Error analysis methods for the fixed-point implementation of linear systems

Thibault Hilaire, Anastasia Volkova

Error analysis methods for the fixed-point implementation of linear systems

by Thibault Hilaire, Anastasia Volkova Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
Email: first_name.last_name@lip6.fr

***Abstract*—In this paper we propose to perform a complete error analysis of a fixed-point implementation of any linear system described by data-flow graph. The system is translated to a matrix-based internal representation that is used to determine the analytical errors-to-output relationship. The error induced by the finite precision arithmetic (for each sum-of-product) of the implementation propagates through the system and perturbs the output.**
The output error is then analysed with three different point of view: classical statistical approach (errors modeled as noises), worst-case approach (errors modeled as intervals) and probability density function. These three approaches allow determining the output error due to the finite precision with respect to its probability to occur and give the designer a complete output error analysis. Finally, our methodology is illustrated with numerical examples.

I. Introduction

Nowadays embedded devices are used every day everywhere by everyone. The progress in microelectronic allows to integrate more and more applications and services in embedded platforms, and Signal Processing is a low level brick required. Signal processing application use blocks made-up of linear digital systems.
The context of our work is the implementation of these systems (Linear Time Invariant filters, described by their transfer function or by a data-flow graph) for signal processing, control, robotic, etc.. The implementation of filters on low-resource devices such as Digital Signal Processors (DSPs) or Field-Programmable Gate Arrays (FPGAs) typically relies on finite precision arithmetic, such as fixed-point arithmetic and thus results in an inherent limitation on the accuracy of the output.

Unfortunately, there is no general tool to automatize and analyze the transformations required to obtain the final code from the filter or controller (as mathematical object). This final code may be executed on a device (software implementation) or it can be translated to some hardware description language. This problem is a difficult one, that is usually composed of several steps.
First, one should be able to deal with the various equivalent algorithms to implement the filter. In addition to the classical direct forms or the state-space, some other interesting realizations can be considered, like the Lattice Wave Digital Filters [1], [2], the ρ-Direct Form II transposed (ρDFIIt) [3], warped filters, and a lot of other specific realizations (LGC, LCW-structures, etc.). The choice of the structure has a large impact on the trade-off cost-accuracy.
Then, a fixed-point algorithm (to be executed on a processor, or to be transformed into a hardware operator) is deduced, and an accurate and meaningful error analysis must be performed.
Finally, this code is mapped to a given processor (or some hardware operators are combined) to produce final implementation.

In this article, we investigate different tools and approaches to perform the error analysis of any linear systems, since the two classical approaches (statistical and worst-case approaches) are not sufficient to embrace the complexity of the output error. The two main contributions of the paper are the comparison of these two approaches and the determination of the probability density function.
We do not consider here the impact of the quantization of the coefficients, but the reader my refer to [4] and [5] for more details.

The paper is organized as follows. Section II recalls the prerequisites concerning the linear systems and data-flow graphs, our internal matrix-based representation and Fixed-Point arithmetic. Section III explores the fixed-point errors due to the quantization operator and their impact to the output by exhibiting the analytical relationship between the errors and the outputs. Section IV focuses on the error propagation using three complementary approaches: statistical approach considering the mean and variance of the errors modeled as noises, worst-case approach where errors are considered as intervals and finally probability density function, that we determine using convolution algorithms. Finally, we illustrate these approaches with two illustrated in section V before conclusion.

Notation: throughout the article matrices are in uppercase boldface, vectors are in lowercase boldface, scalars are in lowercase. The $n \times n$ identity matrix is denoted $\boldsymbol{I}_n$, $\boldsymbol{M}^\top$ is the transposed of $\boldsymbol{M}$ and $\mathbb{1}_n$ is a $n \times 1$ vector full of ones.

II. Prerequisites

A. Linear systems and Data-flow graphs

In this article, we only consider Linear Time Invariant (LTI) systems, like the IIR (Infinite Impulse Response) and FIR (Finite Impulse Response) filters, linear controllers and other linear transforms (FFT, DCT, etc.). They can be described with an input-to-output data-flow graph that only uses the sum operator (additions or subtractions), the constant multiplication and the delay operator (Fig. 1). Or they can be described by appropriate equations (in time or in frequency domain).

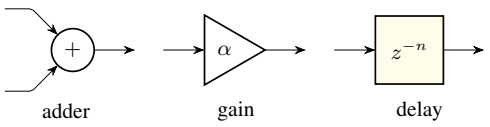

Fig. 1. Linear data-flow graph blocks.

Fig. 2 shows an example of a Lattice Wave Digital Filter [1], [2] expressed as an input-output data-flow graph, using only 4 multiplication-by-constants and some sums and delays.

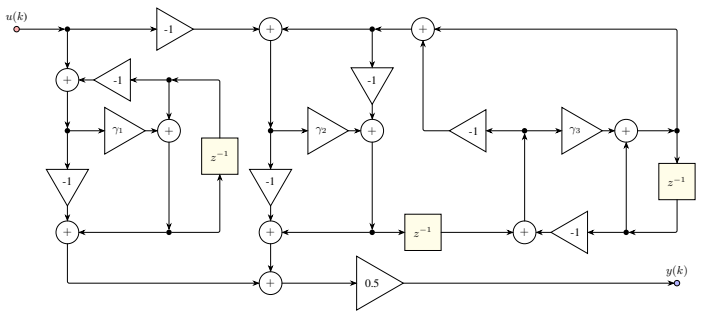

Fig. 2. Lattice Wave Digital Filter data-flow.

This work has been partially sponsored by french ANR agency under grant No ANR-13-INSE-0007-04 MetaLibm.

978-1-5386-0446-5/17/$31.00

B. Specialized Implicit Framework

Though data-flow graph is an appropriate description of the system, we prefer using a matrix-based input-output relationship. In [6], the Specialized Implicit Framework (SIF) has been developed to describe all the possible LTI algorithms in one matrix-based equation.
It has been proposed in order to describe all the possible realizations of a given transfer function (like the direct forms, state-spaces, cascade or parallel decompositions, lattice filters, etc.). It also allows the study and comparison of their finite precision effects. SIF is an extension of the state-space realization, modified in order to express chained Sum-of-Products (SoP) operations.

