Rigorous Roundoff Error Analysis of Probabilistic Floating-Point Computations

George Constantinides[1], Fredrik Dahlqvist[1,2], Zvonimir Rakamarić[3], and Rocco Salvia[3(✉)]

[1] Imperial College London, London, UK
g.constantinides@ic.ac.uk
[2] University College London, London, UK
f.dahlqvist@ucl.ac.uk
[3] University of Utah, Salt Lake City, USA
{zvonimir,rocco}@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 Sect. 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 [34,42], optimization [43], finance [25], fluid dynamics [32], and computer graphics [30]. We

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".

A. Silva and K. R. M. Leino (Eds.): CAV 2021, LNCS 12760, pp. 626–650, 2021.
https://doi.org/10.1007/978-3-030-81688-9_29

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.

As an example, consider the floating-point arithmetic expression $(X+Y) \div Y$, where X and Y are random inputs represented by independent random variables. In Sect. 4, we first show how the computation in *finite-precision* of a single arithmetic operation such as $X + Y$ can be modeled 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 [48], 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 Sect. 6.

We can thus soundly bound the output distribution of any probabilistic computation, such as $(X+Y) \div Y$, performed in 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 Sect. 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, meaning conditioned on the computations not returning an outlier.

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 FPBench [11], and compared its range and error

analysis with the worst-case roundoff error analyzer FPTaylor [46,47] and the probabilistic roundoff error analyzer PrAn [36]. We present the results in Sect. 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 almost all benchmarks.

We summarize our contributions as follows:

(i) We derive a closed-form expression (6) 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 IEEE 754 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,[1] 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 Motivating Example

GPS sensors are inherently noisy. Bornholt [1] shows that the conditional probability of the true coordinates given a GPS reading is distributed according to a Rayleigh distribution. Interestingly, 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 neighborhood of those given by the GPS reading. This leads to errors, and hence the sensed coordinates should be corrected by adding a probabilistic error term which, on average, shifts the observed coordinates into an area of high probability for the true coordinates [1,2]. The latitude correction is given by:

$$\texttt{TrueLat} = \texttt{GPSLat} + ((\texttt{radius} * \texttt{sin}(\texttt{angle})) * \texttt{DPERM}), \tag{1}$$

where `radius` is Rayleigh distributed, `angle` uniformly distributed, `GPSLat` is the latitude, and `DPERM` a constant for converting 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, might want to run a rigorous worst-case floating-point analysis tool to determine which floating-point format is accurate enough to process GPS signals. This is mandatory if the developer requires rigorous error bounds holding with 100% certainty. The problem when analyzing a piece of code involving (1) is that the Rayleigh distribution has $[0, \infty)$ as its support, and *any* worst-case roundoff error analysis will return an infinite error bound in this situation. To get a meaningful (numeric) error bound, we need to truncate the support of the distribution. The most conservative truncation is $[0, max]$, where max is the largest representable number (not causing an overflow) at the target floating-point precision format.

[1] PAF is open source and publicly available at https://github.com/soarlab/paf.

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

Precision	Max	FPTaylor	PAF 100%	PAF 99.9999%	
				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

In Table 1, we report a detailed roundoff error analysis of (1) implemented in IEEE 754 double-, single-, and half-precision formats, with `GPSLat` set to the latitude of the Greenwich observatory. With each floating-point format, we associate the range $[0, max]$ of the truncated Rayleigh distribution. We compute worst-case roundoff error bounds for (1) with the state-of-the-art error analyzer FPTaylor [47] and with our tool PAF by setting the confidence interval to 100%. As expected, the error bounds from the two tools are identical. Finally, we compute the 99.9999% *conditional roundoff error* using PAF. This value is an upper bound to the roundoff error *conditioned* on the computation having landed in an interval capturing 99.9999% of all possible outputs. Column Absolute gives the error in degrees and Meters in meters ($1^\circ \approx$111km).

By looking at the results obtained without our *probabilistic error analysis* (columns FPTaylor and PAF 100%), 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 is many orders of magnitude smaller than the worst-case scenarios. Armed with this information, the developer can conclude that with a roundoff error of roughly 40 cm (4.1e−1 ms) when correcting 99.9999% of GPS latitude readings, working in single-precision is an adequate compromise between efficiency and accuracy of the computation.

