A Probabilistic Approach to Floating Point Arithmetic

Fredrik Dahlqvist
Department of Electrical and Electronic Engineering
Imperial College London
f.dahlqvist09@imperial.ac.uk

Rocco Salvia
School of Computing
University of Utah
rocco@cs.utah.edu

George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London
g.constantinides@imperial.ac.uk

***Abstract*—Finite-precision floating point arithmetic unavoidably introduces rounding errors which are traditionally bounded using a worst-case analysis. However, worst-case analysis might be overly conservative because worst-case errors can be extremely rare events in practice. Here we develop a probabilistic model of rounding errors with which it becomes possible to estimate the likelihood that the rounding error of an algorithm lies within a given interval. Given an input distribution, we show how to compute the distribution of rounding errors. We do this exactly for low precision arithmetic, for high precision arithmetic we derive a simple approximation. The model is then entirely compositional: given a numerical program written in a simple imperative programming language we can recursively compute the distribution of rounding errors at each step of the computation and propagate it through each program instruction. This is done by applying a formalism originally developed by Kozen to formalize the semantics of probabilistic programs. We then discuss an implementation of the model and use it to perform probabilistic range analyses on some benchmarks.**

I. Introduction

IEEE arithmetic [13] is traditionally modelled mathematically as follows [5]: if x, y are two normal floating-point numbers and $\mathrm{op} \in \{+, -, \times, \div\}$ is an infinite-precision arithmetic operation, then the floating-point precision implementation $\mathtt{op}_\mathtt{m}$ of op must satisfy:

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

where u is the unit roundoff for the given precision. Eq. (1) says that the machine implementation of an arithmetic operation can induce a relative error of size δ *for some* $\delta \in [-u, u]$. The '*for some*' is essential: this is a *non-deterministic model*, we have no control whatsoever over which δ appears in Eq. (1). This means that numerical analysis based on this model must consider *all* possible values δ, i.e. numerical analysis based on Eq. (1) is fundamentally a *worst-case analysis*.

It also follows from the perspective of Eq. (1) that any program doing arithmetic is, under this model, a non-deterministic program. Moreover, since the output of such a program might very well turn out to be the input of another program doing arithmetic, one should also consider non-deterministic inputs. This is precisely what happens in practice with tools for numerical analysis like the recent [12], Daisy [4] or FPTaylor [15] which require for each variable of the program a range of possible values in order to perform a worst-case analysis.

However, for a wide variety of programs it makes sense to assume that the inputs are *probabilistic* rather than non-deterministic; that is to say we have some statistical model of the inputs of the program. This situation is in fact incredibly common. The inputs of one numerical routine are frequently generated randomly by another numerical routine, for example in a gradient descent optimization, a Bayesian inference algorithm, or a stochastic ray tracing algorithm. Similarly, sensors on a cyber-physical system can feed analog signals which are very well modelled statistically, to a numerical program processing these signals.

If the inputs of a program have a known distribution, then it becomes possible, at least in principle, to ask the question: *How likely are the inputs generating the worst-case rounding errors obtained from the non-deterministic model of Eq.* (1)*?* Typically, these inputs will occur very infrequently, and in this respect the non-deterministic model can be overly pessimistic since worst-case behaviours might be such rare events that they are never encountered in practice.

In this paper we will explore a quantitative model which formally looks very similar to Eq. (1), namely

$$x \ \mathtt{op}_\mathtt{m} \ y = (x \ \mathrm{op} \ y)(1+\delta), \qquad \delta \sim dist \tag{2}$$

but now δ is *sampled* from $dist$, a probability distribution whose support is $[-u, u]$. In other words we move from a non-deterministic model of rounding errors to a *probabilistic* model of rounding errors. This model will allow us to formalise and answer questions like: *What is the distribution of outputs when rounding errors are taken into account? What is the average rounding error? What is the worst-case error with* 99.9% *probability?*

As was mentioned above, model (1) amounts to saying that any numerical program is a non-deterministic program. Completely analogously, in the perspective of Eq. (2) every numerical program is a *probabilistic program*, that is to say a program which admits sampling as a native instruction. The study of probabilistic programs goes back to Kozen [10] which modelled simple `while` programs containing an *explicit* sampling instruction `random()`. In our setting any numerical program becomes a probabilistic program via an *implicit* sampling operation which takes place whenever an arithmetic operation is performed. This implicit sampling is

the only difference with the standard setting of [10], and we will otherwise understand how programs process randomness by following the framework laid out in [10]. The study of probabilistic programs has recently witnessed a resurgence of interest driven by new applications in machine learning and statistical analysis of large datasets.