In fact, all linear data-flows can be represented with the SIF, and the conversion algorithm has been presented in [7]. This macroscopic description is more suited for the analysis than a data-flow graph as it gives direct analytical formula for the finite precision error analysis, as shown in section III. Both the Single Input Single Output (SISO) and the Multiple-Inputs Multiple Outputs (MIMO) systems can be described with SIF.

Denote k the time (discrete time according to a given sampling period), $\boldsymbol{u}(k)$ and $\boldsymbol{y}(k)$ the vector of q inputs and the vector of p outputs respectively. The n variables that are stored from one step to the other are in the state vector $\boldsymbol{x}(k)$, while the l intermediate results are collected in the vector $\boldsymbol{t}(k)$. Then, a system $\mathcal{H}$ can be described with the SIF thanks to the following set of equations:

$$\mathcal{H}\left\{\begin{array}{rcl}\boldsymbol{J}\boldsymbol{t}(k+1) &=& \boldsymbol{M}\boldsymbol{x}(k)+\boldsymbol{N}\boldsymbol{u}(k)\\ \boldsymbol{x}(k+1) &=& \boldsymbol{K}\boldsymbol{t}(k+1)+\boldsymbol{P}\boldsymbol{x}(k)+\boldsymbol{Q}\boldsymbol{u}(k)\\ \boldsymbol{y}(k) &=& \boldsymbol{L}\boldsymbol{t}(k+1)+\boldsymbol{R}\boldsymbol{x}(k)+\boldsymbol{S}\boldsymbol{u}(k)\end{array}\right. \tag{1}$$

The vector $\boldsymbol{t}(k+1)$ is not used for computations at step k, which characterizes the concept of intermediate variables. The vector $\boldsymbol{t}(k+1)$ is similar to the state vector, except those values are not stored from one step to another, which characterizes the concept of intermediate variables. That vector is denoted $\boldsymbol{t}(k+1)$ (instead of $\boldsymbol{t}(k)$ although it serves at step k) in order to be consistent with implicit state-space theory [8].

The matrix $\boldsymbol{J}$ is lower-triangular with 1 on its diagonal, so the first value of $\boldsymbol{t}(k+1)$ is first computed, then the second one is computed using the first and so on (thus, the computation of $\boldsymbol{J}^{-1}$ is not necessary)). The *implicit* term $\boldsymbol{J}\boldsymbol{t}(k+1)$ naturally serves for describing the specific order of the computation.

The matrices $\boldsymbol{J}$, $\boldsymbol{K}$ and $\boldsymbol{L}$ allow us to describe the sequence of computations. For example, $\boldsymbol{y}=\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{x}$ can be computed as $\boldsymbol{y}=(\boldsymbol{M}_2\boldsymbol{M}_1)\boldsymbol{x}$ or as $\boldsymbol{y}=\boldsymbol{M}_2(\boldsymbol{M}_1\boldsymbol{x})$. The latter expression will be described as

$$\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ -\boldsymbol{M}_2 & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{t}\\ \boldsymbol{y}\end{pmatrix}=\begin{pmatrix}\boldsymbol{M}_1\\ \boldsymbol{0}\end{pmatrix}\boldsymbol{x}, \tag{2}$$

with $\boldsymbol{t}$ holding the intermediate value $\boldsymbol{M}_1\boldsymbol{x}$.

Therefore, transforming a linear data-flow to the SIF formalism corresponds to a description of the computations within the flow in terms of intermediate variables and states while preserving the order of operations [7].

Throughout the paper we consider that the computations associated to (1) are ordered from top to bottom, associated in a one to one manner to the Algorithm 1, where the null multiplications are removed, and trivial parameters ($\pm 2^p$, $p\in\mathbb{Z}$) do not involve any multiplication.

It can be easily established that (1) is equivalent in infinite precision to the state-space filter $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$:

$$\left\{\begin{array}{rcl}\boldsymbol{x}(k+1) &=& \boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k)\\ \boldsymbol{y}(k) &=& \boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{D}\boldsymbol{u}(k)\end{array}\right. \tag{3}$$

Algorithm 1: General algorithm associated to the SIF.

Input: Signal $\boldsymbol{u}(k)$
Output: Signal $\boldsymbol{y}(k)$
begin
  foreach k **do**
    for i *in* $1\ldots l$ **do** `/*Intermediate variables*/`
      $\boldsymbol{t}_i \leftarrow \sum_{j=1}^{n}\boldsymbol{M}_{ij}\boldsymbol{x}_j(k)+\sum_{j=1}^{q}\boldsymbol{N}_{ij}\boldsymbol{u}_j(k)-\sum_{j=1}^{i-1}\boldsymbol{J}_{ij}\boldsymbol{t}_j$
    end
    for i *in* $1\ldots n$ **do** `/*State-vector update*/`
      $\boldsymbol{x}_i(k+1) \leftarrow \sum_{j=1}^{l}\boldsymbol{K}_{ij}\boldsymbol{t}_j+\sum_{j=1}^{n}\boldsymbol{P}_{ij}\boldsymbol{x}_j(k)+\sum_{j=1}^{q}\boldsymbol{Q}_{ij}\boldsymbol{u}_j(k)$
    end
    for i *in* $1\ldots p$ **do** `/*Computation of the outputs*/`
      $\boldsymbol{y}_i(k) \leftarrow \sum_{j=1}^{l}\boldsymbol{L}_{ij}\boldsymbol{t}_j+\sum_{j=1}^{n}\boldsymbol{R}_{ij}\boldsymbol{x}_j(k)+\sum_{j=1}^{q}\boldsymbol{S}_{ij}\boldsymbol{u}_j(k)$
    end
  end
end

with:

$$\begin{array}{ll}\boldsymbol{A}=\boldsymbol{K}\boldsymbol{J}^{-1}\boldsymbol{M}+\boldsymbol{P}, & \boldsymbol{B}=\boldsymbol{K}\boldsymbol{J}^{-1}\boldsymbol{N}+\boldsymbol{Q},\\ \boldsymbol{C}=\boldsymbol{L}\boldsymbol{J}^{-1}\boldsymbol{M}+\boldsymbol{R}, & \boldsymbol{D}=\boldsymbol{L}\boldsymbol{J}^{-1}\boldsymbol{N}+\boldsymbol{S}.\end{array} \tag{4}$$

So this system ((1) or (3)) has the following transfer function:

$$\boldsymbol{H}(z)=\boldsymbol{C}(z\boldsymbol{I}_n-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D},\quad \forall z\in\mathbb{C}. \tag{5}$$

However, (3) corresponds to a different set of coefficients than the one in (1). Therefore, while in infinite precision (1) and (3) are equivalent, in finite precision they have different numerical properties. Thus, the SIF not only describes the input/output relationship but also fully captures properties of computational algorithm.

