Rigorous Roundoff Error Analysis of Probabilistic Floating-Point Computations*

George Constantinides[1], Fredrik Dahlqvist[1,2], Zvonimir Rakamaric[3], and Rocco Salvia[3]

[1] Imperial College London g.constantinides@ic.ac.uk
[2] University College London f.dahlqvist@ucl.ac.uk
[3] University of Utah {rocco,zvonimir}@cs.utah.edu

Abstract. We present a detailed study of roundoff errors in probabilistic floating-point computations. We derive closed-form expressions for the distribution of roundoff errors associated with a random variable, and we prove that roundoff errors are generally close to being uncorrelated with their generating distribution. Based on these theoretical advances, we propose a model of IEEE floating-point arithmetic for numerical expressions with probabilistic inputs and an algorithm for evaluating this model. Our algorithm provides rigorous bounds to the output and error distributions of arithmetic expressions over random variables, evaluated in the presence of roundoff errors. It keeps track of complex dependencies between random variables using an SMT solver, and is capable of providing sound but tight probabilistic bounds to roundoff errors using symbolic affine arithmetic. We implemented the algorithm in the PAF tool, and evaluated it on FPBench, a standard benchmark suite for the analysis of roundoff errors. Our evaluation shows that PAF computes tighter bounds than current state-of-the-art on almost all benchmarks.

1 Introduction

There are two common sources of randomness in a numerical computation (a straight-line program). First, the computation might be using inherently noisy data, for example from analog sensors in cyber-physical systems such as robots, autonomous vehicles, and drones. A prime example is data from GPS sensors, whose error distribution can be described very precisely [2] and which we study in some detail in § 2. Second, the computation itself might sample from random number generators. Such probabilistic numerical routines, known as Monte-Carlo methods, are used in a wide variety of tasks, such as integration [39, 30], optimization [40], finance [21], fluid dynamics [28], and computer graphics [26]. We call numerical computations whose input values are sampled from some probability distributions *probabilistic computations*.

Probabilistic computations are typically implemented using floating-point arithmetic, which leads to roundoff errors being introduced in the computation. To strike the right balance between the performance and energy consumption versus the quality of the computed result, expert programmers rely on either a manual or automated floating-point error analysis to guide their design decisions. However, the current state-of-the-art approaches in this space have primary focused on *worst-case* roundoff error analysis of *deterministic* computations. So what can we say about floating-point roundoff errors in a probabilistic context? Is it possible to probabilistically quantify them by computing confidence intervals? Can we, for example, say with 99% confidence that the roundoff error of the computed result is smaller than some chosen constant? What is the distribution of outputs when roundoff errors are taken into account? In this paper, we explore these and similar questions. To answer them, we propose a rigorous – that is to say *sound* – approach to quantifying roundoff errors in probabilistic computations. Based on this approach, we develop an automatic tool that efficiently computes an overapproximate probabilistic profile of roundoff errors.

* Supported in part by the National Science Foundation awards CCF 1552975, 1704715, the Engineering and Physical Sciences Research Council (EP/P010040/1), and the Leverhulme Project Grant "Verification of Machine Learning Algorithms".

As an example, consider the floating-point arithmetic expression $(X + Y) \div Y$, where X and Y are random inputs described formally as independent random variables (usually defined over some intervals). In §4, we first show how the computation in *finite-precision* of a single arithmetic operation such as $X + Y$ can be modelled as $(X + Y)(1 + \varepsilon)$, where ε is also a random variable. We then show how this random variable can be computed from first principles and why it makes sense to view $(X + Y)$ and $(1 + \varepsilon)$ as independent expressions, which in turn allows us to easily compute the distribution of $(X+Y)(1+\varepsilon)$. The distribution of ε depends on that of $X+Y$, and we therefore need to evaluate arithmetic operations between random variables. When the operands are independent – as in $X + Y$ – this is standard [44], but when the operands are dependent – as in the case of the division in $(X + Y) \div Y$ – this is a hard problem. To solve it, we adopt and improve a technique for soundly bounding these distributions described in [3]. Our improvement comes from the use of an SMT solver to reason about the dependency between $(X + Y)$ and Y and remove regions of the state-space with zero probability. We describe this in § 6.

We can thus soundly bound the output distribution of any probabilistic computation, such as $(X + Y) \div Y$, performed in finite-precision floating-point arithmetic. This gives us the ability to perform *probabilistic range analysis* and prove rigorous assertions like: 99% of the outputs of a floating-point computation are smaller than a given constant bound. In order to perform *probabilistic roundoff error analysis* we develop *symbolic affine arithmetic* in § 5. This technique is combined with probabilistic range analysis to compute *conditional roundoff errors*. Specifically, we over-approximate the maximal error conditioned on the output landing in the 99% range computed by the probabilistic range analysis, i.e. conditioned on the computations not returning an outlier. This allows us to quantify very precisely the idea of the maximal error of a 'typical' computation.

Here we need to make a comment about how to understand the notion of *sound* or *rigorous* error bounds in a probabilistic context. The usual meaning of these words is anchored in a non-deterministic model of computation where inputs are known to belong to some intervals and soundly bounding the error amounts to bounding the error of *all possible computations*. If the inputs are drawn from a distribution, this understanding of soundness corresponds to bounding errors with 100% confidence, and thereby ignoring the probabilistic information completely. This is why the state-of-the-art refers to this non-deterministic model as *worst-case* analysis, because it does not account for the distribution of the inputs. On the other hand, in a probabilistic model of computation, we will say that we are giving a sound (or rigorous) probabilistic guarantee if there is 100% certainty about the probabilistic guarantee. For example when saying *99% of outputs will land in the interval* $[a, b]$ we are making a sound probabilistic statement based on some analytical description of the output distribution, and there is therefore no uncertainty about this statement. An example of *unsound* probabilistic statement is given by making probabilistic statements based on Monte-Carlo sampling. When saying *based on 1 billion runs, 99% of outputs will land in the interval* $[a, b]$ we are making an unsound probabilistic statement which may fail in another run of the experiment. Another example of unsound probabilistic statement would be *with 95% certainly, 99% of outputs will land in the interval* $[a, b]$, as is done in e.g. PAC learning [47]. By working with 'p-boxes' [3] – structures which upper- and lower-bound cumulative distribution functions (see § 3 and § 4) – we will be able to give probabilistic guarantees which are all *sound*.

We implemented our model and algorithms in a tool called PAF (for Probabilistic Analysis of Floating-point errors). We evaluated PAF on the standard floating-point benchmark suite FP-Bench [7], and compared its range and error analysis with the worst-case roundoff error analyzer FPTaylor [43, 42] and the probabilistic roundoff error analyzer PrAn [32]. We present the results in § 7, and show that FPTaylor's worst-case analysis is often overly pessimistic in the probabilistic setting, while PAF also generates tighter probabilistic error bounds than PrAn on all but one benchmark.

We summarize our contributions as follows:

(i) We derive several closed-form expressions for the distribution of roundoff errors associated with a random variable. We prove that roundoff errors are generally close to being uncorrelated with their input distribution.

(ii) Based on these results we propose a model of IEEE754 floating-point arithmetic for numerical expressions with probabilistic inputs.
(iii) We evaluate this model by developing a new algorithm for rigorously bounding the output range and roundoff error distributions of floating-point arithmetic expressions with probabilistic inputs.
(iv) We implement this model in the PAF tool, and perform probabilistic range and roundoff error analysis on a standard benchmark suite. Our comparison with the current state-of-the-art shows the advantages of our approach in terms of computing tighter, and yet still rigorous, probabilistic bounds.

2 Overview Through an Application

GPS sensors provide latitude and longitude coordinates that are inherently noisy. As shown in detail in [1], the conditional probability distribution of the true longitude and latitude coordinates (`TrueLat`, `TrueLong`), given a GPS sensor reading (`ReadLat`, `ReadLong`), is distributed according to a Rayleigh distribution

$$\begin{aligned} & \mathbb{P}\left[\text{Location} = (\texttt{TrueLat}, \texttt{TrueLong}) \mid \text{GPS} = (\texttt{ReadLat}, \texttt{ReadLong})\right] \\ \sim\ & \text{Rayleigh}\left(\|(\texttt{TrueLat}, \texttt{TrueLong}) - (\texttt{ReadLat}, \texttt{ReadLong})\|; \varepsilon/\sqrt{\ln 400}\right) \end{aligned}$$

where ε is a hardware dependent measure of accuracy of the GPS reading known as horizontal accuracy. Interestingly and somewhat surprisingly, since the density of any Rayleigh distribution is always zero at $x = 0$, it is extremely unlikely that the true coordinates lie in a small neighbourhood of those given by the GPS reading. This leads to errors described in [1] and [2] and to the suggestion that the coordinates given by a GPS sensor should be corrected by adding a probabilistic error term which will, on average, shift the observed coordinates into an area of high probability for the true coordinates. Fig. 1 gives the correction of [2], where the value of the variable `radius` is drawn from the Rayleigh distribution with parameter $\frac{\varepsilon}{\sqrt{\ln 400}}$, the value of the variable `angle` is draw from the uniform distribution on $[0, 2\pi]$, and `DEGREES_PER_METER` is a constant (≈ 0.000009009) used to convert the `radius` variable (expressed in meters) into degrees.

A developer trying to strike the right balance between resources, such as energy consumption or execution time, versus the accuracy of the computation, particularly in the context of mobile and embedded devices, might want to run a rigorous worst-case floating-point analysis tool to decide which floating-point format is accurate enough to process GPS signals. This is a mandatory decision if the developer requires rigorous error bounds holding with 100% certainty. The problem such a developer would face when analyzing a piece of code involving the probabilistic correction above, is that the Rayleigh distribution — that is to say the value of the variable `radius` in Fig. 1 — has a semi-infinite support (i.e. $[0, \infty)$), and *any* rigorous (i.e. sound) worst-case roundoff error analysis would return an infinite error bound in the presence of unbounded variables. This is because, in the *worst-case* scenario, the unbounded variable assumes exactly the infinite value, thus resulting in an infinite roundoff error. In order to get a meaningful (numeric) error bound from worst-case roundoff error tools, the range of `radius` will therefore have to be truncated. The main consequence of truncation is that soundness — which is the purpose of these tools — is lost, because regardless of *where* you decide to truncate the distribution, you are going to discard a part of its support with non-zero probability. To mitigate this issue and be 'as sound as possible', the natural truncation is given by the interval $[0, max_{(p,e)}]$ where $max_{(p,e)}$ is the largest representable number not causing an

```
TrueLat = ReadLat + ((radius * sin(angle)) * DEGREES_PER_METER)
TrueLong = ReadLong + ((radius * cos(angle)) * DEGREES_PER_METER)
```

Fig. 1: Probabilistic correction of GPS coordinates from [2]: `radius` is Rayleigh distributed, `angle` is uniformly distributed.

Table 1: Roundoff error analysis for the probabilistic latitude correction of (1).

				PAF 99.9999%	
Precision	Max	FPTaylor	PAF 100%	Absolute	Meters
double	$\approx 10^{307}$	4.3e+286	4.3e+286	4.1e-15	4.5e-10
single	$\approx 10^{38}$	2.1e+26	2.1e+26	3.7e-06	4.1e-1
half	$\approx 10^{4}$	2.5e-2	2.5e-2	2.4e-2	2667

overflow, in a generic floating-point format with p bits for the mantissa and e bits for the exponent representations. As we show next, the main issue with this natural truncation is that all the useful probabilistic information is lost, and regions of high probability are treated in the same way as regions which would never be explored even if we sampled for billions of years.

As an example, let us consider the latitude correction of Fig. 1:

$$\texttt{ReadLat} + ((\texttt{radius} * \texttt{sin}(\texttt{angle})) * \texttt{DEGREES\_PER\_METER}), \tag{1}$$

and set `ReadLat` to 51.4769°, the latitude of the Greenwich observatory, and DEGREES_PER_METER to 0.000009009. In table 1, we report our detailed roundoff error analysis when (1) is implemented in the well-known IEEE-754 double-, single-, and half-precision formats. With each implementation format (reported in the Format column), we associate the range $[0, max_{(p,e)}]$, with $max_{(p,e)}$ reported in the Max column, of the Rayleigh distribution truncated in the natural manner described above. In double-precision, for example, we truncate the Rayleigh distribution to the range $[0, 10^{307}]$ since $10^{307} \approx max_{(53,11)}$. We computed worst-case roundoff error bounds for (1) in each precision format with the state-of-the-art error analyzer FPTaylor [43], with the variable `radius` constrained to the corresponding truncated range. We show the results in the column FPTaylor. Note that in double-precision, FPTaylor returns a roundoff error bound of 4.3e+286. The reason for such an enormous error is the fact that FPTaylor does not take into consideration how `radius` is distributed in the range, i.e. how rare the values contributing to such a large error are. We also computed worst-case roundoff errors with our tool PAF by setting the confidence interval from which conditional roundoff errors are computed to 100%. We report the results in the PAF 100% column; these results are identical to the ones from FPTaylor, thereby confirming empirically that worst-case roundoff error is synonymous with conditional roundoff error based on a 100% confidence interval. Finally, the PAF 99.9999% column gives the 99.9999% *conditional roundoff error* computed by PAF. This value is an upper bound to the roundoff error *conditioned* on the computation having landed in an interval capturing (at least) 99.9999% of all possible outputs. The column Absolute gives the 99.9999% conditional roundoff error of the latitude correction (1), which is expressed in degrees, and the column Meters converts this value into meters, based on the fact that 1° of latitude is roughly equal to 111km.

Our probabilistic error analysis in PAF with 99.9999% confidence interval reduces the 100% (i.e., worst-case) error bound by 300 orders of magnitude in the case of double-precision, and by 30 orders of magnitude in the case of single-precision. The reason behind such dramatic reduction is that PAF runs a sound probabilistic range analysis (described in §4 and §6.2) in order to over-approximate the 99.9999% confidence interval when (1) is implemented in floating-point arithmetic. This 99.9999% interval is then used to compute the *conditional roundoff error* (described in § 6.3). Specifically, using *symbolic affine arithmetic* (see § 5), PAF will maximize the possible roundoff error *for those computations that land in the 99.9999% confidence interval*. In other words, PAF over-approximates the roundoff error of all possible computations except the 0.0001% of outliers that lead to the most abnormal outputs.

Without our probabilistic error analysis, the developer might *erroneously* conclude that half-precision format is the most appropriate to implement (1) because it results in the smallest error bound. However, with the information provided by the 99.9999% conditional roundoff error, the developer can see that the *average* error, without the extremely rare outliers, is many orders of

magnitude smaller than the worst-case scenarios. Furthermore, the developer can also observe that in the case of the half-precision format (and in this case alone), the worst-case errors and the probabilistic errors are almost identical, which means that the *average* roundoff error is of the same order of magnitude as the worst-case error. This is because in half-precision (and lower), the roundoff error of (1) is governed by the rounding errors of the constants, and is thus completely independent from the chosen confidence interval.

Armed with this information, the developer can conclude that with a roundoff error of roughly 40cm (4.1e-1 meters) when correcting 99.9999% of GPS latitude readings, working in single-precision is an adequate compromise between efficiency and accuracy of the computation, since double-precision provides unnecessary accuracy (roundoff error of 4.5e-10 meters) and half-precision is *always* overly imprecise (roundoff error of 2667 meters). As a quantitative intuition of the 99.9999% confidence level, it is not hard to show (by using the Poisson approximation of the binomial distribution) that the probability of encountering at least one outlier after 700,000 latitude corrections given by (1) is about 50%. Assuming one GPS reading every 5 seconds, this corresponds to at least 50% chance that the roundoff error bound is never breached after 40 days of continual use. Of course, we can run PAF at higher levels of confidence if required.

This demonstrates the innovative concept of *probabilistic precision tuning*, supported by this paper, in order to determine which floating-point format is the most appropriate to implement (1). In the current worst-case precision tuners [4, 8], the developer provides the tuner with an algebraic expression $f(x)$ (like (1)) where the arguments x are properly bounded, together with a numerical roundoff error bound ϵ. The output from the tuner is the minimal (in terms of bits) floating-point format required when implementing the expression $f(x)$ that guarantees that the implementation will *never* violate the given error bound. Clearly, worst-case precision tuning suffers from the same limitations we discussed above, meaning we have no information on how rare the inputs violating the given error bound might be.