Related works: The probabilistic model of Eq. (2) is not new, it can be traced back to von Neumann and Goldstine [17] and is very similar to the so-called Monte-Carlo arithmetic of [14]. Within the signal processing community, simple probabilistic models of roundoff error are commonplace. Constantinides *et al.* [2] study the propagation of fixed-point roundoff error through linear time-invariant computation, resulting in propagation of the first and second statistical moments of the distributions. More recently, the model of Eq. (2) has been investigated by Higham [6] and Ipsen [7]. Interestingly, because [6] and [7] are interested in large-dimensional problems, neither work needs to explicitly specify the distribution $dist$ in Eq. (2). Instead, [6] requires that each sample from $dist$ be independent and that $\mathbb{E}[\delta] = 0$, whilst [7] just requires that $\mathbb{E}[\delta] = 0$. By using concentration of measure inequalities the authors then obtain probabilistic bounds which are independent of any particular choice of distribution. These bounds however are only applicable to a small class of problem (inner product, matrix multiplication and matrix factorisation algorithms) with very large inputs. Here we will derive a principled distribution $dist$ for the relative error δ and build a tool implementing the probabilistic model Eq. (2) to any small to medium-sized programs in a systematic way.

In [11] the authors propose a hybrid approach: first, they discretize input distributions and represent them as a dictionary which map each sub-interval to the corresponding probability (focal elements). On each sub-interval they combine probabilistic affine arithmetic, to propagate the error terms through the AST of the program, together with worst-case static analysis to bound the imprecision error term separately for each focal element. Their methodology results in a sound probabilistic error analysis, since the (unknown) error distribution is always bounded between proper upper and lower bounds by the analysis, but they inherit limitations of worst-case analysis in case of overflow and division by zero.
Our approach does not rely on worst-case error estimation, so it is a pure probabilistic model for the floating-point error of an expression.

II. Two probabilistic models of rounding errors

In order to use the probabilistic model given by Eq. (2) we must specify the distribution $dist$ of the random variable δ. In this section we will show how to derive the distribution of rounding errors from first principles. This will yield a distribution which is computable for low precisions (e.g. half-precision and lower) but becomes prohibitively expensive computationally for single- and double-precision. We will then show that very often the rounding error distribution can be approximated remarkably well by a simple distribution which we shall call the *typical distribution*. The quality of this approximation increases with the working precision and we thus derive both an exact, computable model of rounding errors for low-precisions, and a simple but good approximating model of rounding errors for high-precisions.

A. The exact rounding error distribution

Conceptually, the key to our approach is to model the rounding operation probabilistically, i.e. as an operation which adds a probabilistic relative error via

$$x \longrightarrow x(1+\delta) \qquad \delta \sim dist. \tag{3}$$

Since each IEEE arithmetic operation can be understood as implicitly performing a rounding operation on the corresponding infinite-precision operation, the probabilistic rounding above naturally yields Eq. (2). The key is thus to find a good candidate for the distribution $dist$ governing probabilistic rounding.

As discussed in the introduction, we consider numerical programs as probabilistic programs. In particular, all inputs come with probability distributions, and we should consider the variable x in Eq. (3) as a sample from a real random variable X with known probability distribution $\mathbb{P}$. It is then completely natural to require that:

$$\frac{X - \mathrm{Round}(X)}{X} \;\sim\; dist$$

i.e. $dist$ describes the distribution of the actual, deterministic rounding error of samples drawn from X. We will now explicitly compute $dist$. First we introduce some convenient notation. We define

$$\lceil x \rceil \triangleq \sup\{z \in \mathbb{R} \mid \mathrm{Round}(z) = \mathrm{Round}(x)\}$$
$$\lfloor x \rfloor \triangleq \inf\{z \in \mathbb{R} \mid \mathrm{Round}(z) = \mathrm{Round}(x)\}.$$

Whether $\lceil x \rceil$ is the maximal real which rounds to the same value as x, or just the supremum of this set, will in general depend both on x and on the rounding convention, and similarly for $\lfloor x \rfloor$. We also define the sets

$$\left\lfloor x, x \right\rceil \triangleq \{z \in \mathbb{R} \mid \mathrm{Round}(z) = \mathrm{Round}(x)\}$$

In particular if z is a floating-point representable number – notation $z \in \mathbb{F}$ – then $\lfloor z, z \rceil$ is the collection of all reals rounding to z.