The main interest of such a framework is that it allows expressing various algorithms for the computation of the same filter in a unifying form, in order to study and compare the effects of the finite precision arithmetic on them, and then to choose the best Fixed-Point algorithm to implement this filter. All the classical measures used to evaluate the impact of the quantization of the coefficients (sensitivity-based measure like in [4], [5]) and the impact of the roundoff errors ([4], [9], [10], [11]) have been extended to the SIF [6], [12].

C. Fixed-Point Arithmetic

In this paper, only the signed fixed-point arithmetic with 2's complement representation is used. Let z be a w-bit fixed-point number:

$$z=-2^m z_m+\sum_{i=\ell}^{m-1}2^i z_i \tag{6}$$

where m and ℓ are the positions of the most significant bit (MSB) and the least significant bit (LSB) of z, respectively. Let the couple (m,ℓ) denote the Fixed-Point format of z (see Fig. 3). The word-length w can be obtained with:

$$w=m-\ell+1, \tag{7}$$

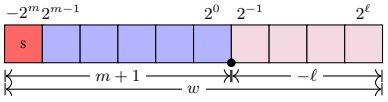

Fig. 3. Representation of a fixed-point number. Here $m=5$ and $\ell=-4$.

The w bits manipulated when using that number can be seen as a 2's complement integer (equal to $z2^\ell$), since the scaling factor 2^ℓ is implicit, and only known by people interpreting those w bits.

Rounding mode	round-to-nearest	Range
Truncation	✗	$[0, 2^\ell[$
Round half up	✓	$[-2^{\ell-1}, 2^{\ell-1}[$
Round to nearest even	✓	$[-2^{\ell-1}, 2^{\ell-1}]$
Von Neumann [16]	✗	$]-2^\ell, 2^\ell[$
Round toward $+\infty$	✗	$]-2^\ell, 0]$
Magnitude truncation	✗	$\begin{cases}[0, 2^\ell[& \forall x \geqslant 0\\ ]-2^\ell, 0] & \forall x < 0\end{cases}$
Round to nearest, ties to 0	✓	$[-2^{\ell-1}, 2^{\ell-1}]$
Round half down	✓	$]-2^{\ell-1}, 2^{\ell-1}]$
Round to nearest, ties away	✓	$[-2^{\ell-1}, 2^{\ell-1}]$

TABLE I
INTERVAL OF ERRORS FOR DIFFERENT ROUNDING MODES (ℓ IS THE REMAINING LSB POSITION).

III. Error analysis

A. Quantization errors

After some arithmetic operations (addition or multiplication), the Fixed-Point Format must be changed and the LSB position reduced. This is called a quantization (or rounding), and several ways to do it exist. The simpliest is the *truncation* and corresponds to simply dropping the extra least significant bits. Some other methods exist, involving more complex operation (mainly a 1-bit addition, see [13]).

Fig. 4. Quantization is equivalent to adding an *roundoff error* term.

A quantization can be modeled as the addition of an extra term, called *roundoff error*, as shown on Fig. 4. Depending on the quantization mode, the range of this error varies as shown in Table I.
These modes are described in the IEE-1666 standard [14] or are equivalent to those described in the floating-point IEEE-754 standard [15]. Round half up, round half down, round to nearest ties away, round to nearest ties to 0 and round to nearest even are five round-to-nearest modes, where the only difference resides in the way the tie-break cases are treated (when the number to round is exactly in the middle of two rounded numbers).
Von Neumann's quantization mode is performed by forcing the value of the LSB to 1 if at least one of the deleted bit is 1 [16]. This rounding mode provides a larger error, but is cheap to implement and leads to a symmetric error, contrary to the truncation mode.

B. Error analysis with feedback

At each step k, the evaluation of the intermediate variables, states and outputs (see Algorithm 1) is composed of sum-of-products (SoP), one per intermediate variable, state and output. Since they are not performed in exact arithmetic, each SoP may include an error, compared to the exact SoP. So (1) is changed to

$$\begin{aligned}\boldsymbol{J}\boldsymbol{t}^*(k{+}1) &\leftarrow \boldsymbol{M}\boldsymbol{x}^*(k) + \boldsymbol{N}\boldsymbol{u}(k) &&+\boldsymbol{\varepsilon_t}(k)\\ \boldsymbol{x}^*(k{+}1) &\leftarrow \boldsymbol{K}\boldsymbol{t}^*(k{+}1) + \boldsymbol{P}\boldsymbol{x}^*(k) + \boldsymbol{Q}\boldsymbol{u}(k) &&+\boldsymbol{\varepsilon_x}(k)\\ \boldsymbol{y}^*(k) &\leftarrow \boldsymbol{L}\boldsymbol{t}^*(k{+}1) + \boldsymbol{R}\boldsymbol{x}^*(k) + \boldsymbol{S}\boldsymbol{u}(k) &&+\boldsymbol{\varepsilon_y}(k)\end{aligned} \tag{8}$$

where $\boldsymbol{\varepsilon_t}(k)$, $\boldsymbol{\varepsilon_x}(k)$ and $\boldsymbol{\varepsilon_y}(k)$ are the vectors of roundoff errors due to the sum-of-products evaluation. Denote $\boldsymbol{\varepsilon}(k)$ the column vector that aggregates those error vectors:

$$\boldsymbol{\varepsilon}(k) \triangleq \begin{pmatrix}\boldsymbol{\varepsilon_t}(k)\\ \boldsymbol{\varepsilon_x}(k)\\ \boldsymbol{\varepsilon_y}(k)\end{pmatrix} \in \mathbb{R}^{l+n+p}. \tag{9}$$

