Arithmetic approaches for rigorous design of reliable Fixed-Point LTI filters

Anastasia Volkova, Thibault Hilaire, Christoph Lauter

Arithmetic approaches for rigorous design of reliable Fixed-Point LTI filters

Anastasia Volkova, Thibault Hilaire, and Christoph Lauter

Abstract—In this paper we target the Fixed-Point (FxP) implementation of Linear Time-Invariant (LTI) filters evaluated with state-space equations. We assume that wordlengths are fixed and that our goal is to determine binary point positions that guarantee the absence of overflows while maximizing accuracy. We provide a model for the worst-case error analysis of FxP filters that gives tight bounds on the output error. Then we develop an algorithm for the determination of binary point positions that takes rounding errors and their amplification fully into account. The proposed techniques are rigorous, i.e. based on proofs, and no simulations are ever used.
In practice, Floating-Point (FP) errors that occur in the implementation of FxP design routines can lead to overestimation/underestimation of resulting parameters. Thus, along with FxP analysis of digital filters, we provide FP analysis of our filter design algorithms. In particular, the core measure in our approach, Worst-Case Peak Gain, is defined as an infinite sum and has matrix powers in it. We provide fine-grained FP error analysis of its evaluation and develop multiple precision algorithms that dynamically adapt their internal precision to satisfy an a priori absolute error bound. Our techniques on multiple precision matrix algorithms, such as eigendecomposition, are of independent interest as a contribution to Computer Arithmetic. All algorithms are implemented as C libraries, integrated into an open-source filter code generator and tested on numerical examples.

Index Terms—Floating-Point Arithmetic, Fixed-Point Arithmetic, Multiple Precision, Interval Arithmetic, Digital Filters, Reliable Computations, Eigendecomposition, Gershgorin circles, Table Maker's Dilemma

1 Introduction

Linear Time-Invariant (LTI) digital filters are ubiquitous in digital signal processing and control. Their applications vary from the simplest audio filters and equalizers to biomedical and autonomous driving systems. LTI filters are often implemented for embedded systems with Fixed-Point (FxP) arithmetic and have strong performance and cost constraints in terms of latency, throughput, area, power consumption, etc. Filter designers make various compromises and simplifications of filter algorithms to achieve satisfactory results. In particular, choices related to the finite-precision arithmetic have strong influence on system performance, e.g. in terms of throughput and area. In this work we are interested in the accuracy vs. performance trade-off.

When implementing signal processing systems with high safety standards, a filter designer must provide guarantees on the numerical quality of the implemented systems: absence of overflows, bounds on the output time-domain errors, etc. Usually such numerical guarantees, if at all provided, come at relatively high cost: algorithms are often implemented with more computational resources than are actually needed.

In this paper we aim at providing numerical guarantees for the implementation of LTI filters at the lowest cost (in terms of performance of the implemented algorithm). In particular, we consider implementation of *recursive* LTI filters with FxP arithmetic and study the rounding errors that occur in the finite-precision implementation. Our goal is to develop a generic approach that: 1/ provides tight bounds on rounding errors that occur in finite-precision computations; 2/ determines in a reliable way efficient parameters for a FxP implementation (i.e. the most and the least significant bit positions for variables) while fully taking into account the impact of rounding errors.

We demonstrate our approach on LTI filters that are evaluated using the state-space algorithm [1, Chapter 6, pp.391]. This algorithm makes the notion of the feedback loop explicit: internal states are updated at each iteration and outputs are computed with the states and input signals. Analysis of recursive filters is highly non-trivial since errors may accumulate and can be amplified at each iteration. Hence, the impact of rounding errors must be taken into account when choosing data formats for all variables.

Existing approaches to LTI filter implementation either cannot provide strong enough guarantees or do not fully support feedback loop systems, mainly because these approaches are based on statistical measures which describe the impact of errors only in terms of mean and variance and not in its absolute value. For details see related work and positioning in section 2.3.

In this work we measure rounding errors in their absolute value to provide tight and guaranteed bounds. We provide a model that rigorously takes into account the error propagation through the feedback loop. This model is based on the so-called Worst-Case Peak Gain (WCPG) measure [2] which provides a bound on a filter's output for stable filters. In contrast to existing approaches that straightforwardly use Floating-Point (FP) arithmetic and generic computation environments such as Matlab, we demonstrate that FP rounding errors themselves represent an additional source of errors that must be dealt with. Overall, we provide two levels of algorithms and error analysis: 1/ a high-level analysis of FxP computations and rounding errors, algorithms for the computation of reliable FxP data formats; and 2/ low-level algorithms for the reliable floating-point evaluation of measures needed for the FxP error analysis, controlling the impact of FP errors upon computed FxP formats.

- *A. Volkova is with University of Nantes, France. E-mail: see http://www.avolkova.org/*
- *T. Hilaire is with Sorbonne Université, Paris, and with Inria and LRI Université Paris-Saclay, Orsay, France. E-mail: see http://www.docmatic.fr/*
- *C. Lauter is with Department of Computer Science & Engineering, UAA College of Engineering, University of Alaska Anchorage, USA. E-mail: see http://www.christoph-lauter.org/*

To summarize the contributions of this paper, we extend the results of [3] and [4] and

- propose a new iterative algorithm which, given a stable recursive filter and wordlength constraints, determines the FxP formats that guarantee absence of overflows;
- prove that the FP evaluation of the FxP formats is, in most cases, exact (formats are overestimated in some rare cases but at most by one bit) and underestimation never occurs;
- enable the above contributions by introducing the first algorithm for arbitrarily accurate evaluation of the WCPG;
- develop multiple precision FP algorithms, such as complex matrix arithmetic that dynamically adapt their precision to satisfy a priori given accuracy; all error bounds are proven;
- develop a technique for multiple precision eigendecomposition and for matrix inversion based on Newton-Raphson iteration;
- identify the FxP format overestimation problem as a Table Maker's Dilemma and propose to solve it using Integer Linear Programming.

Our approaches are integrated into the open-source filter code generator FiXiF[1]. In this paper we demonstrate algorithms on a state-space algorithm but extend them upon any LTI filter in the FiXiF tool.

The paper is organized as follows. We start with recalling basic information on LTI digital filters evaluated with the state-space algorithm. Then, we give definitions related to Fixed-Point arithmetic for filter implementation and justify the choices of arithmetics for analysis. Section 2.3 briefly reviews the related work and clarifies the positioning of the current work w.r.t. existing approaches. In Section 3 we give the error-analysis of the FxP implementation of state-space systems and provide an iterative algorithm for the FxP format choice. Then, in Section 4 we provide an FP rounding error analysis of the iterative algorithm itself. In the core of our approach lies the reliable evaluation of the WCPG measure. We present in Section 5 our core algorithm, for arbitrarily accurate evaluation of the WCPG. Then, in Section 6 we demonstrate efficiency of our approach on numerical examples. Finally, Section 7 gives an overview for applications of the WCPG for new hardware implementations and filter verification before conclusion.

Notation

Throughout this article scalar quantities, vectors and matrices are in lowercase, lowercase boldface and uppercase boldface, respectively (e.g. x, $\boldsymbol{x}$ and $\boldsymbol{X}$). Unless otherwise stated, all matrix absolute values, inequalities and intervals are applied element-by-element. Norms, such as Frobenius norm, notated $\|\boldsymbol{A}\|_F$, stay of course norms on matrices and are not to be understood element-by-element. The conjugate transpose of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^*$, and vector transpose by $\boldsymbol{x}^\top$. Operators $\otimes$ and $\oplus$ denote Floating-Point (FP) multiplication and addition, respectively, $\mathbb{F}$ the set of radix-2 FP numbers. An interval $[x]$ is defined with its lower and upper bounds $[x] := [\underline{x}, \overline{x}]$. An interval matrix is denoted by $[\boldsymbol{M}] := [\underline{\boldsymbol{M}}, \overline{\boldsymbol{M}}]$, where each element $[\boldsymbol{M}_{ij}]$ is an interval $[\boldsymbol{M}_{ij}] = [\underline{\boldsymbol{M}}_{ij}, \overline{\boldsymbol{M}}_{ij}]$. If $\boldsymbol{x} \in \mathbb{R}^n$, then $2^{\boldsymbol{x}}$ denotes the vector $(2^{\boldsymbol{x}_1}, \ldots, 2^{\boldsymbol{x}_n})$.

2 Background and Related Work

2.1 LTI filters

The main objects of this paper are Linear Time Invariant (LTI) digital filters. LTI filters are specified in frequency-domain via $\mathcal{Z}$-transform [1, Chapter 3, pp.105]. For a given frequency-domain description of a digital filter there exist numerous ways to evaluate it in the time domain. However, the questions of choice of the best algorithm are out of scope of this paper.

Without loss of generality, we consider LTI digital filters evaluated via *state-space* equations, which are presented just below. Indeed, in [5], [6] the authors showed that state-space based approaches can be extended to any linear filter algorithm using a unifying framework.

An n^{th} order state-space system $\mathcal{H}$ with q inputs and p outputs is described with

$$\mathcal{H}\begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}, \tag{1}$$

where $k = 0, 1, \ldots$ is the time instance, $\boldsymbol{u}(k) \in \mathbb{R}^q$ is the input, $\boldsymbol{y}(k) \in \mathbb{R}^p$ is the output and $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state vector; matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times q}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{D} \in \mathbb{R}^{p\times q}$ are the state matrices of the system. In the case of a Single Input Single Output (SISO) system, $\boldsymbol{B}$ and $\boldsymbol{C}$ are vectors, and $\boldsymbol{D}$ is a scalar, which we shall indicate appropriately as $\boldsymbol{b}, \boldsymbol{c}$ and d.

In practice, one is interested in Bounded-Input Bounded-Output (BIBO) stable systems, i.e. those that guarantee bounded output sequences for bounded inputs. These systems satisfy the following property:

$$\rho(\boldsymbol{A}) = \max_i |\lambda_i| < 1, \tag{2}$$

where λ_i are the eigenvalues of $\boldsymbol{A}$ and $\rho(\boldsymbol{A})$ is its spectral radius. Throughout the paper we deal only with stable filters.

The output of stable filters can be bounded using the following classic theorem.

Theorem 1 (Worst-Case Peak Gain [1], [2])**.** *Let $\mathcal{H}$ be a BIBO-stable n^{th} order state-space system with q inputs, p outputs. If an input signal is bounded in magnitude, as $|\boldsymbol{u}(k)| \le \bar{\boldsymbol{u}}$ for all $k \ge 0$, then the output $\boldsymbol{y}(k)$ is bounded by*

$$\forall k \ge 0, \quad |\boldsymbol{y}(k)| \le \langle\langle \mathcal{H} \rangle\rangle \bar{\boldsymbol{u}} \tag{3}$$

where $\langle\langle \mathcal{H} \rangle\rangle \in \mathbb{R}^{p\times q}$ is the Worst-Case Peak Gain (WCPG) matrix [2] of the system and can be expressed as:

$$\langle\langle \mathcal{H} \rangle\rangle = |\boldsymbol{D}| + \sum_{k=0}^{\infty} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|. \tag{4}$$

Remark 1. *For each component $\boldsymbol{y}_i(k)$ of the output it is possible to find a finite input signal $\{\boldsymbol{u}(k)\}_{0\le k\le K}$ that makes $\boldsymbol{y}_i(k)$ arbitrarily close to the bound $\langle\langle \mathcal{H} \rangle\rangle \bar{\boldsymbol{u}}$. In the SISO case, such a the worst-case input signal is*

$$u(j) = \begin{cases} \bar{u} \cdot \operatorname{sign}(d) & \text{for } j = 0 \\ \bar{u} \cdot \operatorname{sign}\left(\boldsymbol{c}\boldsymbol{A}^{K-j-1}\boldsymbol{b}\right) & \text{for } 0 < j < K. \end{cases} \tag{5}$$

where $\operatorname{sign}(x)$ *returns* ± 1 *or* 0 *depending on the value of x.*

2.2 Arithmetics

We consider the implementation of digital filters on processors that do not have an FP unit and only use Fixed-Point (FxP) arithmetic. Implementation in this case involves the choice of the FxP formats for algorithm parameters (e.g. coefficients of the state-space) and for algorithm variables. We leave the questions of format choice for filter coefficients out of the scope of this paper and deal here only with the impact of rounding errors in the computations, thus look for the formats for filter variables.

1. https://github.com/fixif

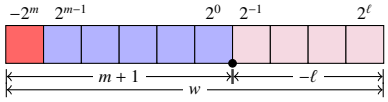


Fig. 1. Fixed-point representation (here, $m = 5$ and $\ell = -4$).

2.2.1 Target arithmetic: Fixed-Point

A radix-2 two's complement FxP number system [7] is a subset of signed real numbers whose elements are w-bit integers scaled by a fixed factor. Such elements have the form $T \cdot 2^\ell$, where $T \in [-2^{w-1}; 2^{w-1} - 1] \cap \mathbb{Z}$ is an integer mantissa and 2^ℓ is an implicit quantization factor.

