Reliable verification of digital implemented filters against frequency specifications

Anastasia Volkova, Christoph Lauter, Thibault Hilaire

Reliable verification of digital implemented filters against frequency specifications

Anastasia Volkova, Christoph Lauter, Thibault Hilaire
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6
4, place Jussieu 75005 Paris, France
Email: first_name.last_name@lip6.fr

Abstract

Reliable implementation of digital filters in finite-precision is based on accurate error analysis. However, a small error in the time domain does not guarantee that the implemented filter verifies the initial band specifications in the frequency domain. We propose a novel certified algorithm for the verification of a filter's transfer function, or of an existing finite-precision implementation. We show that this problem boils down to the verification of bounds on a rational function, and further to the positivity of a polynomial. Our algorithm has reasonable runtime efficiency to be used as a criterion in large implementation space explorations. We ensure that there are no false positives but false negative answers may occur. For negative answers we give a tight bound on the margin of acceptable specifications. We demonstrate application of our algorithm to the comparison of various finite-precision implementations of filters already fully designed.

I. Introduction

The great majority of signal processing or control algorithms are designed on our desktop computer, and then embedded in digital devices, such as general purposes processors, DSPs, FPGAs or ASICs.

Linear filters are basic bricks of such algorithms, and they are usually designed using software like Matlab[1] or SciPy[2] which are used to design filters fulfilling a given frequency specification. These tools rely on algorithms based on double precision Floating-Point arithmetic. Then, the filter is implemented, often with Fixed-Point arithmetic, for cost or power consumption reasons. The filter algorithm used for the implementation may directly round and use the coefficients obtained at design stage, or deduce new coefficients to round and use, from the previous stage. In both cases, the finite precision used in design and implementation stages induces some uncontrolled errors that may make the implemented filter/controllers differ from the initial specification. Currently, engineers have no other choice than to add "some" design margin in their specification, trust these software tools and perform some simulations to somehow test that the implemented filter respects its specifications.

So this article deals with the reliable *a posteriori* verification of the filter implementation against its mathematical frequency specification. We understand by the term *a posteriori* verification that all filter design and implementation is already finished at that point. A filter comes with its (possibly quantized) coefficients and its filter order and it is not up to our algorithm to question these input data.

This problem cannot straightforwardly be solved in a rigorous way by *computing* the maximum and minimum frequency response and comparing it to a bound. Computing such an extremum value rigorously amounts to rigorously optimizing a function on a subset of the complex numbers, which is not an easy task, even using multiple-precision interval arithmetic [1].

We shall rather consider verification only and decompose that verification problem into the following two parts:

First, suppose, we have a transfer function with its coefficients expressed as fixed- or Floating-Point numbers at some precision (integers scaled by powers of 2) and a set of band specifications. Then, we need to verify whether this filter satisfies the specifications. Such a problem boils down to verification whether a rational function is between bounds. We propose an algorithm that guarantees that no false positive answer is ever given.

Secondly, suppose, we have an algorithm realizing the filter, defined as a set of equations or as a data-flow graph (like those in Matlab/Simulink). In general case, the algorithm's coefficients have been quantized and are hence not those of the transfer function. Therefore, to verify whether the filter satisfies band specifications, its transfer function must be recovered. Usually, this cannot be done exactly. We propose a rigorous approach which computes a multiple precision approximation on the filter's transfer function with a reliable error-bound. Then, we can check whether the implemented filter satisfies band specifications while taking into account the error of the transfer function re-computation, out of the actually used coefficients. Moreover, if the filter does not satisfy the band specifications, our algorithm computes a tight extension to the frequency bands that the filter does satisfy.

As a matter of course, filters do not only have to satisfy frequency specifications but also phase specifications. Our method might be amenable to this setting. We consider phase specifications to be beyond the scope of the paper, though.

The paper is organized as follows. Section II reminds some propositions about linear filters and defines the band specifications. In Section III, the reliable verification of a transfer function is exhibited. With the multiple precision approximation on the

[1] http://www.mathworks.com/
[2] https://www.scipy.org/

transfer function of any linear algorithm, Section IV checks whether an implemented filter satisfies its initial specification. Numerical examples that illustrate the method are given in Section V before conclusion.