As an example, let us do a precision tuning exercise for the latitude correction computation (1). We bound the variable `radius` in the interval $[0, 10^{307}]$ and we set the error bound ϵ to 1e-5 (roughly $1m$). We manually perform worst-case precision tuning using FPTaylor to determine that the minimal floating-point format not violating the given error bound ϵ consists of 1022 bits for the mantissa and 11 bits for the exponent. Such custom floating-point format is prohibitively expensive, in particular for devices with an intensive usage of GPS readings, like smart-phones or smart-watches. Thus, worst-case error bounds are not only very pessimistic, but are also of little practical use from an implementation prospective in some cases.

On the other hand, when we manually performed *probabilistic precision tuning* using PAF with a confidence interval set to 99.9999%, we determine that the minimal required floating-point format consists of 22 bits for the mantissa and 11 bits for the exponent. Thanks to PAF, the confidence interval has now become an additional input parameter of the probabilistic precision tuning routine. Thus, in addition to the expression $f(x)$ and the error bound ϵ, the developer can also provide a custom confidence interval of interest and ignore the extremaly unlikely corner cases like the ones we described for (1).

3 Preliminaries

3.1 Floating-Point Arithmetic

Given a *precision* $p \in \mathbb{N}$ and an *exponent range* $[e_{min}, e_{max}] \triangleq \{n \mid n \in \mathbb{N} \wedge e_{min} \leq n \leq e_{max}\}$, we define $\mathbb{F}(p, e_{min}, e_{max})$, or simply $\mathbb{F}$ if there is no ambiguity, as the set of extended real numbers

$$\mathbb{F} \triangleq \left\{ (-1)^s 2^e \left(1 + \frac{k}{2^p}\right) \middle| s \in \{0,1\}, e \in [e_{min}, e_{max}], 0 \leq k < 2^p \right\} \cup \{-\infty, 0, \infty\}$$

Elements $z = z(s, e, k) \in \mathbb{F}$ will be called *floating-point representable numbers* (for the given precision p and exponent range $[e_{min}, e_{max}]$) and we will use the variable z to represent them. The variable s will be called the *sign*, the variable e the *exponent*, and the variable k the *significand* of $z(s, e, k)$.

Next, we introduce a *rounding map* $\mathrm{Round} : \mathbb{R} \to \mathbb{F}$ that rounds to nearest (or to $-\infty/\infty$ for values smaller/greater than the smallest/largest finite element of $\mathbb{F}$) and follows any of the IEEE754 rounding modes in case of a tie. We will not worry about which choice is made since the set of mid-points will always have probability zero for the distributions we will be working with. All choices are thus equivalent, probabilistically speaking, and what happens in a tie can therefore be left unspecified. We will denote the extended real line by $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{-\infty, \infty\}$. The (signed) *absolute error function* $\mathrm{err}_{\mathrm{abs}} : \mathbb{R} \to \overline{\mathbb{R}}$ is defined as: $\mathrm{err}_{\mathrm{abs}}(x) = x - \mathrm{Round}(x)$. We define the sets $\lfloor z, z \rceil \stackrel{\triangle}{=} \{y \in \mathbb{R} \mid \mathrm{Round}(y) = \mathrm{Round}(z)\}$. Thus if $z \in \mathbb{F}$, then $\lfloor z, z \rceil$ is the collection of all reals rounding to z. As the reader will see, the basic result of §4 (Eq. (6)) is expressed entirely using the notation $\lfloor z, z \rceil$ which is parametric in the choice of the Round function. It follows that our results apply to rounding modes other that round-to-nearest with minimal changes. The *relative error function* $\mathrm{err}_{\mathrm{rel}} : \mathbb{R} \setminus \{0\} \to \overline{\mathbb{R}}$ is defined by

$$\mathrm{err}_{\mathrm{rel}}(x) = \frac{x - \mathrm{Round}(x)}{x}.$$

Note that $\mathrm{err}_{\mathrm{rel}}(x) = 1$ on $\lfloor 0, 0 \rceil \setminus \{0\}$, $\mathrm{err}_{\mathrm{rel}}(x) = \infty$ on $\lfloor -\infty, -\infty \rceil$ and $\mathrm{err}_{\mathrm{rel}}(x) = -\infty$ on $\lfloor \infty, \infty \rceil$. Recall also the fact [22] that $-2^{-(p+1)} < \mathrm{err}_{\mathrm{rel}}(x) < 2^{-(p+1)}$ outside of $\lfloor 0, 0 \rceil \cup \lfloor -\infty, -\infty \rceil \cup \lfloor \infty, \infty \rceil$. The quantity $2^{-(p+1)}$ is usually called the *unit roundoff* and will be denoted by u.

For $z_1, z_2 \in \mathbb{F}$ and $\mathrm{op} \in \{+, -, \times, \div\}$ an (infinite-precision) arithmetic operation, the traditional model of IEEE754 floating-point arithmetic [36] [22] states that the finite-precision implementation $\mathtt{op}_{\mathtt{m}}$ of op must satisfy

$$z_1 \ \mathtt{op}_{\mathtt{m}} \ z_2 = (z_1 \ \mathrm{op} \ z_2)(1 + \delta) \qquad |\delta| \leq u, \tag{2}$$

We leave dealing with subnormal floating-point numbers to future work. The model given by Eq. (2) stipulates that the implementation of an arithmetic operation can induce a relative error of magnitude *at most* u. The exact size of the error is, however, not specified and Eq. (2) is therefore a *non-deterministic model of computation*. It follows that numerical analyses based on Eq. (2) must consider *all* possible relative errors δ and are fundamentally *worst-case* analyses. Since the output of such a program might be the input of another, one should also consider non-deterministic inputs, and this is indeed what happens with automated tools for roundoff error analysis, such as Daisy [8] or FPTaylor [43, 42], which require for each variable of the program a (bounded) range of possible values in order to perform a worst-case analysis (*cf.* GPS example in § 2).

In this paper, we study a model formally similar to Eq. (2), namely

$$z_1 \ \mathtt{op}_{\mathtt{m}} \ z_2 = (z_1 \ \mathrm{op} \ z_2)(1 + \delta) \qquad \delta \sim \mathit{dist}. \tag{3}$$

The difference is that δ is now *distributed according to dist*, a probability distribution whose support is $[-u, u]$. In other words, we move from a non-deterministic to a *probabilistic* model of roundoff errors. This is similar to the 'Monte Carlo arithmetic' of [38], but whilst *op. cit. postulates* that *dist* is the uniform distribution on $[-u, u]$, we compute *dist* from first principles in § 4.

3.2 Probability Theory

To fix the notation and be self-contained, we present some basic notions of probability theory which are essential to what follows.

Cumulative Distribution Functions and Probability Density Functions. We assume that the reader is (at least intuitively) familiar with the notion of a (real) random variable. Given a random variable X we define its Cumulative Distribution Function (CDF) as the function $c(t) \triangleq \mathbb{P}[X \leq t]$.

If there exists a non-negative integrable function $d : \mathbb{R} \to \mathbb{R}$ such that

$$c(t) \triangleq \mathbb{P}[X \leq t] = \int_{-\infty}^{t} d(t)\ dt$$

then we call $d(t)$ the Probability Density Function (PDF) of X. If it exists, then it can be recovered from the CDF by differentiation $d(t) = \frac{\partial}{\partial t} c(t)$ by the fundamental theorem of calculus.

Not all random variables have a PDF: consider the random variable which takes value 0 with probability $1/2$ and value 1 with probability $1/2$. For this random variable it is impossible to write $\mathbb{P}[X \leq t] = \int d(t)\ dt$. Instead, we will write the distribution of such a variable using the so-called Dirac delta measure at 0 and 1 as $1/2\delta_0 + 1/2\delta_1$. It is possible for a random variable to have a PDF covering part of its distribution – this will be its *continuous part* – and a sum of Dirac deltas covering the rest of its distribution – this will be its *discrete part*. We will encounter examples of such random variables in § 4. Finally, note that if X is a random variable and $f : \mathbb{R} \to \mathbb{R}$ is a (measurable) function, then $f(X)$ is a random variable. In particular $\mathrm{err}_{\mathrm{rel}}(X)$ is a random variable, which we will describe in detail in § 4.

Arithmetic on Random Variables. Suppose X, Y are *independent* random variables with PDFs f_X and f_Y, respectively. Using the arithmetic operations we can form new random variables $X + Y, X - Y, X \times Y, X \div Y$. The PDFs of these new random variables can be expressed as operations on f_X and f_Y [44]:

$$\begin{aligned} f_{X+Y}(t) &\triangleq \int_{-\infty}^{\infty} f_X(x) f_Y(t-x)\ dx & f_{X-Y}(t) &\triangleq \int_{-\infty}^{\infty} f_X(x) f_Y(x-t)\ dx \\ f_{X\times Y}(t) &\triangleq \int_{-\infty}^{\infty} \frac{1}{|x|} f_X(x) f_Y\left(\frac{t}{x}\right) dx & f_{X\div Y}(t) &\triangleq \int_{-\infty}^{\infty} |x|\, f_X(x) f_Y(tx) dx \end{aligned} \tag{4}$$

It is important to note that these formulas are only valid if X and Y are assumed to be independent. When an arithmetic expression containing variable repetitions is given a random variable interpretation, this independence can no longer be assumed. In the expression $(X + Y) \div Y$ the sub-term $(X + Y)$ can be interpreted by using Eqs. (4) if X, Y are independent. However, the sub-terms $X + Y$ and Y clearly cannot be interpreted as independent random variables, and the division operation can therefore not be evaluated using Eqs. (4).

Soundly Bounding Probabilities. The constraint that the distribution of a random variable must integrate to 1 makes it impossible to order random variables in the 'natural' way: if $\mathbb{P}[X \in A] \leq \mathbb{P}[Y \in A]$, then $\mathbb{P}[Y \in A^c] \leq \mathbb{P}[X \in A^c]$, i.e. we cannot say that $X \leq Y$ if $\mathbb{P}[X \in A] \leq \mathbb{P}[Y \in A]$. This means that we cannot quantify our probabilistic uncertainty about a random variable by sandwiching it between two other random variables as one would do with reals or real-valued functions.

One solution is to restrict the sets used in the comparison, i.e. declare that $X \leq Y$ iff $\mathbb{P}[X \in A] \leq \mathbb{P}[Y \in A]$ for A ranging over a given set of 'test subsets'. Such an order can be defined by taking the 'test subsets' to be all the intervals $(-\infty, x]$ [41]. This order is now known as the *stochastic order*. It follows from the definition of the CDF that this order can most elegantly be defined by saying that $X \leq Y$ iff $c_X \leq c_Y$, where c_X and c_Y are the CDFs of X and Y, respectively. If it is possible to sandwich an unknown random variable X between known lower and upper bounds $X_{lower} \leq X \leq X_{upper}$ using the stochastic order then it becomes possible to give sound bounds to the quantities $\mathbb{P}[X \in [a, b]]$ via

$$\mathbb{P}[X \in [a,b]] = c_X(b) - c_X(a) \leq c_{X_{upper}}(b) - c_{X_{lower}}(a)$$

P-Boxes and DS-Structures. As mentioned above, giving a random variable interpretation to an arithmetic expression containing variable repetitions cannot be done using Eqs. (4). In fact, these interpretations are in general analytically intractable. Hence, a common approach is to give up on soundness and approximate such distributions using Monte-Carlo simulations. We use this approach in our experiments to assess the quality of our sound results. However, we will also provide sound

under- and over-approximations of the distribution of arithmetic expressions over random variables using the stochastic order discussed above. Since $X_{lower} \leq X \leq X_{upper}$ is equivalent to saying that $c_{X_{lower}}(x) \leq c_X(x) \leq c_{X_{upper}}(x)$, the fundamental approximating structure will be a pair of CDFs satisfying $c_1(x) \leq c_2(x)$. Such a structure is known in the literature as a *p-box* [16], and has already been used in the context of probabilistic roundoff errors in related work [3, 32]. The data of a p-box is equivalent to a pair of sandwiching distributions for the stochastic order.

A *Dempster-Shafer structure* (DS-structure) of size N is a set of interval-probability pairs $([x_0, y_0], p_0), ([x_1, y_2], p_1), .., ([x_N, y_N], p_N)\}$ where $\sum_{i=0}^{N} p_i = 1$. The intervals in the collection might overlap. One can always convert a DS-structure to a p-box and back again [16], but arithmetic operations are much easier to perform on DS-structures than on p-boxes ([3]), which is why we will use DS-structures in the algorithm described in § 6.

4 Distribution of Floating-Point Roundoff Errors

We briefly described in the GPS example of § 2 how our tool PAF computes *probabilistic* roundoff errors by conditioning the optimization of symbolic affine form (presented in the next section) on the output of the computation landing in a confidence interval. The purpose of this section is to provide the necessary probabilistic tools to compute these intervals. Specifically, we will show how the probabilistic model of Eq. (3) can be used to compute (or at least bound) the output distribution of a probabilistic computation in finite-precision. In other words, this section provides the foundations of *probabilistic range analysis*.

Our first task is to specify the distribution *dist* of the random variable δ in the probabilistic model given by Eq. (3). Following [6], we use the fact that *dist* can be computed explicitly from first principles and expressed in closed form (Eq. (6)). However, this expression can only be computed in practice at very low precision. We then show that roundoff error distributions in high precision (say, single-precision and higher) can be approximated by another closed-form expression Eq. (7) with an error that can be explicitly bounded, allowing us to remain sound. Moreover, the quality of this approximation increases with the working (i.e. benchmark) precision. In the case where all mantissas are equiprobable, it can be shown that as the working precision increases, the roundoff error distribution always converges to a unique distribution given by Eq. (8), which we call the *typical distribution*. For this class of input distributions, the distribution of errors is therefore asymptotically independent of the way the inputs are distributed. More generally, we show that the covariance between the roundoff errors of Eq. (7) and their generating input distribution is extremely small, that is to say the two processes are almost completely uncorrelated. This will have an important practical application when evaluating our probabilistic model of IEEE 754 arithmetic presented in § 6.1. All proofs can be found in the Appendix.

4.1 Derivation of the Distribution of Rounding Errors

Let us return to the probabilistic model of IEEE 754 arithmetic given by Eq. (3) where op is the infinite-precision arithmetic operation and $\texttt{op}_\texttt{m}$ is its finite-precision implementation:

$$z_1 \ \texttt{op}_\texttt{m} \ z_2 = (z_1 \ \text{op} \ z_2)(1 + \delta) \qquad \delta \sim dist.$$

Let us also assume that z_1, z_2 are random variables with known distributions. Then $z_1 \ \text{op} \ z_2$ is also a random variable which can (in principle) be computed. Since the IEEE 754 standard states that $z_1 \ \texttt{op}_\texttt{m} \ z_2$ should be given by rounding the infinite precision operation $z_1 \ \text{op} \ z_2$, it is a completely natural consequence of the standard to require that δ should simply be given by

$$\delta = \text{err}_{\text{rel}}(z_1 \ \text{op} \ z_2).$$

Thus *dist* should simply be the distribution of the random variable $\text{err}_{\text{rel}}(z_1 \ \text{op} \ z_2)$. More generally, if X is a random variable with know distribution, we will show how to compute the distribution *dist*

of the random variable

$$\mathrm{err}_{\mathrm{rel}}(X) = \frac{X - \mathrm{Round}(X)}{X}.$$

We choose to express the distribution *dist* of relative errors *in multiples of the unit roundoff* u. This choice is arbitrary, but it allows us to work with a distribution on the conceptually and numerically convenient interval $[-1, 1]$, since the absolute value of the relative error is strictly bounded by u (see § 3.1), rather than the interval $[-u, u]$.

