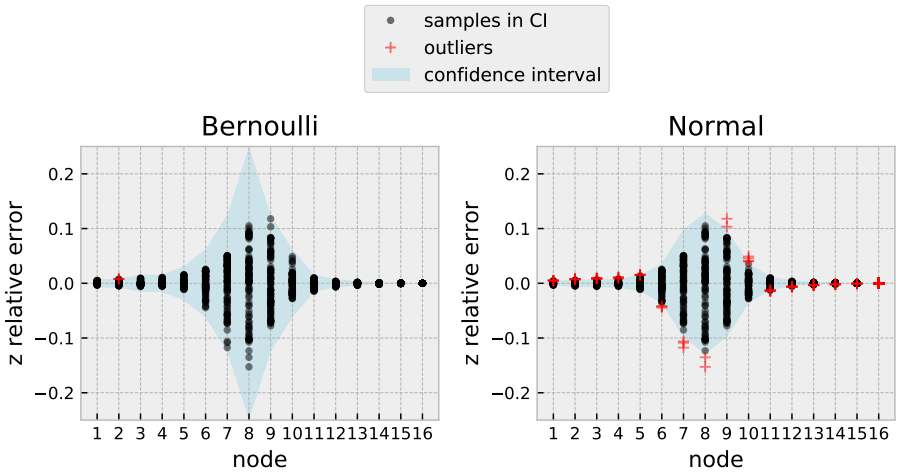

Figure 13: Relative error between the samples and the mean of the z-axis distribution. The blue envelope corresponds to the computed confidence interval with 30 samples. Black dots are samples that fall inside the CI. Red crosses are outliers that fall outside the CI. In the Bernoulli case, only 3 samples out of 70 fall outside of the interval; which is compatible with the 90% probability threshold.

To test the robustness of the proposed confidence interval, we computed the Bernoulli estimate on the first 30 samples of the distribution. This corresponds to a probability of 90% with a confidence of 95%. We also computed the Normal estimate on the first 30 samples with the same probability and confidence.

Figure 12 compares the estimates to the empirical distribution observed on 100 samples. The Bernoulli estimate on 30 samples is fairly precise and accurately predicts the number of significant bits (except for node 2). The clamped node 16 has a fixed position and therefore all its digits are significant. The other nodes have between 2 and 10 significant digits depending on their position.

Figure 13 shows the expected relative error on each node. We see that the Bernoulli estimate is robust and only mispredicts the error on three samples of node 2. On the other hand, as expected, the Normal formula is not a good fit in this case due to the strong non normality of the distribution: the normal estimate is too optimistic and fails to capture the variability of the distribution.

The previous experiments show that the x-axis has no significant digits and that the z-axis distribution has between 2 and 10 significant digits. For example node 6 has 4 significant digits on the z-axis. Therefore, if the practitioner uses the z-axis position in this benchmark as a regression test, she should expect the first four digits of the mantissa to match in 90 % of the runs. If the error is higher than that then a numerical bug has probably been introduced in the code.

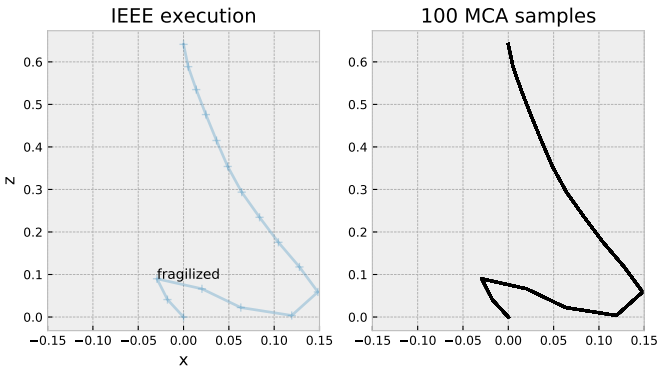

Figure 14: Europlexus buckling simulation with a fragilized node 2. By weakening the column, the physical process becomes reproducible in the presence of small numerical noise.

Another possibility for the practitioner is to adapt slightly the benchmark to make it more robust to numerical noise so it can be used in regression tests. For exemple, we can introduce a small perturbation in the numerical model by slightly moving the node 2 on the x-axis. Then the buckling is expected to always occur in the same direction. Figure 14 shows what happens when node 2 is slightly displaced: the buckling becomes deterministic and robust to numerical noise: 51 bits are significant for the x-axis and z-axis samples with probability 90 %; the two bits of precision lost correspond to the stochastic noise introduced. In this case, stochastic methods allow checking that the benchmark has become deterministic and assessing its resilience to noise.