In order to capture the effects of the FxP implementation we must take into account the propagation of the roundoff errors through the data-flow. The output error $\boldsymbol{\Delta y}(k)$ is defined as the difference between the implemented and the exact systems:

$$\boldsymbol{\Delta y}(k) \triangleq \boldsymbol{y}^*(k) - \boldsymbol{y}(k), \quad \forall k \tag{10}$$

This difference system can be obtained by subtracting (1) to (8). It follows that $\boldsymbol{\Delta y}(k)$ can be seen as the output of the error vector $\boldsymbol{\varepsilon}(k)$ through the filter $\mathcal{H}_{\boldsymbol{\varepsilon}}$ with $(\boldsymbol{A}, \boldsymbol{M}_1, \boldsymbol{C}, \boldsymbol{M}_2)$ as state-space:

$$\boldsymbol{M}_1 \triangleq \begin{pmatrix}\boldsymbol{K}\boldsymbol{J}^{-1} & \boldsymbol{I}_n & \boldsymbol{0}\end{pmatrix}, \quad \boldsymbol{M}_2 \triangleq \begin{pmatrix}\boldsymbol{L}\boldsymbol{J}^{-1} & \boldsymbol{0} & \boldsymbol{I}_p\end{pmatrix}. \tag{11}$$

Equivalently (thanks to the linearity of the considered system), the implemented system can be seen as the sum of the exact system $\mathcal{H}$ and the error system $\mathcal{H}_{\boldsymbol{\varepsilon}}$ with $\boldsymbol{\varepsilon}(k)$ as input, as shown on Figure 5.

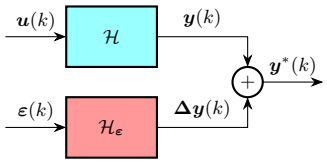

Fig. 5. Equivalent system, with errors separated.

The system $\mathcal{H}_{\boldsymbol{\varepsilon}}$ expresses how the error propagates through the filter, and knowing some properties on the roundoff errors $\boldsymbol{\varepsilon}(k)$ will lead to properties on the output error $\boldsymbol{\Delta y}(k)$, and hence on the accuracy of the implemented algorithm.

C. Finite precision sum-of-products

As already seen, the Fixed-Point arithmetic basic bricks involved here are the evaluation of some sum-of-products, i.e. sums of variables multiplied by constant coefficients. Let us consider here one sum-of-product (SoP):

$$s = \sum_{i=1}^{N} \boldsymbol{c}_i \cdot \boldsymbol{v}_i = \sum_{i=1}^{N} \boldsymbol{p}_i, \tag{12}$$

where $\boldsymbol{p}_i$ denotes the product $\boldsymbol{c}_i \cdot \boldsymbol{v}_i$ of the constants $\boldsymbol{c}_i$ and variables $\boldsymbol{v}_i$ for $1 \leqslant i \leqslant N$.

In our context, the MSB positions of the variables are known (deduced from the magnitude of the variables [17], [18]) and the MSB of the constants are deduced from their value (applying $\log_2$ upon them). In the SIF the sum-of-product results are stored either in the intermediate, state or output variables, so their MSBs are also known. Thus, it is easy to deduce the FxP format of the accumulation in order to guarantee a roundoff error less than a given bound.

Denote (m_s, ℓ_s) the required FxP format for the result. As shown in [12], to perform the multiplication and accumulation with the format $(m_s, \ell_s - g)$ with

$$g = \lceil \log_2 N \rceil, \tag{13}$$

and then discard those extra guard bits guaranties that the computation performs a faithful rounding on the exact result, i.e. the roundoff error is bounded by the weight of the LSB of the result, i.e. 2^{ℓ_s}. Fig. 6 details graphically this operation.

Of course, many other computing schemes are possible (with different rounding modes, using more or less complicated operations), leading to different bounds for the sum-of-product error.

IV. Propagation of errors through a filter

The error analysis is based on the decomposition performed in section III-B. The output error $\boldsymbol{\Delta y}$ depends on the Fixed-Point errors done during the Sum-of-Products evaluation.
Classically, the errors are modeled as noises, and the statistical properties of the final output error are deduced. Otherwise, errors can be

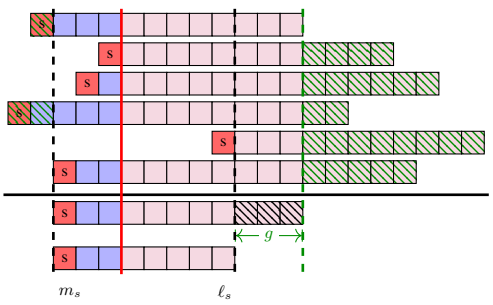

Fig. 6. The sum is first performed with FxP format $(m_s, \ell_s - g)$, and then the g guard bits are discarded.

modeled with intervals, and the Worst-Case Peak-Gain theorem [19] is used to deduce the output error interval.

We denote $|\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC}$ the DC-gain of the system $\mathcal{H}_{\boldsymbol{\varepsilon}}$, i.e. the gain of the system for frequency $\omega = 0$, and obtained by

$$|\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC} = \boldsymbol{C}(\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{M}_1 + \boldsymbol{M}_2. \tag{14}$$

We also denote $\mathbb{h}_{\boldsymbol{\varepsilon}}(k) \in \mathbb{R}^{p\times q}$ the impulse response matrix of $\mathcal{H}_{\boldsymbol{\varepsilon}}$, i.e. $\{(\mathbb{h}_{\boldsymbol{\varepsilon}})_{ij}(k)\}_{k\geqslant 0}$ is the response on the i^{th} output to an impulse on the j^{th} input. We have

$$\mathbb{h}_{\boldsymbol{\varepsilon}}(k) = \begin{cases} \boldsymbol{M}_2 & \text{if } k = 0 \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{M}_1 & \text{if } k > 0 \end{cases}. \tag{15}$$

A. Roundoff noise

In *roundoff noise* context, the roundoff errors are considered as noises that are additive, white and uncorrelated with the signal being quantized. It has been shown that this assumption is realistic and almost true if the number of removed bits is reasonable with respect to the total word-length [9], [20].