This motivates the innovative concept of *probabilistic precision tuning*, evolved from standard worst-case precision tuning [5,12], to determine which floating-point format is the most appropriate for a given computation. As an example, let us do a probabilistic precision tuning exercise for the latitude correction computation of (1). We truncate the Rayleigh distribution in the interval $[0, 10^{307}]$, and assume we can tolerate up to 1e−5 roundoff error (roughly 1 m). First, we manually perform worst-case precision tuning using FPTaylor to determine that the minimal floating-point format not violating the given error bound needs 1022 mantissa and 11 exponent bits. Such large custom format is prohibitively expensive, in particular for devices performing frequent GPS readings, like smartphones or smartwatches. Conversely, when we manually perform probabilistic precision tuning using PAF with a confidence interval set to 99.9999%, we determine we need only 22 mantissa and 11 exponent bits. Thanks to PAF, the developer can provide a custom confidence interval of interest to the probabilistic precision tuning routine to adjust for the extremely unlikely corner cases like the ones we described for (1), and ultimately obtain more optimal tuning results.

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* $\text{Round} : \mathbb{R} \to \mathbb{F}$ 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 IEEE 754 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* $\text{err}_{\text{abs}} : \mathbb{R} \to \overline{\mathbb{R}}$ is defined as: $\text{err}_{\text{abs}}(x) = x - \text{Round}(x)$. We define the sets $\lfloor z, z \rceil \triangleq \{y \in \mathbb{R} \mid \text{Round}(y) = \text{Round}(z)\}$. 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 Sect. 4 (Eq. (5)) 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* $\text{err}_{\text{rel}} : \mathbb{R} \setminus \{0\} \to \overline{\mathbb{R}}$ is defined by

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

Note that $\text{err}_{\text{rel}}(x) = 1$ on $\lfloor 0, 0 \rceil \setminus \{0\}$, $\text{err}_{\text{rel}}(x) = \infty$ on $\lfloor -\infty, -\infty \rceil$ and $\text{err}_{\text{rel}}(x) = -\infty$ on $\lfloor \infty, \infty \rceil$. Recall also the fact [26] that $-2^{-(p+1)} < \text{err}_{\text{rel}}(x) < 2^{-(p+1)}$ 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 $\text{op} \in \{+, -, \times, \div\}$ an (infinite-precision) arithmetic operation, the traditional model of IEEE 754 floating-point arithmetic [26,39] states that the finite-precision implementation op_{m} of op must satisfy

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

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 [12] or FPTaylor [46, 47], 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 Sect. 2).

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

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

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 [41], but whilst *op. cit. postulates* that *dist* is the uniform distribution on $[-u, u]$, we compute *dist* from first principles in Sect. 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 – its *continuous part* – and a sum of Dirac deltas covering the rest of its distribution – its *discrete part*. We will encounter examples of such random variables in Sect. 4. Finally, 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 Sect. 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, which can be found in [48]. It is important to note that these operations 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 the formulas of [48] if X, Y are independent. However, the sub-terms $X+Y$ and Y cannot be interpreted in this way since $X+Y$ and Y are clearly not independent random variables.

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 as 'test subsets' the intervals $(-\infty, x]$ [44]. This order is called the *stochastic order*. It follows from the definition of the CDF that this order can be defined by simply 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 the arithmetic of [48]. 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* [19], and has already been used in the context of probabilistic roundoff errors in related work [3,36]. 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 collection (i.e., 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 [19], 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 Sect. 6.

4 Distribution of Floating-Point Roundoff Errors

Our tool PAF computes *probabilistic* roundoff errors by conditioning the maximization of symbolic affine form (presented in Sect. 5) 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. In other words, this section provides the foundations of *probabilistic range analysis*. All proofs can be found in the extended version [7].

4.1 Derivation of the Distribution of Rounding Errors

Recall the probabilistic model of Eq. (3) where op is an infinite-precision arithmetic operation and $\mathtt{op}_\mathtt{m}$ its finite-precision implementation:

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

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

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

Thus, *dist* is the distribution of the random variable $\mathrm{err}_{\mathrm{rel}}(z_1 \ \mathrm{op} \ z_2)$. 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 Sect. 3.1), rather than the interval $[-u, u]$.