Let t be a signed FxP number. It is written as

$$t = -2^m t_m + \sum_{i=\ell}^{m-1} 2^i t_i, \tag{6}$$

where t_i is the i^{th} bit of t, and m and ℓ are the *Most Significant Bit* (MSB) and *Least Significant Bit* (LSB) positions of t (see Fig. 1) respectively. The wordlength w is related with the MSB and LSB positions via

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

The range of numbers that can be represented with the wordlength w and quantization factor ℓ is the interval $[-2^m; 2^m - 2^\ell]$, called dynamic range.

When determining FxP formats for variables in an algorithm, we need to rigorously determine their ranges. Otherwise, an overflow may occur, which means that some integer mantissa would exceed the range $[-2^{w-1}; 2^{w-1} - 1]$.

A common practice is to suppress overflows using techniques such as wrap-around mode, or saturation that replaces the positive or negative overflows with the largest positive or largest negative representable values of the target format [8], accordingly. Saturated values may give an impression to be correct but introduce non-linear distortions to the output and their impact on the output cannot be analyzed without knowledge of the magnitude of the overflows.

In this paper we claim that one can reliably an efficiently implement filters with a "by construction" guarantee that no overflow occurs.

2.2.2 Toolkit: Floating-Point, Multiple Precision and Interval arithmetics

Classically, in signal processing the attention is brought to the determination of FxP formats, while rounding errors in the evaluation of the MSB and LSB positions themselves are ignored. We use Floating-Point (FP) arithmetic as our main instrument for computation of parameters of filter implementation. In most cases, errors due to FP computations pass unnoticed but can have a drastic effect upon the filter implementation process, leading to incorrect parameters. For instance, as we will see in Section 4, FP errors can easily lead to an off-by-one error in MSB computation.

According to the IEEE 754 [9] standard for FP arithmetic and classical books [10], [11], a normal binary FP number is written as

$$x = (-1)^s \cdot M \cdot 2^{e-p+1}, \tag{8}$$

where $s \in \{0, 1\}$ is the sign bit; the exponent e and the normalized significand M is a p-bit integer such that $2^{p-1} \leq M \leq 2^p - 1$. Any result of FP computation is prone to rounding errors; these can be controlled by varying the compute precision p.

In this paper we will reason in terms of absolute errors, whereas FP arithmetic is optimized for achieving relative error bounds. The issue is that in our FP algorithms we will refer to outputs being computed with an absolute a priori error bound. To connect the relative nature of FP arithmetic and these absolute error bounds, we will use a Multiple Precision (MP) floating-point arithmetic, together with ways to dynamically adapt the compute precision p. We use the GNU MPFR[2] library [12] for all implementations.

We will also require error bounds on the solutions of some linear algebra problems, such as eigendecomposition. Bounding errors on those is a highly non-trivial task. We deal with this by employing Interval Arithmetic (IA) that permits us to compute safe intervals around approximated values that guarantee to contain the exact result.

We denote by $[x] = [\underline{x}, \overline{x}]$ an interval, which is a closed and bounded nonempty set $\left\{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\right\}$. When we compute a function on the interval argument, we seek to determine the intervals around the output such that the inclusion property is satisfied. We use the MPFI [13] library that safely implements IA and ensures that the inclusion property for basic arithmetic operations are maintained even when using FP numbers.

2.3 Related work and positioning

When implementing algorithms in FxP arithmetic, two issues should be addressed:

- *range analysis*, which consists in studying data ranges and choosing MSB to avoid overflows;
- *rounding error analysis*, which consists in studying impact of rounding errors to bound the output error or to choose the LSB positions

A possible overestimation of the MSB leads to an increased implementation cost. Any underestimation of the MSB position, however, may lead to an overflow at some point of the execution and consequently lead to unexpected behavior and modify the signal shape.

Most works in the literature can be seen as either analytical/statistic, or simulation-based.

Simulation-based methods [14], [15] are slower, requiring extensive data sets, and do not provide guarantee beyond the data sets used. Since we aim at providing guarantees for any possible input, we do not use any simulations in our approaches.

A common analytical approach [1, 6.9, pp.454] for rounding error analysis is to view rounding errors as white additive noise uncorrelated with the filter's variables and to analyze their mean and variance. This assumption is not strictly correct but somehow realistic when the number of rounded bits is reasonable with respect to the total wordlength [16]. Then the Signal Quantization Noise Ratio (SQNR), defined as the ratio of the variance of the output signal by the variance of the output noise, serves as a measure of errors [1], [17]. SQNR does not provide a precise measure for the accuracy but merely estimates the power of the noises propagated to the output, and gives an idea of the number of meaningful bits in the result. In [18] authors focus on the noise power analysis and influence of the rounding errors on the frequency-domain behavior.To obtain bounds on the output errors in the time-domain and determine the number of correct bits, rounding errors should be measured by their absolute value instead.

Range analysis and measuring rounding errors by their magnitude can be done using Interval Arithmetic [19], [20], [21],

2. http://www.mpfr.org/

Affine Arithmetic [22], [23], [24], [25] and its generalization to higher-order error polynomials [26] and even SAT-modulo theory (SMT) [20] (IA and AA mostly for rounding error analysis, and SMT and IA for range analysis). The problem is that many previous analytical methods either do not fully support recursive filters (intervals explode due to the wrapping effect, or the number of error terms in affine form is based on heuristic) [21], [24], [25].Generic techniques such as abstract interpretation [27] combined with IA or AA may be used to provide guarantees on programs with loops, but these guarantees will be very pessimistic for sensitive recursive filters.

State-of-the-art work on wordlength optimization, such as work by Sarbishei and Radecka [24], is based on the combination of IA and the WCPG theorem presented in Section 2.1. However, the infinite series in (4) is truncated in a heuristic (and, for sensitive filters, completely unreliable) way and then evaluated in Matlab in FP arithmetic without accuracy guarantees. Besides, the model they use for rounding error analysis is specific to their hardware model and a particular filter evaluation scheme.

In [28], in order to evaluate the WCPG measure Monniaux translates a SISO filter equation to the frequency domain and proposes an approach to bound the rational transfer function using power series development. For the actual evaluation he uses interval arithmetic but to practically bound the tail of the series he stops the computation when the tail term has an indefinite sign, which is an experimentally-based assumption. This approach provides only an a posteriori error bound for the evaluation, not permitting an arbitrarily accurate evaluation (necessary to, for example, provide tight error bounds) and is applicable only to the SISO case.

In our work we also base range analysis on the WCPG theorem but provide the first method on the *reliable* FP evaluation of the WCPG measure with a priori given accuracy that is guaranteed to be satisfied. This contribution is detailed in Section 5. We provide a complete and general methodology in which we measure rounding errors by their absolute values and capture their propagation through the filters using a simple but rigorous model. Our approach provides tight (not uselessly pessimistic) and strong (worst-case) guarantees on the results of LTI filters with feedback loop, even for sensitive filters.

3 Fixed-Point implementation of recursive filters

3.1 Problem statement

The problem of determining the Fixed-Point Formats (FxPF) for an implementation of a filter $\mathcal{H}$ can be formulated under various different hypotheses. Here we formulate it as follows.

Let $\mathcal{H}$ be a filter in a state-space representation (1). Suppose all the inputs to be exact and in an interval bounded by $\bar{\boldsymbol{u}}$. Given the wordlength constraints vector $\boldsymbol{w}_x$ for the state and $\boldsymbol{w}_y$ for the output variables we look for a FxPF for $\boldsymbol{x}(k)$ and $\boldsymbol{y}(k)$ such that for any possible input no overflow occurs. Obviously, we seek to maximize the accuracy of computations, so we search for the **least** (element-by-element) MSB vectors $\boldsymbol{m}_y$ and $\boldsymbol{m}_x$ such that

$$\forall k \geq 0, \quad \boldsymbol{y}(k) \in [-2^{-\boldsymbol{m}_y}; 2^{\boldsymbol{m}_y} - 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}], \tag{9}$$

$$\forall k \geq 0, \quad \boldsymbol{x}(k) \in [-2^{-\boldsymbol{m}_x}; 2^{\boldsymbol{m}_x} - 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1}]. \tag{10}$$

Since the filter $\mathcal{H}$ is linear and the input interval is centered at zero, the output interval is also centered in zero. Therefore, it will be sufficient to determine the least $\boldsymbol{m}_y$ and $\boldsymbol{m}_x$ such that

$$\forall k \geq 0, \quad |\boldsymbol{y}(k)| \leq 2^{\boldsymbol{m}_y} - 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}, \tag{11}$$

$$\forall k \geq 0, \quad |\boldsymbol{x}(k)| \leq 2^{\boldsymbol{m}_x} - 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1}. \tag{12}$$

3.2 Applying WCPG to compute MSB

The idea is to apply the WCPG theorem to compute the ranges of variables. However, WCPG acts only upon the filter's outputs. To extend the result for the state variables, we modify the filter's model by incorporating state variables into the output, as if they were not only stored from one iteration to another, but also given out each time.

Let $\boldsymbol{\zeta}(k) := \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{pmatrix}$ be a new output vector. Then the state-space relationship in 1 takes the form:

$$\mathcal{H}_\zeta \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{\zeta}(k) = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \boldsymbol{u}(k) \end{cases}. \tag{13}$$

This new filter $\mathcal{H}_\zeta$ serves only as a model to apply the WCPG theorem, it is never actually implemented.

Now the corresponding vector $\boldsymbol{w}_\zeta \in \mathbb{Z}^{n+p}$ of wordlength constraints is just a concatenation of $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$.

Applying the WCPG theorem to $\mathcal{H}_\zeta$ yields the following bound:

$$\forall k \geq 0, \quad |\boldsymbol{\zeta}_i(k)| \leq \left(\langle\langle \mathcal{H}_\zeta \rangle\rangle \bar{\boldsymbol{u}} \right)_i, \quad i = 1, \ldots, n+p. \tag{14}$$

We look for the least MSB positions $\boldsymbol{m}_\zeta$ such that

$$\forall k \geq 0, \quad |\boldsymbol{\zeta}(k)| \leq 2^{\boldsymbol{m}_\zeta} - 2^{\boldsymbol{m}_\zeta - \boldsymbol{w}_\zeta + 1}. \tag{15}$$

By applying the above bound on (14), we obtain a simple formula for the computation of MSB positions:

$$\boldsymbol{m}_{\zeta_i} = \left\lceil \log_2 \left((\langle\langle \mathcal{H}_\zeta \rangle\rangle \bar{\boldsymbol{u}})_i \right) - \log_2 \left(1 - 2^{1 - \boldsymbol{w}_{\zeta_i}} \right) \right\rceil \quad \text{for } i = 1, \ldots, n+p. \tag{16}$$

3.3 Taking rounding errors of the implemented filter into account

Problem (16) is usually the one that is solved in wordlength optimization. However, the rounding errors that are induced with each computation may propagate up to the MSB position, changing the dynamic range of signals. This effect should be accounted for.

Due to rounding in finite-precision computations, we model the *implemented* filter as $\mathcal{H}_\zeta^\diamondsuit$:

$$\mathcal{H}_\zeta^\diamondsuit \begin{cases} \boldsymbol{x}^\diamondsuit(k+1) = \diamondsuit_{\boldsymbol{\ell}_x} \left(\boldsymbol{A}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{B}\boldsymbol{u}(k) \right) \\ \boldsymbol{\zeta}^\diamondsuit(k) = \diamondsuit_{\boldsymbol{\ell}_\zeta} \left(\begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}^\diamondsuit(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \boldsymbol{u}(k) \right) \end{cases}, \tag{17}$$

where the Sums-of-Products (accumulation of scalar products on the right-hand side) are computed with some rounding operator $\diamondsuit_\ell$. Suppose this operator ensures faithful rounding [10] so that:

$$|\diamondsuit_\ell(x) - x| < 2^\ell, \tag{18}$$

where ℓ is the LSB position of the operator's output. It was shown in [29] that such an operator can be implemented using a few guard bits for the accumulation.

Let the error vectors due to $\diamondsuit_\ell$ be denoted as $\boldsymbol{\varepsilon}_x(k)$ and $\boldsymbol{\varepsilon}_y(k)$ for the state and output vectors, respectively. Essentially, the vectors $\boldsymbol{\varepsilon}_x(k)$ and $\boldsymbol{\varepsilon}_y(k)$ may be associated with the noise induced by the filter implementation but in contrast to statistical approaches, we

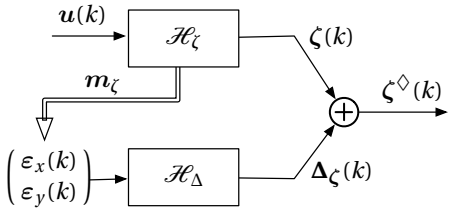


Fig. 2. Implemented filter decomposition.

measure them as intervals. The implemented filter can be rewritten as

$$\mathcal{H}_\zeta^\diamondsuit \begin{cases} \boldsymbol{x}^\diamondsuit(k+1) &= \boldsymbol{A}\boldsymbol{x}^\diamondsuit(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{\varepsilon}_x(k) \\ \boldsymbol{\zeta}^\diamondsuit(k) &= \begin{pmatrix}\boldsymbol{I}\\ \boldsymbol{C}\end{pmatrix}\boldsymbol{x}^\diamondsuit(k) + \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{D}\end{pmatrix}\boldsymbol{u}(k) + \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{pmatrix}\boldsymbol{\varepsilon}_y(k) \end{cases}, \quad (19)$$

where

$$|\boldsymbol{\varepsilon}_x(k)| < 2^{\boldsymbol{\ell}_x}, \quad \left|\boldsymbol{\varepsilon}_y(k)\right| < 2^{\boldsymbol{\ell}_y}.$$

Remark that since the operator $\diamondsuit_\ell$ is applied, $\boldsymbol{\varepsilon}_x(k) \neq \boldsymbol{x}(k) - \boldsymbol{x}^\diamondsuit(k)$ and $\boldsymbol{\varepsilon}_y(k) \neq \boldsymbol{y}(k) - \boldsymbol{y}^\diamondsuit(k)$. As the rounding also affects the filter state, the $\boldsymbol{x}^\diamondsuit(k)$ drifts away from $\boldsymbol{x}(k)$ over time, whereas with $\boldsymbol{\varepsilon}_x(k)$ we consider the error due to *one step only*.