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 normalize the distribution since the absolute value of the relative error is strictly bounded by u. In other words, we express the relative error as a distribution on $[-1, 1]$ rather than $[-u, u]$. In order to compute the density function of $dist$ we proceed in the standard way by first computing the cumulative distribution function $c(t)$ and then taking its derivative. We therefore start by computing

$$\begin{aligned} c(t) &\triangleq \mathbb{P}\left[\frac{X - \widehat{X}}{X} \leq tu\right] \\ &= \mathbb{P}\left[\bigvee_{z \in \mathbb{F}} \left(\frac{X - z}{X} \leq tu \wedge X \in \left\lfloor z, z \right\rceil\right)\right] \end{aligned}$$

We now need to consider three special cases:

1) If $X \in \lfloor 0, 0 \rceil$ then $\frac{X-\widehat{X}}{X} = 1$ and thus (since $tu < 1$):

$$\mathbb{P}\left[\frac{X-0}{X} \leq tu \wedge X \in \lfloor 0, 0 \rceil\right] = 0 \tag{4}$$

2) If $X \in \lfloor -\infty, -\infty \rceil$ then $\frac{X-\widehat{X}}{X} = \infty$ and thus

$$\mathbb{P}\left[\frac{X+\infty}{X} \leq tu \wedge X \in \lfloor -\infty, -\infty \rceil\right] = 0 \tag{5}$$

3) Finally, if $X \in \lfloor \infty, \infty \rceil$ then $\frac{X-\widehat{X}}{X} = -\infty$ and thus

$$\mathbb{P}\left[\frac{X-\infty}{X} \leq tu \wedge X \in \lfloor \infty, \infty \rceil\right] = \mathbb{P}\left[X \in \lfloor \infty, \infty \rceil\right] \tag{6}$$

Using the fact that (4)-(6) yield expressions which are independent of t we get the density

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

where $\mathbb{F}^+$ and $\mathbb{F}^-$ denote the positive (resp. negative) floating-point representable numbers. Suppose now that X is described by a probability density function $f : \mathbb{R} \to \mathbb{R}$, we then get:

$$\begin{aligned} d(t) =& \sum_{z \in \mathbb{F}^- \setminus \{-\infty, 0\}} \frac{\partial}{\partial t} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) \int_{\lfloor z \rfloor}^{\frac{z}{1-tu}} f(s)\, ds + \\ & \sum_{z \in \mathbb{F}^+ \setminus \{0, \infty\}} \frac{\partial}{\partial t} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) \int_{\frac{z}{1-tu}}^{\lceil z \rceil} f(s)\, ds \\ =& \sum_{z \in \mathbb{F}^- \setminus \{-\infty, 0\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right) \frac{-uz}{(1-tu)^2} \\ & + \sum_{z \in \mathbb{F}^+ \setminus \{0. \infty\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right) \frac{uz}{(1-tu)^2} \\ =& \sum_{z \in \mathbb{F} \setminus \{-\infty, 0, \infty\}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z}{1-tu}\right) f\left(\frac{z}{1-tu}\right) \frac{u|z|}{(1-tu)^2} \end{aligned} \tag{7}$$

where $\mathbb{1}_A(x)$ is the usual indicator function which returns 1 if $x \in A$ and 0 otherwise. For low precisions, that is to say up to half-precision (5 bits exponent, 10 bits mantissa), it is perfectly possible to explicitly go through all floating point numbers and compute the density of the rounding error distribution $dist$ by using Eq. (7). However this rapidly becomes too computationally expensive for higher-precision (since the number of floating-point representable numbers grows exponentially).

B. The typical rounding error distribution

Interestingly, when computing the error density Eq. (7) for a wide variety of well-known input distribution, one very often obtains more or less the same curve. This phenomenon is illustrated in Fig. 1 where the half-precision error density computed via Eq. (7) is displayed for uniform distributions over $[-10, 10]$ and $[0, 1]$ and for normal distributions with parameters $\mu = 0, \sigma = 2$ and $\mu = 2, \sigma = 10$ respectively. The reader will notice immediately that all the curves are nearly identical. In this section we will sketch how, under

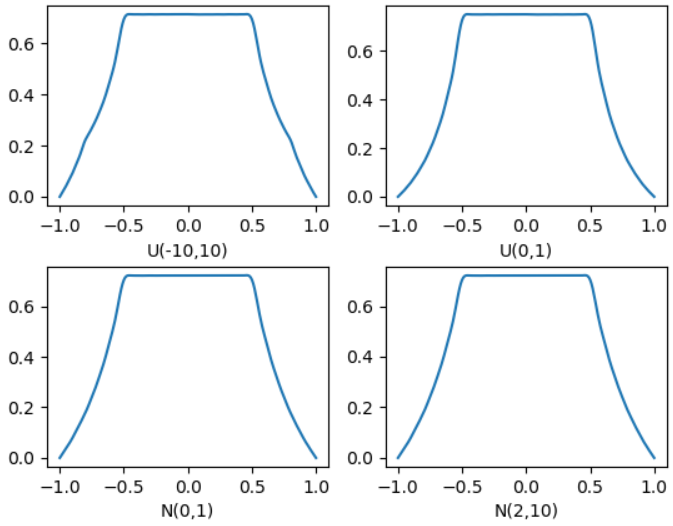

Fig. 1. Half-precision relative error distribution for four typical input distributions

some regularity assumptions about the input distribution, the error density of Eq. (7) can be approximated by a simple, piecewise polynomial curve which we shall call the *typical error distribution*. The precise mathematical derivation of this curve being relatively long, we refer the reader to XXX for the full details. Here we will simply specify conditions under which the error distribution given by Eq. (7) is close to the typical distribution. The key observations which we shall make are that (a) the quality of the approximation increases with the working precision (henceforth p), and (b) the likelihood of the assumptions being satisfied also increases with p.