6.2 Verification of a numerical stability improvements in code_aster

code_aster [Cod18] is an open source scientific computation code dedicated to the simulation of structural mechanics and thermo-mechanics. It has been actively developed since 1989, mainly by the R&D division of EDF, a French electricity utility. It uses the finite elements method to solve models coming from the continuum mechanics theory, and can be used to perform simulations in a wide variety of physical fields such as mechanics, thermal physics, acoustics, seismology and others. code_aster has a very large source code base, with more than 1 500 000 lines of code. It also uses numerous third-party software, such as linear solvers or mesh manipulation tools. Its development team has been dedicated to code quality for a long time, and has accumulated several hundreds of test cases which are run daily as part of the Verification & Validation process.

In a previous study [FL17] of code_aster, the Verrou tool was used to assess the numerical quality of quantities tested in the non-regression database. The

```
if (a.eq.b) then
  area = a
else
  area = (b-a) / (log(b)-log(a))
endif
```

Figure 15: Unstable branch detected in code_aster

```
if (abs(a-b).lt.tol* min(abs(a),abs(b))) then
  area = a
else
  r = a / b
  area = a * (r-1) / log(r)
endif
```

Figure 16: Unstable branch corrected

error localization features of Verrou were used to find the origin of errors in the source code, and improvements were proposed. In this section, we first summarize the results of this study, before using the new estimators described in this paper to confidently assess the benefits of the proposed corrections.

The study focuses on one test case named `sdnl112a` [Ado16], which computes 6 quantities related to the vibrations of steam generator tubes in nuclear reactors. These quantities will be denoted here as a, b, c, d, e and f. In the original implementation of code_aster – which will be referred to as `version0` in the following – the test case successfully completes on some hardware architectures, and fails on others. In this case, failing means that some quantities are computed with relative discrepancies larger than 10^{-6} with respect to some reference values. Using the Verrou tool to assess the numerical quality of these results fails: some runs perturbated by random rounding bail out without producing results. This exhibits somewhat severe instabilities, but does not allow quantifying their impact on the results accuracy.

The methodology then proceeds to finding the origin of such instabilities using Verrou. In a first stage, a test coverage comparison between two samples uncovers an unstable branch which is reproduced, after simplification, in figure 15. A reformulation of the incriminated computation (cf. figure 16) is proposed, which leads to a first corrected implementation, which will be referred to as `version1`. This version still does not pass automated non-regression testing. However, this time, assessing the stability of the results using Verrou and the standard MCA estimator (with 6 samples) yields meaningful results. Quantity e is evaluated to be the most problematic, with only 5 reliable significant decimal digits (19 bits), when users expect at least 6. Other computed quantities seem to meet the expected precision. For example, 9 decimal digits (30 bits) are estimated to be reliably computed for quantity a. In the following, we will use notations consistent with the rest of this paper, the computed result x being either quantity a or e.

In a second stage, the delta-debugging feature of Verrou is used to locate the origin of remaining instabilities in `version1`. The delta-debugging pinpoints the dot product in a sparse matrix vector product (routine `mrmmvr`). A first approach to mitigate the loss of precision, implemented in `version2`, consists in using a compensated summation algorithm in the dot product: algorithm `sum2` from [ORO05]. In `version3`, a fully compensated dot product algorithm is implemented: algorithm `dot2` from [ORO05].

Implementation	$\hat{s}_{\text{MCA}}$		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 1: Summary of the numerical quality assessment of 4 versions of code_aster, using Verrou and the standard MCA estimator with 6 samples.

Still using the standard MCA estimator, the study concludes that `version2` slightly improves results precision: respectively 30 and 19 reliable bits for quantities a and e). Only `version3` meets user thresholds, with respectively 32 and 21 reliable bits for a and e. This version is also the only one to pass automated non-regression and verification tests.

These different variants of code_aster, along with the estimated quality of their results, is summarized in table 1.

Therefore, `version3` should be considered as a good candidate to fix the implementation of code_aster. However, this raises many questions. Are we confident enough that this version really produces more accurate results? Should results produced by "older" versions of code_aster be considered invalid?