At each instance of time both input and error vectors are propagated through the filter. Thanks to the linearity of filters, we model the output of the implemented filter $\mathcal{H}_\zeta^\diamondsuit$ as the sum of the output of the exact filter and a special "error-filter", denoted by $\mathcal{H}_\Delta$, which describes the propagation of the error vectors. This decomposition is illustrated in Figure 2. Note that this "error-filter" is an artificial one; it is not required to be implemented and serves exclusively for error-analysis purposes.

More precisely, the filter $\mathcal{H}_\Delta$ is obtained by computing the difference between $\mathcal{H}_\zeta^\diamondsuit$ and $\mathcal{H}_\zeta$. This filter takes the rounding errors $\boldsymbol{\varepsilon}(k) := \begin{pmatrix}\boldsymbol{\varepsilon}_x(k)\\ \boldsymbol{\varepsilon}_y(k)\end{pmatrix}$ as input and returns the result of their propagation through the filter:

$$\mathcal{H}_\Delta \begin{cases} \boldsymbol{\Delta}_x(k+1) &= \boldsymbol{A}\boldsymbol{\Delta}_x(k) + \begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{\varepsilon}(k) \\ \boldsymbol{\Delta}_\zeta(k) &= \begin{pmatrix}\boldsymbol{I}\\ \boldsymbol{C}\end{pmatrix}\boldsymbol{\Delta}_x(k) + \begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{pmatrix}\boldsymbol{\varepsilon}(k) \end{cases}, \quad (20)$$

where $\boldsymbol{\varepsilon}(k)$ is guaranteed to be bounded by $\bar{\boldsymbol{\varepsilon}} := 2^{\boldsymbol{\ell}_\zeta}$.

Once the decomposition is done, we can apply the WCPG theorem on the "error-filter" $\mathcal{H}_\Delta$ and deduce the output interval of the computational errors propagated through the filter:

$$\forall k \geq 0, \quad \left|\boldsymbol{\Delta}_\zeta(k)\right| \leq \langle\langle \mathcal{H}_\Delta \rangle\rangle \bar{\boldsymbol{\varepsilon}}. \quad (21)$$

Hence, the output of the implemented filter is bounded with

$$\left|\boldsymbol{\zeta}^\diamondsuit(k)\right| = \left|\boldsymbol{\zeta}(k) + \boldsymbol{\Delta}_\zeta(k)\right| \leq |\boldsymbol{\zeta}(k)| + \left|\boldsymbol{\Delta}_\zeta(k)\right|. \quad (22)$$

Remark 2. *When applying the triangle inequality in* (22) *we actually overestimate the bound. From a practical point of view, it can be interpreted as an assumption that the input signal that leads to the worst-case output also leads to the worst-case rounding errors. This is not generally true. Also, the error inputs themselves cannot be exercised concurrently but only element-wise. Thus, the triangle inequality bound is not generally attained. Consequently, the "least" MSB positions that we compute further are not the least possible but the least for our way to model the errors and their propagation. In Section 4.2 we propose an approach for dealing with this potential overestimation.*

Applying the WCPG theorem to the implemented filter and using (22) we can computed the MSB vector $\boldsymbol{m}_\zeta^\diamondsuit$ as

$$\boldsymbol{m}_{\zeta_i}^\diamondsuit = \left\lceil \log_2\left(\left(\langle\langle \mathcal{H}_\zeta \rangle\rangle \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta \rangle\rangle \bar{\boldsymbol{\varepsilon}})_i\right) - \log_2\left(1 - 2^{1-w_{\zeta_i}}\right)\right\rceil, \quad (23)$$
$$\text{for } i = 1, \ldots, n+p.$$

Therefore, the FxP formats $(\boldsymbol{m}_\zeta^\diamondsuit, \boldsymbol{\ell}_\zeta^\diamondsuit)$, where the LSB $\boldsymbol{\ell}_\zeta^\diamondsuit$ is computed via (7), guarantee that *no overflows occur for the implemented filter*.

3.4 Complete algorithm for reliable MSB

Since the input of the error filter $\mathcal{H}_\Delta$ depends on the error-bound vector $\boldsymbol{\varepsilon}_\zeta$ of the FxP formats chosen for implementation, we cannot directly use (23). The idea is to first compute the FxP formats of the variables in the exact filter $\mathcal{H}_\zeta$, where computational errors are not taken into account, and then use them as an initial guess for the implemented filter $\mathcal{H}_\zeta^\diamondsuit$. Hence, we obtain the following two-step algorithm:

Step 1: Determine the FxP formats $(\boldsymbol{m}_\zeta, \boldsymbol{\ell}_\zeta)$ for the exact filter $\mathcal{H}_\zeta$
Step 2: Construct the "error-filter" $\mathcal{H}_\Delta$, which gives the propagation of the computational errors induced by format $(\boldsymbol{m}_\zeta, \boldsymbol{\ell}_\zeta)$; then, using (23) compute the FxP formats $(\boldsymbol{m}_\zeta^\diamondsuit, \boldsymbol{\ell}_\zeta^\diamondsuit)$ of the actually implemented filter $\mathcal{H}_\zeta^\diamondsuit$.

The above algorithm takes into account the filter implementation errors. However, the MSB computation via (23) itself is implemented in finite-precision and can suffer from rounding errors, which influence the output result.

All operations in the MSB computation may induce errors, so the quantities we actually compute are only floating-point approximations $\widehat{\boldsymbol{m}_\zeta}$ and $\widehat{\boldsymbol{m}_\zeta^\diamondsuit}$. In Section 4 we propose an error-analysis of the approximations in (23) and (16). We prove that by controlling the accuracy of the WCPG evaluation, we can compute $\boldsymbol{m}_\zeta^\diamondsuit$ exactly in most cases, otherwise overestimate by one, while being sure that we never underestimate.

In most cases the MSB vectors $\widehat{\boldsymbol{m}}_\zeta$ (computed at Step 1) and $\widehat{\boldsymbol{m}}_\zeta^\diamondsuit$ (computed at Step 2) are the same. When they are not, it is because one of the following happened:

- the accumulated rounding errors due to the initially computed FxPF $(\widehat{\boldsymbol{m}}_\zeta, \widehat{\boldsymbol{\ell}}_\zeta)$ makes the magnitude output of the implemented filter require one more bit; or
- the floating-point approximation $\widehat{\boldsymbol{m}}_\zeta^\diamondsuit$ is off by one due to our use of the triangle inequality in (22).

Moreover, if the MSB position is increased, then the LSB position moves along and increases the error (since the wordlengths are fixed here). Consequently, the modified format must be re-checked to be valid. Obviously, there is a risk that the check fails and the MSB position is to be increased yet again. To avoid an infinite loop we propose to use a rather natural exit condition: when the LSB position of the actually implemented filter reaches the position of the initially determined MSB position, nothing but noise is computed by the implemented filter. An interpretation of the above situation is that the filter simply cannot be reliably implemented with the given wordlengths. This information is quite helpful for the filter designers and, to the best of our knowledge, is not provided in state-of-the-art tools like Matlab.

We formalize the complete iterative approach in Algorithm 1.

4 Error analysis of the MSB computation formula

In this Section we state the requirements on the accuracy of the WCPG such that the computed MSB positions are either computed

Algorithm 1: Reliable determination of the MSBs

Input: system $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$;
input interval bound $\bar{\boldsymbol{u}}$;
wordlength constraints $\boldsymbol{w}_x, \boldsymbol{w}_y$
Output: Formats $(\boldsymbol{m}_x, \boldsymbol{m}_y)$ or an error
$\boldsymbol{w}_\zeta \longleftarrow$ concatenate $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$
$\mathcal{H}_\zeta \longleftarrow$ incorporate states into the filter via (13)
$\mathcal{H}_\Delta \longleftarrow$ model of the error filter via (20)
Step 1: **for** $i = 1, \ldots, n+p$ **do**
  $[\boldsymbol{m}_{\zeta_i}] \longleftarrow$ interval evaluation of MSB via (16)
  $\boldsymbol{m}_{\max_i} \longleftarrow \overline{\boldsymbol{m}}_{\zeta_i} + \boldsymbol{w}_{\zeta_i} + 1$
end
do
Step 2: **for** $i = 1, \ldots, n+p$ **do**
  $\overline{\boldsymbol{\varepsilon}}_{\zeta_i} \longleftarrow 2^{\overline{\boldsymbol{m}}_{\zeta_i} - \boldsymbol{w}_{\zeta_i} + 1}$
  $[\boldsymbol{m}^{\diamondsuit}_{\zeta_i}] \longleftarrow$interval evaluation of MSB via (23)
end
Check: **if** $[\boldsymbol{m}^{\diamondsuit}_{\zeta_i}] == [\boldsymbol{m}_{\zeta_i}]$ *for* $i = 1, \ldots, n+p$ **then**
  return $\overline{\boldsymbol{m}_\zeta}$
end
else
  $[\boldsymbol{m}_{\zeta_i}] \longleftarrow [\boldsymbol{m}_{\zeta_i}] + 1$ for $i = 1, \ldots, n+p$
end
while $\overline{\boldsymbol{m}}^{\diamondsuit}_\zeta < \boldsymbol{m}_{\max}$;
return *"Impossible to implement"*

exactly or overestimated by one. In the rare cases of overestimation by one, we face an instance of a Table Maker's Dilemma [10]. We shall propose an approach to overcome this issue.

4.1 Controlling the accuracy of the Worst-Case Peak Gain

Let us consider the case of $\widehat{\boldsymbol{m}_{\zeta_i}}^{\diamondsuit}$. For readability, let

$$\mathfrak{m} := \log_2\left(\left(\langle\langle\mathcal{H}_\zeta\rangle\rangle\bar{\boldsymbol{u}}\right)_i + (\langle\langle\mathcal{H}_\Delta\rangle\rangle\bar{\boldsymbol{\varepsilon}})_i\right) - \log_2\left(1 - 2^{1-\boldsymbol{w}_{\zeta_i}}\right). \quad (24)$$

The approach described below can be applied to the computation of $\widehat{\boldsymbol{m}}_{\zeta_i}$ with the terms concerning filter $\mathcal{H}_\Delta$ set to zero.

Handling floating-point analysis of multiplications and additions in (23) is trivial using a Higham's [11] approach. The difficulty comes from the approximations to WCPG matrices, which cannot be computed exactly. Both approximations $\widehat{\langle\langle\mathcal{H}_\zeta\rangle\rangle}$ and $\widehat{\langle\langle\mathcal{H}_\Delta\rangle\rangle}$, even if computed with arbitrary precision, bear some errors ε_{WCPG_ζ} and $\varepsilon_{WCPG_\Delta}$ that satisfy

$$0 \leq \widehat{\langle\langle\mathcal{H}_\Delta\rangle\rangle} - \langle\langle\mathcal{H}_\Delta\rangle\rangle \leq \varepsilon_{WCPG_\zeta} \quad (25)$$

$$0 \leq \widehat{\langle\langle\mathcal{H}_\zeta\rangle\rangle} - \langle\langle\mathcal{H}_\zeta\rangle\rangle \leq \varepsilon_{WCPG_\Delta} \quad (26)$$

Introducing the errors on the WCPG computations into the formula (23) we obtain that what we actually compute is

$$\widehat{\boldsymbol{m}_{\zeta_i}}^{\diamondsuit} \leq \left\lceil \mathfrak{m} + \underbrace{\log_2\left(1 + \frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j + \varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{\left(\langle\langle\mathcal{H}_\zeta\rangle\rangle\bar{\boldsymbol{u}}\right)_i + (\langle\langle\mathcal{H}_\Delta\rangle\rangle\bar{\boldsymbol{\varepsilon}})_i}\right)}_{\delta} \right\rceil. \quad (27)$$

The error term δ in (27) cannot be zero (apart from trivial case with zero $\bar{\boldsymbol{u}}$). However, assuming that we can control the accuracy of the WCPG matrices, we can deduce conditions for the approximation $\widehat{\boldsymbol{m}_{\zeta_i}}^{\diamondsuit}$ to be off by at most one.