Let $z(e, s, k)$ denote the floating-point representable real with exponent e, sign $(-1)^s$ and mantissa k, and let e_{min} and e_{max} denote the smallest and largest exponents respectively. Given a $t \in [-1, 1]$, one can show that the mantissas such that

$$\mathbb{1}_{\lfloor z(e,s,k), z(e,s,k) \rceil}\left(\frac{z(e,s,k)}{1-tu}\right) = 1 \tag{8}$$

are given by

$$k \leq 2^p \left(\frac{1}{|t|} - 1\right) - \frac{1}{2} \tag{9}$$

Note that for $t \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ Eq. (9) always holds, i.e. all mantissa are compatible with Eq. (8). We can now specify our assumptions.

Assumption 0. The probability density function is constant at the scale of the intervals between floating point numbers, i.e.

$$\begin{aligned}\mathbb{P}\left[\mathrm{Round}(x) = z\right] &= \int_{\lfloor z \rfloor}^{\lceil z \rceil} f(x)\ dx \\ &\approx f\left(\frac{z}{1-tu}\right)\left(\lceil z \rceil - \lfloor z \rfloor\right)\end{aligned}$$

for all values of t such that Eq. (8) holds.

Assumption 1. Writing $z(s,k)$ for $z(e_{min}, s, k)$ assume that:

$$\sum_{\substack{0 \le k < 2^p \\ s \in \{0,1\}}} \mathbb{1}_{\lfloor z, z \rceil}\left(\frac{z(s,k)}{1-tu}\right) f\left(\frac{z(s,k)}{1-tu}\right)\left(\lceil z(s,k) \rceil - \lfloor z(s,k) \rfloor\right) \approx 0$$

and similarly for $z(s,k) = z(e_{max}, s, k)$. Under Assumption 0, this condition says that the probability under f of sampling a number whose rounding has exponent e_{min} or e_{max} is close to zero. This condition is certainly met for the distributions of Fig. 1 and half-precision.

Assumption 2. Given $t \in [-1, 1]$, for every k satisfying Eq. (9) we assume

$$\sum_{\substack{e_{min} < e < e_{max} \\ s \in \{0,1\}}} f\left(\frac{z(e,s,k)}{1-tu}\right)\left(\lceil z(e,s,k) \rceil - \lfloor z(e,s,k) \rfloor\right) \approx \frac{1}{2^p}$$

Under assumption 0, this condition means that when rounding a sample drawn from the distribution f, all mantissas are equally likely.

Under assumptions 0-2 one can show that for $t \in \left[-\frac{1}{2}, \frac{1}{2}\right]$

$$d(t) \approx \frac{1}{2^p(1-tu)^2}\left(\frac{2}{3} + \frac{3(2^p-1)}{4}\right) \tag{10}$$

Similarly, under assumptions 0-2 one has for $|t| > \frac{1}{2}$:

$$\begin{aligned}d(t) \approx \frac{1}{2^p(1-tu)^2}\Bigg(&\frac{2}{3} + \frac{1}{2}\lfloor 2^p\left(\frac{1}{t} - 1\right) - \frac{1}{2}\rfloor + \\ &\frac{1}{2^{p+2}}\left(\lfloor 2^p\left(\frac{1}{t} - 1\right) + \frac{1}{2}\rfloor\right)^2\Bigg)\end{aligned} \tag{11}$$

where $\lfloor 2^p\left(\frac{1}{t} - 1\right) + \frac{1}{2}\rfloor$ here denotes the usual floor function. Combining Eq. (10) and Eq. (11) we get under assumptions 0-2 that as $p \to \infty$ the error density $d(t)$ is well approximated by the *typical density*:

$$d_{typ}(t) = \begin{cases} \frac{3}{4} & |t| \le \frac{1}{2} \\ \frac{1}{2}\left(\frac{1}{t} - 1\right) + \frac{1}{4}\left(\frac{1}{t} - 1\right)^2 & |t| > \frac{1}{2} \end{cases} \tag{12}$$

which is represented in Fig. 2 and is clearly a good approximation of the exact densities of Fig. 1.

Remark: A remark to the effect that assumptions 0-2 are increasingly likely to be met as $p \to \infty$.