To compute the density function of *dist*, we proceed as described in § 3.2 by first computing the CDF $c(t)$ and then taking its derivative. Recall first from § 3.1 that $\mathrm{err}_{\mathrm{rel}}(x) = 1$ if $x \in \lfloor 0, 0 \rceil \setminus \{0\}$, $\mathrm{err}_{\mathrm{rel}}(x) = \infty$ if $x \in \lfloor -\infty, -\infty \rceil$, $\mathrm{err}_{\mathrm{rel}}(x) = -\infty$ if $x \in \lfloor \infty, \infty \rceil$, and $-u \leq \mathrm{err}_{\mathrm{rel}}(x) \leq u$ elsewhere. Thus:

$$\mathbb{P}\left[\mathrm{err}_{\mathrm{rel}}(X) = -\infty\right] = \mathbb{P}\left[X \in \left\lfloor \infty, \infty \right\rceil\right] \qquad \mathbb{P}\left[\mathrm{err}_{\mathrm{rel}}(X) = 1\right] = \mathbb{P}\left[X \in \left\lfloor 0, 0 \right\rceil\right]$$
$$\mathbb{P}\left[\mathrm{err}_{\mathrm{rel}}(X) = \infty\right] = \mathbb{P}\left[X \in \left\lfloor -\infty, -\infty \right\rceil\right]$$

In other words, the probability measure corresponding to $\mathrm{err}_{\mathrm{rel}}$ has three discrete components at $\{-\infty\}, \{1\}$, and $\{\infty\}$, which cannot be accounted for by a PDF (see § 3.2). It follows that the probability measure *dist* is given by

$$dist_c + \mathbb{P}\left[X \in \left\lfloor 0, 0 \right\rceil\right] \delta_1 + \mathbb{P}\left[X \in \left\lfloor -\infty, -\infty \right\rceil\right] \delta_\infty + \mathbb{P}\left[X \in \left\lfloor \infty, \infty \right\rceil\right] \delta_{-\infty} \tag{5}$$

where $dist_c$ is a continuous measure that is not quite a probability measure since its total mass is $1 - \mathbb{P}\left[X \in \lfloor 0, 0 \rceil\right] - \mathbb{P}\left[X \in \lfloor -\infty, -\infty \rceil\right] - \mathbb{P}\left[X \in \lfloor \infty, \infty \rceil\right]$. In general, $dist_c$ integrates to 1 in machine precision since $\mathbb{P}\left[X \in \lfloor 0, 0 \rceil\right]$ is of the order of the smallest positive floating-point representable number, and the PDF of X rounds to 0 way before it reaches the smallest/largest floating-point representable number. For example, Python's `scipy` implementation of the PDF of the standard Gaussian rounds to 0 from $x = 39$. However in order to be sound, we must in general include these three discrete components in our computations. The density $dist_c$ can be computed explicitly and is given by the following result whose proof can essentially be found in [6].

Theorem 1. *Let X be a real random variable with PDF f. The continuous part $dist_c$ of the distribution of $\mathrm{err}_{\mathrm{rel}}(X)$ has a PDF given by*

$$d(t) = \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1 - tu}\right) f\left(\frac{z}{1 - tu}\right) \frac{u\,|z|}{(1 - tu)^2}, \tag{6}$$

where $\mathbb{1}_A(x)$ is the indicator function which returns 1 if $x \in A$ and 0 otherwise.

Fig. 2 (i) and (ii) shows an implementation of Eq. (6) applied to the distribution $\mathsf{Unif}(2, 4)$, first in very low precision (3 bit exponent, 4 bit significand) and then in half-precision. The theoretical density is plotted alongside a histogram of the relative error incurred when rounding 100,000 samples to low precision (computed in double-precision). The reported statistic is the K-S (Kolmogorov-Smirnov) test which measures the likelihood that a collection of samples were drawn from a given distribution. This test reports that we cannot reject the hypothesis that the samples are drawn from the corresponding density. Note how in low precision the term in $\frac{1}{(1-tu)^2}$ induces a visible asymmetry on the central section of the distribution. This effect is much less pronounced in half-precision.

For low precisions, say up to half-precision, it is computationally feasible to explicitly go through all floating-point numbers and compute the density of the roundoff error distribution *dist* directly from Eq. (6). However, this rapidly becomes prohibitively computationally expensive for higher precisions (since the number of floating-point representable numbers grows exponentially).

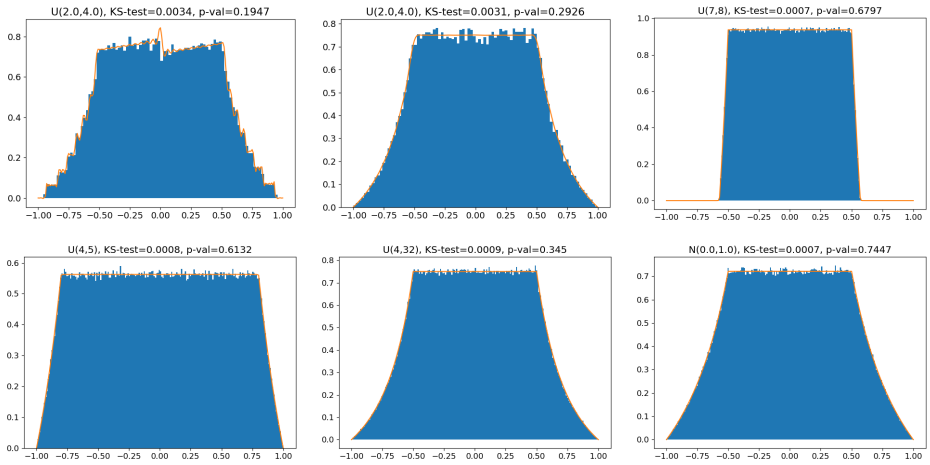

Fig. 2: Error distribution versus the corresponding empirical error distributions, clockwise from top-left: (i) Eq. (6) for $\mathsf{Unif}(2,4)$ 3 bit exponent, 4 bit significand, (ii) Eq. (6) for $\mathsf{Unif}(2,4)$ in half-precision, (iii) Eq. (7) for $\mathsf{Unif}(7,8)$ in single-precision, (iv) Eq. (7) for $\mathsf{Unif}(4,5)$ in single-precision, (v) Eq. (7) for $\mathsf{Unif}(4,32)$ in single-precision, (vi) Eq. (7) for $\mathsf{Norm}(0,1)$ in single-precision.

4.2 High-Precision Case

As the working precision increases, a regime changes occurs: on the one hand it becomes practically impossible to enumerate all floating-point representable numbers as done in Eq. (6), but on the other hand sufficiently well-behaved density functions are numerically close to being constant at the scale of an interval between two floating-point representable numbers. We exploit this smoothness to overcome the combinatorial limit imposed by Eq. (6).

Theorem 2. *Let X be a real random variable with PDF f. The continuous part dist_c of the distribution of $\mathrm{err}_{\mathrm{rel}}(X)$ has a PDF given by $d_c(t) = d_{hp}(t) + R(t)$ where $d_{hp}(t)$ is the function on $[-1,1]$ defined by*

$$d_{hp}(t) = \begin{cases} \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(2-u)} \frac{|x|}{2^{e+1}} f(x)\, dx & |t| \le \frac{1}{2} \\ \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(\frac{1}{|t|}-u)} \frac{|x|}{2^{e+1}} f(x)\, dx & \frac{1}{2} < |t| \le 1 \end{cases} \tag{7}$$

and $R(t)$ is an error whose total contribution $|R| \triangleq \int_{-1}^{1} |R(t)|dt$ can be bounded by

$$|R| \le \mathbb{P}\left[\mathrm{Round}(X) = z(s, e_{min}, k)\right] + \mathbb{P}\left[\mathrm{Round}(X) = z(s, e_{max}, k)\right] + \frac{3}{4}\left(\sum_{s, e_{min}<e<e_{max}} |f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})| \frac{2^{2e}}{2^p}\right)$$

where for each exponent e and sign s, $\xi_{e,s}$ is a point in $[z(s,e,0), z(s,e,2^p-1)]$ if $s=0$ and in $[z(s,e,2^p-1), z(s,e,0)]$ if $s=1$.

Note how Eq. (7) reduces the sum over *all* floating-point representable numbers in Eq. (6) to a sum over *the exponents* by exploiting the regularity of f. This can often be reduced further since one only needs to consider the exponent on the support of f. Note also that since f is a PDF, it usually decreases very quickly away from 0, and its derivative decreases even quicker (or vanishes altogether for uniform distributions) and $|R|$ thus tends to be very small. This is the case for all benchmarks in §7. As an example, in single-precision for $X \sim \mathsf{Norm}(0,1)$ we get $|R| < \mathsf{3.2e-7}$, for $X \sim \mathsf{Unif}(-2,2)$

we get $|R| < \texttt{1.2e}-7$; in both cases very close to the smallest floating-point representable increment to 1. Moreover, $|R| \to 0$ as the precision $p \to \infty$.

Eq. (7) is easy to implement, and we present some results in Fig. 2 where we have chosen as input: (i) a distribution $\mathsf{Unif}(7,8)$ where large significands are more likely, (ii) a distribution $\mathsf{Unif}(4,5)$ where small significands are more likely, (iii) a distribution $\mathsf{Unif}(4,32)$ where all significands are equally likely, and (iv) a distribution $\mathsf{Norm}(0,1)$ with infinite support. The graphs show the density function given by Eq. (7) in single-precision versus a histogram of the relative error incurred when rounding 1,000,000 samples to single-precision (computed in double-precision). The K-S test reports that we cannot reject the hypothesis that the samples are drawn from the corresponding distributions.

4.3 Typical Distribution

The distributions depicted in graphs (ii), (v) and (vi) of Fig. 2 are extremely similar, despite being computed from very different input distributions. What they have in common is that their input distributions have the property that that all significands in their supports are equally likely. In fact, we show that under this assumption, the distribution of roundoff errors given by Eq. (6) converges to a unique density as the precision increases, irrespective of the input distribution! Since all significands are in practice often equiprobable (it is the case for a third of the benchmarks presented in § 7), this density is of great practical importance. If one had to choose 'the' canonical distribution for roundoff errors, we claim that the density given below should be this distribution, and we therefore call it the *typical distribution*; we depict it in Fig. 3 and formalize it with the following theorem, which makes precise ideas from [6].

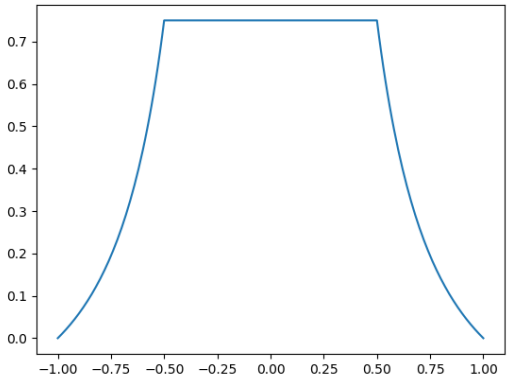

Fig. 3: Typical distribution.

Theorem 3. *If X is a random variable such that $\mathbb{P}\left[\mathrm{Round}(X) = z(s,e,k_0)\right] = \frac{1}{2^p}$ for any significand k_0, then*

$$d_{typ}(t) \triangleq \lim_{p\to\infty} d(t) = \begin{cases} \frac{3}{4} & |t| \le \frac{1}{2} \\ \frac{1}{2}\left(\frac{1}{t}-1\right) + \frac{1}{4}\left(\frac{1}{t}-1\right)^2 & |t| > \frac{1}{2} \end{cases} \tag{8}$$

where $d(t)$ is the exact density given by Eq. (6)*.*

4.4 Covariance Structure

The result above can be interpreted as saying that if X is such that all mantissas are equiprobable, then X and $\mathrm{err}_{\mathrm{rel}}(X)$ are asymptotically independent (as $p \to \infty$). Much more generally, we now show that if a random variable X has a sufficiently regular PDF, it is close to being uncorrelated from $\mathrm{err}_{\mathrm{rel}}(X)$. Formally, we prove that the covariance

$$\mathrm{Cov}(X, \mathrm{err}_{\mathrm{rel}}(X)) = \mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right] \tag{9}$$

is small, specifically of the order of u. Note that the expectation in the first summand above is taken w.r.t. the joint distribution of X and $\mathrm{err}_{\mathrm{rel}}(X)$.

The main technical obstacles to proving that the expression above is small are that $\mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right]$ turns out to be difficult to compute (we only manage to bound it) and that the joint distribution

$\mathbb{P}[X \in A \wedge \mathrm{err}_{\mathrm{rel}}(X) \in B]$ does not have a PDF since it is not continuous w.r.t. the Lebesgue measure on $\mathbb{R}^2$. Indeed, it is supported by the graph of the function $\mathrm{err}_{\mathrm{rel}}$ which has a Lebesgue measure of 0. This does not mean that it is impossible to compute the expectation

$$\mathbb{E}[X.\mathrm{err}_{\mathrm{rel}}(X)] = \int_{\mathbb{R}^2} xut \; d\mathbb{P} \tag{10}$$

but it is necessary to use some more advanced probability theory. We will make the simplifying assumption that the density of X is constant at the level of each interval $\lfloor z, z \rceil$ in order to keep the proof manageable. In practice this is an extremely good approximation. Without this assumption, we would need to add an error term similar to that of Theorem 2 to the expression below. This is not conceptually difficult, but is is rather messy, and it would distract from the main aim of the following theorem which is to bound $\mathbb{E}[\mathrm{err}_{\mathrm{rel}}(X)]$, compute $\mathbb{E}[X.\mathrm{err}_{\mathrm{rel}}(X)]$, and show that the covariance between X and $\mathrm{err}_{\mathrm{rel}}(X)$ is typically of the order of u.

Theorem 4. *If the density of X is piecewise constant on intervals $\lfloor z, z \rceil$, then*

$$\left(L - \mathbb{E}[X]\, K \frac{u}{6}\right) \leq \mathrm{Cov}(X, \mathrm{err}_{\mathrm{rel}}(X)) \leq \left(L - \mathbb{E}[X]\, K \frac{4u}{3}\right)$$

where $L = \sum_{s,e} f((-1)^s 2^e)(-1)^s 2^{2e} \frac{3u^2}{2}$ *and* $K = \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e (1-u)}^{(-1)^s 2^e (2-u)} \frac{|x|}{2^{e+1}} f(x)\; dx$

If the distribution of X is centered (i.e., $\mathbb{E}[X] = 0$) then L is the exact value of the covariance, and it is worth noting that L is fundamentally an artefact of the floating-point representation and is due to the fact that the intervals $\lfloor 2^e, 2^e \rceil$ are not symmetric. More generally, for $\mathbb{E}[X]$ of the order of, say, 2, the covariance will be small (of the order of u) as $K \leq 1$ (since $|x| \leq 2^{e+1}$ in each summand). For very large values of $\mathbb{E}[X]$ it is worth noting that there is a high chance that L is also be very large, partially canceling $\mathbb{E}[X]$. An illustration of this is given by the *doppler* benchmark examined in § 7, an outlier as it has an input variable with range $[20,\ 20000]$. Nevertheless, even for this benchmark the bounds of Theorem 4 still give a small covariance of the order of 0.001.

4.5 Error Terms and P-Boxes

In low-precision we can use the exact formula Eq. (6) to compute the error distribution. However, in high-precision, approximations (typically extremely good) like Eqs. (7) and (8) must be used. In order to remain sound in the implementation of our model (see § 6) we must account for the error made by this approximation. We have not got the space to discuss the error made by Eq. (8), but taking the term $|R|$ of Theorem 2 as an illustration, we can use the notion of p-box described in § 3.2 to create an object which soundly approximates the error distribution. We proceed as follows: since $|R|$ bounds the total error accumulated over all $t \in [-1, 1]$, we can soundly bound the CDF $c(t)$ of the error distribution given by Eq. (7) by using the p-box

$$c^-(t) = \max(0, c(t) - |R|) \qquad \text{and} \qquad c^+(t) = \min(1, c(t) + |R|)$$

This p-box is used as the (independent) roundoff error term for the probabilistic model of IEEE arithmetic given by Eq. (12), which we describe in § 6.