Lemma 1. *[4] If the WCPG matrices $\langle\langle\mathcal{H}_\zeta\rangle\rangle$ and $\langle\langle\mathcal{H}_\Delta\rangle\rangle$ are computed such that* (25) *and* (26) *hold with*

$$\varepsilon_{WCPG_\Delta} < \frac{1}{2}\frac{\left(\overline{\langle\langle\mathcal{H}_\Delta\rangle\rangle}\cdot\bar{\boldsymbol{\varepsilon}}\right)_i}{\sum_{j=1}^{p+n}\bar{\boldsymbol{\varepsilon}}_j}, \quad \forall i \quad (28)$$

$$\varepsilon_{WCPG_\zeta} < \frac{1}{2}\frac{\left(\overline{\langle\langle\mathcal{H}_\zeta\rangle\rangle}\cdot\bar{\boldsymbol{u}}\right)_i}{\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j}, \quad \forall i \quad (29)$$

where $\overline{\langle\langle\mathcal{H}\rangle\rangle} := |\boldsymbol{D}| + |\boldsymbol{CB}| + |\boldsymbol{CAB}|$ is a very low-accuracy lower bound on $\langle\langle\mathcal{H}\rangle\rangle$, then

$$0 \leq \widehat{\boldsymbol{m}_{\zeta_i}}^{\diamondsuit} - \boldsymbol{m}_{\zeta_i}^{\diamondsuit} \leq 1, \quad \forall i. \quad (30)$$

We propose to perform arithmetic operations in (23) and (16) in multiple precision interval arithmetic in order to account for floating-point rounding errors. In most cases, the interval evaluation $[\widehat{\mathfrak{m}}]$ of (24) is not going to contain an integer, thus the final ceil operation will yield a point-interval. However, when it does, we have to choose between two possible MSB positions that are off by one. In the following section we discuss on how to refine the computations in order to overcome this potential overestimation.

4.2 Off-by-One problem and Table Maker's Dilemma

Let $[\widehat{\mathfrak{m}}]$ be an interval estimation of (24), where the WCPG matrices were computed with the error bounds deduced in Lemma 1. The integer MSB positions are computed as $[\boldsymbol{m}_\zeta] = \left\lceil[\widehat{\mathfrak{m}}]\right\rceil$. However, if the exact value $\mathfrak{m}$ is very close to the integer $\lceil\mathfrak{m}\rceil$, rounding the interval's bounds up to the nearest integer may yield and interval that will contain both $\lceil\mathfrak{m}\rceil$ and $\lceil\mathfrak{m}\rceil + 1$ and one must choose between two possible MSB positions. Then, we need to determine the smallest accuracy of the WCPG such that we do not overestimate the MSB position, i.e. the upper bound of interval $\left\lceil[\widehat{\mathfrak{m}}]\right\rceil$ is the same as $\lceil\mathfrak{m}\rceil$. This problem is an instance of the Table Maker's Dilemma (TMD) [10], which usually occurs during the implementation of transcendental functions.

One of the strategies of solving the TMD is performing Ziv's iteration [30]. In this approach we reduce the width of the interval $[\widehat{\mathfrak{m}}]$ by iteratively increasing the accuracy of the WCPG computation. However, even after numerous iterations the interval may still contain the integer $\lceil\mathfrak{m}\rceil$. This may be due to the following:

- the interval is still too large due to the rounding errors;
- the propagation of the rounding errors indeed yields the larger MSB position.

Thus, we cannot simply continue increasing the precision of the computations. We propose the following strategy:

(i) increase the accuracy of the WCPG several times;
(ii) if the interval $[\widehat{\mathfrak{m}}]$ still contains the integer z, try to find whether there exist a state and input vector that yield an overflow if the eventual MSB position is set to z. Roughly said, we try to use the smaller format and prove that an overflow is not possible nevertheless.

To prove that an overflow is not possible, we solve an instance of the following Integer Linear Programming [31] problem.

4.2.1 *Integer Linear Programming as a way to solve the TMD*

Let the input signal $\boldsymbol{u}$ be represented in some FxP Format. Suppose that we determine the FxP Formats for the state and output variables and, in case of the off-by-one problem, we choose the smaller MSB positions. Let $\underline{\boldsymbol{x}}$, $\underline{\boldsymbol{y}}$, $\underline{\boldsymbol{u}}$ be the minimal and $\overline{\boldsymbol{x}}$, $\overline{\boldsymbol{y}}$, $\overline{\boldsymbol{u}}$ the maximum authorized values for the state, output and input vectors respectively.

Then, our goal is to find $\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix}$ that are in the deduced FxP formats but for which

$$\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix} = \begin{pmatrix}\overline{\boldsymbol{x}}\\ \overline{\boldsymbol{y}}\end{pmatrix} + \begin{pmatrix}\boldsymbol{\delta}_x\\ \boldsymbol{\delta}_y\end{pmatrix} \tag{31}$$

with $\begin{pmatrix}\boldsymbol{\delta}_x\\ \boldsymbol{\delta}_y\end{pmatrix} \geq 0$. In other words, we search for $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}_x, \boldsymbol{\delta}_y$ such that

$$\begin{aligned}
\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix} &\leq \begin{pmatrix}\overline{\boldsymbol{x}}\\ \overline{\boldsymbol{y}}\end{pmatrix} + \begin{pmatrix}\boldsymbol{\delta}_x\\ \boldsymbol{\delta}_y\end{pmatrix}\\
\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix} &\geq \begin{pmatrix}\underline{\boldsymbol{x}}\\ \underline{\boldsymbol{y}}\end{pmatrix} + \begin{pmatrix}\boldsymbol{\delta}_x\\ \boldsymbol{\delta}_y\end{pmatrix}\\
\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix} &\leq \begin{pmatrix}\overline{\boldsymbol{x}}\\ \overline{\boldsymbol{u}}\end{pmatrix}\\
\begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{pmatrix} &\geq \begin{pmatrix}\underline{\boldsymbol{x}}\\ \underline{\boldsymbol{u}}\end{pmatrix}
\end{aligned} \tag{32}$$

Denote $\boldsymbol{x} := \underline{\boldsymbol{x}} + \boldsymbol{x}'$ and $\boldsymbol{u} := \underline{\boldsymbol{u}} + \boldsymbol{u}'$. To formalize the optimization problem, we need to bring the above inequalities to the canonical form, i.e. bring all inequalities to the direction "$\leq$".

Then, the optimization problem is the following:

$$\text{maximize } \boldsymbol{t}^\top \boldsymbol{\xi} \tag{33}$$

subject to the following constraints:

$$\boldsymbol{\mathfrak{F}}\boldsymbol{\xi} \leq \boldsymbol{r} \tag{34}$$

where

$$\boldsymbol{\xi} = \begin{pmatrix}\boldsymbol{x}'\\ \boldsymbol{u}'\\ \boldsymbol{\delta}_x\\ \boldsymbol{\delta}_y\end{pmatrix} \geq 0, \quad \boldsymbol{t} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{1}\\ \boldsymbol{1}\end{pmatrix} \tag{35}$$

$$\boldsymbol{\mathfrak{F}} = \begin{pmatrix}\boldsymbol{A} & \boldsymbol{B} & -\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{0} & -\boldsymbol{I}\\ -\boldsymbol{A} & -\boldsymbol{B} & \boldsymbol{I} & \boldsymbol{0}\\ -\boldsymbol{C} & -\boldsymbol{D} & \boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix}\left(\begin{pmatrix}\overline{\boldsymbol{x}}\\ \overline{\boldsymbol{y}}\end{pmatrix} - \begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}\begin{pmatrix}\underline{\boldsymbol{x}}\\ \underline{\boldsymbol{y}}\end{pmatrix}\right)\\ \left(\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}\begin{pmatrix}\underline{\boldsymbol{x}}\\ \underline{\boldsymbol{y}}\end{pmatrix} - \begin{pmatrix}\overline{\boldsymbol{x}}\\ \overline{\boldsymbol{y}}\end{pmatrix}\right)\\ \begin{pmatrix}\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}\\ \overline{\boldsymbol{u}} - \underline{\boldsymbol{u}}\end{pmatrix}\end{pmatrix}. \tag{36}$$

In our case we are actually interested in *existence* of a state and an output in the feasible set constrained by (34), hence the cost function essentially does not matter. We chose the cost function that, among feasible results, if they exist, selects the one with an average overflow. Another possible cost function, for example, is maximization of individual δ_{x_i} and δ_{y_i}. Then, one could solve $n + p$ instances of such optimization problem in order to obtain, if they exist, maximal errors that are possible.

Since filter implementation is performed in FxP arithmetic, $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ are actually integer matrices scaled by some factor. Thus, the above optimization problem becomes an instance of an Integer Linear Programming (ILP) problem.

To ensure that the exact solution is found and not just an approximation, we suggest using a solver over the rational numbers, such as the SCIP Optimization Suite[3] [32].

If there does not exist any solution to the above problem, then the overestimation of the MSB position was due to the application of triangular inequality in (22) (see Remark 2) and it is safe to take the smaller MSB positions, i.e. $\lceil \mathfrak{m} \rceil$ instead of $\lceil \mathfrak{m} \rceil + 1$.

Neither of our experiments showed that the ILP instance had feasible solutions. Thus, we were actually able to solve the TMD in our cases and return the smaller formats.

3. http://scip.zib.de/

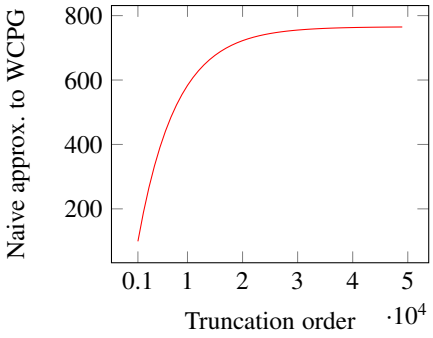


Fig. 3. The approximations of the WCPG with the increase of truncation order for a certain SISO filter.

5 Accurate evaluation of the Worst-Case Peak Gain

As we have seen in Section 3, the WCPG measure is required not only for bounding the dynamics of the variables in a digital filter, but also for bounding the impact of the rounding errors that occur in FxP implementation. In this section we present an algorithm for the floating-point evaluation of the WCPG matrix with an a priori given error bound.

5.1 Problem statement

Given an LTI filter as in (1) and a small $\varepsilon > 0$, we seek to evaluate a FP approximation $\boldsymbol{S}$ on the WCPG matrix $\langle\langle \mathcal{H} \rangle\rangle$ such that element-by-element

$$|\langle\langle \mathcal{H} \rangle\rangle - \boldsymbol{S}| \leq \varepsilon. \tag{37}$$

5.2 Naive approach and related work

Of course, we need to first truncate the sum in (4) to some finite number of terms N (further called truncation order). In other approaches [33], some "sufficiently large" truncation order is often chosen, e.g. 500 or 1000 terms. The following example demonstrates that this may be very dangerous.

Example 1. *Consider a certain stable 5^{th} order random SISO filter[4]. A naive computation of the WCPG in double precision with 1000 terms in the sum* (4) *yields* $\langle\langle \mathcal{H} \rangle\rangle_{naive} = 105.66$. *Suppose all the inputs are in the interval* $[-1, 1]$. *Then, according to the WCPG theorem, outputs must be in the interval* $[-105.66, 105.66]$.
Now, consider the input signal from Remark 1, i.e. the one that yields the worst-case output. With this input signal, the output of a double-precision FP filter reaches the value 192.2 *in just* 2000 *iterations. Obviously, the WCPG was underestimated.*

In [2] Balakrishnan and Boyd propose lower and upper bounds on the truncation order. Their iterative algorithm is proven to work in exact arithmetic, however its implementation in FP arithmetic does not. First, it is based on matrix exponentiation, which would require a non-trivial error analysis. Second, on each iteration (the quantity of which may reach a high order) solving Lyapunov equations [34] is required for which there exists no ready-to-use solution with rigorous error bounds on the result. Therefore, *numerically computing* a guaranteed lower bound on the truncation order N seems to be difficult with this approach as it is.

A competing approach, similar to the one in [28], would be not to start with truncation order determination but to immediately

4. Its double-precision coefficients are given in appendix.

Algorithm 2: Floating-point evaluation of the WCPG

Input: $\boldsymbol{A} \in \mathbb{F}^{n\times n}, \boldsymbol{B} \in \mathbb{F}^{n\times q}, \boldsymbol{C} \in \mathbb{F}^{p\times n}, \boldsymbol{D} \in \mathbb{F}^{p\times q}, \varepsilon > 0$

Output: $\boldsymbol{S}_N \in \mathbb{F}^{p\times q}$ such that $|\langle\langle \mathcal{H} \rangle\rangle - \boldsymbol{S}| \leq \varepsilon$

Step 1: Compute N

Step 2: Compute $\boldsymbol{V}$ from an eigendecomposition of $\boldsymbol{A}$

$\boldsymbol{T} \leftarrow \mathrm{inv}(\boldsymbol{V}) \otimes \boldsymbol{A} \otimes \boldsymbol{V}$

if $\|\boldsymbol{T}\|_2 > 1$ **then return** $\perp$

Step 3: $\boldsymbol{B}' \leftarrow \mathrm{inv}(\boldsymbol{V}) \otimes \boldsymbol{B}$

$\boldsymbol{C}' \leftarrow \boldsymbol{C} \otimes \boldsymbol{V}$

$\boldsymbol{S}_{-1} \leftarrow |\boldsymbol{D}|, \quad \boldsymbol{P}_{-1} \leftarrow \boldsymbol{I}_n$

for k **from** 0 **to** N **do**

Step 4: $\quad \boldsymbol{P}_k \leftarrow \boldsymbol{T} \otimes \boldsymbol{P}_{k-1}$

Step 5: $\quad \boldsymbol{L}_k \leftarrow \boldsymbol{C}' \otimes \boldsymbol{P}_k \otimes \boldsymbol{B}'$

Step 6: $\quad \boldsymbol{S}_k \leftarrow \boldsymbol{S}_{k-1} \oplus \mathrm{abs}(\boldsymbol{L}_k)$

end

return $\boldsymbol{S}_N$

go for summation and to stop when adding more terms does not improve accuracy. For example, if we increase the truncation order in Example 1, we obtain the dynamic of the WCPG approximations shown in Figure 3.