While 6 samples were enough for the purpose of debugging code_aster, and in light of what is at stake with a change in the actual implementation of code_aster, a higher degree of confidence should be required here.

Following the rules of Bernoulli experiments without failure and choosing $p = 1 - \alpha = 0.995$, the required number of samples is $N = 1058$. Table 2 reports accuracy estimations of the three versions of code_aster for this new number of samples. Four estimators are compared:

1. $\hat{s}_{\text{B}}^{\hat{\mu}}$: estimator based on Bernoulli experiments without failure, with a reference result taken to be the average of all samples.

2. $\hat{s}_{\text{B}}^{\text{IEEE}}$: estimator similar to the previous one, but taking the IEEE computation as reference.

3. $\hat{s}_{\text{CNH}}$: estimator based on the Centered Normal Hypothesis. $\hat{\mu}$ is taken as the reference to satisfy the centered hypothesis. This estimator should be used together with a normality test, such as Shapiro-Wilk, of which the p-value is displayed in parentheses.

4. $\hat{s}_{\text{MCA}}$: standard MCA estimator.

With the 4 estimators, `version3` is more accurate for both variables. As the normality test sometimes fails, this version can be selected only based on $\hat{s}_{\text{B}}^{\hat{\mu}}$ and $\hat{s}_{\text{B}}^{\text{IEEE}}$, which give similar results.

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

When results follow a Gaussian distribution[6], the estimate provided by $\hat{s}_{\mathrm{CNH}}$ is slightly less conservative than $\hat{s}_{\mathrm{b}}$ while remaining, by construction, sound and more conservative than $\hat{s}_{\mathrm{MCA}}$.

It is interesting to note here that the Bernoulli indicator $\hat{s}_{\mathrm{B}}$ is by definition an integer, which can somewhat limiting. Looking for example at $\hat{s}_{\mathrm{B}}^{\hat{\mu}}$ for quantity e, as computed by `version2` and `version3` and reported in Table 2b, is the difference between 18 and 19 bits really significant, or does it only come from definition (16) restricting $\hat{s}_{\mathrm{B}}$ to integer values? Using the centered normality hypothesis estimator, the accuracy improvement between `version2` and `version3` can be estimated to 1.66 bits. It is also possible to generalize definition (16) to fractional values of k. In this example, this would yield estimated numbers of significant bits of 18.16 for `version2` and 19.75 for `version3` respectively, again estimating the gain in accuracy to approximately 1.6 bits.

In any case, the results produced by `version3` are confidently estimated to be computed with 20 reliable bits for quantity e (*i.e.* relative error in the order of $2^{-20} \approx 9.5 \times 10^{-7}$), which now satisfies the user requirement of 6 decimal digits (although barely). Quantity a is estimated to be computed with at least 31 reliable bits, or approximately 4.7×10^{-10} relative error. And it is also safe to conclude that this implementation is the most robust among all three tested version. `version3` can thus be safely introduced in the code_aster code base.

One could also wonder, in retrospect, whether results produced by the IEEE execution of the original version of code_aster (`version0`) were valid. The comparison of IEEE results produced by `version0` and `version3` yields relative

[6]The normality of `version1` results might be subject to caution, but `version2` and `version3` clearly produce normal results, with Shapiro-Wilk p-values significantly higher than 0.1.

discrepancies of 4.29×10^{-10} and 9.84×10^{-7} respectively for quantities a and e. These discrepancies have the same order of magnitude as the uncertainties estimated above, an observation which does not invalidate the IEEE results produced by `version0`.

7 Limitations and Future Works

Stochastic arithmetic methods like MCA or CESTAC suffer from some limitations. Some of these are addressed by the current work; some others are not and should therefore be addressed in future work.

First, stochastic arithmetic methods provide accuracy assessments which are valid only for the specific set of input data that were used at run-time. Similarly to other run-time based analysis, the robustness of the conclusions can only be obtained through the multiplication of test cases to maximize the coverage of the analysis.

One way to overcome this limitation consists in using formal tools based on the static analysis of the source code. The aim of such tools is to prove that a given program follows its specifications for all possible input data in its admissible range. However, although much progress has been done in this area over the last few years [BCF+13], the use of such methods is, to the authors' knowledge, limited to small programs, especially in the field of scientific computing codes.