Notation: throughout the article matrices are in uppercase boldface, vectors are in lowercase boldface, scalars are in lowercase. The complex unit is notated j ($j^2 = -1$). The conjugation operator on a complex number z is notated $z^\star$.

II. Pre-requisites

A. Digital filters and LTI systems

Digital filters are computational blocks that transform a signal u (i.e. a sequence $u(k)$, where $k \in \mathbb{Z}$ stands for the step-time, according to the sampling period) into a signal y. In this article, we focus on discrete-time Linear Time Invariant (LTI) filters only, i.e. filters that are linear and for which a time shift (a delay) of the input sequence causes a corresponding shift in the output sequence [2], [3]. They are standard basic bricks of digital filtering (and control) and include FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters.

The output of a LTI filter can be expressed by

$$y(k) = \sum_{i=0}^{n} b_i u(k-i) - \sum_{i=1}^{n} a_i y(k-i), \quad \forall k \in \mathbb{Z}, \tag{1}$$

where $\{a_i\}_{1 \leq i \leq n}$ and $\{b_i\}_{0 \leq i \leq n}$ are real numbers defining the filter (n is the order of the filter).

Instead of an input-output relationship in time (the output $y(k)$ with respect to the previous and current inputs and previous outputs), such filters are more often described in the frequency-domain, by means of the $\mathcal{Z}$-transform [3]. For a given signal x, its $\mathcal{Z}$-transform is defined as:

$$X(z) = \sum_{k=-\infty}^{+\infty} x(k) z^{-k}. \tag{2}$$

This transform is an equivalent of the Laplace transform but for discrete-time signals and systems.

Applied to (1), we obtain a relationship between the $\mathcal{Z}$-transform of the input and the output, denoted U and Y respectively:

$$Y(z) = \sum_{i=0}^{n} b_i z^{-i} U(z) - \sum_{i=1}^{n} a_i z^{-i} Y(z) \tag{3}$$

and then $Y(z) = H(z)U(z)$, with

$$H(z) \triangleq \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}. \tag{4}$$

H is called the *transfer function* of the filter, i.e. its input-output relationship in the z-domain. Its restriction to the unit circle ($\{z = e^{j\omega} \mid \forall \omega \in [0, 2\pi]\}$) is the Discrete-Time Fourier Transform (DTFT), so $H(e^{j\omega})$ gives the frequency response of the filter (if a sinusoid signal with normalized frequency ω_0 is applied as input of the filter, then the output will be a sinusoid with the same normalized frequency ω_0, but amplified by $\left|H(e^{j\omega_0})\right|$, and dephased by $\arg\left(H(e^{j\omega_0})\right)$).

B. Filter specifications

In a classical signal processing flow, filters are designed from specifications in the frequency domain, in order to amplify (or preserve) signals in some frequency bands, and attenuate them in other bands.

A filter specification is then composed of several *passbands* (i.e. the gain of the filter for these frequencies should be bounded, often around 1) and *stopbands* (the gain should be lower than a given bound), formally described as:

$$\underline{\beta} \leq \left|H(e^{j\omega})\right| \leq \overline{\beta}, \quad \forall \omega \in [\omega_1, \omega_2]. \tag{5}$$

The lower bound $\underline{\beta}$ is equal to 0 for stopbands, and usually $\underline{\beta} = 1 - \delta$ and $\overline{\beta} = 1 + \delta$ for passbands.

For instance, Figure 1 exhibits a lowpass filter specification, mathematically described by

$$\begin{cases} 1 - \delta_p \leq \left|H(e^{j\omega})\right| \leq 1 + \delta_p, & \forall \omega \in [0, \omega_p] \quad \text{(passband)} \\ \left|H(e^{j\omega})\right| \leq \delta_s, & \forall \omega \in [\omega_s, \pi] \quad \text{(stopband)} \end{cases}$$

For sake of generality, a filter specification will be described in this paper as a set of inequalities as in (5). This way of specifying filters is not the only conceivable way: instead of constant bounds $\underline{\beta}$ and $\overline{\beta}$, one might consider upper and lower bounds be given as polynomials varying in the normalized frequency or even measures allowing spectral densities and partial violations of bounds to be taken into account. Similary, the filter specification only concerns here the magnitude, but not the phase. Considering these alternate ways of filter specifications shall be left to future work.