However, naive computation of the terms in (4) with FP arithmetic in some precision, set at the start of the algorithm, may yield significant rounding errors and would not allow the final approximation error to be bounded in an a priori way by an arbitrary ε.

Therefore, in the following we propose a new approach on the evaluation of the WCPG in multiple precision. Our goal is to not only perform rigorous error analysis of approximations but also to deduce the required accuracy for each computation in the evaluation of the WCPG. By adapting the precision of intermediate computations we achieve an a priori bound on the overall approximation error.

5.3 Algorithm for the Worst-Case Peak Gain evaluation

In this Section we give an overview of the proposed algorithm and detail our analysis in Sections 5.4, 5.5 and 5.6.

We propose a new direct formula for the bound on the truncation order N. Then, instead of directly computing the infinite sum $\left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|$ for any $k < N$, we will use an approximate eigenvalue decomposition of $\boldsymbol{A}$ (i.e. $\boldsymbol{A} \approx \boldsymbol{V}\boldsymbol{T}\boldsymbol{V}^{-1}$) and compute the floating-point sum $\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|$ for $0 \leq k \leq N$. We assume $\boldsymbol{A}$ to be diagonalizable with simple eigenvalues.

Our approach to compute the approximation $\boldsymbol{S}_N$ of $\langle\langle \mathcal{H} \rangle\rangle$ is summarized in algorithm 2 where all matrix operations ($\otimes$, $\oplus$, inv, abs, etc.) are floating-point multiple precision operations done at various precisions to be determined such that the overall error is at most ε.

The overall error analysis is decomposed into 6 steps, where each one expresses the impact of a particular approximation (or truncation), and provides the accuracy requirements for the associated operations.

Step 1: Let $\langle\langle \mathcal{H} \rangle\rangle_N$ be the truncated sum

$$\langle\langle \mathcal{H} \rangle\rangle_N := |\boldsymbol{D}| + \sum_{k=0}^{N} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|. \tag{38}$$

We compute a truncation order N of the infinite sum $\langle\langle \mathcal{H} \rangle\rangle$ such that the truncation error is less than ε_1:

$$|\langle\langle \mathcal{H} \rangle\rangle - \langle\langle \mathcal{H} \rangle\rangle_N| \leq \varepsilon_1. \tag{39}$$

Step 2: Error analysis for computing the powers $\boldsymbol{A}^k$ of a full matrix $\boldsymbol{A}$, when the k reaches several hundreds, is a significant problem, especially when the eigenvalues of $\boldsymbol{A}$ are close to the unit circle. However, if $\boldsymbol{A}$ can be represented as $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$ with $\boldsymbol{E} \in \mathbb{C}^{n\times n}$ strictly diagonal and $\boldsymbol{X} \in \mathbb{C}^{n\times n}$, then powering of $\boldsymbol{A}$ reduces to powering an approximation to $\boldsymbol{E}$.

Suppose we have a matrix $\boldsymbol{V}$ approximating $\boldsymbol{X}$. We require this approximation to be just quite accurate so that we are able to discern the different associated eigenvalues and be sure their absolute values are less than 1.

We may then consider the matrix $\boldsymbol{V}$ to be exact and compute an approximation $\boldsymbol{T}$ to $\boldsymbol{V}^{-1}\,\boldsymbol{A}\,\boldsymbol{V}$ with sufficient accuracy such that the error of computing $\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}$ instead of matrix $\boldsymbol{A}^k$ is less than ε_2:

$$\left|\langle\langle \mathcal{H} \rangle\rangle_N - \sum_{k=0}^{N} \left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|\right| \leq \varepsilon_2. \tag{40}$$

Step 3: We compute approximations $\boldsymbol{B}'$ and $\boldsymbol{C}'$ to $\boldsymbol{V}^{-1}\boldsymbol{B}$ and $\boldsymbol{C}\boldsymbol{V}$, respectively. We require that the propagated error committed in using $\boldsymbol{B}'$ instead of $\boldsymbol{V}^{-1}\boldsymbol{B}$ and $\boldsymbol{C}'$ instead of $\boldsymbol{C}\boldsymbol{V}$ be less than ε_3:

$$\left|\sum_{k=0}^{N} \left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right| - \sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}'\right|\right| \leq \varepsilon_3. \tag{41}$$

Step 4: We compute in $\boldsymbol{P}_k$ the powers $\boldsymbol{T}^k$ of $\boldsymbol{T}$ with a certain accuracy. We require that the propagated error be less than ε_4:

$$\left|\sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}'\right| - \sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'\right|\right| \leq \varepsilon_4. \tag{42}$$

Step 5: We compute in $\boldsymbol{L}_k$ each summand $\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'$ with an error small enough such that the overall approximation error induced by this step is less than ε_5:

$$\left|\sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'\right| - \sum_{k=0}^{N} |\boldsymbol{L}_k|\right| \leq \varepsilon_5. \tag{43}$$

Step 6: Finally, we sum $\boldsymbol{L}_k$ in $\boldsymbol{S}_N$ with enough precision so that the absolute error bound for summation is bounded by ε_6:

$$\left|\sum_{k=0}^{N} |\boldsymbol{L}_k| - \boldsymbol{S}_N\right| \leq \varepsilon_6. \tag{44}$$

By ensuring that each step verifies its bound ε_i, and taking $\varepsilon_i = \frac{1}{6}\varepsilon$, we get $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \varepsilon_5 + \varepsilon_6 = \varepsilon$, hence (37) will be satisfied if inequalities (39) through (44) are.

Our approach hence determines first a truncation order N and then performs summation up to that truncation order, whilst adjusting precision in the different summation steps.

5.4 Truncation order

In this Section we propose a direct formula for the lower bound on N along with a reliable evaluation algorithm.

Step 1: The goal is to determine a lower bound on the truncation order N of the infinite sum (4) such that its tail is smaller than the given ε_1. Obviously, $\langle\langle \mathcal{H} \rangle\rangle_N$ is a lower bound on $\langle\langle \mathcal{H} \rangle\rangle$ and increases monotonically to $\langle\langle \mathcal{H} \rangle\rangle$ with increasing N. Hence the truncation error is

$$|\langle\langle \mathcal{H} \rangle\rangle - \langle\langle \mathcal{H} \rangle\rangle_N| = \sum_{k>N} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|. \tag{45}$$

Many simple bounds on (45) are possible. For instance, if the eigendecomposition of $\boldsymbol{A}$ is computed

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1} \tag{46}$$

where $\boldsymbol{X} \in \mathbb{C}^{n\times n}$ is the right hand eigenvector matrix, and $\boldsymbol{E} \in \mathbb{C}^{n\times n}$ is a diagonal matrix holding the eigenvalues $\boldsymbol{\lambda}_l$, the terms $\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}$ can be written

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{\Phi}\boldsymbol{E}^k\boldsymbol{\Psi} = \sum_{l=1}^{n} \boldsymbol{R}_l\boldsymbol{\lambda}_l^k \tag{47}$$

where $\boldsymbol{\Phi} \in \mathbb{C}^{p\times n}$, $\boldsymbol{\Psi} \in \mathbb{C}^{n\times q}$ and $\boldsymbol{R}_l \in \mathbb{C}^{p\times q}$ are defined by

$$\boldsymbol{\Phi} := \boldsymbol{C}\boldsymbol{X}, \qquad \boldsymbol{\Psi} := \boldsymbol{X}^{-1}\boldsymbol{B}, \qquad (\boldsymbol{R}_l)_{ij} := \boldsymbol{\Phi}_{il}\boldsymbol{\Psi}_{lj}. \tag{48}$$

In this setting, we obtain

$$|\langle\langle\mathcal{H}\rangle\rangle - \langle\langle\mathcal{H}\rangle\rangle_N| \leq= \sum_{k>N}\sum_{l=1}^{n} \left|\boldsymbol{R}_l\boldsymbol{\lambda}_l^k\right|. \tag{49}$$

As only stable filters are considered, it is guaranteed that all eigenvalues $\boldsymbol{\lambda}_l$ of matrix $\boldsymbol{A}$ lie in the unit circle. We may therefore notice that the outer sum is in geometric progression with a common ratio $|\boldsymbol{\lambda}_l| < 1$. So the following bound is possible:

$$\begin{aligned} |\langle\langle\mathcal{H}\rangle\rangle - \langle\langle\mathcal{H}\rangle\rangle_N| &\leq \sum_{k=N+1}^{\infty}\sum_{l=1}^{n} |\boldsymbol{R}_l|\left|\boldsymbol{\lambda}_l^k\right| && (50)\\ &\leq \sum_{l=1}^{n} |\boldsymbol{R}_l| \frac{\left|\boldsymbol{\lambda}_l^{N+1}\right|}{1-|\boldsymbol{\lambda}_l|} \\ &= \rho(\boldsymbol{A})^{N+1} \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1-|\boldsymbol{\lambda}_l|}\left(\frac{|\boldsymbol{\lambda}_l|}{\rho(\boldsymbol{A})}\right)^{N+1}. && (51)\end{aligned}$$

Since $\frac{|\boldsymbol{\lambda}_l|}{\rho(\boldsymbol{A})} \leq 1$ holds for all terms, we may leave out the powers. Let be

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1-|\boldsymbol{\lambda}_l|}\frac{|\boldsymbol{\lambda}_l|}{\rho(\boldsymbol{A})} \in \mathbb{R}^{p\times q}. \tag{52}$$

The tail of the infinite sum is hence bounded by

$$|\langle\langle\mathcal{H}\rangle\rangle - \langle\langle\mathcal{H}\rangle\rangle_N| \leq \rho(\boldsymbol{A})^{N+1}\boldsymbol{M}. \tag{53}$$

In order to get (53) bounded by ε_1, it is required that element-by-element

$$\rho(\boldsymbol{A})^{N+1}\boldsymbol{M} \leq \varepsilon_1.$$

Solving this inequality for N leads us to the following bound:

$$N \geq \left\lceil \frac{\log\frac{\varepsilon_1}{m}}{\log\rho(\boldsymbol{A})} \right\rceil \tag{54}$$

where m is defined as $m := \min_{i,j} \boldsymbol{M}_{i,j}$.

However we cannot compute exact values for all quantities occurring in (54) when using finite-precision arithmetic. We only have approximations for them. Thus, in order to reliably determine a lower bound on N, we must compute lower bounds on m and $\rho(\boldsymbol{A})$, from which we can deduce an upper bound on $\log\frac{\varepsilon_1}{m}$ and a lower bound on $\log\rho(\boldsymbol{A})$ to eventually obtain a lower bound on N.

Due to the implementation of (46) and (48) with finite-precision arithmetic, only approximations on $\boldsymbol{\lambda}, \boldsymbol{X}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{R}_l$ can be obtained. To provide guaranteed inclusions on the results of these computations, we combine LAPACK floating-point arithmetic with Interval Arithmetic enhanced with Rump's Theory of Verified Inclusions [35], [36] which provide guaranteed inclusions on the solutions of various linear algebra problems. The verification process is performed by means of checking an interval fixed point and yields to a trusted interval for the solution. Then, the intervals for (48), (52) and (54) are computed with Interval Arithmetic. Our complete algorithm to determine a reliable lower bound on N can be found in [3] (Algorithm 3).

5.5 Multiple precision eigendecomposition

As seen, in each step of the summation, a matrix power, $\boldsymbol{A}^k$, must be computed. In [11] Higham devotes a chapter to error analysis of matrix powers but this theory is in most cases inapplicable for state matrices $\boldsymbol{A}$ of linear filters, as the requirement $\rho(|\boldsymbol{A}|) < 1$ does not necessarily hold here. Therefore, despite taking $\boldsymbol{A}$ to just a finite power k, the sequence of computed matrices may explode in norm since k may take an order of several hundreds or thousands. Thus, even extending the precision is not a solution, as an enormous number of bits would be required.

For stable systems, the state matrices are diagonalizable , i.e. there exists a matrix $\boldsymbol{X} \in \mathbb{C}^{n\times n}$ and diagonal $\boldsymbol{E} \in \mathbb{C}^{n\times n}$ such that $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$. Then $\boldsymbol{A}^k = \boldsymbol{X}\boldsymbol{E}^k\boldsymbol{X}^{-1}$. A good choice of $\boldsymbol{X}$ and $\boldsymbol{E}$ are the eigenvector and eigenvalue matrices obtained using eigendecomposition (46). However, with LAPACK we can compute only approximations of them and we cannot control their accuracy. Therefore, we propose the following method to *almost* diagonalize matrix $\boldsymbol{A}$. The method does not make any assumptions on $\boldsymbol{V}$ except for it being *some* approximation to $\boldsymbol{X}$. Therefore, for simplicity of further reasoning we treat $\boldsymbol{V}$ as an exact matrix.