Second, even when it is valid only for a given input dataset, a quality assessment can be guaranteed or not. Interval Arithmetic [Rum10] is an example of guaranteed run-time analysis method, which provides a sound interval for the result of a floating point computation. However, the use of such a method is not always tractable, especially for complex programs with data dependent control paths. In that case the solution rapidly diverge to dramatic overestimation of the error and therefore does not bring useful information.

On the other hand, stochastic arithmetic methods model round-off and cancellation errors with a Monte Carlo simulation. Stochastic methods scale well to complex programs and do not suffer from the intractability problems of other. However their estimates are based on a limited number of random samples for each particular use-case input data. Therefore, as shown in this paper, stochastic methods do not provide formal guarantees but statistical confidence and probability.

Finally, there remains an important limitation to stochastic arithmetic methods, even using our methodology: the practical usefulness of quality assessments provided by stochastic methods relies on the implied hypothesis that MCA or CESTAC correctly model the mechanisms causing accuracy losses in industrial calculations. Indeed, practical objectives of numerical verification include for example:

- assessing the robustness of a scientific computation code with respect to uncontrolled changes in its run-time environment, such as changes in the threads and/or processes scheduling for concurrent programs, or

- assessing the accumulated loss of accuracy due floating-point arithmetic and affecting a computed result.

If a stochastic arithmetic method fails to reproduce the effect of floating-point round-offs, threads scheduling changes or other important factors affecting the accuracy of the computed result, then the quality assessment will be affected not only by *sampling* errors as seen above, but also by *model* errors. CESTAC and MCA have been designed to model realistically the round-off and cancellation errors in IEEE-754 floating-point arithmetic, and in practice they seem to correctly simulate floating-points bugs in many cases. However, it is possible to find corner cases where CESTAC or MCA diverge from the IEEE-754 computation, producing results which can be more accurate [CV93] or less accurate [DdOCP16].

Since the confidence intervals introduced in this paper tackle sampling errors, the next step consists in preventing and detecting model errors. This requires in depth understanding of the limitations and hypotheses of each stochastic arithmetic model, documentation informing the practitioner of these hypotheses and, when possible, automated run-time checks raising warnings when these hypotheses are not met in a particular calculation. Some of the stochastic arithmetic software tools, such as CADNA, implement runtime checks that detect some cases violating the hypotheses of the model [JC08].

In the meantime, model errors can be mitigated with simple sanity checks. For example, taking the IEEE-754 computation of the program as a reference value is a good sanity check. Indeed, if the stochastic model diverges from the IEEE-754 computation, the large error to the reference value will raise a red flag.

8 Conclusion

This paper introduces a methodology to compute confidence intervals in stochastic arithmetic computations. Numerical debugging tools such as Verificarlo, Verrou or Cadna use stochastic arithmetic samples to find and help fix numerical bugs. Thanks to the confidence intervals, such tools can provide a confidence probability when estimating the number of significant digits of a computation. This paper also introduces a novel measure for the precision of a result, the number of contributing digits, after which digits are random noise. This metric can help choosing how many digits to keep when forwarding a result to another tool or storing a result during a checkpoint.

The results of this paper are applicable to both centered normal distribution and general distributions. In the case of centered normal distributions, we show that to reach a given confidence level it is enough to apply a constant shift to the number of significant digits computed using the standard MCA estimator. In the case of general distribution, we provide a simple formula to decide how many samples are needed to reach a given confidence level. Appendix A provides

tables for the shifts and number of samples in the centered normal and general cases.

We demonstrate the applicability of the methodology to the two state of the art EuroPlexus and code_aster engines. In the EuroPlexus test case we show how the confidence intervals help to decide the numerical reproducibility. In the code_aster test case we show how the confidence intervals help to compare different versions of the code to select the most numerically robust.

An accompanying Jupyter notebook [7] provides a reference Python implementation that can be used by the practitioner to compute the confidence intervals from this paper in the normal and Bernoulli cases.

Acknowledgements

The authors thank Yohan Chatelain for helpful reviews and feedbacks when writing this paper.

References

[Ado16] André Adobes. Code_Aster : Sdnl112. https://www.code-aster.org/V2/doc/v14/en/man_v/v5/v5.02.112.pdf, 2016.