To compute the density function of *dist*, we proceed as described in Sect. 3.2 by first computing the CDF $c(t)$ and then taking its derivative. Recall first from Sect. 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:

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

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 Sect. 3.2). It follows that the probability measure *dist* is given by

$$\mathit{dist}_c + \mathbb{P}\left[X \in \lfloor 0, 0 \rceil\right] \delta_1 + \mathbb{P}\left[X \in \lfloor -\infty, -\infty \rceil\right] \delta_\infty + \mathbb{P}\left[X \in \lfloor \infty, \infty \rceil\right] \delta_{-\infty} \quad (4)$$

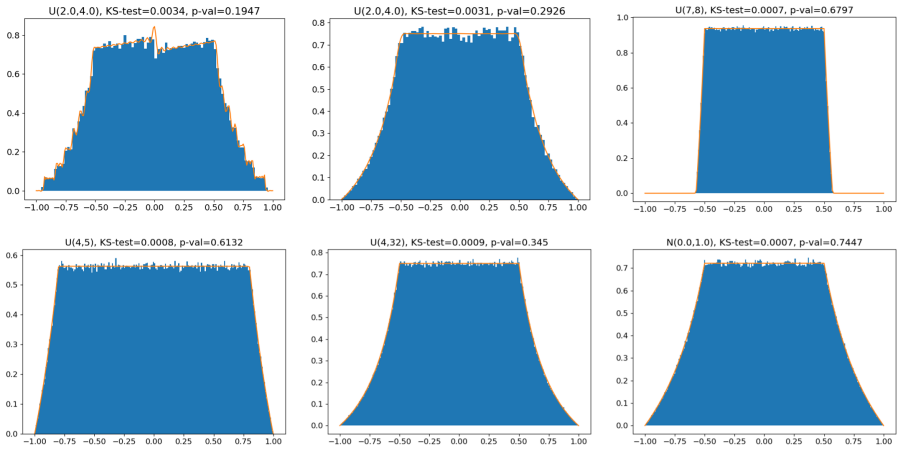

Fig. 1. Theoretical vs. empirical error distribution, clockwise from top-left: (i) Eq. (5) for $\mathsf{Unif}(2,4)$ 3 bit exponent, 4 bit significand, (ii) Eq. (5) for $\mathsf{Unif}(2,4)$ in half-precision, (iii) Eq. (6) for $\mathsf{Unif}(7,8)$ in single-precision, (iv) Eq. (6) for $\mathsf{Unif}(4,5)$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

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. However in order to be sound, we must in general include these three discrete components to our computations. The density $dist_c$ is given explicitly by the following result whose proof can already be found in [9].

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*

$$d(t) = \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right) \frac{u\,|z|}{(1-tu)^2}, \tag{5}$$

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

Figure 1 (i) and (ii) shows an implementation of Eq. (5) 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. (5). However, this rapidly becomes prohibitively computationally expensive for higher precisions (since the number of floating-point representable numbers grows exponentially).

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. (5), 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. (5).

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*

$$d_{hp}(t) = \begin{cases} \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(2-u)} \frac{|x|}{2^{e+1}} f(x)\, dx & |t| \le \frac{1}{2} \\ \frac{1}{1-tu} \sum_{s,e=e_{min}+1}^{e_{max}-1} \int_{(-1)^s 2^e(1-u)}^{(-1)^s 2^e(\frac{1}{|t|}-u)} \frac{|x|}{2^{e+1}} f(x)\, dx & \frac{1}{2} < |t| \le 1 \end{cases} \tag{6}$$

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. (6) reduces the sum over *all* floating-point representable numbers in Eq. (5) to a sum over *the exponents* by exploiting the regularity of f. Note also that since f is a PDF, it usually decreases very quickly away from 0, and its derivative decreases even quicker and $|R|$ thus tends to be very small and $|R| \to 0$ as the precision $p \to \infty$.