Step 2: Using our multiprecision algorithms for matrix inverse and multiplication we may compute a $\boldsymbol{T} \in \mathbb{C}^{n\times n}$:

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2, \tag{55}$$

where $\boldsymbol{V} \in \mathbb{C}^{n\times n}$ is an approximation to $\boldsymbol{X}$, $\boldsymbol{\Delta}_2 \in \mathbb{C}^{n\times n}$ is a matrix representing the element-by-element errors due to the two matrix multiplications and the inversion of matrix $\boldsymbol{V}$.

Although the matrix $\boldsymbol{E}$ is strictly diagonal, $\boldsymbol{V}$ is not exactly the eigenvector matrix and consequently $\boldsymbol{T}$ is a dense matrix. However it has its prevailing elements on the main diagonal. Thus $\boldsymbol{T}$ is an approximation to $\boldsymbol{E}$.

We require for matrix $\boldsymbol{T}$ to satisfy $\|\boldsymbol{T}\|_2 \leq 1$. This condition is stronger than $\rho(\boldsymbol{A}) < 1$, and Section 5.5.1 provides a way to test it. In other words, this condition means that there exists some margin for computational errors between the spectral radius and 1.

Let $\boldsymbol{\Xi}_k := (\boldsymbol{T}+\boldsymbol{\Delta}_2)^k - \boldsymbol{T}^k$. Hence $\boldsymbol{\Xi}_k \in \mathbb{C}^{n\times n}$ represents the error matrix which captures the propagation of error $\boldsymbol{\Delta}_2$ when powering $\boldsymbol{T}$. Since

$$\boldsymbol{A}^k = \boldsymbol{V}(\boldsymbol{T}+\boldsymbol{\Delta}_2)^k\boldsymbol{V}^{-1}, \tag{56}$$

we have

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B} + \boldsymbol{C}\boldsymbol{V}\boldsymbol{\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}. \tag{57}$$

Thus the error of computing $\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}$ instead of $\boldsymbol{A}^k$ in (38) is bounded by

$$\left|\sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right| - \sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|\right| \leq \tag{58}$$

$$\sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} - \boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right| \leq \sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}\right|. \tag{59}$$

Here and further on each step of the algorithm we use inequalities with left side in form (59) rather than (58), i.e. we will instantly use the triangular inequality $\big||a|-|b|\big| \leq |a-b|\ \forall a, b$ applied element-by-element to matrices.

In order to determine the accuracy of the computations on this step such that (59) is bounded by ε_2, we need to perform detailed analysis of $\boldsymbol{\Xi}_k$, with spectral-norm. Using the definition of $\boldsymbol{\Xi}_k$ the following recurrence can be easily obtained:

$$\|\boldsymbol{\Xi}_k\|_2 \leq \|\boldsymbol{\Xi}_{k-1}\|_2 + \|\boldsymbol{\Delta}_2\|_2 \left(\|\boldsymbol{\Xi}_{k-1}\|_2 + 1\right) \tag{60}$$

If $\|\boldsymbol{\Xi}_{k-1}\|_2 \leq 1$, which must hold in our case since $\boldsymbol{\Xi}_k$ represent an error-matrix, then

$$\|\boldsymbol{\Xi}_k\|_2 \leq \|\boldsymbol{\Xi}_{k-1}\|_2 + 2\,\|\boldsymbol{\Delta}_2\|_2 \tag{61}$$

In the following, we bound the matrices with respect to their Frobenius norm, which is easy to compute and has the following useful properties: $\left|\boldsymbol{K}_{ij}\right| \leq \|\boldsymbol{K}\|_F$ and $\|\boldsymbol{K}\|_2 \leq \|\boldsymbol{K}\|_F \leq \sqrt{n}\,\|\boldsymbol{K}\|_2$ for $\boldsymbol{K} \in \mathbb{C}^{n\times n}$.

As $\|\boldsymbol{\Xi}_1\|_2 = \|\boldsymbol{\Delta}_2\|_2$ we can get the desired bound capturing the propagation of $\boldsymbol{\Delta}_2$ with Frobenius norm:

$$\|\boldsymbol{\Xi}_k\|_F \leq 2\sqrt{n}(k+1)\,\|\boldsymbol{\Delta}_2\|_F\,. \tag{62}$$

Substituting this bound to (59) and folding the sum, we obtain

$$\sum_{k=0}^{N} \left|\boldsymbol{CV}\boldsymbol{\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}\right| \leq \beta\,\|\boldsymbol{\Delta}_2\|_F\,\|\boldsymbol{CV}\|_F\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F, \tag{63}$$

with $\beta = \sqrt{n}(N+1)(N+2)$. Thus, we get a bound on the error of approximation to $\boldsymbol{A}$ by $\boldsymbol{VTV}^{-1}$. Since we require it to be less than ε_2 we obtain a condition for the error of the inversion and two matrix multiplications:

$$\|\boldsymbol{\Delta}_2\|_F \leq \frac{1}{\beta}\frac{\varepsilon_2}{\|\boldsymbol{CV}\|_F\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F}. \tag{64}$$

Using this bound we can deduce the desired accuracy of our multiprecision algorithms for complex matrix multiplication and inverse as a function of ε_2.

5.5.1 Checking whether $\|\boldsymbol{T}\|_2 \leq 1$

Since $\|\boldsymbol{T}\|_2^2 = \rho(\boldsymbol{T}^*\boldsymbol{T})$, we study the eigenvalues of $\boldsymbol{T}^*\boldsymbol{T}$. According to Gershgorin's circle theorem [37], each eigenvalue μ_i of $\boldsymbol{T}^*\boldsymbol{T}$ is in the disk centered in $(\boldsymbol{T}^*\boldsymbol{T})_{ii}$ with radius $\sum_{j\neq i}\left|(\boldsymbol{T}^*\boldsymbol{T})_{ij}\right|$.

Let us decompose $\boldsymbol{T}$ into $\boldsymbol{T} = \boldsymbol{F} + \boldsymbol{G}$, where $\boldsymbol{F}$ is diagonal and $\boldsymbol{G}$ contains all the other terms ($\boldsymbol{F}$ contains the approximate eigenvalues, $\boldsymbol{G}$ contains small terms and is zero on its diagonal). Let be $\boldsymbol{Y} := \boldsymbol{T}^*\boldsymbol{T} - \boldsymbol{F}^*\boldsymbol{F} = \boldsymbol{F}^*\boldsymbol{G} + \boldsymbol{G}^*\boldsymbol{F} + \boldsymbol{G}^*\boldsymbol{G}$. Then

$$\begin{aligned}\sum_{j\neq i}\left|(\boldsymbol{T}^*\boldsymbol{T})_{ij}\right| &= \sum_{j\neq i}\left|\boldsymbol{Y}_{ij}\right| \\ &\leq (n-1)\,\|\boldsymbol{Y}\|_F \\ &\leq (n-1)\left(2\,\|\boldsymbol{F}\|_F\,\|\boldsymbol{G}\|_F + \|\boldsymbol{G}\|_F^2\right) \\ &\leq (n-1)\left(2\sqrt{n} + \|\boldsymbol{G}\|_F\right)\|\boldsymbol{G}\|_F\,. \end{aligned} \tag{65}$$

Each eigenvalue of $\boldsymbol{T}^*\boldsymbol{T}$ is in the disk centered in $(\boldsymbol{F}^*\boldsymbol{F})_{ii}+(\boldsymbol{Y})_{ii}$ with radius γ, where γ is equal to $(n-1)\left(2\sqrt{n} + \|\boldsymbol{G}\|_F\right)\|\boldsymbol{G}\|_F$, computed in a rounding mode that makes the result become an upper bound (round-up).

As $\boldsymbol{G}$ is zero on its diagonal, the diagonal elements of $\boldsymbol{Y}$ are equal to the diagonal elements of $\boldsymbol{G}^*\boldsymbol{G}$. They can hence be bounded as follows:

$$|\boldsymbol{Y}_{ii}| = |(\boldsymbol{G}^*\boldsymbol{G})_{ii}| \leq \|\boldsymbol{G}\|_F^2\,. \tag{66}$$

Then, Gershgorin circles enclosing the eigenvalues of $\boldsymbol{F}^*\boldsymbol{F}$ can be increased, meaning that if $(\boldsymbol{F}^*\boldsymbol{F})_{ii}$ is such that

$$\forall i, \quad |(\boldsymbol{F}^*\boldsymbol{F})_{ii}| \leq 1 - \|\boldsymbol{G}\|_F^2 - \gamma, \tag{67}$$

it holds that $\rho(\boldsymbol{T}^*\boldsymbol{T}) \leq 1$ and $\|\boldsymbol{T}\|_2 \leq 1$.

This condition can be tested by using floating-point arithmetic with directed rounding modes (round-up for instance).

After computing $\boldsymbol{T}$ out of $\boldsymbol{V}$ and $\boldsymbol{A}$ according to (55), the condition on $\boldsymbol{T}$ should be tested in order to determine if $\|\boldsymbol{T}\|_2 \leq 1$. This test failing means that $\boldsymbol{V}$ is not a sufficiently accurate approximation to $\boldsymbol{X}$ or that the error $\boldsymbol{\Delta}_2$ committed when computing (55) is too large, i.e. the accuracy of our multiprecision algorithm for complex matrix multiplication and inverse should be increased. The test is required for rigor only. We do perform the test in the implementation of our WCPG method, and, on the examples we tested, never saw it give a negative answer. In case where the matrix $\boldsymbol{T}$ does not pass the check, our algorithm is designed to return an error message.

5.6 Summation

Now that the truncation order is determined and $\boldsymbol{A}$ is replaced by $\boldsymbol{VTV}^{-1}$, we must perform floating-point matrix operations with an a priori absolute error bound.

Step 3: we compute approximations to the matrices $\boldsymbol{CV}$ and $\boldsymbol{V}^{-1}\boldsymbol{B}$ with a certain precision and need to determine the required accuracy of these multiplications such that the overall error of this step is less than ε_3.

Let $\boldsymbol{C'} := \boldsymbol{CV} + \boldsymbol{\Delta}_{3_C}$ and $\boldsymbol{B'} := \boldsymbol{V}^{-1}\boldsymbol{B} + \boldsymbol{\Delta}_{3_B}$, where $\boldsymbol{\Delta}_{3_C} \in \mathbb{C}^{p\times n}$ and $\boldsymbol{\Delta}_{3_B} \in \mathbb{C}^{n\times q}$ are error matrices containing the errors of the two matrix multiplications and the inversion.

Using the Frobenius norm, we can bound the error in the approximation to $\boldsymbol{CV}$ and $\boldsymbol{V}^{-1}\boldsymbol{B}$ by $\boldsymbol{C'}$ and $\boldsymbol{B'}$ as follows:

$$\begin{aligned}&\sum_{k=0}^{N}\left|\boldsymbol{CVT}^k\boldsymbol{V}^{-1}\boldsymbol{B} - \boldsymbol{C'T}^k\boldsymbol{B'}\right| \leq \\ &\sum_{k=0}^{N}\left\|\boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{B'} + \boldsymbol{C'T}^k\boldsymbol{\Delta}_{3_B} + \boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B}\right\|_F.\end{aligned} \tag{68}$$

Since $\|\boldsymbol{T}\|_2 \leq 1$ holds we have

$$\begin{aligned}&\left\|\boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{B'} + \boldsymbol{C'T}^k\boldsymbol{\Delta}_{3_B} + \boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B}\right\|_F \leq \\ &\sqrt{n}\left(\left\|\boldsymbol{\Delta}_{3_C}\right\|_F\left(\left\|\boldsymbol{B'}\right\|_F + \left\|\boldsymbol{\Delta}_{3_B}\right\|_F\right) + \left\|\boldsymbol{C'}\right\|_F\left\|\boldsymbol{\Delta}_{3_B}\right\|_F\right).\end{aligned} \tag{69}$$

This bound represents the impact of our approximations for each $k = 0 \ldots N$. If (69) is bounded by $\frac{1}{N+1}\cdot\varepsilon_3$, then the overall error is less than ε_3. Hence, bounds on the two error-matrices are:

$$\left\|\boldsymbol{\Delta}_{3_C}\right\|_F \leq \frac{1}{3\sqrt{n}}\cdot\frac{1}{N+1}\frac{\varepsilon_3}{\|\boldsymbol{B'}\|_F} \tag{70}$$

$$\left\|\boldsymbol{\Delta}_{3_B}\right\|_F \leq \frac{1}{3\sqrt{n}}\cdot\frac{1}{N+1}\frac{\varepsilon_3}{\|\boldsymbol{C'}\|_F}. \tag{71}$$

Therefore, using bounds on $\left\|\boldsymbol{\Delta}_{3_C}\right\|_F$ and $\left\|\boldsymbol{\Delta}_{3_B}\right\|_F$, we can deduce the required accuracy of our multiprecision matrix multiplication and inversion according to ε_3.