5 Symbolic Affine Arithmetic

In this section, we introduce *symbolic affine arithmetic*, which we employ to generate the symbolic form for the roundoff error that we use in § 6.3. Affine arithmetic [5] is a model for range analysis that extends classic interval arithmetic [37] with information about linear correlations between operands.

Symbolic affine arithmetic extends standard affine arithmetic by keeping the coefficients of the noise terms *symbolic*. We define a *symbolic affine form* as

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i, \qquad \text{where } \epsilon_i \in [-1, 1]. \tag{11}$$

We call x_0 the central symbol of the affine form, while x_i are the symbolic coefficients for the noise terms ϵ_i. We can always convert a symbolic affine form to its corresponding interval representation. This can be done using interval arithmetic or, to avoid precision loss, using a global optimizer.

Affine operations between symbolic forms follow the usual rules, such as

$$\alpha\hat{x} + \beta\hat{y} + \zeta = \alpha x_0 + \beta y_0 + \zeta + \sum_{i=1}^{n} (\alpha x_i + \beta y_i)\epsilon_i$$

Non-linear operations cannot be represented exactly using an affine form. Hence, we approximate them like in standard affine arithmetic [45].

Sound Error Analysis with Symbolic Affine Arithmetic. We now show how the roundoff errors get propagated through the four basic arithmetic operation. We apply these propagation rules to an arithmetic expression to accurately keep track of the roundoff errors. Since the (absolute) roundoff error directly depends on the range of a computation, we describe range and error together as a pair `(range: Symbol, ` $\widehat{err}$`: Symbolic Affine Form)`. Here, `range` represents the infinite-precision range of the computation, while $\widehat{err}$ is the symbolic affine form for the roundoff error in floating-point precision. Unary operators (e.g., rounding) take as input a (range, error form) pair, and return a new output pair; binary operators take as input two pairs, one per operand. For linear operators, the ranges and errors get propagated using the standard rules of affine arithmetic.

For the multiplication, we distribute each term in the first operand to every term in the second operand:

$$(\mathtt{x},\ \widehat{err}_x) * (\mathtt{y},\ \widehat{err}_y) = (\mathtt{x*y},\ \mathtt{x} * \widehat{err}_y + \mathtt{y} * \widehat{err}_x + \widehat{err}_x * \widehat{err}_y)$$

The output range is the product of the input ranges and the remaining terms contribute to the error. Only the last (quadratic) expression cannot be represented exactly in symbolic affine arithmetic; we bound such non-linearities using a global optimizer. The division is computed as the term-wise multiplication of the numerator with the inverse of the denominator. Hence, we need the inverse of the denominator error form, and then we can proceed as for multiplication. To compute the inverse, we leverage the symbolic expansion used in FPTaylor [42].

Finally, after every operation we apply the unary rounding operator from Eq. (2). The infinite-precision range is not affected by rounding. The rounding operator appends a fresh noise term to the symbolic error form. The coefficient for the new noise term is the (symbolic) floating-point range given by the sum of the input range with the input error form.

6 Algorithm and Implementation

In this section, we describe our probabilistic model of floating-point arithmetic and how we implement it in a prototype named PAF (for Probabilistic Analysis of Floating-point errors). Fig. 4 shows the toolflow of PAF.

6.1 Probabilistic model

PAF takes as input a text file describing a probabilistic floating-point computation and its input distributions. The kinds of computations we support are captured with this simple grammar:

$$\mathtt{t} ::= \mathtt{z} \mid \mathtt{x_i} \mid \mathtt{t}\ \mathtt{op_m}\ \mathtt{t} \qquad \mathtt{z} \in \mathbb{F}, \mathtt{i} \in \mathbb{N},\ \mathtt{op_m} \in \{+, -, \times, \div\}$$

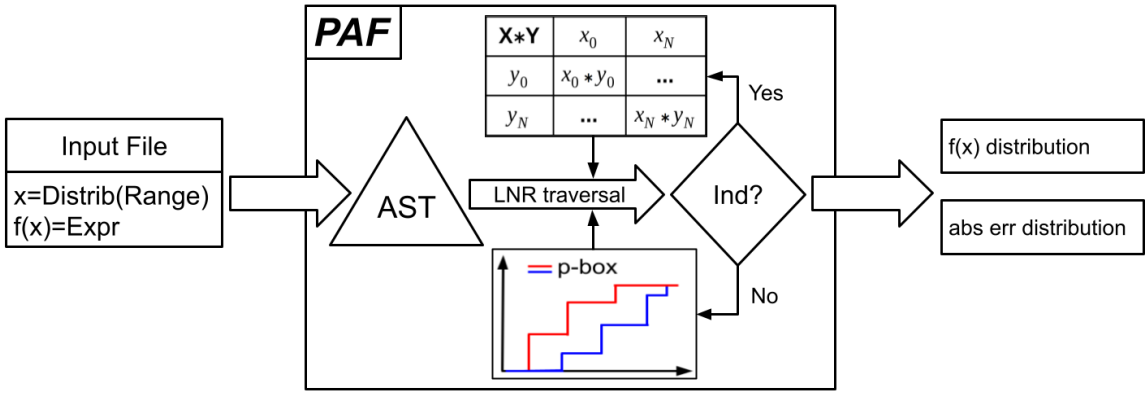

Fig. 4: Toolflow of PAF.

Following [27], we interpret each computation $\mathtt{t}$ given by the grammar as a random variable. We define the interpretation map $[\![-]\!]$ over the computation tree inductively. The base case is given by $[\![\mathtt{z}(s,e,k)]\!] \triangleq (-1)^s 2^e (1+k2^{-p})$ and $[\![\mathtt{x_i}]\!] \triangleq X_i$, where the real numbers $[\![\mathtt{z}(s,e,k)]\!]$ are understood as constant random variables and each X_i is a random input variable with a user-specified distribution. Currently, PAF supports several well-known distributions out-of-the-box (e.g., uniform, normal, exponential, beta), and a user can also define custom distributions as piecewise functions. For the inductive case $[\![\mathtt{t_1}\ \mathtt{op_m}\ \mathtt{t_2}]\!]$, we put the lessons from § 4 to work. Recall first the probabilistic model Eq. (3):

$$x\ \mathtt{op_m}\ y = (x\ \mathrm{op}\ y)(1+\delta), \qquad \delta \sim dist$$

In § 4.1, we showed that *dist* should be taken as the distribution of the actual roundoff errors of the random elements $(x\ \mathrm{op}\ y)$. We therefore define:

$$[\![\mathtt{t_1}\ \mathtt{op_m}\ \mathtt{t_2}]\!] \triangleq ([\![\mathtt{t_1}]\!]\ \mathrm{op}\ [\![\mathtt{t_2}]\!]) \times (1 + \mathrm{err}_{\mathrm{rel}}([\![\mathtt{t_1}]\!]\ \mathrm{op}\ [\![\mathtt{t_2}]\!])). \tag{12}$$

To evaluate the model of Eq. (12), we first use the appropriate closed-form expression Eqs. (6) to (8) derived in § 4 to evaluate the (distribution of the) random variable $\mathrm{err}_{\mathrm{rel}}([\![\mathtt{t_1}]\!]\ \mathrm{op}\ [\![\mathtt{t_2}]\!])$ — or the corresponding p-box as described in § 4.5. We then use Theorem 4 to justify evaluating the multiplication operation in Eq. (12) *independently* — that is to say by using Eq. (4) — since the roundoff process is very close to being uncorrelated to the process generating it. The validity of this assumption is also confirmed experimentally by the remarkable agreement of Monte-Carlo simulations with this analytical model (see § 7).

We now introduce the algorithm for evaluating the model given in Eq. (12). The evaluation performs an in-order (LNR) traversal of the *Abstract Syntax Tree* (AST) of a computation given by our grammar, and it feeds the results to the parent level along the way. At each node, it computes the probabilistic range of the intermediate result using the probabilistic ranges computed for its children nodes (i.e., operands). We first determine whether the operands are independent or not (Ind? branch in the toolflow), and we either apply a cheaper (i.e., no SMT solver invocations) algorithm if they are independent (see below) or a more involved one (see § 6.2) if they are not. We describe our methodology at a generic intermediate computation in the AST of the expression.

We consider two distributions X and Y discretized into DS-structures DS_X and DS_Y (§ 3.2), and we want to derive the DS-structure DS_Z for $Z = X\ \mathrm{op}\ Y$, $\mathrm{op} \in \{+,-,\times,\div\}$.Together with the DS-structures of the operands, we also need the traces $trace_X$ and $trace_Y$ containing the history of the operations performed so far, one for each operand. A trace is constructed at each leaf of the AST with the input distributions and their range. It is then propagated to the parent level and populated at each node with the current operation. Such history traces are critical when dealing with dependent operations since they allow us to interrogate an SMT solver about the feasibility of the current operation, as we describe in the next section. When the operands are independent, we simply use the arithmetic operations on independent DS-structures [3] (see Appendix B).

Algorithm 1 Dependent Operation $Z = X$ op Y

```
1: function DEP_OP(DS_X, op, DS_Y, trace_X, trace_Y)
2:     DS_Z = list()
3:     for all ([x_1, x_2], p_x) ∈ DS_X do
4:         for all ([y_1, y_2], p_y) ∈ DS_Y do
5:             [z_1, z_2] = [x_1, x_2] op [y_1, y_2]          ▷ operation between intervals
6:             [z'_1, z'_2] = SMT.prune([z_1, z_2])
7:             if SMT.check(trace_X ∧ trace_Y ∧ [x_1, x_2] ∧ [y_1, y_2]) is SAT then
8:                 p_Z = unknown-probability
9:             else
10:                p_Z = 0
11:            DS_Z.append(([z'_1, z'_2], p_Z))
12:    trace_Z = trace_X ∪ trace_Y ∪ {Z = X op Y}
13:    return DS_Z, trace_Z
```

6.2 Computing Probabilistic Ranges for Dependent Operands

When the operands are dependent, we start by assuming that the dependency is unknown. This assumption is sound because the dependency of the operation is included in the set of unknown dependencies, while the result of the operation is no longer a single distribution but a p-box. Due to this "unknown assumption", the CDFs of the output p-box are a very pessimistic over-approximation of the operation, i.e., they are far from each other. Our key insight is to then leverage an SMT solver to prune infeasible combinations of intervals from the input DS-structures, which prunes regions of zero probability from the output p-box. This probabilistic pruning using a solver squeezes together the CDFs of the output p-box, often resulting in a much more accurate over-approximation. Thanks to using a solver, we move from an unknown to a *partially known* dependency between the operands. Currently, PAF supports the Z3 [13] and dReal [19] SMT solvers.

Algorithm 1 shows the pseudocode of our algorithm for computing the probabilistic output range (i.e., DS-structure) for dependent operands. When dealing with dependent operands, interval arithmetic (line 5) might not be as precise as in the independent case. Hence, we use an SMT solver to prune away any over-approximations introduced by interval arithmetic when computing with dependent ranges (line 6); this use of the solver is orthogonal to the one dealing with probabilities. On line 7, we check with an SMT solver whether the current combination of ranges $[x_1, x_2]$ and $[y_1, y_2]$ is compatible with the traces of the operands. If the query is satisfiable, the probability is strictly greater than zero but currently unknown (line 8). If the query is unsatisfiable, we assign a probability of zero to the range in DS_Z (line 10). Finally, we append a new range to the DS-structure DS_Z (line 11). Note that the loops are independent, and hence in our prototype implementation we run them in parallel.

After this algorithm terminates, we still need to assign probability values to all the unknown-probability ranges in DS_Z. Since we cannot assign an exact value, we compute a range of potential values $[p_{z_{min}}, p_{z_{max}}]$ instead. This computation can be encoded as a *linear programming* routine exactly as described in previous work [3] (see Appendix B.1).

6.3 Computing Conditional Roundoff Error

The final step of our toolflow computes the conditional roundoff error by combining the symbolic affine arithmetic error form of the computation (see §5) with the probabilistic range analysis described above. The symbolic error form gets maximized conditioned on the results of all the intermediate operations landing in the given confidence interval (e.g., 99%) of their respective ranges (computed as described in the previous section). Note that conditioning only on the last operation of the computation tree (i.e., the AST root) would lead to extremely pessimistic over-approximation since all the outliers in the intermediate operations would be part of the maximization routine. This would lead to our tool PAF computing pessimistic error bounds typical of worst-case analyzers.

Algorithm 2 Conditional Roundoff Error Computation

```
1: function COND_ERR(DSS, errorForm, confidence)
2:     allRanges = list()
3:     for all DS_i ∈ DSS do
4:         focals = sorted(DS_i, key = prob, order = descending)
5:         accumulator = 0
6:         ranges = ∅
7:         for all ([x_1, x_2], p_x) ∈ focals do
8:             accumulator = accumulator + p_x
9:             ranges = ranges ∪ [x_1, x_2]
10:            if accumulator ≥ confidence then
11:                allRanges.append(ranges)
12:                break
13:    error = maximize(errorForm, allRanges)
14:    return error
```

Algorithm 2 shows the pseudocode of the roundoff error computation algorithm. The algorithm takes as input a list DSS of DS-structures (one for each intermediate result range in the computation), the generated symbolic error form, and a confidence interval. It iterates over all the intermediate DS-structures (line 3), and for each it determines the ranges needed to support the chosen confidence intervals (lines 4–12). In each iteration, it first sorts the list of range-probability pairs (i.e., focal elements) of the current DS-structure by their probability value in a descending order (line 4). This is a heuristic that prioritizes the focal elements with most of the probability mass (aka *typicals*) to avoid including the unlikely outliers (with low probability) that cause large roundoff errors into the final roundoff error computation. With the help of an accumulator (line 8), we keep collecting focal elements (line 9) until the accumulated probability satisfies the confidence interval (line 10). Finally, we maximize the error form conditioned to the collected ranges of all intermediate operations (line 13). In PAF, the maximization is done using the rigorous global optimizer Gelpia [20].

7 Experimental Evaluation

We evaluate PAF on the standard FPBench benchmark suite [7, 17] that uses the four basic operations we currently support $\{+, -, \times, \div\}$. Many of these benchmarks were also used in recent related work [32] that we compare against. The benchmarks come from a variety of domains: embedded software (*bsplines*), linear classifications (*classids*), physics computations (*dopplers*), filters (*filters*), controllers (*traincars*, *rigidBody*), polynomial approximations of functions (*sine*, *sqrt*), solving equations (*solvecubic*), and global optimizations (*trids*). Since FPBench has been primarily used for worst-case roundoff error analysis, the benchmarks come with ranges for input variables, but they do not specify input distributions. We instantiate the benchmarks with three well-known distributions for all the inputs: uniform, standard normal distribution, and double exponential (i.e. Laplace) distribution with $\sigma = 0.01$ which we will call 'exp'. The normal and exp distributions get truncated to the given range. We assume single-precision floating-point format for all operands and operations, but it is straightforward to extend PAF to mixed-precision computations.

To assess the accuracy and performance of PAF, we compare it with PrAn [32], the current state-of-the-art tool for automated analysis of probabilistic roundoff errors. PrAn currently supports only uniform and normal input distributions. It offers six different tool configurations; for each benchmark, we run all of them and report the best result. We fix the number of intervals in each discretization to 50 to match PrAn. We choose 99% as the confidence interval for the computation of our conditional roundoff error (§ 6.3) and of PrAn's probabilistic error. We also compare our probabilistic error bounds against FPTaylor which performs worst-case roundoff error analysis, and hence it does not take into account the distributions of the input variables. We ran our experiments in parallel on a 4-socket 2.2 GHz 8-core Intel Xeon E5-4620 machine with 128 GB of memory.