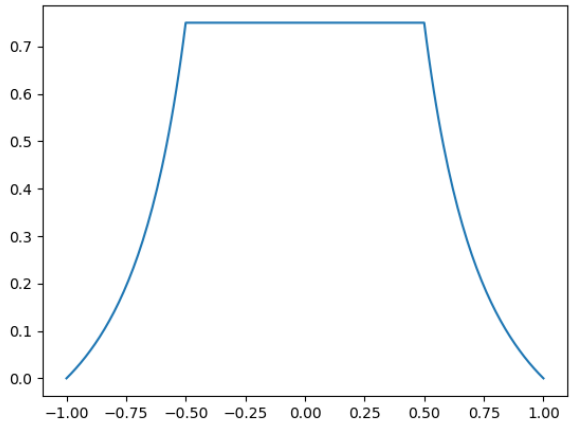

Fig. 2. Typical distribution of rounding errors (in unit roundoffs)

III. Probabilistic interpretation of simple expressions

In this section we present a class of numerical programs for which it is possible to *formally* compute an output distribution given an input distribution, whilst implementing the probabilistic model of IEEE arithmetic operations given by Eq. (2). In § IV we will present some initial steps towards *numerically* implementing the model presented in this section.

A. A simple syntax

Our class of numerical program given by the following simple grammar.

Terms:

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

Tests:

$\mathtt{b} ::= \mathtt{t} < \mathtt{r} \mid \mathtt{t} > \mathtt{r} \mid \mathtt{t} == \mathtt{r} \qquad \mathtt{r} \in \mathbb{F}$

Programs:

$\mathtt{p} ::= \mathtt{skip} \mid \mathtt{x_i} := \mathtt{t} \mid \mathtt{p}\ ;\ \mathtt{p} \mid \mathtt{if\ b\ then\ p\ else\ p}$

For every expression $\mathtt{p}$, we consider the list $(\mathtt{x_1}, \ldots, \mathtt{x_n})$ of variables appearing in $\mathtt{p}$. We view all variables as public variables and we associate with the list $(\mathtt{x_1}, \ldots, \mathtt{x_n})$ a multivariate random variable (random vector) $\mathbf{X}$ modelling its (probabilistic) state. We will call this data the *probabilistic context* and denote it $\{(\mathtt{x_1}, \ldots, \mathtt{x_n}) \sim \mathbf{X}\}$. We will denote by X_i the i^{th} marginal of $\mathbf{X}$. The cumulative distribution function of $\mathbf{X}$ will be given by

$$\mathbb{P}\left[X_1 \le x_1, \ldots, X_n \le x_n\right].$$

We assume a fixed exponent range e_{min}, e_{max} and precision level p throughout.

B. Random variable arithmetic

Whilst the probabilistic interpretation of programs will be defined in terms of an operator updating the probabilistic

context, i.e. an operator sending random vectors to random vectors, the probabilistic interpretation of *terms* will be defined in terms of arithmetic operation on (univariate) random variables. We briefly review arithmetic operations on random variables which posses a density function (w.r.t. the Lebesgue measure) translate into operations on these densities [16]. In particular the density of the sum of two independent random variables is given by the convolution of the densities. In more detail one has the following correspondence:

$$X + Y \sim f_X \oplus f_Y(t) = \int_{-\infty}^{\infty} f_X(x) f_Y(t-x)\ dx \tag{13}$$

$$X - Y \sim f_X \ominus f_Y(t) = \int_{-\infty}^{\infty} f_X(x) f_Y(x-t)\ dx \tag{14}$$

$$X \times Y \sim f_X \otimes f_Y(t) = \int_{-\infty}^{\infty} \frac{1}{|x|} f_X(x) f_Y\left(\frac{t}{x}\right) dx \tag{15}$$

$$X \div Y \sim f_X \oslash f_Y(t) = \int_{-\infty}^{\infty} |x| f_X(x) f_Y(tx) dx \tag{16}$$

Similarly, addition and multiplication by a scalar correspond to

$$\alpha + X \sim (\alpha \oplus f_X)(t) = f(t+\alpha)$$
$$\alpha X \sim (\alpha \otimes f_X)(t) = \alpha f(\alpha t)$$

C. Probabilistic interpretation

For any univariate random variable X with density f_X, we define the *error of* X – notation $\mathcal{E}(X)$ – as the random variable $\frac{X-\widehat{X}}{X}$ whose density is given by Eq. (7). For any n-dimensional multivariate random variable $\mathbf{X}$ we write $X_i, 1 \leq i \leq n$ for its i^{th} *marginal* distribution. The probabilistic interpretation developed below dates back to [10].

1) Input quantization: Given a probabilistic context, the first question is to decide whether we need to model an initial quantization. This would correspond for example to modelling the quantization of an analog input (e.g. a sensor) or of an input generated at a higher precision level (e.g. a routine at half-precision level receiving input in double-precision). If we choose to model this step, then we need to add a probabilistic error term to each input, this is achieved by the probabilistic quantization sending $\mathbf{X}$ to the multivariate distribution with cumulative distribution function

$$\mathbb{P}\left[X_1(1+\mathcal{E}(X_1)) \leq x_1 \ldots \leq X_n(1+\mathcal{E}(X_n))\right]$$

where the explicit computation of the random variables $X_1(1+\mathcal{E}(X_i))$ can be performed using the densities derived in § II and the operations on densities defined in § III-B.