Figure 1 shows Eq. (6) for: (i) the distribution $\mathsf{Unif}(7, 8)$ where large significands are more likely, (ii) the distribution $\mathsf{Unif}(4, 5)$ where small significands are more likely, (iii) the distribution $\mathsf{Unif}(4, 32)$ where significands are equally

likely, and (iv) the distribution $\mathsf{Norm}(0,1)$ with infinite support. The graphs show the density function given by Eq. (6) 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. 1 are very similar, despite being computed from very different input distributions. What they have in common is that their input distributions have the property that all significands in their supports are equally likely. We show that under this assumption, the distribution of roundoff errors given by Eq. (5) converges to a unique density as the precision increases, irrespective of the input distribution! Since significands are frequently equiprobable (it is the case for a third of our benchmarks), 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. 2 and formalize it with the following theorem, which can mostly be found in [9].

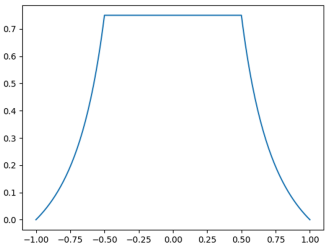

Fig. 2. Typical distribution.

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*

$$d_{typ}(t) \triangleq \lim_{p\to\infty} d(t) = \begin{cases} \frac{3}{4} & |t| \leq \frac{1}{2} \\ \frac{1}{2}\left(\frac{1}{t}-1\right) + \frac{1}{4}\left(\frac{1}{t}-1\right)^2 & |t| > \frac{1}{2} \end{cases} \tag{7}$$

where $d(t)$ is the exact density given by Eq. (5).

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

$$\mathrm{Cov}(X, \mathrm{err}_{\mathrm{rel}}(X)) = \mathbb{E}\left[X.\mathrm{err}_{\mathrm{rel}}(X)\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[\mathrm{err}_{\mathrm{rel}}(X)\right] \tag{8}$$

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

$$\mathbb{E}[X.\mathrm{err}_{\mathrm{rel}}(X)] = \int_{\mathbb{R}^2} xut \; d\mathbb{P} \tag{9}$$

but it is necessary to use some more advanced probability theory. We will make the simplifying assumption that the density of X is constant on 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 it is messy, and 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 artifact 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 Sect. 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. (5) to compute the error distribution. However, in high-precision, approximations (typically extremely good) like Eqs. (6) and (7) must be used. In order to remain sound in the implementation of our model (see Sect. 6) we must account for the error made by this approximation. We have not got the space to discuss the error made by Eq. (7), but taking the term $|R|$ of Theorem 2 as an illustration, we can use the notion of p-box described in Sect. 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. (6) by using the p-box

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

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 Sect. 6.3. Affine arithmetic [6] is a model for range analysis that extends classic interval arithmetic [40] 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

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i, \qquad \text{where } \epsilon_i \in [-1, 1]. \tag{10}$$

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 [49].

Sound Error Analysis with Symbolic Affine Arithmetic. We now show how the roundoff errors get propagated through the four arithmetic operations. 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 [46].

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.

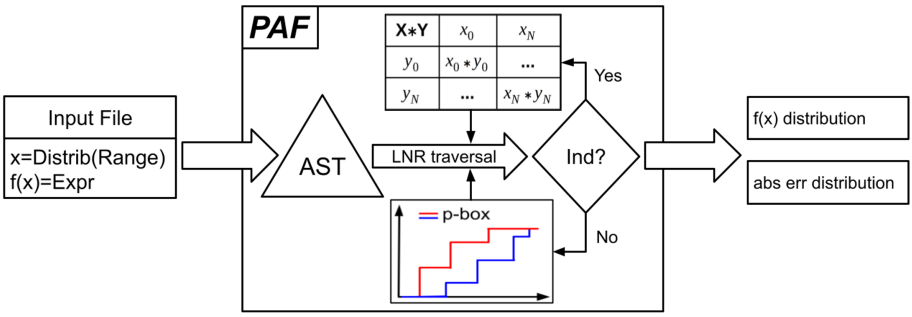

Fig. 3. Toolflow of PAF.

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). Figure 3 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:

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

Following [8,31], 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), and the 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 Sect. 4 to work. Recall first the probabilistic model from Eq. (3):

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

In Sect. 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:

$$[\![\mathtt{t}_1 \ \mathtt{op_m} \ \mathtt{t}_2]\!] \triangleq ([\![\mathtt{t}_1]\!] \ \mathrm{op} \ [\![\mathtt{t}_2]\!]) \times (1 + \mathrm{err}_{\mathrm{rel}}([\![\mathtt{t}_1]\!] \ \mathrm{op} \ [\![\mathtt{t}_2]\!])) \tag{11}$$

To evaluate the model of Eq. (11), we first use the appropriate closed-form expression Eqs. (5) to (7) derived in Sect. 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 Sect. 4.5. We then use Theorem 4 to justify evaluating the multiplication operation in Eq. (11) *independently*—that is to say by using [48]—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.