Let us denote (μ) and $(\boldsymbol{\Psi})$ the first and second order moments of the noise $\boldsymbol{\varepsilon}(k)$, respectively:

$$\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \triangleq E\{\boldsymbol{\varepsilon}(k)\} \in \mathbb{R}^{(l+n+p)\times 1} \tag{16}$$

$$\boldsymbol{\Psi}_{\boldsymbol{\varepsilon}} \triangleq E\left\{(\boldsymbol{\varepsilon}(k) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}})(\boldsymbol{\varepsilon}(k) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}})^\top\right\} \in \mathbb{R}^{(l+n+p)\times(l+n+p)}$$

$$\sigma^2_{\boldsymbol{\varepsilon}} \triangleq E\left\{(\boldsymbol{\varepsilon}(k) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}})^\top(\boldsymbol{\varepsilon}(k) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}})\right\} = tr(\boldsymbol{\Psi}_{\boldsymbol{\varepsilon}}) \in \mathbb{R}, \tag{17}$$

where $E\{.\}$ is the *expectation* operator (over time), and $tr(.)$ the trace operator.

Lemma 1 (Noise through a filter) *We consider the noise $\boldsymbol{\varepsilon}(k)$ to be stationary white noise such that $\boldsymbol{\varepsilon}(k)$ and $\boldsymbol{\varepsilon}(l)$ are not correlated for $k \neq l$ and independent ($\boldsymbol{\Psi}_{\boldsymbol{\varepsilon}}$ is diagonal).*
Then, the output $\boldsymbol{\Delta y}(k)$ through $\mathcal{H}_{\boldsymbol{\varepsilon}}$ has the following properties:

$$\boldsymbol{\mu}_{\boldsymbol{\Delta y}} = |\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC}\,\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \quad \sigma^2_{\boldsymbol{\Delta y}} = \|\mathcal{H}_{\boldsymbol{\varepsilon}}\boldsymbol{\Phi}_{\boldsymbol{\varepsilon}}\|_2^2 \tag{18}$$

where $\|.\|_2$ the ℓ_2-norm and $\boldsymbol{\Phi}_{\boldsymbol{\varepsilon}}$ the $(l+n+p)\times(l+n+p)$ diagonal matrix such that $(\boldsymbol{\Phi}_{\boldsymbol{\varepsilon}})_{ii} = \sqrt{(\boldsymbol{\Psi}_{\boldsymbol{\varepsilon}})_{ii}}$.

The ℓ_2-norm $\|\mathcal{H}_{\boldsymbol{\varepsilon}}\boldsymbol{\Phi}_{\boldsymbol{\varepsilon}}\|_2$ can be obtained from the observability Gramian $\boldsymbol{W}_o$ of $\mathcal{H}_{\boldsymbol{\varepsilon}}$:

$$\sigma^2_{\boldsymbol{t}} = tr\left(\boldsymbol{\Psi}_{\boldsymbol{\varepsilon}}(\boldsymbol{M}_2^\top M_2 + \boldsymbol{M}_1^\top \boldsymbol{W}_o \boldsymbol{M}_1)\right) \tag{19}$$

where $\boldsymbol{W}_o$ is the solution of the Lyapunov equation $\boldsymbol{W}_o = \boldsymbol{A}^\top \boldsymbol{W}_o \boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{C}$, or equivalently $\boldsymbol{W}_o = \sum_{k=O}^{\infty} \boldsymbol{A}^{\top k}\boldsymbol{C}^\top \boldsymbol{C}\boldsymbol{A}^k$.

Proof: The output error $\boldsymbol{\Delta y}(k)$ can be obtained from the error $\{\boldsymbol{\varepsilon}(l)\}_{0\leqslant l\leqslant k}$ and the impulse response $\mathbb{h}_{\boldsymbol{\varepsilon}}(k)$ of $\mathcal{H}_{\boldsymbol{\varepsilon}}$ with

$$\boldsymbol{\Delta y}(k) = \sum_{l=0}^{k} \mathbb{h}_{\boldsymbol{\varepsilon}}(l)\boldsymbol{\varepsilon}(k-l). \tag{20}$$

This last equation is used to compute $\boldsymbol{\mu}_{\boldsymbol{\Delta y}}$ and $\sigma^2_{\boldsymbol{\delta y}}$ from (16) and (17) using DC-gain and ℓ_2-norm classical definitions and properties. ■

Usually, the Signal-to-Noise Ratio (SNR) is used to determine the roundoff noise impact. It is defined as the ratio of the output signal power to output error power: $SQNR \triangleq \frac{\sigma_{\boldsymbol{y}}{}^2}{\sigma^2_{\boldsymbol{\Delta y}}}$.

B. Roundoff error

A more Computer Arithmetic approach would be to modeling errors with intervals. Now, we consider the interval to which $\boldsymbol{\varepsilon}$ and $\boldsymbol{\Delta y}$ belong. Usually, the lower and upper bound of the roundoff error are considered, as in Table I. But in the following, intervals are represented in mid-rad representation $\langle x_m, x_r\rangle$ where x_m is the center of the interval, and x_r its radius.

Lemma 2 (Worst-Case Peak Gain theorem) *Suppose that for all k, $\boldsymbol{\varepsilon}(k)$ is (element-by-element) in the interval $\langle \boldsymbol{\varepsilon}_m, \boldsymbol{\varepsilon}_r\rangle$, then, for any $\delta > 0$, it exists $K > 0$ such that $\forall k > K$, $\boldsymbol{\Delta y}(k)$ is (element-by-element) in the interval $\langle \boldsymbol{\Delta y}_m, \boldsymbol{\Delta y}_r\rangle$ with:*

$$\boldsymbol{\Delta y}_m = |\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC}\,\boldsymbol{\varepsilon}_m, \quad \boldsymbol{\Delta y}_r = \langle\!\langle \mathcal{H}_{\boldsymbol{\varepsilon}} \rangle\!\rangle\, \boldsymbol{\varepsilon}_r + \delta\mathbb{1} \tag{21}$$

where $\langle\!\langle \mathcal{H}_{\boldsymbol{\varepsilon}} \rangle\!\rangle$ is the Worst-Case Peak-Gain (WCPG) matrix[19] of the system $\mathcal{H}_{\boldsymbol{\varepsilon}}$, classically computed with the ℓ_1-norm of the impulse response $\mathbb{h}_{\boldsymbol{\varepsilon}}$:

$$\langle\!\langle \mathcal{H}_{\boldsymbol{\varepsilon}} \rangle\!\rangle \triangleq \sum_{k=0}^{\infty} |\mathbb{h}_{\boldsymbol{\varepsilon}}(k)| = |\boldsymbol{M}_2| + \sum_{k=0}^{\infty} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{M}_1\right|. \tag{22}$$

In [21] an efficient algorithm to evaluate it at arbitrary accuracy was proposed.
This lemma means that, except of the first terms, the output error $\boldsymbol{\Delta y}(k)$ is, in the steady state, in the interval $\langle |\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC}\,\boldsymbol{\varepsilon}_m; \langle\!\langle \mathcal{H}_{\boldsymbol{\varepsilon}} \rangle\!\rangle\, \boldsymbol{\varepsilon}_r\rangle$ (with an error δ).