2) Terms: will be modelled as (univariate) random variables using the following inductive definition:

$$\begin{aligned} model(\mathtt{r}) &= r, \text{ the constant r.v.} \\ model(\mathtt{x_i}) &= X_i \\ model(\mathtt{t_1\ op_m\ t_2}) &= (model(\mathtt{t_1})\ \text{op}\ model(\mathtt{t_2}))\cdot \\ &\quad (1+\mathcal{E}(model(\mathtt{t_1})\ \text{op}\ model(\mathtt{t_2}))) \end{aligned} \tag{17}$$

Note how we model the arithmetic operations in accordance with the fundamental model of Eq. (2): we first compute the random variable $model(\mathtt{t_1})$ op $model(\mathtt{t_2})$ (typically using the operations on densities defined in § III-B) which corresponds to the infinite-precision operation, and we then add a probabilistic error term whose distribution is computed from the distribution of $model(\mathtt{t_1})$ op $model(\mathtt{t_2})$ itself using the densities derived in § II.

3) Tests: correspond to the obvious subset of $\mathbb{R}^n$ generated by the comparisons. Thus if $\mathtt{t}(x_1, \ldots, x_n)$ is a term in n variables

$$model(\mathtt{t} < \mathtt{r}) = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid \mathtt{t}(x_1, \ldots, x_n) < r\}$$

and similarly for $\mathtt{t} > \mathtt{r}$ and $\mathtt{t} == \mathtt{r}$.

4) Expressions: will be modelled as operations sending multivariate random variables to multivariate random variables. In effect, updating the probabilistic context.

(i) `skip`:

$$model(\mathtt{skip})(\mathbf{X}) = \mathbf{X}$$

(ii) Assignments: $model(\mathtt{x_i} := \mathtt{t})(\mathbf{X})$ is the multivariate random variable whose cumulative distribution function is given by

$$\begin{aligned} &\mathbb{P}[X_1 \leq x_1, \ldots, X_{i-1} \leq x_{i-1}, model(\mathtt{t}) \leq x_i, \\ &\quad X_{i+1} \leq x_{i+1}, \ldots, X_n \leq x_n] \end{aligned}$$

(iii) Sequential composition:

$$model(\mathtt{p_1}\ ;\ \mathtt{p_2})(\mathbf{X}) = model(\mathtt{p_2})(model(\mathtt{p_1})(\mathbf{X}))$$

For `if then else` statements we need to introduce the following notation: if $B \subseteq \mathbb{R}^n$ is a measurable subset of $\mathbb{R}^n$ and $\mathbf{X}$ is a multivariate random variable in R^n, then $\mathbf{X}_B$ will denote the multivariate random variable with cumulative distribution function

$$\mathbb{P}\left[X_1 \leq x_1, \ldots, X_n \leq x_n \wedge (X_1, \ldots, X_n) \in B\right]$$

With this notation we can now define

(iv) Conditionals:

$$\begin{aligned} &model(\mathtt{if\ b\ then\ p_1\ else\ p_2})(\mathbf{X}) = \\ &\quad model(\mathtt{p_1})\big(\mathbf{X}_{model(\mathtt{b})}\big) + model(\mathtt{p_2})\big(\mathbf{X}_{model(\mathtt{b})^c}\big) \end{aligned}$$

where $model(\mathtt{b})^c$ is the complement of $model(\mathtt{b})$

The rules given above allows us (in principle) to compute the output distribution of any expression given by the grammar of § III-A. By construction this output will include the cumulative and combined effect of every probabilistic rounding occurring through arithmetic operations via the model (2).

IV. Experimental results

We have implemented the probabilistic interpretation of the class of terms (defined in § III-A) whose syntax has the structure of a *tree*, that is to say terms where variables are not repeated (in which case the syntax would be a DAG). We can thus compute the probabilistic interpretation of $(\mathtt{x}+\mathtt{y})/(\mathtt{z}*\mathtt{t})$ but not of $(\mathtt{x}+\mathtt{y})/(\mathtt{x}*\mathtt{y})$. The reason for this current

limitation is that two occurrences of the same variable must be interpreted as two perfectly correlated random variables, i.e. a probability distribution whose support lies on the diagonal of $\mathbb{R}^2$. Such a distribution cannot have a two-dimensional density function, and this renders their representation highly-non trivial. For the time being we leave this problem to future work.

As shown in § III-C, the interpretation of terms is computed by performing arithmetic operations on random variables. For random variables whose probability distribution is representable by density functions, § III-B shows how these arithmetic operations can be implemented in practice. A sophisticated version of Eq. (13)-Eq. (16) is implemented in the Python library PaCAL [8][9] on which we have based the evaluation of the probabilistic interpretation of terms.