We now introduce the algorithm for evaluating the model given in Eq. (11). 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 Sect. 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 (Sect. 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].

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 use an

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 ([x1, x2], p_x) ∈ DS_X do
4:         for all ([y1, y2], p_y) ∈ DS_Y do
5:             [z1, z2] = [x1, x2] op [y1, y2]          ▷ operation between intervals
6:             [z1', z2'] = SMT.prune([z1, z2])
7:             if SMT.check(trace_X ∧ trace_Y ∧ [x1, x2] ∧ [y1, y2]) is SAT then
8:                 p_Z = unknown-probability
9:             else
10:                p_Z = 0
11:            DS_Z.append(([z1', z2'], p_Z))
12:    trace_Z = trace_X ∪ trace_Y ∪ {Z = X op Y}
13:    return DS_Z, trace_Z
```

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. With the solver, we move from an unknown to a *partially known* dependency between the operands. Currently, PAF supports the Z3 [17] and dReal [23] 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 is encoded as a *linear programming* routine exactly as in [3].

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 Sect. 5) with the probabilistic range analysis described above. The symbolic error form gets maximized conditioned on the results of all the intermediate operations

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 = Ø
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
```

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 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 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 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 and avoids the unlikely outliers that cause large roundoff errors into the final 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 intermediate operations (line 13). The maximization is done using the rigorous global optimizer Gelpia [24].

7 Experimental Evaluation

We evaluate PAF (version 1.0.0) on the standard FPBench benchmark suite [11, 20] that uses the four basic operations we currently support $\{+, -, \times, \div\}$. Many of these benchmarks were also used in recent related work [36] 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 (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.

To assess the accuracy and performance of PAF, we compare it with PrAn (commit 7611679 [10]), the current state-of-the-art tool for automated analysis of probabilistic roundoff errors [36]. PrAn currently supports only uniform and normal distributions. We run all 6 tool configurations and report the best result for each benchmark. 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 (Sect. 6.3) and of PrAn's probabilistic error. We also compare our probabilistic error bounds against FPTaylor (commit efbbc83 [21]), 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.

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 24 out of 27 benchmarks, for 2 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 all the benchmarks. Unlike PrAn, PAF supports probabilistic output range analysis as well. We present these results in the extended version [7].

In Table 2, of particular interest are benchmarks (10 for normal and 18 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, since all inputs are equally likely. The standard normal distribution concentrates more than 99% of probability mass in the interval $[-3, 3]$, 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.

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.

Benchmark	Uniform		Normal		Exp	FpTaylor
	PAF	PrAn	PAF	PrAn	PAF	
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

However, when the normal distribution gets truncated to a wider range (e.g., $[-100, 100]$ for *trids*), 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.