Proof: Equation (20) gives $\boldsymbol{\Delta y}_m = \sum_{k=0}^{\infty} \mathbb{h}_{\boldsymbol{\varepsilon}}(k)\boldsymbol{\varepsilon}_m$ and (14) is used. For $\boldsymbol{\Delta y}_r$, the idea is to consider $\boldsymbol{\varepsilon}(k)$ as $\boldsymbol{\varepsilon}_r + \boldsymbol{\varepsilon}'(k)$ with $|\boldsymbol{\varepsilon}'| \leqslant \boldsymbol{\varepsilon}_m$, and use (20) to bound $|\boldsymbol{\Delta y}(k) - \boldsymbol{\Delta y}_m|$. ■

C. Comparison between roundoff noise and roundoff errors

It is interesting to note that the two lemmas are quite similar in the sense that the mean $\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}$ and the middle $\boldsymbol{\varepsilon}_m$ of the input $\boldsymbol{\varepsilon}$ are multiplied by the DC-gain $|\mathcal{H}_{\boldsymbol{\varepsilon}}|_{DC}$, whereas the 2nd order moment and the radius are both multiplied by a norm of the system $\mathcal{H}_{\boldsymbol{\varepsilon}}$ (ℓ_2-norm and WCPG). Note that the ℓ_2-norm of the system is also the ℓ_2-norm of its impulse response (Parseval's theorem). So Lemma 1 (ℓ_2-norm of $\mathbb{h}_{\boldsymbol{\varepsilon}}$) gives the amplification of the variance of the roundoff errors when they propagate in the filter, while Lemma 2 (ℓ_1-norm) gives how their magnitude are amplified in the worst cases.

D. Probability Density Function

These two pieces of information, worst-case range and statistical moments of the output error, are precious results to characterize the overall error and to determine if its level is acceptable or not. However, worst cases may be or may not be really representative of real life output error, whereas SQNR is by definition more representative, but gives no usable information about the error magnitude.

A more interesting information would be the probability density function (pdf) or the cumulative distribution. It could let the designer determines for which probability the output error level is considered as representative [22]. Depending on the application, the worst cases, even extremely rare, is a must have, whereas only output error with reasonable probability will be considered for some other applications.

This has already been studied using Karhunen-Loève expansion [23], [24], but this rather complicated method has approximation (number of terms in the expansion, etc.) for which we do not know how to bound the associated error.

Remark that the signal to be quantized is already discrete, so are the quantization errors $\boldsymbol{\varepsilon}(k)$ and one should rather consider probability mass (pmf) function (also called discrete probability density function).

Assuming the roundoff errors are independent and uniformly distributed in their range (see Table I), we want to compute the probability mass distribution of the output error. From (20):

$$\boldsymbol{\Delta y}(k) = \sum_{l=0}^{k} \sum_{i=1}^{l+n+p} (\mathbb{h}_{\boldsymbol{\varepsilon}})_i(l)\boldsymbol{\varepsilon}_i(k-l) \tag{23}$$

so $\boldsymbol{\Delta y}(k)$ is the sum of $(l+m+p)(k+1)$ weighted discrete uncorrelated random variables.

Lemma 3 *Let X and Y be two uncorrelated discrete random variables, with pdf f_X and f_Y. If $Z = \alpha X + \beta Y$, then its pdf f_Z is given by the convolution of αf_X by βf_Y, i.e.:*

$$f_Z(x) = (\alpha f_X * \beta f_Y)(x) = \int_{-\infty}^{+\infty} \alpha f_X(u)\beta f_Y(x-u)du. \tag{24}$$

Denote $f_{\boldsymbol{\varepsilon}_i}$ the discrete pdf of the error $\boldsymbol{\varepsilon}_i$:

$$f_{\boldsymbol{\varepsilon}_i}(x) = \begin{cases} \frac{1}{2\boldsymbol{\varepsilon}_{i,r}} & \text{if } |x| \leqslant \boldsymbol{\varepsilon}_{i,r} \\ 0 & \text{elsewhere} \end{cases}. \tag{25}$$

Then applying Lemma 3 to (23) gives:

$$\forall k, \quad f_{\boldsymbol{\Delta y}(k)}(x) = \mathop{\ast}_{l=0}^{k} \left(\mathop{\ast}_{i=1}^{l+n+p} |(\mathbb{h}_{\boldsymbol{\varepsilon}})_i(l)|\, f_{\boldsymbol{\varepsilon}_i}(x) \right) \tag{26}$$

where $*$ is the convolution operator.

In [25], the problem of the distribution of the sum of non-identically distributed uniform random variables has been covered, but unfortunately the exact formula of the pdf (and pmf) are not usable in practice since they require the sum of 2^k terms (the pdf is a piece-wise function, and the number of piece doubles every convolution per $f_{\boldsymbol{\varepsilon}}$). Denote

$$f_{\boldsymbol{\Delta y}}(x) = \lim_{k\to\infty} f_{\boldsymbol{\Delta y}(\boldsymbol{k})}(x) \tag{27}$$

the final pdf of the output error. Denote $\rho(\boldsymbol{A})$ is the spectral radius of A, i.e. the maximum moduli of the poles of the systems $\mathcal{H}$ and $\mathcal{H}_{\boldsymbol{\varepsilon}}$. Since $|\mathbb{h}_{\boldsymbol{\varepsilon}}(k)|$ decrease as $\rho(\boldsymbol{A})^k$ when k tends to $+\infty$, (27) exists and can be approximated by a convolution of finite terms[1].