A. Rounding error distribution

Our implementation focuses on computations performed in low precision, that is to say half-precision or lower. We therefore compute the distribution of relative errors using the exact analytic formula Eq. (7).

PaCAL concretely implements random variables as a density function represented by a piecewise Chebyshev interpolation polynomial. We therefore use the same representation and concretely represent the density Eq. (7) as a Chebyshev interpolating polynomial. The polynomial interpolation is performed using the library `pychebfun`, a Python implementation of the library `chebfun` [1] developed to perform accurate and fast computations on functions represented as Chebyshev polynomial interpolations.

Fig. 3 shows a plot of the half-precision error distribution for a random variable distributed uniformly on $[0, 1]$. The red line is the density represented as a Chebyshev interpolating polynomial, the blue area is a histogram of the relative rounding error on one million samples. Note again the similarity with the typical density function represented in Fig. 2.

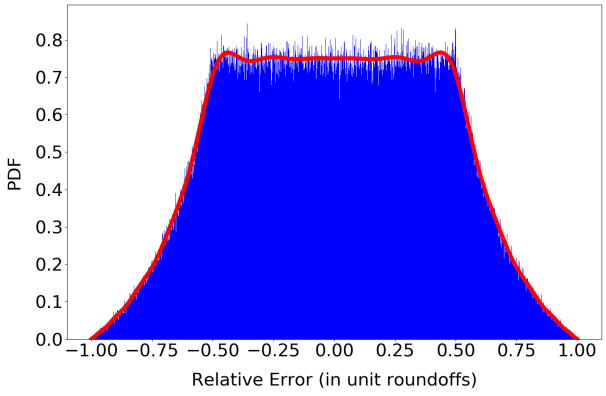

Fig. 3. Implemented density function vs Monte-Carlo simulation.

B. Range analysis

With a concrete representation of the error distribution associated with an input distribution we can recursively evaluate the probabilistic interpretation of terms following (17). This computes the final output distribution (including probabilistic rounding errors) of a term, given input distributions for each of its variables. A simple application of this output distribution is *range analysis*.

We first consider a simple application of range analysis, namely the detection of overflows. To illustrate our probabilistic approach consider two variables $\texttt{x0}, \texttt{x1}$ with `x0` distributed uniformly on $[10, 15.5]$ and `x1` distributed uniformly on $[0.97, 2]$. Suppose that the working precision is 3 bits for the exponent and 3 bits for the mantissa and that we are interested in the term $\texttt{x0}/\texttt{x1}$. In infinite precision $15.5/0.97 < 16$, the largest representable number at the given precision level, and there is no overflow. In reduced precision however overflow can occur. In this case, an analyser like FPTaylor will correctly detect the overflow and return an infinite range, but will not be able to quantify the *likelihood of overflow*. Fig. 4 shows the output distribution of the term $\texttt{x0}/\texttt{x1}$ (red line), the support of the output distribution ('PM' in the legend), the output range of FPTaylor ('FPT' in the legend), and a histogram of the computation $\texttt{x0}/\texttt{x1}$ performed on one million samples in reduced precision. Analytically, the probability of overflow is 0.0775%, and empirically 0.0642% of the samples overflow.

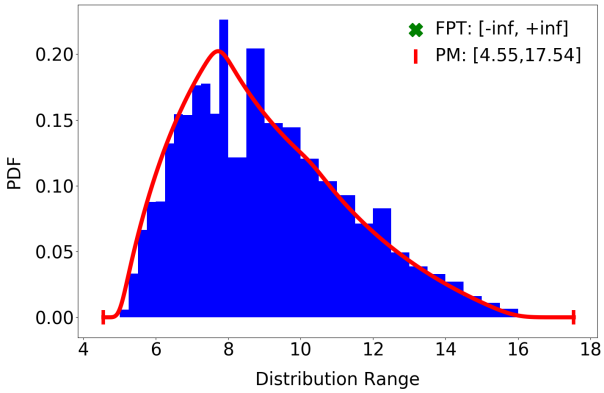

Fig. 4. Analytical and simulated output distribution for $\texttt{x0}/\texttt{x1}$.

Next, we consider two benchmarks from FPBench [3] which our current setup can handle. In both cases we take *half-precision* as the working precision. The first benchmark is `test02_sum8less` which is given by the term:

$$((((((\texttt{x0} + \texttt{x1}) + \texttt{x2}) + \texttt{x3}) + \texttt{x4}) + \texttt{x5}) + \texttt{x6}) + \texttt{x7}$$

with each `xi` assumed to be uniformly distributed on the interval $[1, 2]$. The result of the range analysis are shown in Fig. 5. The support of the output distribution provides marginally worse error bounds than FPTaylor and is in good agreement with one million Monte-Carlo simulations, that is to say one million evaluations of the term `test02_sum8less` in reduced precision where is input value is sampled uniformly from $[1, 2]$. Crucially, we can probabilistically tighten these bound: at 99.99% confidence we can say that the output range lies in the interval $[9.0, 15.0]$.