Table 2: Roundoff error bounds reported by PAF, PrAn, and FPTaylor given uniform (uni), normal (norm), and Laplace (exp) input distributions. We set the confidence interval to 99% for PAF and PrAn, and mark the smallest reported roundoff errors for each benchmark in bold. Asterisk (*) highlights a difference of more than one order of magnitude between PAF and FPTaylor.

	uniform		normal		exp	
Benchmark	PAF	PrAn	PAF	PrAn	PAF	FpTaylor
bspline0	**5.71e-08**	6.12e-08	**5.71e-08**	6.12e-08	**5.71e-08**	5.72e-08
bspline1	**1.86e-07**	2.08e-07	**1.86e-07**	2.08e-07	**6.95e-08**	1.93e-07
bspline2	**1.94e-07**	2.13e-07	**1.94e-07**	2.13e-07	**2.11e-08**	2.10e-07
bspline3	**4.22e-08**	4.65e-08	**4.22e-08**	4.65e-08	**7.62e-12***	4.22e-08
classids0	**6.93e-06**	8.65e-06	**4.45e-06**	8.64e-06	**1.70e-06**	6.85e-06
classids1	**3.71e-06**	4.63e-06	**2.68e-06**	4.62e-06	**7.62e-07**	3.62e-06
classids2	**5.23e-06**	7.32e-06	**3.85e-06**	7.32e-06	**1.46e-06**	5.15e-06
doppler1	**7.95e-05**	1.17e-04	**5.08e-07***	1.17e-04	**4.87e-07***	6.10e-05
doppler2	**1.43e-04**	2.45e-04	**6.61e-07***	2.45e-04	**6.28e-07***	1.11e-04
doppler3	**4.55e-05**	5.12e-05	**9.11e-07***	5.12e-05	**8.95e-07***	3.41e-05
filter1	**1.25e-07**	2.03e-07	**1.25e-07**	2.03e-07	**5.43e-09***	1.25e-07
filter2	**7.93e-07**	1.01e-06	**6.13e-07**	1.01e-06	**2.90e-08***	7.93e-07
filter3	**2.34e-06**	2.86e-06	**2.05e-06**	2.87e-06	**1.09e-07***	2.23e-06
filter4	**4.15e-06**	5.20e-06	**4.15e-06**	5.20e-06	**4.61e-07**	3.81e-06
rigidbody1	1.74e-04	**1.58e-04**	**6.14e-06***	1.58e-04	**4.80e-07***	1.58e-04
rigidbody2	1.96e-02	**9.70e-03**	**5.99e-05***	9.70e-03	**9.55e-07***	1.94e-02
sine	**2.37e-07**	2.40e-07	**2.37e-07**	2.40e-07	**1.49e-08***	2.38e-07
solvecubic	**1.78e-05**	1.83e-05	**6.84e-06**	1.83e-05	**2.76e-06**	1.60e-05
sqrt	**1.54e-04**	**1.54e-04**	**1.10e-06***	1.54e-04	**2.46e-07***	1.51e-04
traincars1	**1.76e-03**	1.96e-03	**8.26e-04**	1.96e-03	**4.50e-04**	1.74e-03
traincars2	**1.04e-03**	1.36e-03	**3.61e-04**	1.36e-03	**2.83e-05***	9.46e-04
traincars3	**1.75e-02**	2.29e-02	**9.56e-03**	2.29e-02	**8.95e-04***	1.80e-02
traincars4	**1.81e-01**	2.30e-01	**8.87e-02**	2.30e-01	**7.33e-03***	1.81e-01
trid1	**6.01e-03**	6.03e-03	**1.58e-05***	6.03e-03	**1.58e-05***	6.06e-03
trid2	**1.03e-02**	1.17e-02	**2.42e-05***	1.17e-02	**2.43e-05***	1.03e-02
trid3	**1.75e-02**	1.95e-02	**6.80e-05***	1.95e-02	**6.77e-05***	1.75e-02
trid4	**2.69e-02**	2.88e-02	**2.64e-04***	3.03e-02	**2.64e-04***	2.66e-02

Table 2 compares roundoff errors reported by PAF, PrAn, and FPTaylor. PAF outperforms PrAn by computing tighter probabilistic error bounds on almost all benchmarks, occasionally by orders of magnitude. In the case of uniform input distributions, PAF provides tighter bounds for 20 out of 27 benchmarks, for 6 benchmarks the bounds from PrAn are tighter, while for *sqrt* they are the same. In the case of normal input distributions, PAF provides tighter bounds for 26 out of 27 benchmarks, while for *bspline3* the bounds from PrAn are tighter. Unlike PrAn, PAF supports probabilistic output range analysis as well. Table 3 in Appendix C presents detailed range analysis results.

In Table 2, of particular interest are benchmarks (6 for normal and 10 for exp) where the error bounds generated by PAF for the 99% confidence interval are at least an order of magnitude tighter than the worst-case bounds generated by FPTaylor. For such a benchmark and input distribution, PAF's results inform a user that there is an opportunity to optimize the benchmark (e.g., by reducing precision of floating-point operations) if their use-case can handle at most 1% of inputs generating roundoff errors that exceed a user-provided bound. FPTaylor's results, on the other hand, do not allow for a user to explore such fine-grained trade-offs since they are worst-case and do not take probabilities into account.

In general, we see a gradual reduction of the errors transitioning from uniform to normal to exp. When the input distributions are uniform, there is a significant chance of generating a roundoff error of the same order of magnitude as the worst-case error, because all the inputs, including the extrema, are equally likely. The standard normal distribution concentrates more than 99% of probability mass in the interval $[-3, 3]$, thus resulting in the *long tail* phenomenon, where less than 0.5% of mass spreads in the interval $[3, \infty]$.When the normal distribution gets truncated in a neighborhood of zero (e.g., $[0, 1]$ for *bsplines* and *filters*) nothing changes with respect to the uniform case — there is still a high chance of committing errors close to the worst-case. However, when the normal distribution gets

truncated in a wider range (e.g., $[-100, 100]$ for *trids*) and we do have long tails, then the outliers causing large errors are very rare events, not included in the 99% confidence interval. The exponential distribution further compresses the 99% probability mass in the tiny interval $[-0.01, 0.01]$, so the long tails effect is common among all the benchmarks.

The runtimes of PAF vary between 10 minutes for small benchmarks, such as *bsplines*, to several hours for benchmarks with more than 30 operations, such as *trid4*; they are always less than two hours, except for *trids* with 11 hours and *filters* with 6 hours. The runtime of PAF is usually dominated by Z3 invocations, and the long runtimes are caused by numerous Z3 timeouts that the respective benchmarks induce. The runtimes of PrAn are comparable to PAF since they are always less than two hours, except for *trids* with 3 hours, *sqrt* with 3 hours, and *sine* with 11 hours. Note that neither PAF nor PrAn are memory intensive, and hence memory consumption is not an issue.

To assess the quality of our rigorous (i.e., sound) results, we implement Monte Carlo sampling to generate both roundoff error and output range distributions. The procedure consists of randomly sampling from the provided input distributions, evaluating the floating-point computation in both the specified and high-precision (e.g., double-precision) floating-point regimes to measure the roundoff error, and finally partitioning the computed errors into bins to get an approximation (i.e., histogram) of the PDF. Of course, Monte Carlo sampling does not provide rigorous bounds, but is a useful tool to assess how far the rigorous bounds computed statically by PAF are from an empirical measure of the error.

Fig.5 shows the effects of the input distributions on the output and roundoff error ranges of the *traincars3* benchmark. In the error graphs (right column), we show the Monte Carlo sampling evaluation (yellow line) together with the error bounds from PAF with 99% confidence interval (red plus symbol) and FPTaylor's worst-case bounds (green crossmark). In the range graphs (left column), we also plot PAF's p-box over-approximations. We can observe that in the case of uniform inputs the computed p-boxes overlap at the extrema of the output range. This phenomenon makes it impossible to distinguish between 99% and 100% confidence intervals, and hence as expected the bound reported by PAF is almost identical to FPTaylor's. This is not the case for normal and exponential distributions, where we can observe the long tail phenomenon, and PAF can significantly improve both the output range and error bounds over FPTaylor. Hence, we again illustrate how pessimistic the bounds from worst-case tools can be when the information about the input distributions is not taken into account. Finally, the graphs illustrate how the rigorous p-boxes and error bounds from PAF follow their respective empirical estimations, showing that PAF adjusts them based on the shape of the input distribution.

8 Related Work

Our work draws inspiration from the seminal probabilistic affine arithmetic approach of Bouissou et al. [3]. The fundamental object in this approach is the *probabilistic affine form*, which is an affine form whose error symbols are associated with p-boxes (see §3.2). The authors propose an innovative technique to perform dependent operations between random variables based on affine arithmetic. However, their approach can only detect affine dependencies between the operands. On the other hand, in PAF, we detect dependencies between operands using an SMT solver. Our approach allows us to handle not only polynomial dependencies, but also any non-linearities the solver supports. Finally, their approach only computes output ranges, while we went a step further and can compute roundoff error bounds as well, which is a non-trivial extension.

The most similar approach to our work is the recent static probabilistic roundoff error analysis called PrAn [32]; to the best of our knowledge, this is also the only work apart from this paper that presents a rigorous and general probabilistic roundoff error analysis. PrAn builds on the probabilistic affine arithmetic of Bouissou et al. [3], and inherits the same limitations in dealing with dependent operations we described above. Like us, their method hinges on a discretization scheme that builds p-boxes for both the input and error distributions and propagates them through the computation.

The question of how these p-boxes are chosen is left open. In contrast, we take the input variables to be random variables that the user specifies and show how the distribution of each error terms can be computed directly and exactly from the random variable generating it (see § 4). Furthermore, unlike PrAn, PAF leverages the noncorrelation between random variables and the corresponding error distribution we described in § 4.4. The immediate consequence is that PAF performs the rounding in Eq. (3) as an *independent* operation, thus without any uncertainty. Putting these together leads to PAF computing tighter probabilistic roundoff error bounds than PrAn, as our experimental results show (see § 7).

The idea of using a probabilistic model of rounding errors to analyze *deterministic* computations can be traced back to [48]. Parker's so-called 'Monte Carlo arithmetic' [38] is probably the most detailed description of this approach. We, however, consider *probabilistic* computations, and are therefore able to derive the error distribution from first principles. For this reason, the famous critique of the probabilistic approach to roundoff errors [25] does not apply to this work, since it only considers deterministic computations. When dealing with probabilistic computations however, everything – including rounding errors – has a probabilistic behavior and the probabilistic approach is therefore a necessity.

More recently, probabilistic roundoff error models have been investigated by Higham and Mary [23] as well as Ipsen and Zhou [24]. Interestingly, because these authors are interested in large-dimensional problems, neither approach needs to explicitly specify the distribution of errors. However, this means that these approaches are only applicable to large-dimensional problems, and are completely unsuited for the domains (e.g., control systems, filters, cyber-physical systems) captured by the FPBench benchmark suite. Finally, unlike us, neither approach is sound since they rely on concentration of measure inequalities.

Unlike probabilistic roundoff error analysis, worst-case analysis of roundoff errors has been an active research area with numerous published approaches
[12, 31, 9, 14, 34, 46, 8, 43, 42, 10, 33, 29, 18, 11]. As expected, none of them can deal with probabilistic inputs. Affine arithmetic [5] is a well-known technique that extends interval arithmetic [37] with information about linear correlations between operands [45]. Rigorous affine arithmetic [35] takes into consideration numerical errors stemming from the use of floating-point arithmetic when computing with affine forms, and it has been implemented in Rosa [10] to perform rigorous worst-case roundoff error analysis. Our symbolic affine arithmetic (see § 5) used in PAF evolved from rigorous affine arithmetic by keeping the coefficients of the noise terms symbolic, which often leads to improved precision. These symbolic terms are very similar to the first-order Taylor approximations of the roundoff error expressions used in FPTaylor [43, 42]; our experimental results also support this since the bounds generated by PAF for the 100% confidence interval are almost always equal to the worst-case bounds computed by FPTaylor.

9 Conclusions and Future Work

In this paper, we presented our approach to computing rigorous (i.e., sound) probabilistic roundoff error bounds for arithmetic floating-point computations involving random variables. First, we showed how to explicitly compute the error distribution of floating-point arithmetic operations directly from the distributions of the input random variables. Then, we demonstrated how the random variable and the corresponding error distribution are close to being uncorrelated. We leverage this to be able to rigorously compute operations between random floating-point variables, and thus perform rigorous probabilistic range analysis. We showed how to do this even in the complex case of dependent operands using our novel combination of p-boxes and SMT solving. We use our probabilistic range analysis to compute conditional roundoff errors where, as opposed to the worst-case approach, the (symbolic) error expression gets maximized constrained to the output landing in a given confidence interval of interest (e.g., 99%). We implemented our approach in a prototype tool named PAF, and we compared PAF on a popular benchmark suite with state-of-the-art tools for probabilistic as well

as worst-case error analysis. Our results show that PAF almost always outperforms the state-of-the-art probabilistic analysis tool PrAn in terms of generating tighter, but still rigorous, error bounds. Moreover, we observe that worst-case roundoff errors can be very pessimistic in some cases, and that PAF can reduce such error bounds by several orders of magnitude.

As future work, we plan to explore the potential use of our probabilistic range analysis to perform probabilistic overflow detection. Together with a warning about a potential overflow, a user would also get from PAF a rigorous bound on the probability of the overflow happening. This would allow users to perform a more detailed risk analysis in order to decide whether a mitigation effort is needed. We envision a similar use case for the probabilistic division-by-zero detection. Finally, in our motivating example (see § 2), we performed manual probabilistic precision tuning to further drive the point that probabilistic error analysis is needed in many settings. In fact, PAF can be used as a back-end roundoff error analyzer as a part of an existing precision tuner (such as FPTuner [4]). In such a setup, the confidence interval becomes a new key parameter in the precision tuning process, thereby allowing for programmers to explore a richer space of trade-offs. We plan to explore this line of work in the future.

References

1. James Bornholt. *Abstractions and Techniques for Programming with Uncertain Data.* Undergraduate honours thesis, Australian National University., 2013.
2. James Bornholt, Todd Mytkowicz, and Kathryn S McKinley. Uncertain<T>: a first-order type for uncertain data. In *ASPLOS*, 2014.
3. Olivier Bouissou, Eric Goubault, Jean Goubault-Larrecq, and Sylvie Putot. A generalization of p-boxes to affine arithmetic. *In: Computing*, 2012.
4. Wei-Fan Chiang, Mark Baranowski, Ian Briggs, Alexey Solovyev, Ganesh Gopalakrishnan, and Zvonimir Rakamarić. Rigorous floating-point mixed-precision tuning. In *POPL*, 2017.
5. Joao Luiz Dihl Comba and Jorge Stolfi. Affine arithmetic and its applications to computer graphics, 1993.
6. Fredrik Dahlqvist, Rocco Salvia, and George A Constantinides. A probabilistic approach to floating-point arithmetic (non-peer-reviewed extended abstract). In *ASILOMAR*, 2019.
7. Nasrine Damouche, Matthieu Martel, Pavel Panchekha, Chen Qiu, Alexander Sanchez-Stern, and Zachary Tatlock. Toward a standard benchmark format and suite for floating-point analysis. In *NSV*, 2016.
8. Eva Darulova, Anastasiia Izycheva, Fariha Nasir, Fabian Ritter, Heiko Becker, and Robert Bastian. Daisy-framework for analysis and optimization of numerical programs (tool paper). In *TACAS*, 2018.
9. Eva Darulova and Viktor Kuncak. Trustworthy numerical computation in scala. In *OOPSLA*, 2011.
10. Eva Darulova and Viktor Kuncak. Sound compilation of reals. In *POPL*, 2014.
11. Arnab Das, Ian Briggs, Ganesh Gopalakrishnan, Sriram Krishnamoorthy, and Pavel Panchekha. Scalable yet rigorous floating-point error analysis. In *SC*, 2020.
12. Marc Daumas and Guillaume Melquiond. Certification of bounds on expressions involving rounded operators. *In: ACM Transactions on Mathematical Software*, 2010.
13. Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *TACAS*, 2008.
14. David Delmas, Eric Goubault, Sylvie Putot, Jean Souyris, Karim Tekkal, and Franck Védrine. Towards an Industrial Use of FLUCTUAT on Safety-Critical Avionics Software. In *FMICS*. 2009.
15. Richard M. Dudley. *Real Analysis and Probability.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2002.
16. Scott Ferson, Vladik Kreinovich, Lev Grinzburg, Davis Myers, and Kari Sentz. Constructing probability boxes and Dempster-Shafer structures. Technical report, Sandia National Lab, 2015.
17. FPBench: standards and benchmarks for floating-point research. https://fpbench.org.
18. Zhoulai Fu, Zhaojun Bai, and Zhendong Su. Automated backward error analysis for numerical code. In *OOPSLA*, 2015.
19. Sicun Gao, Soonho Kong, and Edmund M. Clarke. Dreal: An smt solver for nonlinear theories over the reals. In *CADE*, 2013.
20. Gelpia.
21. Paul Glasserman. *Monte Carlo methods in financial engineering.* Springer, 2013.
22. Nicholas J Higham. *Accuracy and stability of numerical algorithms.* SIAM, 2002.
23. Nicholas J Higham and Theo Mary. A new approach to probabilistic rounding error analysis. *In: SISC*, 2019.
24. Ilse C. F. Ipsen and Hua Zhou. Probabilistic error analysis for inner products. *In: SIMAX*, 2019.
25. William Kahan. The improbability of probabilistic error analyses for numerical computations, 1996.
26. James T Kajiya. The rendering equation. In *SIGGRAPH*, 1986.
27. Dexter Kozen. Semantics of probabilistic programs. *In: JCSS*, 1981.
28. David P Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics.* Cambridge University Press, 2014.
29. Wonyeol Lee, Rahul Sharma, and Alex Aiken. Verifying bit-manipulations of floating-point. In *PLDI*, 2016.
30. G Peter Lepage. Vegas-an adaptive multi-dimensional integration program. Technical report, 1980.
31. Michael D. Linderman, Matthew Ho, David L. Dill, Teresa H. Meng, and Garry P. Nolan. Towards program optimization through automated analysis of numerical precision. In *CGO*, 2010.
32. Debasmita Lohar, Milos Prokop, and Eva Darulova. Sound probabilistic numerical error analysis. In *IFM*, 2019.
33. Victor Magron, George Constantinides, and Alastair Donaldson. Certified roundoff error bounds using semidefinite programming. *In: TOMS*, 2017.