Let $\epsilon > 0$ be any arbitrary tolerance, we found N such that $\sum_{k=N+1}^{\infty} |\mathbb{h}_{\boldsymbol{\varepsilon}}(k)| < \epsilon$ by performing a pole-residue decomposition of $\mathcal{H}_{\boldsymbol{\varepsilon}}$ (or equivalent eigenvalue decomposition of $\boldsymbol{A}$) such that for all k, $\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{M}_1 = \sum_{i=1}^{n} \boldsymbol{R}_i\lambda_i^k$, where the λ_i are the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{R}_i \in \mathbb{R}^{p\times q}$ its associated residue. Then

$$\sum_{k=N+1}^{\infty} |\mathbb{h}_{\boldsymbol{\varepsilon}}(k)| \leqslant \sum_{i=1}^{n} \sum_{k=N+1}^{\infty} |\boldsymbol{R}_i|\,|\lambda_i|^k \tag{28}$$

$$\leqslant \sum_{i=1}^{n} |\boldsymbol{R}_i| \frac{|\lambda_i|^{N+1}}{1-|\lambda_i|} \leqslant \rho(\boldsymbol{A})^{N+1}\boldsymbol{M} \tag{29}$$

with $\boldsymbol{M} \triangleq \sum_{i=1}^{n} \frac{|\boldsymbol{R}_i|}{1-|\lambda_i|} \frac{|\lambda_i|}{\rho(\boldsymbol{A})} \in \mathbb{R}^{p\times q}$. Then N must satisfy

$$N \geqslant \left\lceil \frac{\log \frac{\epsilon}{m}}{\log(\rho(\boldsymbol{A}))} \right\rceil \tag{30}$$

[1]Remark that it is not possible to apply the Central Limit Theorem here, because $\sum_{k=0}^{\infty} \|\mathbb{h}_{\boldsymbol{\varepsilon}}(k)\|_F^2$ converges and thus the Lindeberg condition is not satisfied [26].

	Example 1	Example 2
$\rho(\boldsymbol{A})$	0.7387	0.3801
N	42	12
$\sigma_{\boldsymbol{\varepsilon}}^2$	1.242e-9	1.242e-9
$\sigma_{\boldsymbol{\Delta y}}^2$	8.9605e-10	3.169e-9
$\boldsymbol{\Delta y}_r$	2.205e-4	5.742e-4
time (s)	228.5	35.1

TABLE II
ROUNDOFF ERROR RANGES AND MOMENTS FOR THE TWO EXAMPLES.

with $m = \min_{i,j} |\boldsymbol{M}_{ij}|$. Using interval arithmetic and Theory of Verified Inclusions [27], a rigorous evaluation of $\boldsymbol{M}$ and N can be done [21]. Finally, computing $f_{\boldsymbol{\Delta y}(N)}$ with such a N is an approximation of $f_{\boldsymbol{\Delta y}}$.

Unfortunately, exact evaluation of $f_{\boldsymbol{\Delta y}(N)}$ is actually not possible using [25], [28], [29] when N is over about 25 (due to the 2^N terms involved). For the same reason, it is not possible either to use some probabilistic programming languages such as Edward[2] or PyMC3[3] to perform simulation-based pdf evaluation.
But it is possible to use some algorithms dedicated to fast convolution (based on Legendre series) like the one implemented in Chebfun[4] [30], and to reduce the complexity of the pdf (number of pieces) using the `simplify` and `merge` methods in Chebfun.

V. EXAMPLES

We start with the data-flow graph of Fig. 2, corresponding to a 3rd order Butterworth filter with 0.2 as cutoff frequency (the parameters γ_1, γ_2 and γ_3 of Fig. 2 are set to 0.4905, 0.4543 and 0.1910 respectively). Using [7] a SIF system has been deduced, and so the error system $\mathcal{H}_{\boldsymbol{\varepsilon}}$. Assuming a 16-bit Fixed-Point format $(0,-15)$ (so the input is in $[-1, 1-2^{-15}]$), and Sum-of-Products with faithful rounding as in section III-C, the interval of the error vector $\boldsymbol{\varepsilon}(k)$ is deduced ($\boldsymbol{\varepsilon}$ is centered). Using Propositions 1 and 2, the output error range and moments are determined. Table II sums up these properties, and gives the spectral radius of $\boldsymbol{A}$ and the minimum number of terms N to consider for the finite convolution of (27) with $\epsilon = 10^{-2}$.
Using the Chebfun convolution, the following probability density function for the output error is plot in Figure 7 (Chebfun simplifies it in 3 subdivisions of 515 pieces). The distribution of the output error $\boldsymbol{\Delta y}$ over a simulation of 10,000 uniformly distributed errors $\boldsymbol{\varepsilon}_i$ is superposed, with the worst-case interval error. We can see that the pdf determined with (26) matches quite nicely with the simulation.

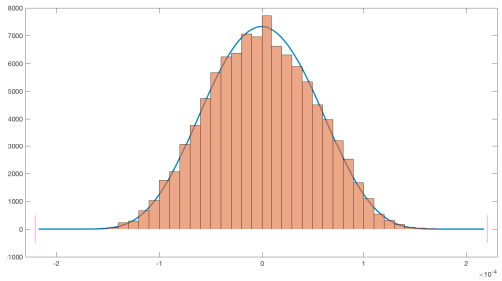

Fig. 7. Example 1: pdf of the output error $\boldsymbol{\Delta y}$, with histogram from the simulation.

The second example is a stable 5th order random filter (obtained using `dss` Matlab's function). Its low spectral radius implies to use few terms ($N = 12$) to approximate the output error.

The two pdf plotted in Figures 7 and 8 look like Gaussian distribution (as we can expect) but with finite support, since the pdf

[2]http://edwardlib.org
[3]https://github.com/pymc-devs/pymc3
[4]http://www.chebfun.org

is null outside of the range $\boldsymbol{\Delta y}_r$ (plotted with vertical bars). These two figures, or equivalently the cumulative distribution (not plotted here due to lack of space) can be used to determine the probability to have an output error bounded by a certain value.