Steps 4 to 6: we proceed in a similar manner, each time bounding the propagated error and expressing the requirements for the accuracy of matrix operations. We refer the reader to [3] for more details.

5.7 Matrix arithmetic with a priori accuracy

5.7.1 Basic brick methods

In order for our WCPG evaluation algorithm to work, we require the following three basic floating-point algorithms: *multiplyAndAdd*, *sumAbs* and *inv*, computing, respectively, the product followed

by a sum ($\boldsymbol{AB}+\boldsymbol{C}$), the accumulation of an absolute value in a sum ($\boldsymbol{A}+|\boldsymbol{B}|$) and the inverse of matrices ($\boldsymbol{A}^{-1}$). Each of these operators was required to satisfy an absolute error bound $|\boldsymbol{\Delta}| < \delta$ to be ensured by the absolute-error matrix $\boldsymbol{\Delta}$ with respect to scalar δ, given in argument to the algorithm.

Ensuring such an *absolute* error bound is not possible in general when fixed-precision floating-point arithmetic is used. Any such algorithm, when returning its result, must round into that fixed-precision floating-point format. Hence, when the output grows sufficiently large, the unit in the last place of that format and hence the final rounding error in fixed-precision floating-point arithmetic grows larger than a previously set absolute error bound.

We develop algorithms that will generically determine the output precision of the floating-point variables they return their results in, such that a user-given absolute error bound is guaranteed. In contrast to classical floating-point arithmetic, such as Higham's analyzes, there is no longer any clear, overall *compute precision*, though. Variables just bear the precision that had been determined for them by the previous computation step. This preliminary clarification being made, a general description of our three basic bricks *sumAbs*, *inv* and *multiplyAndAdd* is easy.

For $\mathit{sumAbs}(\boldsymbol{A},\boldsymbol{B},\delta) = \boldsymbol{A}+|\boldsymbol{B}|+\boldsymbol{\Delta}$, we can reason element by element. We need to approximate $\boldsymbol{A}_{ij} + \sqrt{\left(\Re\boldsymbol{B}_{ij}\right)^2 + \left(\Im\boldsymbol{B}_{ij}\right)^2}$ with absolute error no larger than δ, where $\Re z$ and $\Im z$ are the real and imaginary parts of the complex number z. This can be ensured by considering the floating-point exponents of each of $\boldsymbol{A}_{ij}$, $\Re\boldsymbol{B}_{ij}$ and $\Im\boldsymbol{B}_{ij}$ with respect to the floating-point exponent of δ.

For $\mathit{multiplyAndAdd}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\delta) = \boldsymbol{A}\cdot\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{\Delta}$, we can reason in terms of scalar products between $\boldsymbol{A}$ and $\boldsymbol{B}$. The scalar products boil down to summation of products which, in turn, can be done exactly, as we can determine the precision of the $\boldsymbol{A}_{ik}$ and $\boldsymbol{B}_{kj}$. As a matter of course the very same summation can capture the matrix elements $\boldsymbol{C}_{ij}$. Finally, multiple precision floating-point summation with an absolute error bound can be performed with a modified, software-simulated Kulisch accumulator [38], which does not need to be exact but bear just enough precision to satisfy the absolute accuracy bound δ.

Finally, once the *multiplyAndAdd* operator is available, it is possible to implement the matrix inversion algorithm *inv* using a Newton-Raphson-like iteration [39]:

$$\begin{aligned} \boldsymbol{U}_0 &\leftarrow \text{some seed inverse matrix for } \boldsymbol{V}^{-1} \\ \boldsymbol{R}_k &\leftarrow \boldsymbol{V}\boldsymbol{U}_k - \boldsymbol{I}_n \\ \boldsymbol{U}_{k+1} &\leftarrow \boldsymbol{U}_k - \boldsymbol{U}_k\boldsymbol{R} \end{aligned} \tag{72}$$

where the iterated matrices $\boldsymbol{U}_k$ converge to $\boldsymbol{V}^{-1}$ provided certain conditions are met. See [3] and the appendix for details.

5.7.2 Error analysis of MP basic bricks

One of the core algorithms in our approach is the sum of the elements of a real vector with an a priori given absolute error bound. The idea is first to provide a Higham-like error analysis of computations and to determine the bound on the rounding error; and then to use just enough additional precision in computations to account for that error. We give an example of the proof for a simple basic brick algorithm: summation of vector elements.

Let $\boldsymbol{v}$ be a vector of n MP positive floating-point numbers (no NaNs or Infinities) whose exponents are bounded by $\bar{e}$ and let $s = \sum_{i=1}^{n} \boldsymbol{v}_i$ be the sum of its elements.

Lemma 2. *Let $p = \bar{e} + 2\left\lceil \log_2 n \right\rceil + k + 1$. If the MP floating-point approximation $\hat{s}$ to s is computed with the precision at least p bits, then $|s-\hat{s}| \le 2^{-k}$.*

In addition, the only case when the precision of the final result $\hat{s}$ is larger than the precision p_{sum} of the original sum variable is when the exponent e of $\hat{s}$ satisfies $e > p_{sum} - k - 1$.

Proof. Since $|\boldsymbol{v}_i| < 2^{\bar{e}}$, the sum is bounded as $|s| \le n2^{\bar{e}} \le 2^{\bar{e}+\left\lceil \log_2 n \right\rceil}$.

When performing one addition in MP with precision p, an error of at most $2^{-p}|s|$ is induced. When performing n additions, we obtain [40]:

$$|s-\hat{s}| \le n2^{-p}|s| \le 2^{\left\lceil \log_2 n \right\rceil - p + \bar{e} + \left\lceil \log_2 n \right\rceil} \le 2^{-k-1}. \tag{73}$$

Thus, the accumulated sum $\hat{s}$ satisfies

$$\hat{s} = s + \delta, \quad |\delta| \le 2^{-k-1}. \tag{74}$$

Let p_{sum} be the precision of the original sum variable in the summation algorithm. It may happen that $p > p_{\text{sum}}$, i.e. we need to round $\hat{s}$ to the precision closest to p_{sum} while maintaining the original error bound 2^{-k}.

Let e be the exponent of the accumulated sum $\hat{s}$. Then, when rounding $\hat{s}$ to p_{sum} bits, we obtain the value $\hat{s}'$ such that

$$\left|\hat{s}-\hat{s}'\right| \le 2^{e-p_{\text{sum}}}. \tag{75}$$

Thus we actually obtain

$$\hat{s}' = \hat{s} + \delta' = s + \delta + \delta' \tag{76}$$

and

$$\left|s-\hat{s}'\right| \le |\delta| + \left|\delta'\right| \le 2^{k-1} + 2^{e-p_{\text{sum}}}. \tag{77}$$

If $e - p_{\text{sum}} \le -k-1$, then $|s-\hat{s}'| \le 2^{-k-1} + 2^{-k-1} = 2^{-k}$.

Otherwise, we round $\hat{s}$ to $e+k+1$ bits, inducing an error δ' bounded by 2^{-k-1} but still maintaining the overall error bound 2^{-k}. This is the only case where the final precision of the sum variable is not equal to the original precision. □

6 Numerical examples

The algorithms discussed above were implemented in C, using GNU MPFR version 3.1.12, GNU MPFI[5] version 1.5.2 and CLAPACK[6] version 3.2.1. A Python `ctypes` wrapper was also provided in order to integrate the libraries into the FiXiF tool. All the code is released as open-source.

We use FiXiF (and Python 2.7.10) as a front end providing the necessary infrastructure for LTI filter manipulation. The timings reported are those measured for the C code.

Experiments were done on a laptop computer with an Intel Core i5 processor running at 2.8 GHz and 16 GB of RAM.

Here we demonstrate our tool on the following three examples:

- $\mathcal{H}_1$ comes from Control Theory: the LTI system is extracted from an active controller of vehicle longitudinal oscillation [41].
- $\mathcal{H}_2$ is a highly sensitive lowpass filter. It has very short passband (between 0 and 1e-3 in normalized frequencies) and short transition band (stopband starts at 1.8e-3) with minimum attenuation of 40dB.
- $\mathcal{H}_3$ is a simple lowpass MIMO filter (with 3 outputs).

Table 1 gathers the evaluations of corresponding WCPG measures (first with a fixed $\delta_{\text{WCPG}} = 2^{-53}$) and numerical results

5. https://gforge.inria.fr/projects/mpfi/
6. http://www.netlib.org/clapack/

TABLE 1
Experimental results for two SISO and one MIMO filters ($q = 3$).

filter	n	$1-\rho(\boldsymbol{A})$	$\langle\langle\mathcal{H}\rangle\rangle$ (fixed $\delta_{\text{WCPG}} = 2^{-53}$)			w=10				w=16			
			time	N	$\langle\langle\mathcal{H}\rangle\rangle$	δ_{WCPG}	# steps	$\bar{\Delta}_\zeta$	time	δ_{WCPG}	# steps	$\bar{\Delta}_\zeta$	time
$\mathcal{H}_1$	10	1.39e-2	1.17 s	2147	0.8584939	1.81e-3	2	8.85e-03	3.06 s	6.10e-4	2	2.77e-4	0.76 s
$\mathcal{H}_2$	5	4.13e-4	5.95 s	90169	1.5118965	1.16e-4	9	-	20.32 s	2.33e-4	4	2.07e-1	14.94 s
$\mathcal{H}_3$	6	7.49e-2	0.06 s	395	4.9136028 7.2209045 2.0428813	2.57e-2	3	3.41e-1 5.31e-1 1.32e-1	0.05 s	1.03e-2	2	1.43e-3 3.70e-3 5.78e-3	0.03 s

for implementations with wordlength constraints (homogeneously) set to 10 and 16 bits. Here δ_{WCPG} is the smallest a priori error for WCPG evaluation, $\bar{\Delta}_\zeta$ is the bound on the output errors and "# steps" denotes number of steps in Algorithm 1 (2 steps + possibly additional iterations).

Our algorithm successfully captures when, for a given wordlength, implementation guaranteeing the absence of overflows is impossible. For the filter $\mathcal{H}_2$ and $w = 10$, it took 9 iterations for our algorithm to determine that implementation is impossible, due to condition as in Section 3.4.

We observe that the WCPG truncation order N varies significantly for different filters, when WCPG is evaluated accurately to double precision. However, the FxP format determination algorithm in most cases does not actually require low error bound on the approximations of the WCPG matrices (Table 1 gathers maximum target precisions). Still, Algorithm 1 is highly dependent on the WCPG evaluation time. In case of $\mathcal{H}_1$ and $w = 16$, it takes 0.76s for our algorithm to determine reliable formats after evaluating WCPG up to 6.10e-4 for a 10th order system. In the meantime, for $\mathcal{H}_1$ our algorithm takes relatively more time, around 15s, to evaluate the WCPG measures to roughly the same accuracy for a twice smaller, yet much more sensitive, system. In our experiments, the internal precision during the WCPG evaluation is set to at most a few hundred bits: maximum 174 bits, for the case $\delta_{\text{WCPG}} = 2^{-53}$.

Overall, our algorithm is quite fast and can be used in a design space exploration. A straightforward exploration is by iteration over increasing wordlengths in order to find an implementation with a suitable error bound. We discuss a better approach on exploiting our algorithms to directly determine minimum wordlengths for a required error bound in Section 7.2.

7 Other applications of WCPG: hardware generation and frequency checks

Our new algorithm for the reliable evaluation of the WCPG that guarantees an a priori error bound opens up numerous possibilities for the design and verification of new arithmetic hardware dedicated to signal processing.

7.1 Numerical verification of frequency specifications

A posteriori validation of the frequency behavior of implemented filters is an integral part of the design of reliable systems. In practice, filter coefficients are rounded to lower precision, which inherently influences the frequency-domain behavior of the filter [1, Chapter 6.7, pp.433]. Then, in order to rigorously prove that a filter with coefficients expressed with given precision satisfies certain frequency specifications for *any frequency* (and not just on a finite subset), the WCPG can be used to bound the error of approximation to the filter's frequency response, as done in [42].

7.2 Design of faithfully-rounded hardware implementations

Consider the following problem: given filter coefficients and FxP formats for input and output signals, generate *at minimal cost* a hardware implementation that guarantees that the output is always faithfully rounded. Thus, the precision of internal computations must be chosen in a way that guarantees an a priori bound on the output error while not wasting area. These precision choices must rely on worst-case rounding error analysis but not be uselessly pessimistic. WCPG plays a central role for such hardware design.

Indeed, if the errors due to arithmetic operations must be bounded by a certain value $2^{\boldsymbol{\ell}}$, then in (21) we get

$$\langle\langle\mathcal{H}_\Delta\rangle\rangle 2^{\boldsymbol{\ell}_\zeta} \leq 2^{\ell}. \tag{78}$$

This can be easily transposed into requirements on the precision of internal computations, namely

$$\boldsymbol{\ell}_\zeta \leq \log_2\langle\langle\mathcal{H}_\Delta\rangle\rangle - \ell \tag{79}$$

We use the above relation in our recent work [43] to deduce an accuracy constraint for a new Sum-of-Product-based hardware code generator for recursive filters. In this context, WCPG is key to deducing architectures guaranteeing faithful rounding at minimal hardware cost.