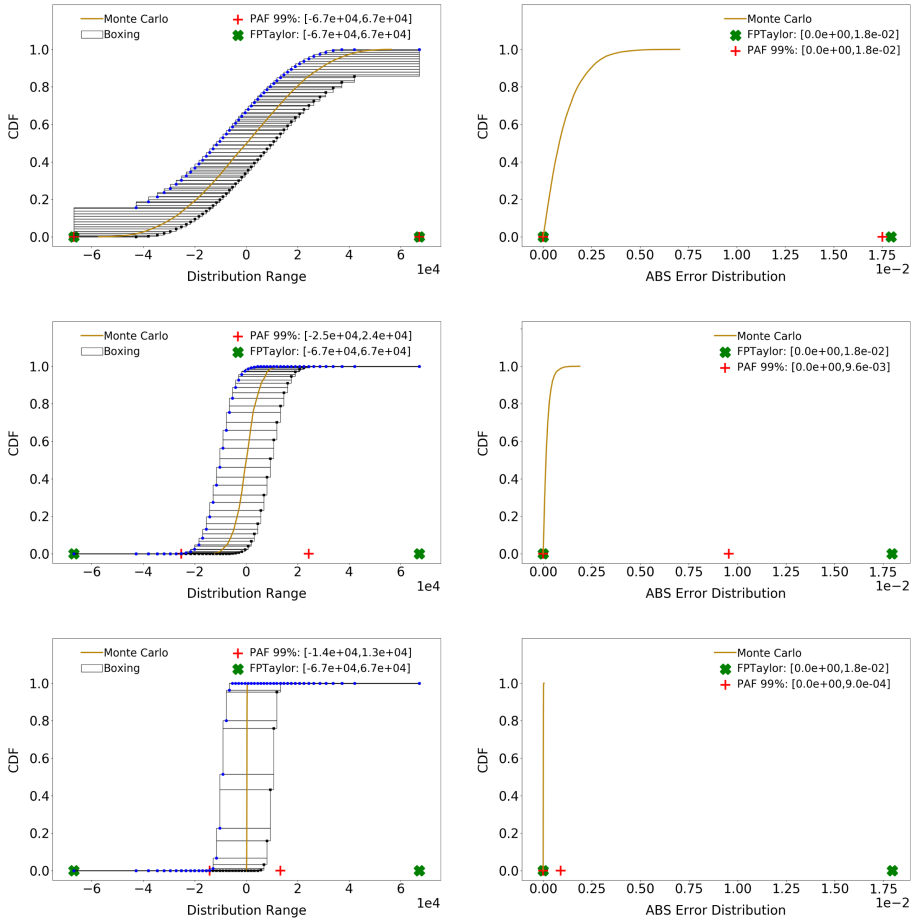

Fig. 4. CDFs of the range (left) and error (right) distributions for the benchmark *traincars3* for uniform (top), normal (center), and exp (bottom).

The runtimes of PAF vary between 10 min 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 h and *filters* with 6 h. 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 h, *sqrt* with 3 h, and *sine* with 11 h. Note that neither PAF nor PrAn are memory intensive.

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.

Figure 4 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 PAF can significantly improve both the output range and error bounds over FPTaylor. This again illustrates 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 p-boxes and error bounds from PAF follow their respective empirical estimations.

8 Related Work

Our work draws inspiration from *probabilistic affine arithmetic* [3,4], which aims to bound probabilistic uncertainty propagated through a computation; a similar goal to our probabilistic range analysis. This was recently extended to polynomial dependencies [45]. On the other hand, PAF detects any non-linear dependency supported by the SMT solver. While these approaches show how to bound moments, we do not consider moments but instead compute conditional roundoff error bounds, a concern specific to the analysis of floating-point computations. Finally, the concentration of measure inequalities [4,45] provides bounds for (possibly very large) problems that can be expressed as sums of random variables, for example multiple increments of a noisy dynamical system, but are unsuitable for typical floating-point computations (such as FPBench benchmarks).

The most similar approach to our work is the recent static probabilistic roundoff error analysis called PrAn [36]. PrAn also builds on [3], and inherits the same limitations in dealing with dependent operations. Like us, PrAn 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 the PrAn approach. In contrast, we take the input variables to be user-specified random variables, and show how the distribution of each error term can be computed directly and exactly from the random variables generating it (Sect. 4). Furthermore, unlike PrAn, PAF leverages the non-correlation between random variables and the corresponding error distribution (Sect. 4.4). Thus, PAF performs the rounding in Eq. (3) as an *independent*

operation. Putting these together leads to PAF computing tighter probabilistic roundoff error bounds than PrAn, as our experiments show (Sect. 7).

The idea of using a probabilistic model of rounding errors to analyze *deterministic* computations can be traced back to Von Neumann and Goldstine [51]. Parker's so-called 'Monte Carlo arithmetic' [41] is probably the most detailed description of this approach. We, however, consider *probabilistic* computations. For this reason, the famous critique of the probabilistic approach to roundoff errors [29] does not apply to this work. Our preliminary report [9] presents some early ideas behind this work, including Eqs. (5) and (7) and a very rudimentary range analysis. However, this early work manipulated distributions *unsoundly*, could not handle any repeated variables, and did not provide any roundoff error analysis. Recently, probabilistic roundoff error models have also been investigated using the concentration of measure inequalities [27,28]. Interestingly, this means that the distribution of errors in Eq. (3) can be left almost completely unspecified. However, as in the case of related work from the beginning of this section [4,45], concentration inequalities are very ill-suited to the applications captured by the FPBench benchmark suite.