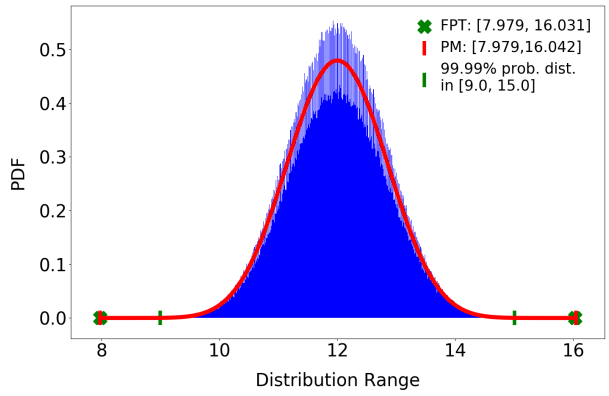

Fig. 5. Analytical and simulated output distribution for `test02_sum8less`.

The second benchmark provides an even stronger case for considering probabilistic range analysis. Consider the multiplicative cousin of `test02_sum8less` given by the term `test02_mul8less`:

$$((((((\texttt{x0} * \texttt{x1}) * \texttt{x2}) * \texttt{x3}) * \texttt{x4}) * \texttt{x5}) * \texttt{x6}) * \texttt{x7}$$

with each `xi` assumed to be uniformly distributed on the interval $[-3, 3]$. The range analysis `test02_mul8less` is displayed in Fig. 6. Again the support of the output distribution is marginally wider that the one provided by FPTaylor, but with 99.99% confidence we can tighten the output range by about five orders of magnitude in base 2 to $[-206, 206]$.

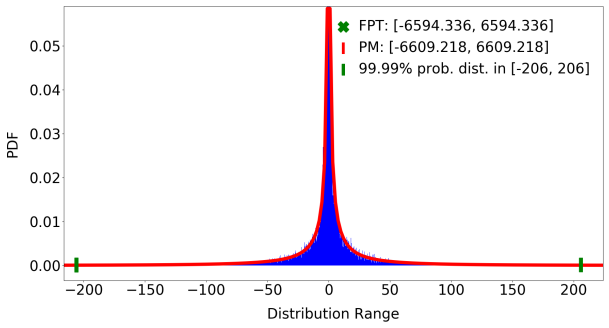

Fig. 6. Analytical and simulated output distribution for `test02_mul8less`

References

[1] Zachary Battles and Lloyd N Trefethen. An extension of matlab to continuous functions and operators. *SIAM Journal on Scientific Computing*, 25(5):1743–1770, 2004.

[2] George Constantinides, Peter YK Cheung, and Wayne Luk. *Synthesis and optimization of DSP algorithms*. Springer Verlag, 2004.

[3] Nasrine Damouche, Matthieu Martel, Pavel Panchekha, Jason Qiu, Alex Sanchez-Stern, and Zachary Tatlock. Toward a standard benchmark format and suite for floating-point analysis. 2016.

[4] 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 *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 270–287. Springer, 2018.

[5] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

[6] Nicholas J Higham and Theo Mary. A new approach to probabilistic rounding error analysis. *SIAM Journal on Scientific Computing*, 41(5):A2815–A2835, 2019.

[7] Ilse CF Ipsen and Hua Zhou. Probabilistic error analysis for inner products. *arXiv preprint arXiv:1906.10465*, 2019.

[8] Szymon Jaroszewicz and Marcin Korzeń. Arithmetic operations on independent random variables: A numerical approach. *SIAM Journal on Scientific Computing*, 34(3):A1241–A1265, 2012.

[9] Marcin Korzen and Szymon Jaroszewicz. PaCAL: A python package for arithmetic computations with random variables. *Journal of Statistical Software*, 57(10):5, 2014.

[10] Dexter Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, June 1981.

[11] Debasmita Lohar, Milos Prokop, and Eva Darulova. Sound probabilistic numerical error analysis. In *International Conference on Integrated Formal Methods*, pages 322–340. Springer, 2019.

[12] Victor Magron, George Constantinides, and Alastair Donaldson. Certified roundoff error bounds using semidefinite programming. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):34, 2017.

[13] Microprocessor Standards Committee of the IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic, June 2019.

[14] Douglass Stott Parker. *Monte Carlo Arithmetic: exploiting randomness in floating-point arithmetic*. University of California (Los Angeles). Computer Science Department, 1997.

[15] 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. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 41(1):20, 2018.

[16] M.D. Springer. *The algebra of random variables*. Probability and Statistics Series. Wiley, 1979.

[17] John Von Neumann and Herman H Goldstine. Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53(11):1021–1099, 1947.