[BCD01] Lawrence D Brown, T Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical science*, pages 101–117, 2001.

[BCF+13] Sylvie Boldo, François Clément, Jean-Christophe Filliâtre, Micaela Mayero, Guillaume Melquiond, and Pierre Weis. Wave Equation Numerical Resolution: a Comprehensive Mechanized Proof of a C Program. *Journal of Automated Reasoning*, 50(4):423–456, April 2013.

[CdOCP+18] Yohan Chatelain, Pablo de Oliveira Castro, Eric Petit, David Defour, Jordan Bieder, and Marc Torrent. VeriTracer: Context-enriched tracer for floating-point arithmetic analysis. In *25th IEEE Symposium on Computer Arithmetic, ARITH 2018, Amherst, MA, USA. June 25th-27th, 2018*, page (to appear), 2018.

[Cha88] Françoise Chatelin. On the general reliability of the cestac method. *C. R. Acad.Sci. Paris*, 1:851–854, 1988.

[Cod18] Code_Aster. Structures and thermomechanics analysis for studies and research. http://www.code-aster.org/, 2018.

[7] `https://mybinder.org/v2/gh/interflop/stochastic-confidence-intervals/master?filepath=Intervals.ipynb`

[CV88] Jean-Marie Chesneaux and Jean Vignes. On the robustness of the cestac method. *C. R. Acad.Sci. Paris*, 1:855–860, 1988.

[CV93] Jean-Marie Chesneaux and Jean Vignes. L'algorithme de gauss en arithmétique stochastique. *Comptes rendus de l'Académie des sciences. Série 2, Mécanique, Physique, Chimie, Sciences de l'univers, Sciences de la Terre*, 316(2):171–176, 1993.

[DdOCP16] Christophe Denis, Pablo de Oliveira Castro, and Eric Petit. Verificarlo: Checking floating point accuracy through monte carlo arithmetic. In *23nd IEEE Symposium on Computer Arithmetic, ARITH 2016, Silicon Valley, CA, USA, July 10-13, 2016*, pages 55–62, 2016.

[ELKT95] Ernst Eypasch, Rolf Lefering, C K Kum, and Hans Troidl. Probability of adverse events that have not yet occurred: a statistical reminder. *BMJ*, 311(7005):619–620, 1995.

[Eur18] Europlexus. Project web page, 2018.

[FL15] Michael Frechtling and Philip H.W. Leong. Mcalib: Measuring sensitivity to rounding error with monte carlo programming. *ACM Transactions on Programming Languages and Systems*, 37(2):5, 2015.

[FL16] François Févotte and Bruno Lathuilière. VERROU: Assessing Floating-Point Accuracy Without Recompiling. working paper or preprint, October 2016.

[FL17] François Févotte and Bruno Lathuilière. Studying the numerical quality of an industrial computing code: A case study on code_aster. In *International Workshop on Numerical Software Verification*, pages 61–80. Springer, 2017.

[Hig02] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.

[JC08] Fabienne Jézéquel and Jean-Marie Chesneaux. Cadna: a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12):933 – 955, 2008.

[Kah66] William Kahan. Numerical linear algebra. In *Canadian Mathematical Bulletin*, pages 756–801, 1966.

[Kah96] William Kahan. The improbability of probabilistic error analyses for numerical computations. In *UCB Statistics Colloquium, Evans Hall edition*, page 20, 1996.

[LCJ10] Jean-Luc Lamotte, Jean-Marie Chesneaux, and Fabienne Jézéquel. Cadna_c: A version of cadna for use with c or c++ programs. *Computer Physics Communications*, 181(11):1925–1926, 2010.

[Li13] Wenbin Li. *Numerical accuracy analysis in simulations on hybrid high-performance computing systems*. PhD thesis, University of Stuttgart, 2013.

[NS07] Nicholas Nethercote and Julian Seward. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI)*, 2007.

[ORO05] Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi. Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26:1955–1988, 2005.

[Rum10] Siegfried M Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.

[Sap11] Gilbert Saporta. *Probabilités, analyse de données et statistiques (3eme édition)*. Editions Technip, 2011.

[SP97] Douglas Stott Parker. Monte carlo arithmetic: exploiting randomness in floating-point arithmetic. Technical Report CSD-970002, UCLA Computer Science Dept., 1997.