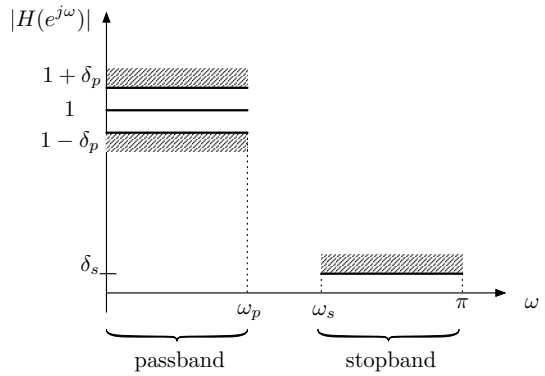


Figure 1: A lowpass filter specification.

We shall consider all frequency specification bounds as hard constraints. Our algorithm will return –in the first place– a boolean answer whether the specification is satisfied or not. Users of the algorithm might nevertheless decide to live with weaker bounds, allowing for margins that cancel out from a systems perspective. They may use our algorithm by adding their possible margins –possibly in a second run of the algorithm– to the specifications to allow for larger frequency specifications.

Sometimes, filter designers prefer to give the bounds in decibels (x dB means $10^{\frac{x}{20}}$ as a bound) and frequencies (normalized frequency ω and frequency f are linked by $\omega = 2\pi\frac{f}{F_s}$ where F_s is the sampling frequency of the filter).

Due to Nyquist-Shannon theorem [3], it is only necessary to specify the frequency up to $\frac{F_s}{2}$ (or the normalized frequency up to π instead of 2π). Due to lack of space, this article considers only filters with real coefficients, setting aside filters with complex coefficients.

III. Verifying bounds on a transfer function

The purpose of this Section is to detail our method that verifies that the modulus of a transfer function H stays between two bounds $\underline{\beta}$ and $\overline{\beta}$ for all z taken on a segment of the unit circle, corresponding to a certain frequency band, i.e. $z = e^{j\omega}$ for all $\omega \in \Omega \subseteq [0, 2\pi]$. In the case when the given bounds cannot be verified, our intention is to compute approximations to problematic frequencies for which the bounds are violated.

We proceed in three steps. In Section III-A, we show how we can reduce the given problem to showing that a rational function with real coefficients stays between two bounds for real arguments taken in a subset of $[0, 1]$. In Section III-B, we then further reduce the problem to showing that a polynomial stays non-negative over a subset of $[0, 1]$. In Section III-C, we briefly describe our approach to computing problematic frequencies in the case when the verification does not succeed.

A. Reducing the problem to a real rational function

We wish to verify that $\underline{\beta} \leq |H(z)| \leq \overline{\beta}$ for all $z = e^{j\omega}$ with $\omega \in \Omega \subseteq [0, 2\pi]$. We suppose that H is given as a rational function $H(z) = \frac{b(z)}{a(z)}$ with real coefficients. Since we can suppose without lack of generality that $\underline{\beta} \geq 0$, this is equivalent to showing that

$$\underline{\beta}^2 \leq |H(z)|^2 \leq \overline{\beta}^2, \quad \forall z = e^{j\omega}, \omega \in \Omega. \tag{6}$$

Since $z = e^{j\omega}$ and the numerator and denominator polynomials a and b have real coefficients, conjugation of z yields $z^\star = 1/z$ and conjugation of the polynomials has no effect. So we have

$$\begin{aligned} |H(z)|^2 &= \frac{b(z)\, b^\star(z^\star)}{a(z)\, a^\star(z^\star)} \\ &= \frac{b(z)\, b(1/z)}{a(z)\, a(1/z)} \\ &= \frac{v(z)}{w(z)}, \end{aligned} \tag{7}$$

where v and w also are polynomials with real coefficients, obtained by simplifying the fraction $\frac{b(z)\, b(1/z)}{a(z)\, a(1/z)}$.