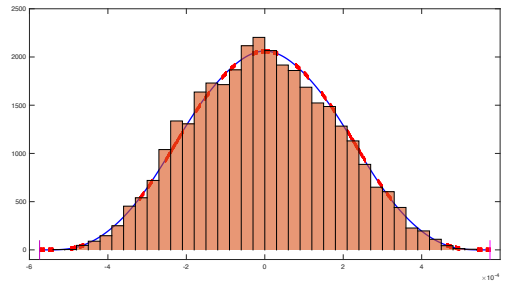

Fig. 8. Example 2: pdf of the output error $\boldsymbol{\Delta y}$, with histogram from the simulation.

VI. Conclusion

We gave an overview and comparison of three different approaches to analyze the output error of a linear system implemented in fixed-point arithmetic. They are all based on a specific analysis of the quantization errors that occur at each time step for each sum-of-product, and the analysis of the propagation of these errors through an error filter.

Statistical approach gives the level of the output error from a power point of view, whereas using intervals we focus on worst-case errors. Being able to approximate and evaluate the probability density function is a real issue to tackle the complexity of the error analysis. This approach could also be used with different models of error. For example, approximate computing and operators leads to an error non uniformly distributed. This is due to the fact that some errors, with large amplitude, may sometime appear with a low probability. In such cases, worst-case and statistical approaches cannot handle completely the complexity of these rare but large errors.

We should continue our work in order to analyze these fast convolution algorithms and to understand how we can more easily use them when we require the number of convolutions to be much larger (up to 100 or 1000). A comparison with the Karhunen-Loève expansion should be explored.

References

[1] A. Fettweiss, "Wave digital filters: Theory and practice," *Proc. of the IEEE*, vol. 74, no. 2, 1986.

[2] L. Gazsi, "Explicit formulas for lattice wave digital filters," *IEEE Trans. Circuits & Systems*, vol. 32, no. 1, 1985.

[3] G. Li and Z. Zhao, "On the generalized DFIIt structure and its state-space realization in digital filter implementation," *IEEE Trans. on Circuits and Systems*, vol. 51, no. 4, pp. 769–778, April 2004.

[4] M. Gevers and G. Li, *Parametrizations in Control, Estimation and Filtering Probems*. Springer-Verlag, 1993.

[5] R. Istepanian and J. Whidborne, Eds., *Digital Controller implementation and fragility*. Springer, 2001.

[6] T. Hilaire, P. Chevrel, and J. Whidborne, "A unifying framework for finite wordlength realizations," *IEEE Trans. on Circuits and Systems*, vol. 8, no. 54, pp. 1765–1774, August 2007.

[7] T. Hilaire, A. Volkova, and M. Ravoson, "Reliable fixed-point implementation of linear data-flows," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, 2016.

[8] J. Aplevich, *Implicit Linear Systems*. Springer-Verlag, 1991.

[9] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge, UK: Cambridge University Press, 2008.

[10] C. Mullis and R. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," in *IEEE Trans. on Circuits and Systems*, vol. 23, no. 9, Sept. 1976.

[11] D. Menard and O. Sentieys, "A methodology for evaluating the precision of fixed-point systems," in *ICASSP*, 2002, pp. 3152–3155.

[12] B. Lopez, T. Hilaire, and L.-S. Didier, "Formatting bits to better implement signal processing algorithms," in *4th int. Conf. on Pervasive and Embedded Computing and Communication Systems*, Jan. 2014.

[13] J. Cavanagh, *Computer Arithmetic and Verilog HDL Fundamentals*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.

[14] "IEEE Standard for Standard SystemC Language Reference Manual," *IEEE Std 1666-2011*, pp. 1–638, Jan 2012.

[15] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.

[16] J. v. Neumann, "First draft of a report on the edvac," the United States Army Ordinance Department and the University of Pennsylvania Moore School of Electrical Engineering, Tech. Rep., 1945.

[17] J. Lopez, C. Carreras, and O. Nieto-Taladriz, "Improved interval-based characterization of fixed-point LTI systems with feedback loops," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 11, pp. 1923–1933, November 2007.

[18] A. Volkova, T. Hilaire, and C. Lauter, "Determining fixed-point formats for a digital filter implementation using the worst-case peak-gain measure," in *Asilomar Conf. on Signals, Systems and Computers*, Nov. 2015.

[19] V. Balakrishnan and S. Boyd, "On computing the worst-case peak gain of linear systems," *Systems & Control Letters*, vol. 19, 1992.

[20] J. Proakis and D. Manolakis, *Digital Signal Processing principles,algorithms, and applications*, 3rd ed. Upper Saddle River, New Jersey: Prentice All, 1996.

[21] A. Volkova, T. Hilaire, and C. Q. Lauter, "Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision," in *22nd IEEE Symposium on Computer Arithmetic*, 2015.

[22] A. Banciu, E. Casseau, D. Menard, and T. Michel, "A case study of the stochastic modeling approach for range estimation," in *Conf. on Design and Architectures for Signal and Image Processing (DASIP)*, 2010.

[23] A. Banciu, "A stochastic approach for the range evaluation," Ph.D. dissertation, Université de Rennes I, 2012.

[24] B. Wu, J. Zhu, and F. N. Najm, "An analytical approach for dynamic range estimation," in *Proceedings. 41st Design Automation Conference, 2004.*, July 2004, pp. 472–477.

[25] D. M. Bradley and R. C. Gupta, "On the distribution of the sum of n non-identically distributed uniform random variables," *Annals of the Institute of Statistical Mathematics*, pp. 689–70, 2002.

[26] W. Feller, *An Introduction to Probability Theory and Its Applications, 2nd edition*. Wiley, January 1991, vol. 2.

[27] S. M. Rump, "New results on verified inclusions," in *Accurate Scientific Computations, Symposium, Proceedings*, 1985.

[28] F. Killmann and E. von Collani, "A note on the convolution of the uniform and related distributions and their use in quality control," *Economic Quality Control*, no. 1, pp. 17–41, 2001.

[29] J. S. Kang, S. L. Kim, Y. H. Kim, and Y. S. Jang, "Generalized convolution of uniform distributions," *J. Applied Mathematic & Informatics*, vol. 28, no. 5-6, pp. 1573–1581, 2010.

[30] N. Hale and A. Townsend, "An algorithm for the convolution of Legendre series," *SIAM J. Scientific Computing*, vol. 36, no. 3, 2014.