[Ver18a] Verificarlo. Project repository, 2018.

[Ver18b] Verrou. Project repository, 2018.

[Vig04] Jean Vignes. Discrete stochastic arithmetic for validating results of numerical software. *Numerical Algorithms*, 37(1-4):377–390, 2004.

[VLP74] Jean Vignes and Michel La Porte. Error analysis in computing. In *Proceedings of IFP 1974*, pages 610–614. IFP, 1974.

[Zel09] Andreas Zeller. *Why Programs Fail*. Morgan Kaufmann, Boston, second edition, 2009.

A Tabulated numbers of samples and variance shifts

This appendix presents tabulated values allowing the practitioner to easily choose a desired number of samples (table 3) in the case of a general distribution of results (Bernoulli framework), or determine the shifting levels to build sound estimators from the sampled variance in the centered normality hypothesis (table 4).

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 3: 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).

N_{samples}	p	0.66	0.66	0.75	0.75	0.75	0.9	0.9	0.9	0.95	0.95	0.95	0.99	0.99	0.99	0.995	0.995	0.995	0.999	0.999
	$1-\alpha$	0.66	0.75	0.66	0.75	0.9	0.75	0.9	0.95	0.9	0.95	0.99	0.95	0.99	0.995	0.99	0.995	0.999	0.995	0.999
3		1.145	1.385	1.415	1.655	2.345	2.171	2.861	3.370	3.114	3.623	4.791	4.017	5.186	5.687	5.310	5.811	6.972	6.040	7.202
4		0.817	0.990	1.086	1.260	1.749	1.776	2.264	2.617	2.517	2.870	3.665	3.264	4.059	4.396	4.183	4.520	5.298	4.749	5.527
5		0.650	0.790	0.919	1.060	1.449	1.576	1.965	2.241	2.218	2.494	3.108	2.888	3.502	3.759	3.626	3.883	4.473	4.112	4.703
6		0.546	0.667	0.816	0.936	1.266	1.452	1.781	2.013	2.034	2.266	2.772	2.660	3.167	3.377	3.291	3.501	3.981	3.730	4.210
8		0.423	0.519	0.693	0.789	1.048	1.305	1.564	1.744	1.817	1.997	2.383	2.391	2.777	2.935	2.901	3.059	3.415	3.288	3.645
9		0.382	0.470	0.652	0.740	0.977	1.256	1.493	1.656	1.746	1.909	2.258	2.303	2.652	2.794	2.776	2.918	3.236	3.147	3.465
10		0.350	0.432	0.620	0.701	0.921	1.217	1.437	1.587	1.690	1.840	2.159	2.234	2.553	2.682	2.677	2.806	3.095	3.036	3.324
12		0.301	0.373	0.570	0.643	0.835	1.159	1.351	1.482	1.604	1.735	2.011	2.129	2.405	2.516	2.529	2.640	2.886	2.869	3.115
14		0.265	0.330	0.535	0.600	0.773	1.116	1.289	1.406	1.542	1.659	1.905	2.054	2.299	2.397	2.423	2.521	2.737	2.750	2.967
15		0.250	0.313	0.520	0.583	0.748	1.099	1.264	1.376	1.517	1.629	1.862	2.023	2.256	2.349	2.380	2.473	2.678	2.702	2.907
20		0.197	0.250	0.467	0.520	0.657	1.036	1.173	1.265	1.426	1.518	1.708	1.912	2.102	2.177	2.226	2.301	2.465	2.530	2.695
22		0.182	0.232	0.452	0.502	0.631	1.018	1.147	1.234	1.400	1.486	1.664	1.881	2.059	2.129	2.183	2.253	2.406	2.482	2.635
25		0.164	0.210	0.434	0.479	0.599	0.995	1.115	1.195	1.368	1.448	1.611	1.842	2.005	2.070	2.129	2.194	2.333	2.423	2.562
29		0.144	0.186	0.414	0.456	0.566	0.972	1.081	1.154	1.334	1.407	1.555	1.801	1.950	2.008	2.074	2.132	2.257	2.361	2.487