Worst-case analysis of roundoff errors has been an active research area with numerous published approaches [12–16,18,22,33,35,37,38,46,47,50]. Our symbolic affine arithmetic used in PAF (Sect. 5) evolved from rigorous affine arithmetic [14] 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 [46,47]. Hence, PAF with the 100% confidence interval leads to the same worst-case roundoff error bounds as computed by FPTaylor (Sect. 7).

Acknowledgments. We thank Ian Briggs and Mark Baranowski for their generous and prompt support with Gelpia. We also thank Alexey Solovyev for his detailed feedback and suggestions for improvements. Finally, we thank the anonymous reviewers for their insightful reviews that improved our final version, and program chairs for carefully guiding the review process.

References

1. Bornholt, J.: Abstractions and techniques for programming with uncertain data. Undergraduate honours thesis, Australian National University (2013)
2. Bornholt, J., Mytkowicz, T., McKinley, K.S.: Uncertain <T>: a first-order type for uncertain data. In: ASPLOS (2014)
3. Bouissou, O., Goubault, E., Goubault-Larrecq, J., Putot, S.: A generalization of p-boxes to affine arithmetic. Computing **89**, 189–201 (2012). https://doi.org/10.1007/s00607-011-0182-8
4. Bouissou, O., Goubault, E., Putot, S., Chakarov, A., Sankaranarayanan, S.: Uncertainty propagation using probabilistic affine forms and concentration of measure inequalities. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 225–243. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_13

5. Chiang, W.F., Baranowski, M., Briggs, I., Solovyev, A., Gopalakrishnan, G., Rakamarić, Z.: Rigorous floating-point mixed-precision tuning. In: POPL (2017)
6. Comba, J.L.D., Stolfi, J.: Affine arithmetic and its applications to computer graphics. In: SIBGRAPI (1993)
7. Constantinides, G., Dahlqvist, F., Rakamarić, Z., Salvia, R.: Rigorous roundoff error analysis of probabilistic floating-point computations (2021). arXiv:2105.13217
8. Dahlqvist, F., Kozen, D.: Semantics of higher-order probabilistic programs with conditioning. In: POPL (2019)
9. Dahlqvist, F., Salvia, R., Constantinides, G.A.: A probabilistic approach to floating-point arithmetic. In: ASILOMAR (2019). Non-peer-reviewed extended abstract
10. Daisy. https://github.com/malyzajko/daisy
11. Damouche, N., Martel, M., Panchekha, P., Qiu, C., Sanchez-Stern, A., Tatlock, Z.: Toward a standard benchmark format and suite for floating-point analysis. In: Bogomolov, S., Martel, M., Prabhakar, P. (eds.) NSV 2016. LNCS, vol. 10152, pp. 63–77. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54292-8_6
12. Darulova, E., Izycheva, A., Nasir, F., Ritter, F., Becker, H., Bastian, R.: Daisy - framework for analysis and optimization of numerical programs (tool paper). In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10805, pp. 270–287. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89960-2_15
13. Darulova, E., Kuncak, V.: Trustworthy numerical computation in Scala. In: OOPSLA (2011)
14. Darulova, E., Kuncak, V.: Sound compilation of reals. In: POPL (2014)
15. Das, A., Briggs, I., Gopalakrishnan, G., Krishnamoorthy, S., Panchekha, P.: Scalable yet rigorous floating-point error analysis. In: SC (2020)
16. Daumas, M., Melquiond, G.: Certification of bounds on expressions involving rounded operators. ACM Trans. Math. Softw. **37**, 1–20 (2010)
17. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
18. Delmas, D., Goubault, E., Putot, S., Souyris, J., Tekkal, K., Védrine, F.: Towards an industrial use of FLUCTUAT on safety-critical avionics software. In: Alpuente, M., Cook, B., Joubert, C. (eds.) FMICS 2009. LNCS, vol. 5825, pp. 53–69. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04570-7_6
19. Ferson, S., Kreinovich, V., Grinzburg, L., Myers, D., Sentz, K.: Constructing probability boxes and dempster-shafer structures. Technical report, Sandia National Lab (2015)
20. FPBench: standards and benchmarks for floating-point research. https://fpbench.org
21. FPTaylor. https://github.com/soarlab/fptaylor
22. Fu, Z., Bai, Z., Su, Z.: Automated backward error analysis for numerical code. In: OOPSLA (2015)
23. Gao, S., Kong, S., Clarke, E.M.: dReal: an SMT solver for nonlinear theories over the reals. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 208–214. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_14
24. Gelpia: a global optimizer for real functions. https://github.com/soarlab/gelpia
25. Glasserman, P.: Monte Carlo Methods in Financial Engineering. Springer, Heidelberg (2013). https://doi.org/10.1007/978-0-387-21617-1
26. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. SIAM (2002)
27. Higham, N.J., Mary, T.: A New Approach to Probabilistic Rounding Error Analysis. SISC (2019)