7.3 Input-aware rigorous FxP design

FxP filter design based on the WCPG does not make any assumptions on the spectrum of the input signal and provides worst-case bounds. However, often a digital filter's input signal is the output of an existing signal processing system, or describes particular physical process, dynamics of which can be expressed as frequency (spectrum) specifications. For example, an input signal that describes temperature usually lies in low frequencies (assuming high enough sampling rate), with higher frequencies dedicated to possible measurement noise.

Taking into account this information on the input spectrum can result in a lower WCPG measure and, hence, yield smaller hardware designs.

We will model the frequency specification by a function G of the normalized frequency ω bounding the Discrete-Time Fourier Transform $U(e^{i\omega})$ of the input signal:

$$|U(e^{i\omega})| \leq G(\omega), \quad \forall\omega \in [0,\pi]. \tag{80}$$

Our idea is to model the initial filter as a cascade of two filters: (1) a system $\mathcal{G}$ that produces an output with frequency response $G(\omega)$; (2) the initial filter. While the first filter is not going to be actually implemented, it permits to take into account the dynamics of the initial input signal when the WCPG theorem is applied upon the cascaded system.

We propose to proceed along following steps:
Step 1. Using some classical approach [44], design a digital filter $\mathcal{G}^*$ that corresponds to the specifications $G(\omega)$ relaxed by a small margin Δ (to account for approximation errors in the filter design);
Step 2. Use our WCPG-based verification algorithm [42] to ensure that $\mathcal{G}^*$ satisfies $G(\omega) + \Delta$;
Step 3. Cascade $\mathcal{G}^*$ with the initial filter and apply the WCPG theorem to deduce the ranges of variables of the initial filter;
Step 4. Apply our FxP algorithm (or, for FPGA implementations, our techniques from [43]) upon the cascaded filter. We slightly modify our approaches to account for the fact that $\mathcal{G}^*$ will not be part of the implemented filter, and thus no errors will propagate through it.

8 Conclusion

We have proposed an algorithm for the reliable determination of the FxP formats for all the variables involved in a recursive filter. We assume that the wordlength constraints and a bound on the input interval are given. We take computational errors as well as their propagation over time fully into account. We achieve this by decomposing the actually implemented filter into a sum of the exact filter and a special error-filter. By applying the WCPG theorem upon the error filter we get a bound on the worst-case error. We take this bound into account while computing the MSB positions for the variables.

We provided an error analysis of the MSB computation formula and showed that by adjusting the accuracy of the WCPG, the computed positions are either exact or overestimated by one. Our approach is fully reliable and we do not use any simulations anywhere in our algorithms. We identified the off-by-one problem as an instance of a Table Maker's Dilemma and proposed an Integer Linear Programming-based approach to deal with it. Even with the off-by-one issue, to our knowledge, our algorithm is the first existing approach that given wordlength constraints provides reliable MSB positions along with a rigorous bound on the computational errors. Moreover, it is straightforward to turn the problem the other way around and, given some output error bound, determine the least MSB positions that ensure this bound. We also support multiple wordlength paradigm, i.e. wordlengths are not necessarily the same for all variables.

The core algorithm that enables our approach is the evaluation of the WCPG measure to arbitrary precision. Our reliable algorithm relies on multiple precision eigendecomposition to perform matrix powering, some multi-precision basic bricks developed to satisfy a priori absolute error bounds and detailed step-by-step error analysis. We consider that our techniques for multiple precision approximation to eigenvalues are of interest independently of the context. We demonstrated that our multiple precision MPFR/MPFI-based implementation does not usually use precisions beyond a few hundred bits and is quite fast for our needs.

The execution time of our algorithm for FxP formats is dominated by the computation of the WCPG. In most cases, we do not require high accuracy for the WCPG. On the contrary, we often need the WCPG to be accurate to even less than double precision which speeds up the computations. Overall, the execution time of our algorithm permits us to use it repeatedly, for instance as part of optimization routines.

Accurate WCPG evaluation opens up numerous possibilities for the design of new hardware arithmetic and verification. In [42], [43] we present mature work on the application of the WCPG to the verification of digital filters against frequency specifications and on the automatic generation of optimal architectures for faithfully-rounded recursive filters. We also propose in this paper a four-step procedure that can be used for the input-aware FxP implementation.

Some efforts are still required on both the FxP error-analysis and FP error analysis sides. First, we left the question of quantization of filter coefficients out of scope. In the future we plan to extend the work in [45] to adapt our FxP format algorithm to consider both computational and quantization errors. Second, integrating a multiple precision eigenvalue decomposition such as the one available in `mpmath` [7] could accelerate our matrix powering techniques. Finally, we leave to future work further development of our input-aware FxP design techniques for the design of efficient arithmetic operators on reconfigurable hardware.

References

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. NJ, USA: Prentice Hall Press, 2009.
[2] V. Balakrishnan and S. Boyd, "On Computing the Worst-Case Peak Gain of Linear Systems," *Systems & Control Letters*, vol. 19, pp. 265–269, 1992.
[3] A. Volkova, T. Hilaire, and C. Lauter, "Reliable evaluation of the worst-case peak gain matrix in multiple precision," in *IEEE Symposium on Computer Arithmetic*, 2015, pp. 96–103.
[4] ——, "Determining fixed-point formats for a digital filter implementation using the worst-case peak gain measure," in *Asilomar Conference on Signals, Systems & Computers*, 2015, pp. 737–741.
[5] T. Hilaire, P. Chevrel, and J. F. Whidborne, "A Unifying Framework for Finite Wordlength Realizations," *IEEE Transactions on Circuits and Systems*, vol. 8, no. 54, pp. 1765–1774, 2007.
[6] A. Volkova, "Towards reliable implementation of digital filters," Ph.D. dissertation, Sorbonne Universités – University of Pierre and Marie Curie, 2017.
[7] W. Padgett and D. Anderson, *Fixed-Point Signal Processing*, ser. Synthesis lectures on signal processing. Morgan & Claypool, 2009.
[8] R. Oshana, *DSP Software Development Techniques for Embedded and Real-Time Systems*. Elsevier Science, 2006.
[9] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.
[10] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating-Point Arithmetic*. 2nd ed. Birkhäuser, 2018.
[11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms (2 ed.)*. SIAM, 2002.
[12] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A Multiple-precision Binary Floating-point Library with Correct Rounding," *ACM Transactions on Mathematical Software*, vol. 33, no. 2, 2007.
[13] N. Revol and F. Rouillier, "Motivations for an arbitrary precision interval arithmetic and the MPFI library," *Reliable Computing*, vol. 11, no. 4, pp. 275–290, 2005.
[14] D. Báez-López, D. Báez-Villegas, R. Alcántara, J. J. Romero, and T. Escalante, "Package for filter design based on MATLAB," *Comp. Applic. in Engineering Education*, vol. 9, no. 4, pp. 259–264, 2001.
[15] L. D. Coster, M. Adé, R. Lauwereins, and J. A. Peperstraete, "Code generation for compiled bit-true simulation of DSP applications," in *Proceedings of the 11th International Symposium on System Synthesis, ISSS '98, Hsinchu, Taiwan*, 1998, pp. 9–14.
[16] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge, UK: Cambridge University Press, 2008.
[17] G. Constantinides, P. Cheung, and L. Wayne, *Synthesis and Optimization of DSP Algorithms*. Kluwer, 2004.
[18] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Roundoff-noise shaping in filter design," in *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, May 2000, pp. 57–60 vol.4.
[19] A. Benedetti and P. Perona, "Bit-width optimization for configurable DSP's by multi-interval analysis," in *Asilomar Conference on Signals, Systems & Computers*, vol. 1, 2000, pp. 355–359.

7. http://mpmath.org

[20] A. B. Kinsman and N. Nicolici, "Bit-width allocation for hardware accelerators for scientific computing using SAT-modulo theory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 405–413, 2010.

[21] J. A. 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, 2007.

[22] S. Vakili, J. M. P. Langlois, and G. Bois, "Enhanced precision analysis for accuracy-aware bit-width optimization using affine arithmetic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1853–1865, 2013.

[23] D. U. Lee, A. A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[24] O. Sarbishei and K. Radecka, "On the fixed-point accuracy analysis and optimization of polynomial specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 831–844, 2013.

[25] C. F. Fang, R. A. Rutenbar, M. Püschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Design Automation Conference*, 2003, pp. 496–501.

[26] D. Boland and G. Constantinides, "Bounding variable values and round-off effects using Handelman representations," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1691 –1704, 2011.

[27] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *POPL*, 1977, pp. 238–252.

[28] D. Monniaux, "Applying the z-transform for the static analysis of floating-point numerical filters," *CoRR*, vol. abs/0706.0252, 2007.

[29] F. de Dinechin, M. Istoan, and A. Massouri, "Sum-of-product architectures computing just right," in *IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*, 2014, pp. 41–47.

[30] A. Ziv, "Fast evaluation of elementary mathematical functions with correctly rounded last bit," *ACM Transactions Math. Softw.*, vol. 17, no. 3, pp. 410–423, 1991.

[31] I. Zelinka, V. Snasel, and A. Abraham, *Handbook of Optimization: From Classical to Modern Approach*. Springer Berlin Heidelberg, 2012.

[32] W. Cook, T. Koch, D. E. Steffy, and K. Wolter, "A hybrid branch-and-bound approach for exact rational mixed-integer programming," Konrad-Zuse-Zentrum für Informationstechnik Berlin, Tech. Rep., 2012.

[33] O. Sarbishei, K. Radecka, and Z. Zilic, "Analytical optimization of bit-widths in fixed-point LTI systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 343–355, 2012.

[34] S. Hammarling, "Numerical solution of the discrete-time, convergent, non-negative definite Lyapunov equation," *Syst. Control Lett.*, vol. 17, no. 2, pp. 137–139, 1991.

[35] S. M. Rump, "Solution of Linear Systems with Verified Accuracy," *Applied numerical mathematics*, vol. 3, no. 3, pp. 233–241, 1987.

[36] ——, "Guaranteed inclusions for the complex generalized eigenproblem," *Computing*, vol. 42, no. 2-3, pp. 225–238, 1989.

[37] S. Gershgorin, "Über die Abgrenzung der Eigenwerte einer Matrix." *Bull. Acad. Sci. URSS*, no. 6, pp. 749–754, 1931.

[38] U. Kulisch and V. Snyder, "The Exact Dot Product As Basic Tool for Long Interval Arithmetic," *Computing*, vol. 91, no. 3, pp. 307–313, 2011.

[39] V. Pan and J. Reif, "Efficient Parallel Solution of Linear Systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. ACM, 1985, pp. 143–152.

[40] S. M. Rump, "Error estimation of floating-point summation and dot product," *BIT Numerical Mathematics*, vol. 52, no. 1, pp. 201–220, 2012.

[41] D. Lefebvre, P. Chevrel, and S. Richard, "An $\mathcal{H}_\infty$ based control design methodology dedicated to the active control of longitudinal oscillations," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 6, pp. 948–956, 2003.

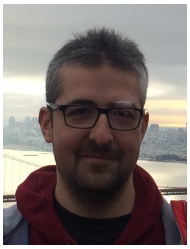
[42] A. Volkova, T. Hilaire, and C. Lauter, "Reliable verification of digital implemented filters against frequency specifications," in *2017 IEEE 24th Symposium on Computer Arithmetic*, 2017.

[43] A. Volkova, M. Istoan, F. De Dinechin, and T. Hilaire, "Towards hardware iir filters computing just right: Direct form i case study," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 597–608, April 2019.

[44] T. W. Parks and J. H. McClellan, "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase," *IEEE Transactions on Circuit Theory*, vol. 19, no. 2, pp. 189–194, Mar 1972.

[45] B. Lopez, "Implémentation optimale de filtres linéaires en arithmétique virgule fixe," Ph.D. dissertation, Sorbonne Universités – University of Pierre and Marie Curie, 2015.

Anastasia Volkova was born in Odessa, Ukraine in 1991. She obtained a PhD in Computer Science from Sorbonne Université in Paris, France in 2017. She did a postdoc at Inria, France and was an AI research resident at Intel Corporation. In 2019 she joined University of Nantes as an Associate professor. Her research interests include computer arithmetic, validated numerical computing and design of optimized software/hardware for Floating- and Fixed-Point algorithms.



Thibault Hilaire was born in 1977, obtained a master's degree from École Centrale de Nantes in 2002, then a PhD from University of Nantes in 2006. After two postdoctoral positions at Université de Rennes 1 and Technische Universität Wien, he joined Sorbonne Université as an associate professor. His research interests include fixed-point arithmetic (and more generally computer arithmetic), software and hardware implementation of signal processing and control algorithms.

Christoph Lauter was born in Amberg, Germany, in 1980. He received this Ph.D. in Computer Science from École Normale Supérieure de Lyon, France, in 2008. He worked as a Software Engineer in the Numerics Team at Intel Corporation. Since 2010, he has been *Maître de Conférences* (Associate Professor) at Sorbonne Université, Paris, France. In 2018, he joined University of Alaska Anchorage (UAA), where he holds an Assistant Professor position. His research interests are Floating-Point Arithmetic and Validated Numerical Computing.