34. Matthieu Martel. Rangelab: A static-analyzer to bound the accuracy of finite-precision computations. In *SYNASC*, 2011.
35. Frédéric Messine. Extentions of affine arithmetic: Application to unconstrained global optimization. *J. UCS*, 8(11):992–1015, 2002.
36. Microprocessor Standards Committee of the IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic., 2019.
37. R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
38. D Stott Parker, Brad Pierce, and Paul R Eggert. Monte Carlo arithmetic: how to gamble with floating point and win. *In: Computing in Science & Engineering*, 2000.
39. William H Press and Glennys R Farrar. Recursive stratified sampling for multidimensional monte carlo integration. *In: Computers in Physics*, 1990.
40. William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*. Cambridge University Press, 1988.
41. Michael Rothschild and Joseph E Stiglitz. Increasing risk: I. a definition. *In: Journal of Economic theory*, 1970.
42. Alexey Solovyev, Marek S Baranowski, Ian Briggs, Charles Jacobsen, Zvonimir Rakamarić, and Ganesh Gopalakrishnan. Rigorous estimation of floating-point round-off errors with symbolic taylor expansions. *In: TOPLAS*, 2018.
43. Alexey Solovyev, Charles Jacobsen, Zvonimir Rakamarić, and Ganesh Gopalakrishnan. Rigorous estimation of floating-point round-off errors with symbolic taylor expansions. In *FM*, 2015.
44. Melvin D. Springer. *The algebra of random variables*. Wiley, 1979.
45. Jorge Stolfi and Luiz Henrique De Figueiredo. Self-validated numerical methods and applications, 1997.
46. Laura Titolo, Marco A. Feliú, Mariano Moscato, and César A. Muñoz. An abstract interpretation framework for the round-off error analysis of floating-point programs. In *VMCAI*, 2018.
47. Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
48. John Von Neumann and Herman H Goldstine. Numerical inverting of matrices of high order. *In: Bulletin of the American Mathematical Society*, 1947.

A Proofs

We define $\lceil x \rceil \triangleq \sup\{y \in \mathbb{R} \mid \text{Round}(y) = \text{Round}(x)\}$ and $\lfloor x \rfloor \triangleq \inf\{z \in \mathbb{R} \mid \text{Round}(z) = \text{Round}(x)\}$. Recall also that

$$\left\lfloor z, z \right\rceil \triangleq \{y \in \mathbb{R} \mid \text{Round}(y) = \text{Round}(z)\}\,.$$

A.1 Proofs for § 4.1.

Proof of Theorem 1: For $t \in]-1, 1[$, the cumulative distribution function of the measure associated with the random variable $\text{err}_{\text{rel}}(X)$ is given by:

$$c(t) \stackrel{\triangle}{=} \mathbb{P}\left[\text{err}_{\text{rel}}(X) \leq tu\right] = \mathbb{P}\left[\bigvee_{z \in \mathbb{F}} \left(\frac{X - z}{X} \leq tu \wedge X \in \left\lfloor z, z \right\rceil\right)\right]$$

As explained in § 4.1 we exclude the floating-point representable numbers $\{-\infty, 0, \infty\}$ which correspond to the discrete components $\mathbb{P}\left[X \in \lfloor 0, 0 \rceil\right] \delta_1 + \mathbb{P}\left[X \in \lfloor -\infty, -\infty \rceil\right] \delta_\infty + \mathbb{P}\left[X \in \lfloor \infty, \infty \rceil\right] \delta_{-\infty}$ of the distribution $dist$.

Using the σ-additivity of measures (see § 3.2) we get the following density for $dist_c$:

$$\begin{aligned} d(t) =& \frac{\partial}{\partial t} \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{P}\left[\frac{X - z}{X} \leq tu \wedge X \in \left\lfloor z, z \right\rceil\right] \\ =& \sum_{z \in \mathbb{F}^+_{0,-\infty}} \frac{\partial}{\partial t} \mathbb{P}\left[\frac{z}{1 - tu} \geq X \wedge X \in \left\lfloor z, z \right\rceil\right] + \sum_{z \in \mathbb{F}^-_{0,\infty}} \frac{\partial}{\partial t} \mathbb{P}\left[\frac{z}{1 - tu} \leq X \wedge X \in \left\lfloor z, z \right\rceil\right] \end{aligned}$$

where $\mathbb{F}^+_{0,\infty}$ (resp. $\mathbb{F}^-_{0,-\infty}$) denotes the strictly positive (resp. negative) finite floating-point representable numbers. Since X is described by a probability density function $f : \mathbb{R} \to \mathbb{R}$, we get:

$$\begin{aligned} d(t) =& \sum_{z \in \mathbb{F}^+_{0,-\infty}} \frac{\partial}{\partial t} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1 - tu}\right) \int_{\lfloor z \rfloor}^{\frac{z}{1-tu}} f(s)\, ds + \sum_{z \in \mathbb{F}^-_{0,\infty}} \frac{\partial}{\partial t} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1 - tu}\right) \int_{\frac{z}{1-tu}}^{\lceil z \rceil} f(s)\, ds \\ =& \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1 - tu}\right) f\left(\frac{z}{1 - tu}\right) \frac{u\,|z|}{(1 - tu)^2} \end{aligned}$$

□

Proofs for § 4.2.

We now give some useful results about floating-point numbers. The following lemma collects explicit representations of $\lceil z \rceil$ and $\lfloor z \rfloor$ which will be useful in what follows, the proof is by direct computation.

Lemma 1. *For $z = z(0, e, k) \in \mathbb{F}$ the values of $\lceil z \rceil$ and $\lfloor z \rfloor$ are given by:*

$$\lfloor z \rfloor = \begin{cases} 2^{e-1} & \text{if } e = e_{min}, k = 0 \\ 2^{e-1}\left(1 + \frac{2^{p+1}-1}{2^{p+1}}\right) & \text{if } e > e_{min}, k = 0 \\ 2^e\left(1 + \frac{2k-1}{2^{p+1}}\right) & \text{otherwise} \end{cases}$$

$$\lceil z \rceil = \begin{cases} z & \text{if } e = n, k = 2^p - 1 \\ 2^e\left(1 + \frac{2k+1}{2^{p+1}}\right) & \text{otherwise} \end{cases}$$

For $z = z(1, e, k)$ we use the identities $\lfloor -z \rfloor = -\lceil z \rceil$ and $\lceil -z \rceil = -\lfloor z \rfloor$.

Using this lemma we can prove a result relating the length $\tau(z) \triangleq \lceil z \rceil - \lfloor z \rfloor$ of the interval corresponding to a representable number $z \in \mathbb{F}$ with its absolute value $|z|$. Note that $\lceil z \rceil - \lfloor z \rfloor$ is always a positive quantity, hence the relation with $|z|$. The following result will be used heavily in the proof of Theorem 2 . Again, the proof can be obtained by direct computation.

Lemma 2. *For any $z(s,e,k) \in \mathbb{F}, z \neq 0$ it is the case that*

$$u\,|z| = C(e,k)\,(\lceil z \rceil - \lfloor z \rfloor)$$

where the coefficients $C(e,k)$ are given by:

$$C(e_{min},0) = \frac{2^{p+1}+1}{2^p(2^{p+1}-1)} \qquad C(e,0) = \frac{2}{3}, \quad e > e_{min}$$
$$C(e_{max},2^p-1) = \frac{3(2^{p+1}-1)}{2^{p+1}} \qquad C(e,k) = \frac{2^p+k}{2^{p+1}}, \quad \textit{otherwise.}$$

Finally, we will need the following technical lemma which can also be shown by direct computation.

Lemma 3. *If $z = z(s,e,k)$ with $e \neq e_{min}, e_{max}$ then*

$$\frac{z}{1-tu} \in \lfloor z, z \rceil \text{ iff } \begin{cases} \frac{-2^{p+1}}{2^{p+2}-1} \leq t \leq \frac{2^{p+1}}{2^{p+1}+1} & k = 0 \\ \frac{-2^{p+1}}{2^{p+1}+2k-1} \leq t \leq \frac{2^{p+1}}{2^{p+1}+2k+1} & \textit{otherwise} \end{cases}$$

Note in particular that $\frac{z}{1-tu} \in \lfloor z, z \rceil$ whenever $|t| \leq \frac{1}{2}$. We are now ready to prove the theorem of § 4.2.

Proof of Theorem 2: We start by plugging the coefficients of Prop. 2 into Eq. (6):

$$d(t) = \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{1}_{\lfloor z,z \rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right) \frac{C(e,k)(\lceil z \rceil - \lfloor z \rfloor)}{(1-tu)^2}.$$

We start by considering the case where $|t| \leq \frac{1}{2}$. As shown in Lemma 3 if $|t| \leq \frac{1}{2}$ then $\mathbb{1}_{\lfloor z,z \rceil}\left(\frac{z}{1-tu}\right) = 1$, and $d(t)$ thus simplifies to

$$d(t) = \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} f\left(\frac{z}{1-tu}\right) \frac{C(e,k)(\lceil z \rceil - \lfloor z \rfloor)}{(1-tu)^2}.$$

We now restrict ourselves to floating-point representable whose exponents are not extremal, i.e. $z(s,e,k)$ with $k \notin \{e_{min}, e_{max}\}$, and put the usually minuscule contribution made by these numbers in the error term R, i.e. we will take

$$\mathbb{P}\left[\text{Round}(X) = z(s, e_{min}, k)\right] + \mathbb{P}\left[\text{Round}(X) = z(s, e_{max}, k)\right]$$

to be part of the error term R. We do this chiefly for mathematical expediency (to avoid the special behaviour of Lemma 2). We are now left with the sum

$$\begin{aligned}
d(t) =& \frac{1}{(1-tu)^2} \sum_{s,e=e_{min}+1}^{e_{max}-1} \left(\frac{2}{3} f\left(\frac{z(s,e,0)}{1-tu}\right) (\lceil z(s,e,0)\rceil - \lfloor z(s,e,0)\rfloor) \right. \\
&\left. + \sum_{k=1} \frac{2^p+k}{2^{p+1}} f\left(\frac{z(s,e,k)}{1-tu}\right) (\lceil z(s,e,k)\rceil - \lfloor z(s,e,k)\rfloor) \right) \\
\overset{(1)}{=} & \frac{1}{(1-tu)^2} \sum_{s,e=e_{min}+1}^{e_{max}-1} \left(\frac{1}{2} f\left(\frac{z(s,e,0)}{1-tu}\right) (\lceil z(s,e,0\rceil - z(1-u)) \right. \\
&\left. + \sum_{k=1}^{2^{p-1}} \frac{z(s,e,k)}{2^{e+1}} f\left(\frac{z(s,e,k)}{1-tu}\right) (\lceil z(s,e,k)\rceil - \lfloor z(s,e,k)\rfloor) \right) \\
\overset{(2)}{=} & \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \frac{1}{2^{e+1}} \left(\frac{z(s,e,0)}{1-tu} f\left(\frac{z(s,e,0)}{1-tu}\right) (\lceil z(s,e,0\rceil - z(1-u)) \right. \\
&\left. + \sum_{k=1}^{2^{p-1}} \frac{z(s,e,k)}{1-tu} f\left(\frac{z(s,e,k)}{1-tu}\right) (\lceil z(s,e,k)\rceil - \lfloor z(s,e,k)\rfloor) \right) \\
\overset{(3)}{=} & \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \frac{1}{2^{e+1}} \left(\int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(2-u)} |x|\, f(x)\, dx \right) + S(e,s,t)
\end{aligned} \tag{13}$$

where (1) follows by computing the distance $\lceil z(s,e,0)\rceil - \lfloor z(s,e,0)\rfloor$ with Lemma 1 and by substituting k for its value in terms of z via $z = 2^e(1+\frac{k}{2^p})$, and (2) follows from writing $\frac{1}{2}$ as $\frac{z(s,e,0)}{2^{e+1}}$. The last step follows by noticing that the sum in step (2) approximates the integral in (3) (where the absolute value $|x|$ takes care of the fact that the bounds of the integral will be the 'wrong way round' when $s=1$). Indeed, when $t=0$ it corresponds to the so-called *mid-point rule* of numerical integration. We can explicitly compute the error $S(e,s,t)$ incurred from this approximation using standard technique from numerical analysis. Specifically, for a fixed $z(s,e,k)$ we write the function $xf(x)$ as a Taylor expansion around $\frac{z}{1-tu} f\left(\frac{z}{1-tu}\right)$ with a first-order remainder in Lagrange form to get

$$xf(x) = \frac{z}{1-tu} f\left(\frac{z}{1-tu}\right) + \left(x - \frac{z}{1-tu}\right) (f'(\xi_x)\xi_x + f(\xi_x)) \tag{14}$$

for some ξ_x between x and $\frac{z}{1-tu}$. Using this representation we get the error term

$$\begin{aligned}
S(z,t) \triangleq & \int_{z_0}^{\lceil z\rceil} f\left(\frac{z(s,e,k)}{1-tu}\right) (\lceil z(s,e,k)\rceil - \lfloor z(s,e,k)\rfloor) - \int_{z_0}^{\lceil z\rceil} xf(x)\, dx \\
&= (f'(\xi_k)\xi_k + f(\xi_k)) \frac{-ztu}{1-tu} (\lceil z\rceil - z_0)
\end{aligned}$$

where $z_0 = z(1-u)$ when the significand of z is 0 and $z_0 = \lfloor z\rfloor$ otherwise. The last step uses the Mean Value Theorem extract a unique $\xi_k \in [z_0, \lceil z\rceil]$.. Note that when $t=0$ the error term above disappears. This is completely expected since $t=0$ corresponds to the mid-point rule which is a second-order numerical integration technique (it has no first-order error). We should therefore expand Eq. (14) to the second order in the case where $t=0$. however, since we will integrate over every t, what happens at $t=0$ can be ignored since the Lebesgue measure is continuous. Now summing over every

significand and using the Mean Value Theorem once more we get and cumulative error

$$\begin{aligned} S(e,s,t) &\triangleq \sum_{k=0}^{2^p-1} S(z(s,e,k),t) \\ &= (f'(\xi_{e,s})\xi_{e,s} + f(\xi_e)) \sum_{k=0}^{2^p-1} \frac{-z(s,e,k)tu}{1-tu} \left(\lceil z(s,e,k) \rceil - z_0\right) \end{aligned}$$

for some $\xi_{e,s}$ in $[2^e(1-u), 2^e(2-u)]$ when $s=0$ and $[-2^e(2-u), -2^e(1-u)]$ when $s=1$, and for z_0 defined as above. We can bound $|S(e,s,t)|$ as follows. We first compute

$$\begin{aligned} \left|\frac{-ztu}{1-tu}\left(\lceil z \rceil - z_0\right)\right| &= \left|\frac{t}{1-tu}\right| |z|\, u\,(z-z_0) \\ &\overset{(1)}{=} \left|\frac{t}{1-tu}\right| \frac{2^p+k}{2^{p+1}} (z-z_0)^2 \\ &\overset{(2)}{=} \left|\frac{t}{1-tu}\right| \frac{2^p+k}{2^{p+1}} \frac{2^{2e}}{2^{2p}} \end{aligned}$$

where (1) follows by Lemma 2 and (2) follows from the fact that all the intervals $z - z_0$ have the same size given by $2^e/2^p$. We can now bound $S(e,s,t)$ by

$$\begin{aligned} |S(e,s,t)| &= |f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})| \sum_{k=0}^{2^p-1} \left|\frac{t}{1-tu}\right| \frac{2^p+k}{2^{p+1}} \frac{2^{2e}}{2^{2p}} \\ &= |f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})| \left|\frac{t}{1-tu}\right| \frac{3}{4}(2^p-1)\frac{2^{2e}}{2^{2p}} \\ &= |f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})| \left|\frac{t}{1-tu}\right| \frac{3}{4}(1-u)\frac{2^{2e}}{2^{p}} \\ &\leq |f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})| \frac{3}{4}\frac{2^{2e}}{2^p} \end{aligned} \tag{15}$$

since $|t| \leq 1$.