28. Ipsen, I.C.F., Zhou, H.: Probabilistic error analysis for inner products. SIMAX (2019)
29. Kahan, W.: The improbability of probabilistic error analyses for numerical computations (1996)
30. Kajiya, J.T.: The rendering equation. In: SIGGRAPH (1986)
31. Kozen, D.: Semantics of probabilistic programs. In: JCSS (1981)
32. Landau, D.P., Binder, K.: A Guide to Monte Carlo Simulations in Statistical Physics. Cambridge University Press, Cambridge (2014)
33. Lee, W., Sharma, R., Aiken, A.: Verifying bit-manipulations of floating-point. In: PLDI (2016)
34. Lepage, G.P.: VEGAS – an adaptive multi-dimensional integration program. Technical report, Cornell (1980)
35. Linderman, M.D., Ho, M., Dill, D.L., Meng, T.H., Nolan, G.P.: Towards program optimization through automated analysis of numerical precision. In: CGO (2010)
36. Lohar, D., Prokop, M., Darulova, E.: Sound probabilistic numerical error analysis. In: IFM (2019)
37. Magron, V., Constantinides, G., Donaldson, A.: Certified roundoff error bounds using semidefinite programming. TOMS **43**, 1–31 (2017)
38. Martel, M.: RangeLab: a static-analyzer to bound the accuracy of finite-precision computations. In: SYNASC (2011)
39. Microprocessor Standards Committee of the IEEE Computer Society: IEEE Standard for Floating-Point Arithmetic (2019)
40. Moore, R.E.: Interval Analysis. Prentice-Hall, Hoboken (1966)
41. Parker, D.S., Pierce, B., Eggert, P.R.: Monte Carlo arithmetic: how to gamble with floating point and win. Comput. Sci. Eng. **2**, 58–68 (2000)
42. Press, W.H., Farrar, G.R.: Recursive stratified sampling for multidimensional Monte Carlo integration. Comput. Phys. **4**, 190–195 (1990)
43. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, Cambridge (1988)
44. Rothschild, M., Stiglitz, J.E.: Increasing risk: I. A definition. J. Econ. Theory **2**, 225–243 (1970)
45. Sankaranarayanan, S., Chou, Y., Goubault, E., Putot, S.: Reasoning about uncertainties in discrete-time dynamical systems using polynomial forms. In: NeurIPS (2020)
46. Solovyev, A., Baranowski, M.S., Briggs, I., Jacobsen, C., Rakamarić, Z., Gopalakrishnan, G.: Rigorous estimation of floating-point round-off errors with Symbolic Taylor Expansions. TOPLAS **41**, 1–39 (2018)
47. Solovyev, A., Jacobsen, C., Rakamarić, Z., Gopalakrishnan, G.: Rigorous estimation of floating-point round-off errors with Symbolic Taylor Expansions. In: FM (2015)
48. Malik, H.J.: The Algebra of Random Variables (Springer, MD). Wiley (1979)
49. Stolfi, J., Figueiredo, L.H.D.: Self-validated numerical methods and applications. In: IMPA (1997)
50. Titolo, L., Feliú, M.A., Moscato, M., Muñoz, C.A.: An abstract interpretation framework for the round-off error analysis of floating-point programs. In: VMCAI (2018)
51. Von Neumann, J., Goldstine, H.H.: Numerical inverting of matrices of high order. Bull. Am. Math. Soc. **53**, 1021–1099 (1947)