30		0.140	0.181	0.410	0.451	0.558	0.967	1.074	1.145	1.327	1.398	1.543	1.792	1.938	1.994	2.062	2.118	2.241	2.348	2.471
40		0.108	0.143	0.378	0.413	0.504	0.929	1.019	1.079	1.272	1.332	1.453	1.726	1.847	1.895	1.971	2.019	2.120	2.248	2.349
45		0.097	0.129	0.366	0.399	0.484	0.915	1.000	1.056	1.253	1.308	1.421	1.703	1.815	1.859	1.939	1.983	2.077	2.212	2.306
50		0.087	0.118	0.357	0.388	0.468	0.904	0.984	1.036	1.236	1.289	1.395	1.683	1.789	1.830	1.913	1.954	2.042	2.183	2.271
59		0.073	0.102	0.343	0.371	0.444	0.887	0.960	1.008	1.213	1.261	1.356	1.655	1.751	1.788	1.875	1.912	1.991	2.141	2.220
75		0.056	0.081	0.326	0.350	0.414	0.866	0.930	0.972	1.183	1.224	1.308	1.619	1.702	1.734	1.826	1.858	1.927	2.087	2.156
90		0.044	0.067	0.314	0.336	0.394	0.852	0.910	0.947	1.163	1.200	1.275	1.595	1.670	1.698	1.794	1.822	1.884	2.052	2.113
100		0.038	0.059	0.308	0.329	0.383	0.845	0.899	0.935	1.152	1.188	1.258	1.582	1.652	1.680	1.776	1.804	1.861	2.033	2.091
200		0.005	0.020	0.275	0.290	0.327	0.806	0.843	0.868	1.096	1.120	1.169	1.515	1.563	1.581	1.687	1.705	1.744	1.935	1.973
299		-0.008	0.004	0.261	0.273	0.304	0.789	0.820	0.839	1.072	1.092	1.131	1.486	1.525	1.540	1.649	1.664	1.695	1.893	1.924
300		-0.008	0.003	0.261	0.273	0.304	0.789	0.819	0.839	1.072	1.092	1.131	1.486	1.525	1.540	1.649	1.664	1.695	1.893	1.924
459		-0.020	-0.011	0.250	0.259	0.284	0.775	0.799	0.815	1.052	1.068	1.099	1.462	1.493	1.505	1.617	1.629	1.653	1.858	1.883
500		-0.022	-0.013	0.248	0.257	0.280	0.773	0.796	0.811	1.049	1.064	1.093	1.458	1.487	1.499	1.611	1.623	1.646	1.852	1.876
528		-0.023	-0.015	0.246	0.255	0.278	0.771	0.794	0.808	1.047	1.061	1.090	1.455	1.484	1.495	1.608	1.619	1.642	1.848	1.871
750		-0.031	-0.023	0.239	0.247	0.265	0.762	0.781	0.793	1.034	1.046	1.070	1.441	1.464	1.473	1.588	1.597	1.616	1.827	1.846
919		-0.034	-0.028	0.235	0.242	0.259	0.758	0.775	0.786	1.028	1.039	1.060	1.433	1.455	1.463	1.579	1.587	1.604	1.816	1.833
1000		-0.036	-0.029	0.234	0.241	0.257	0.756	0.773	0.783	1.026	1.036	1.057	1.430	1.451	1.459	1.575	1.583	1.599	1.812	1.828
1058		-0.037	-0.030	0.233	0.239	0.255	0.755	0.771	0.781	1.024	1.034	1.054	1.428	1.448	1.456	1.572	1.580	1.596	1.809	1.825
1379		-0.040	-0.035	0.229	0.235	0.249	0.751	0.764	0.773	1.017	1.026	1.044	1.420	1.438	1.444	1.562	1.568	1.582	1.798	1.812
5296		-0.054	-0.051	0.216	0.219	0.226	0.735	0.742	0.746	0.995	0.999	1.008	1.393	1.402	1.405	1.526	1.529	1.536	1.759	1.766
6905		-0.055	-0.053	0.214	0.217	0.223	0.733	0.739	0.743	0.992	0.996	1.003	1.390	1.397	1.400	1.521	1.524	1.530	1.754	1.760
10000		-0.057	-0.055	0.212	0.214	0.219	0.730	0.735	0.739	0.988	0.991	0.998	1.386	1.392	1.394	1.516	1.518	1.523	1.748	1.753

Table 4: Tabulated values of the shift δ_{CNH} used to compute sound lower bounds in the Centered Normality Hypothesis, for various sets of parameters.