We now consider the case where $\frac{1}{2} < |t| \leq 1$. We can follow the derivation leading to Eq. (13) by simply appending a coefficient given by $\mathbb{1}_{\lfloor z,z \rceil}$ at every step up to the penultimate one. Now using Lemma 3 and substituting k by its value in terms of z via $z = 2^e(1 + \frac{k}{2^p})$ we get:

$$\mathbb{1}_{\lfloor z,z \rceil}\left(\frac{z(s,e,k)}{1-tu}\right) = 1 \Leftrightarrow \begin{cases} z \leq 2^e(\frac{1}{t} - u) & z > 0 \\ z \leq 2^e(-\frac{1}{t} + u) & z < 0. \end{cases}$$

This means that in the last step of the derivation of Eq. (13) we simply need to replace the bounds of the integrals by $2^e(\frac{1}{t} - u)$ and $2^e(\frac{1}{t} + u)$, which leads to the final expression

$$\frac{1}{(1-tu)} \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(\frac{1}{|t|}-u)} \frac{|x|}{2^{e+1}} f(x)\, dx$$

In terms of error, it is clear that since we're approximating the same integrals $\int x f(x)\, dx$ in the same way, but on a more narrow interval, the error term Eq. (15) also provides an upper bound the error made in the integrals $\int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(\frac{1}{|t|}-u)} \frac{|x|}{2^{e+1}} f(x)\, dx$, in other words

$$|S(e,s,t)| \leq |f'(\xi_e)\xi_e + f(\xi_e)| \frac{3}{4}\frac{2^{2e}}{2^p}$$

also when $1/2 \leq |t| \leq 1$. It thus follows by the triangular inequality that we can bound the total error made from converting discrete sums into integrals by

$$\frac{3}{4}\left(\sum_{s,e_{min}<e<e_{max}} \left(f'(\xi_{e,s})\xi_{e,s} + f(\xi_{e,s})\right)\frac{2^{2e}}{2^p}\right)$$

as claimed. □

Proofs for § 4.3.

Proof of Theorem 3: We start with the exact expression given in Eq. (6), namely

$$d(t) = \sum_{z\in\mathbb{F}\setminus\{-\infty,0,\infty\}} \mathbb{1}_{\lfloor z,z\rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right)\frac{u\,|z|}{(1-tu)^2}$$

We are now going to sum over significands (since they are assumed to be equiprobable). Using Lemma 2 to re-write $u\,|z|$ and ignoring the terms with maximal exponents as we did in the proof of Theorem 2 (this is not strictly necessary, but it make the proof a lot less cumbersome) we get:

$$\begin{aligned}
d(t) &= \sum_{k=0}^{2^p-1}\frac{C(e,k)}{(1-tu)^2}\sum_{e=emin+1}^{emax-1}\sum_{s=0}^{1} f\left(\frac{z(e,s,k)}{1-tu}\right)\left(\lceil z(e,s,k)\rceil - \lfloor z(e,s,k)\rfloor\right)\\
&= \sum_{k=0}^{2^p-1}\frac{C(e,k)}{(1-tu)^2}\mathbb{P}\left[\text{Round}(X) = z(s,e,k)\right]) + S(k,p,t)\\
&= \frac{1}{2^p(1-tu)^2}\left(\frac{2}{3} + \sum_{k=1}^{2^p-1}\frac{2^p+k}{2^{p+1}}\right) + S(k,p,t)\\
&= \frac{1}{2^p(1-tu)^2}\left(\frac{2}{3} + \frac{3(2^p-1)}{4}\right) + S(k,p,t) \qquad (16)
\end{aligned}$$

where $S(k,p,t)$ is an error term quantifying the error made by the numerical integration scheme evaluating $\mathbb{P}\left[\text{Round}(X) = z(s,e,k)\right])$ as the sum in the first step of the derivation above. It can be quantified in the same way as in the proof of Theorem 2. Crucially $\lim_{p\to\infty} S(k,p,t) = 0$ for every k,t. We thus get that for $|t| < 1/2$

$$\lim_{p\to\infty} d(t) = \lim_{p\to\infty}\frac{1}{2^p(1-t2^{-p})^2}\left(\frac{2}{3} + \frac{3(2^p-1)}{4}\right) + S(k,p) = \frac{3}{4}.$$

As shown by Lemma 3, when $\frac{1}{2} < t \leq 1$ not all mantissas in the sum of Eq. (6) are possible,. Using the explicit characterisation of Lemma 3, we get that for $1/2 < |t| \leq 1$ (16) becomes

$$\begin{aligned}
d(t) &= \frac{1}{2^p(1-tu)^2}\left(\frac{2}{3} + \sum_{k=1}^{\alpha(t)}\frac{2^p+k}{2^{p+1}}\right) + S(k,p,t)\\
&= \frac{1}{2^p(1-tu)^2}\left(\frac{2}{3} + \frac{1}{2}\alpha(t) + \frac{1}{2^{p+2}}(\alpha(t)^2)\right) + S(k,p,t)
\end{aligned}$$

where $\alpha(t) = \lfloor 2^p(\frac{1}{t}-1) - \frac{1}{2}\rfloor$ is the usual floor function applied to $2^p(\frac{1}{t}-1) - \frac{1}{2}$. Taking the limit of the equation above we get

$$\begin{aligned}
\lim_{p\to\infty} d(t) &= \lim_{p\to\infty}\frac{1}{(1-t2^{-p})^2}\left(\frac{1}{2}\left(\frac{1}{t}-1\right) + \frac{1}{4}\left(\frac{1}{t}-1\right)^2\right) + S(k,p,t)\\
&= \frac{1}{2}\left(\frac{1}{t}-1\right) + \frac{1}{4}\left(\frac{1}{t}-1\right)^2 \qquad (17)
\end{aligned}$$

The case where $-1 \leq t < -\frac{1}{2}$ is treated in the same way and yields the same asymptotic distribution. Combining 16 and 17 we get

$$\lim_{p\to\infty} d(t) = \begin{cases} \frac{3}{4} & \text{if } t \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\ \frac{1}{2}\left(\frac{1}{t}-1\right) + \frac{1}{4}\left(\frac{1}{t}-1\right)^2 & \text{else} \end{cases}$$

as announced. □

Proofs for § 4.4.

We assume throughout this section that $\mathrm{err}_{\mathrm{rel}}(X)$ is distributed according to d_{hp} in Eq. (7). We start by bounding $\mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right]$. The proof of the following Proposition is a little technical but not conceptually difficult and can be found in the Appendix.

Proposition 1. *$K\frac{u}{6} \leq \mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right] \leq K\frac{4u}{3}$ where*

$$K = \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(2-u)} \frac{|x|}{2^{e+1}} f(x)\ dx$$

Proof. Consider the definition Eq. (7), when $\frac{1}{2} < |t| \leq 1$. Note that the terms

$$K(t) \triangleq \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(\frac{1}{|t|}-u)} \frac{|x|}{2^{e+1}} f(x)\ dx$$

are symmetric around 0 due to the dependence on $|t|$ (note also that the constant $K = K(\frac{1}{2})$). It follows by a simple change of variable that

$$\begin{aligned} \int_{-1}^{-\frac{1}{2}} \frac{t}{(1-tu)^2} K(t)\ dt &= \int_{1}^{\frac{1}{2}} \frac{-s}{(1+su)^2} K(-s)\ (-ds) \\ &= \int_{\frac{1}{2}}^{1} \frac{-s}{(1+su)^2} K(s)\ ds \end{aligned}$$

and thus

$$\begin{aligned} \int_{-1}^{-\frac{1}{2}} d(t)t\ dt + \int_{\frac{1}{2}}^{1} d(t)t\ dt &= -\int_{\frac{1}{2}}^{1} \frac{t}{(1+tu)^2} K(t)\ dt + \int_{\frac{1}{2}}^{1} d(t)t\ dt \\ &\geq -\int_{\frac{1}{2}}^{1} \frac{t}{(1-tu)^2} K(t)\ dt + \int_{\frac{1}{2}}^{1} d(t)t\ dt = 0 \end{aligned}$$

From this it follows that the net contribution $\mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right]$ of the two 'wings' of the distribution is positive and in particular that

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} d(t)t\ dt \leq \int_{-1}^{1} d(t)t\ dt = \mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right].$$

Moreover, by definition of K we have

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} d(t)t\ dt = K \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{t}{(1-tu)^2}\ dt$$

We now integrate by parts to get

$$\begin{aligned} K\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{t}{(1-tu)^2}\ dt &= K\ \frac{t}{u(1-tu)}\Big|_{-\frac{1}{2}}^{\frac{1}{2}} - \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{u(1-tu)}\ dt \\ &= \frac{K}{u}\left(\frac{t}{u(1-tu)} - \frac{\ln(1-tu)}{u}\right)\Big|_{-\frac{1}{2}}^{\frac{1}{2}} \\ &= \frac{K}{u}\left(\frac{1}{1-\frac{u^2}{4}} - \frac{2}{u}\text{arctanh}\left(\frac{u}{2}\right)\right) \end{aligned}$$

For small values of u the term above is very nearly linear with slope $\frac{1}{6}$ and intercept 0. To an extremely good approximation we therefore get for small values of u (say below 2^{-2}) that

$$K\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{t}{(1-tu)^2}\ dt = K\frac{u}{6}$$

which proves the lower bound.

For the upper bound we proceed in a similar way. Note that

$$\begin{aligned} \mathbb{E}\left[\text{err}_{\text{rel}}(X)\right] &= \int_{-1}^{-\frac{1}{2}} d(t)t\ dt + \int_{-\frac{1}{2}}^{\frac{1}{2}} d(t)t\ dt + \int_{\frac{1}{2}}^{1} d(t)t\ dt \\ &= \int_{-1}^{-\frac{1}{2}} \frac{t}{(1-tu)^2}K(t)\ dt + K\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{t}{(1-tu)^2}\ dt + \int_{\frac{1}{2}}^{1} \frac{t}{(1-tu)^2}K(t)\ dt \\ &\leq K\int_{-1}^{1} \frac{t}{(1-tu)^2}\ dt \qquad\qquad (\text{since } K(t) \leq K\ ) \\ &= \frac{K}{u}\left(\frac{2}{(1-u^2)} - \frac{2}{u}\text{arctanh}(u)\right) \end{aligned}$$

Again, the expression above is very nearly linear in u for small values of u (say below 2^{-2}), with slope $\frac{4}{3}$ and intercept 0. It follows that for small values of u

$$\mathbb{E}\left[\text{err}_{\text{rel}}(X)\right] \leq K\frac{4u}{3}$$

proving the upper bound.

□

Note that $K \leq 1$ since $|x| \leq 2^{e+1}$ in each summand. The *expected relative error* $\mathbb{E}\left[\text{err}_{\text{rel}}(X)\right]$ *will therefore always be at most of the order of* u by the previous result. This makes good intuitive sense.

We now turn to $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right]$. The proof of the following result is conceptually more involved. It is based on the fundamental definition of Lebesgue integration as the limit of a monotone sequence of so-called simple functions.

Proposition 2. $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right] = \sum_{s,e} f((-1)^s 2^e)(-1)^s 2^{2e}\frac{3u^2}{2}$

In particular $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right] = 0$ if f is symmetric. More generally, the value of $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right]$ is determined by the competing terms 2^{2e} and u^2. A quick calculation shows that as long as the distribution f assigns most of its mass to values below $2^{\frac{p}{2}}$, $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right]$ will be of order u. In fact, for all the benchmarks considered in § 7 bar one, $\mathbb{E}\left[X.\text{err}_{\text{rel}}(X)\right]$ is of the order of u or smaller.

Proof. In order to (over)approximate the support of the joint distribution $\mathbb{P}$ and the integral Eq. (10), we consider for each $n \in \mathbb{N}$ the following collection of sets

$$A^n(z,k) \triangleq \left[\lfloor z \rfloor + \frac{k}{n}\tau(z), \lfloor z \rfloor + \frac{k+1}{n}\tau(z)\right] \times \left[\mathrm{err}_{\mathrm{rel}}\left(\lfloor z \rfloor + \frac{k}{n}\tau(z)\right), \mathrm{err}_{\mathrm{rel}}\left(\lfloor z \rfloor + \frac{k+1}{n}\tau(z)\right)\right]$$

where $z \in \mathbb{F}, 0 \leq k < n$, and $\tau(z) \triangleq \lceil z \rceil - \lfloor z \rfloor$ is the size of the interval $\lfloor z, z \rceil$. These sets contain pairs $(x, \mathrm{err}_{\mathrm{rel}}(x))$ where x is restricted to a sub-interval of $\lfloor z, z \rceil$ whose size is controlled by n. For any n, the support of $\mathbb{P}$ is included in the union $\bigcup_{z,k} A^n(z,k)$, by definition of $\mathbb{P}$. Moreover, by definition of the joint probability $\mathbb{P}$ and the assumption that f is constant on $\lfloor z, z \rceil$, we have

$$\begin{aligned}\mathbb{P}\left[A^n(z,k)\right] &= \int_{\lfloor z \rfloor + \frac{k}{n}\tau(z)}^{\lfloor z \rfloor + \frac{k+1}{n}} f(x)\ dx \\ &= f(z)\frac{\tau(z)}{n}\end{aligned}$$

The sets $A^n(k,z)$ were chosen is such a way that the *simple functions* (see [15]) $h_n : \mathbb{R} \times [-1,1] \to \mathbb{R}$ given by

$$h_n(x,t) = \sum_{z,k} \sup\{yus : (y,s) \in A^n(z,k)\}.\mathbb{1}_{A^n(z,k)}(x,t)$$

provide a monotonically decreasing (over)approximation of the function

$$h(x,t) = xut$$

which we want to integrate. Using the common notation of integration theory: $h_n \downarrow h$. It now follows from the definition of the Lebesgue integral that

$$\begin{aligned}\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] &= \int_{\mathbb{R}\times[-1,1]} xut\ d\mathbb{P} \\ &\triangleq \int_{\mathbb{R}\times[-1,1]} h(x,t)\ d\mathbb{P} \\ &\leq \int_{\mathbb{R}\times[-1,1]} h_n(x,t)\ d\mathbb{P} \\ &= \sum_{z,k} \sup_{(x,t)\in A^n(z,k)} xut\ \mathbb{P}\left[A^n(z,k)\right]\end{aligned}$$

The supremum is easily seen to be

$$\sup_{(x,t)\in A^n(z,k)} xut = \begin{cases}\left(\lfloor z \rfloor + \frac{k+1}{n}\tau(z)\right)\mathrm{err}_{\mathrm{rel}}(\lfloor z \rfloor + \frac{k+1}{n}\tau(z)) & z > 0 \\ \left(\lfloor z \rfloor + \frac{k}{n}\tau(z)\right)\mathrm{err}_{\mathrm{rel}}(\lfloor z \rfloor + \frac{k}{n}\tau(z)) & z < 0\end{cases}$$

Clearly, multiplying any real by its own relative error will yield its absolute error:

$$\sup_{(x,t)\in A^n(z,k)} xut = \begin{cases}(\lfloor z \rfloor + \frac{k+1}{n}\tau(z)) - z & z > 0 \\ (\lfloor z \rfloor + \frac{k}{n}\tau(z)) - z & z < 0\end{cases}$$

and we thus have

$$\begin{aligned}\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] \leq & \sum_{k,z>0}(\lfloor z\rfloor + \frac{k+1}{n}\tau(z)) - z)f(z)\frac{\tau(z)}{n} + \\ & \sum_{k,z<0}(\lfloor z\rfloor + \frac{k}{n}\tau(z)) - z)f(z)\frac{\tau(z)}{n} \\ = & \sum_{z>0} f(z)\sum_{k}(\lfloor z\rfloor + \frac{k+1}{n}\tau(z)) - z)\frac{\tau(z)}{n} + \\ & \sum_{z<0} f(z)\sum_{k}(\lfloor z\rfloor + \frac{k}{n}\tau(z)) - z)\frac{\tau(z)}{n}.\end{aligned}$$

It is not too hard to see that as $n \to \infty$ the expression $\sum_k(\lfloor z\rfloor + \frac{k+1}{n}\tau(z)) - z)\frac{\tau(z)}{n}$ and $\sum_k(\lfloor z\rfloor + \frac{k}{n}\tau(z)) - z)\frac{\tau(z)}{n}$ both approach the Riemann integral

$$\int_{\lfloor z\rfloor}^{\lceil z\rceil}(x-z)\ dx$$

In other words:

$$\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] \leq \sum_{z} f(z)\int_{\lfloor z\rfloor}^{\lceil z\rceil}(x-z)\ dx$$

There are now two cases. If the significand of z is zero then the interval $\lfloor z, z\rceil$ is not symmetric around z, it is in fact divided in a $\frac{1}{3}$:$\frac{2}{3}$ ratio around z and we therefore have

$$\begin{aligned}\int_{\lfloor z\rfloor}^{\lceil z\rceil}(x-z)\ dx &= \int_{\lfloor z\rfloor}^{z+\frac{\tau(z)}{3}}(x-z)\ dx + \int_{z+\frac{\tau(z)}{3}}^{\lceil z\rceil}(x-z)\ dx \\ &= 0 + \int_{z+\frac{\tau(z)}{3}}^{\lceil z\rceil}(x-z)\ dx\end{aligned}$$

For $z = z(0, e, 0) = 2^e$ the expression above yields:

$$\int_{2^e(1+u)}^{2^e(1+2u)}(x-2^e)\ dx = 2^{2e}\frac{3u^2}{2}.$$

Similarly for $z = z(1, e, 0) = -2^e$ we get

$$\int_{-2^e(1+2u)}^{-2^e(1+u)}(x+2^e)\ dx = -2^{2e}\frac{3u^2}{2}.$$

All other intervals $\lfloor z, z\rceil$ are centred around z and therefore

$$\int_{\lfloor z\rfloor}^{\lceil z\rceil}(x-z)\ dx = 0$$

We thus have the following bound for the expectation $\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right]$:

$$\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] \leq \sum_{s,e} f((-1)^s 2^e)(-1)^s 2^{2e}\frac{3u^2}{2}$$

A completely analogous argument using an under-approximating sequence of simple function $h_n \uparrow h$ defined by $\inf(A^n(z,k))$ instead of $\sup(A^n(z,k))$ shows that

$$\mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] \geq \sum_{s,e} f((-1)^s 2^e)(-1)^s 2^{2e} \frac{3u^2}{2}$$

and the equality follows.

□

Proof of Theorem 4: This is a direct corollary of Prop. 1 and Prop. 2. □

B Implementation

Independent Operation. We report the pseudo-code in Algorithm 3. When the operands X, Y are independent, we can compute $Z = X$ op Y with a simple pairwise operation (line 3, 4) between focal elements using interval arithmetic (line 5), and we multiply the corresponding probabilities (line 6). No matter the independence, we still populate and propagate the trace for future dependent operations (line 8). This is very much similar to the pen-and-paper approach using a joint table.

Algorithm 3 Independent Operation $Z = X$ op Y

1: **function** IND_OP(DS_X, op , $DS_Y, trace_x, trace_y$)
2: $DS_Z = list()$
3: **for all** $([x_1, x_2], p_x) \in DS_X$ **do**
4: **for all** $([y_1, y_2], p_y) \in DS_Y$ **do**
5: $[z_1, z_2] = [x_1, x_2]\ op\ [y_1, y_2]$ ▷ operation between intervals
6: $p_z = p_x * p_y$
7: $DS_Z.append(([z_1, z_2], p_z))$
8: $trace_Z = trace_X \cup trace_Y \cup \{Z = X \text{ op } Y\}$
9: **return** DS_Z, $trace_Z$

B.1 Linear Programming Routine

We can create a linear programming (LP) routine to delineate upper bound and lower bounds for the p-box. In the following we report the pseudo-code for the evaluation of the upper bound of a p-box. The lower bound is similar.

$$\begin{aligned}
\textbf{maximize} \quad & \sum_{evp \in z_{i,j}} p_{z_{i,j}} \\
\textbf{subject to} \quad & 0 \leq p_{z_{i,j}} \leq 1 \\
& \forall i \in [1, N], \sum_{1 \leq j \leq N} p_{z_{i,j}} = p_{y_i} \\
& \forall j \in [1, N], \sum_{1 \leq i \leq N} p_{z_{i,j}} = p_{x_i}
\end{aligned}$$

The probabilities in the DS structures of the operands (p_{y_i}, p_{x_i}) are called the *marginals*. The focal elements we use to populate DS_Z are called the *insiders* ($p_{z_{i,j}}$). The constraints in the LP

Table 3: Comparison between PAF, given uniform (uni), normal (norm) and exponential (exp) input distributions, and FPTaylor. The PAF columns report 99% of the support of the output range distribution. The FPTaylor columns report the worst-case output ranges. The asterisk (*) highlights a difference of more than one order of magnitude between PAF and FPTaylor.

Benchmark	PAF (uni)	PAF (norm)	PAF (exp)	FPTaylor
bspline0	[-5.85e-08, 1.67e-01]	[-5.85e-08, 1.67e-01]	[1.22e-01, 1.67e-01]*	[-5.72e-08, 1.67e-01]
bspline1	[1.65e-01, 6.67e-01]	[1.65e-01, 6.67e-01]	[6.47e-01, 6.67e-01]	[1.55e-01, 6.75e-01]
bspline2	[1.67e-01, 6.67e-01]	[1.67e-01, 6.68e-01]	[1.67e-01, 2.34e-01]	[1.59e-01, 6.78e-01]
bspline3	[-1.67e-01, 5.22e-08]	[-1.67e-01, 5.22e-08]	[-4.04e-04, 5.22e-08]*	[-1.67e-01, 4.22e-08]
classids0	[-2.32e01, 2.47e01]	[-2.32e01, 7.22e00]	[-1.43e01, -8.93e-01]*	[-2.32e01, 2.47e01]
classids1	[-1.02e01, 1.20e01]	[-3.81e00, 1.21e01]	[-2.80e-01, 5.18e00]*	[-1.02e01, 1.21e01]
classids2	[-1.67e01, 1.33e01]	[-9.80e00, 1.33e01]	[-8.06e-01, 8.91e00]*	[-1.68e01, 1.34e01]
doppler1	[-1.43e02, -3.39e-02]	[-7.56e-02, -3.39e-02]*	[-7.81e00, -3.39e-02]*	[-1.39e02, -3.16e-02]
doppler2	[-2.45e02, -2.27e-02]	[-9.29e00, -2.27e-02]*	[-1.07e01, -2.27e-02]*	[-2.32e02, -2.08e-02]
doppler3	[-8.62e-01, -5.05e-01]	[-7.98e00, -5.05e-01]*	[-7.39e00, -5.06e-01]*	[-8.35e01, -5.03e-01]
filter1	[-1.40e00, 1.40e00]	[-1.40e00, 1.40e00]	[-1.12e-01, 1.12e-01]*	[-1.40e00, 1.40e00]
filter2	[-2.04e00, 2.04e00]	[-2.04e00, -2.04e00]	[-9.54e-01, 8.81e-01]*	[-2.06e00, 2.06e00]
filter3	[-2.36e00, 2.36e00]	[-2.36e00, 2.36e00]	[-2.36e00, 2.36e00]	[-2.39e00, 2.39e00]
filter4	[-3.26e00, 3.26e00]	[-3.26e00, 3.26e00]	[-3.27e00, 3.27e00]	[-3.32e00, 3.32e00]
rigidbody1	[-7.05e02, 7.05e02]	[-5.05e01, 5.69e01]*	[-4.45e01, 6.00e01]*	[-7.05e02, 7.05e02]
rigidbody2	[-5.60e04, 5.87e04]	[-4.77e03, 8.70e03]*	[-1.44e02, 3.74e02]*	[-5.63e04, 5.87e04]
sine	[-1.00e00, 1.00e00]	[-1.00e00, 1.00e00]	[-2.61e-01, 4.21e-01]	[-1.02e00, 1.02e00]
solvecubic	[-9.27e-01, 4.81e01]	[-9.27e-01, 1.37e01]	[-9.27e-01, 9.87e00]	[-9.34e-01, 4.82e01]
sqrt	[-3.35e02, 1.48e00]	[-7.57e-01, 1.48e00]*	[-2.11e-01, 1.46e00]*	[-3.38e02, 1.50e00]
traincars1	[-2.67e03, 5.44e03]	[2.03e03, 5.44e03]*	[3.36e03, 5.44e03]*	[-2.67e03, 5.44e03]
traincars2	[-3.65e03, 3.85e03]	[-1.39e03, 1.45e03]	[-6.41e02, 6.57e02]	[-3.65e03, 3.85e03]
traincars3	[-6.70e04, 6.73e04]	[-2.52e04, 2.43e04]	[-1.42e04, 1.33e04]	[-6.70e04, 6.73e04]
traincars4	[-5.84e05, 5.69e05]	[-2.33e05, 5.69e05]	[-1.15e05, 1.27e05]	[-5.84e05, 5.69e05]
trid1	[-1.28e04, 1.26e04]	[-5.55e03, 7.72e03]	[-6.91e02, 7.68e02]	[-1.29e04, 1.27e04]
trid2	[-3.26e04, 1.33e04]	[-1.65e03, 1.59e03]*	[-1.96e03, 1.81e03]*	[-3.27e04, 1.35e04]
trid3	[-5.28e04, 1.42e04]	[-2.98e03, 3.95e03]*	[-4.06e03, 4.27e03]*	[-5.30e04, 1.40e04]
trid4	[-7.26e04, 1.92e04]	[-6.10e03, 1.17e04]*	[-8.13e03 1.05e04]*	[-7.28e04, 1.54e04]

program force the insiders to have a probability between 0 and 1, and they relate the insiders with the marginals. The insiders have to sum up to the marginals. This is very similar to a so called *joint table*.

The LP program takes in input an *evaluation point* (evp) and returns a probability value. In order to construct the DS_Z exactly we should pick one evaluation point per focal element. This has quadratic complexity. A good trade-off between accuracy and execution time consists in picking only N out of the N^2 evaluation points (e.g. using some heuristics), at the price of a slightly over-approximation. We run the linear programming routines in parallel, because the analysis of a single evaluation point is completely independent from the others.

C Experimental Evaluation

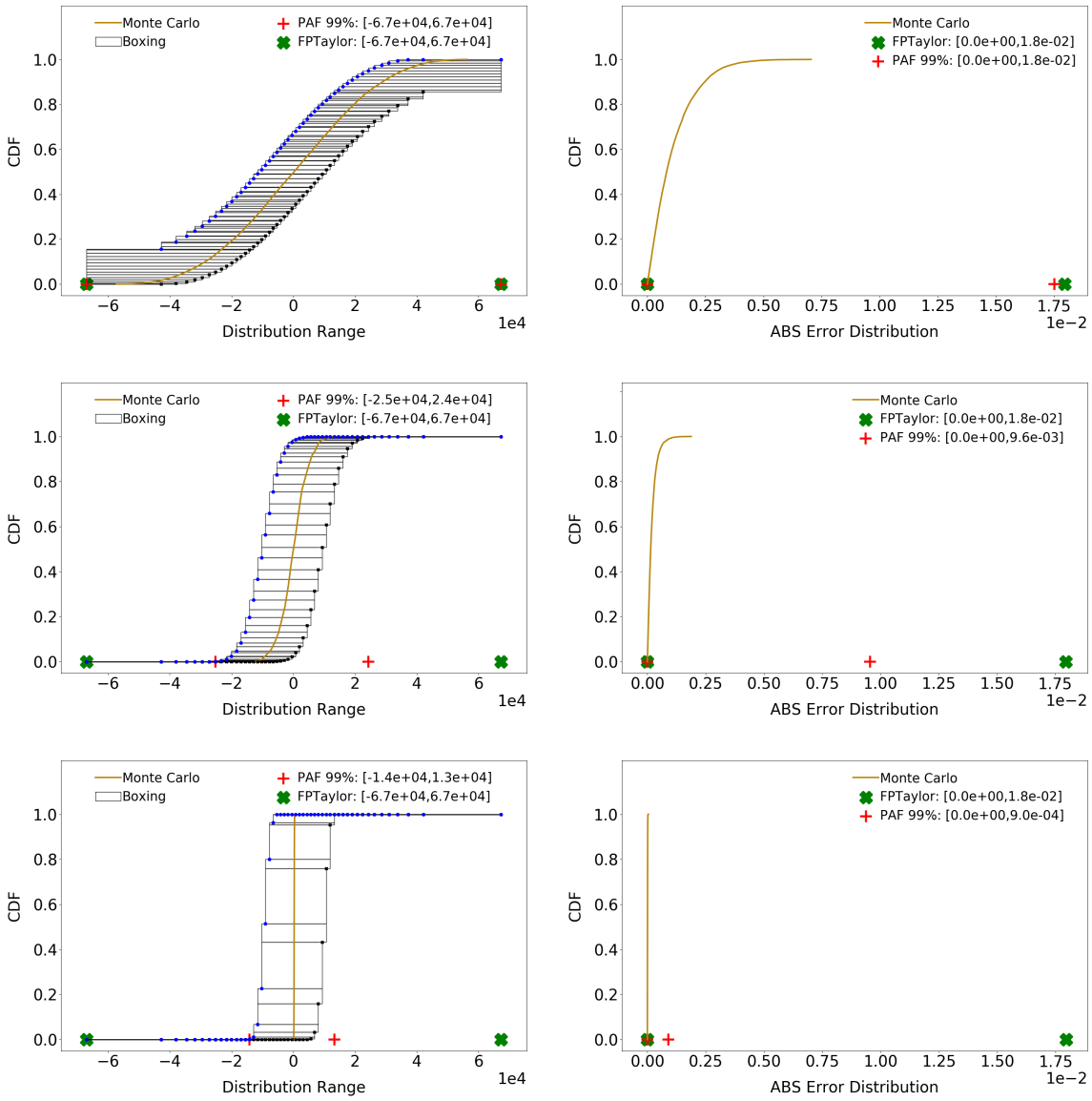

Fig. 5: CDFs of the range (left) and error (right) distributions for the benchmark *train3* for uniform (top), normal (center), and exp (bottom).