We have hence reduced the problem to verifying that

$$\underline{\beta}^2 \leq \frac{v(z)}{w(z)} \leq \overline{\beta}^2, \quad \forall z = e^{j\omega}, \omega \in \Omega. \tag{8}$$

Taking now $t = \tan\frac{\omega}{2}$, we can write $z = e^{j\omega}$ as

$$z = e^{j\omega} = \cos\omega + j\,\sin\omega = \frac{1-t^2}{1+t^2} + j\,\frac{2t}{1+t^2}. \tag{9}$$

By formally composing v and w with the expression $z = \frac{1-t^2}{1+t^2} + j\,\frac{2t}{1+t^2}$, for example by formal evaluation with Horner's scheme, and clearing numerators and denominators, we can hence obtain four polynomials r, s, ж, щ, all with real coefficients, such that

$$|H(z)|^2 = \frac{v(z)}{w(z)} = \frac{v\left(\frac{1-t^2}{1+t^2} + j\,\frac{2t}{1+t^2}\right)}{w\left(\frac{1-t^2}{1+t^2} + j\,\frac{2t}{1+t^2}\right)} = \frac{r(t) + j\,\text{ж}(t)}{s(t) + j\,\text{щ}(t)}. \tag{10}$$

We can now observe that $|H(z)|^2$ is a real number and that the ratio $\frac{r(t)}{s(t)}$ is hence equal to the complex ratio $\frac{r(t)+j\,\text{ж}(t)}{s(t)+j\,\text{щ}(t)}$. We may therefore drop ж and щ. We have now reduced the problem to verifying that

$$\underline{\beta}^2 \leq \frac{r(t)}{s(t)} \leq \overline{\beta}^2, \quad \forall t = \tan\frac{\omega}{2}, \omega \in \Omega \subseteq [0, 2\pi]\,, \tag{11}$$

where the both polynomials r and s have real coefficients and all other quantities, $\underline{\beta}^2, \overline{\beta}^2, \omega$ and t are all real numbers. We must hence no longer deal with complex ratios and complex numbers and have reduced the problem to verifying the bounds of a real rational function over an interval, subset of the reals.

Unfortunately, the mapping $t = \tan\frac{\omega}{2}$ maps the possible frequencies $\omega \in \Omega \subseteq [0, 2\pi]$ onto to the whole real axis. In our experiments, we found this difficult to handle, partly because the tool we used, Sollya, has very little support for unbounded intervals and partly because having unbounded intervals meant searching for the zeros of certain functions over such unbounded intervals, which we found numerically unstable (see Section III-C for more details). We hence apply a second mapping: $t = \frac{1-2\xi}{\xi\,(1-\xi)}$. Still by formally composing the polynomials r and s with the expression $t = \frac{1-2\xi}{\xi\,(1-\xi)}$, we obtain two polynomials p and q with real coefficients such that

$$|H(z)|^2 = \frac{r(t)}{s(t)} = \frac{p(\xi)}{q(\xi)}. \tag{12}$$

In the same step, we reduce the resulting rational function to its least terms to obtain $\frac{p(\xi)}{q(\xi)}$. In order to do so, we extended the tool we used, Sollya, with an algorithm to compute the gcd of two polynomials [4].

As the inverse mapping $\xi = \frac{t+2-\sqrt{t^2+4}}{2t}$ maps the reals onto the interval $[0, 1]$, we have hence reduced our problem to verifying that

$$\underline{\beta}^2 \leq \frac{p(\xi)}{q(\xi)} \leq \overline{\beta}^2, \quad \forall \xi \in \Xi \subseteq [0, 1]\,. \tag{13}$$

B. Verifying the bounds of a rational function by showing the non-negativity of a polynomial

In order to verify an instance of (13), we can suppose that the interval Ξ the arguments ξ vary in is not reduced to a point and that $\underline{\beta}^2 \neq \overline{\beta}^2$ (otherwise a simple evaluation or a check whether p and q are constant polynomials suffice). We can hence reduce the problem further to obtain:

$$-1 \leq \frac{2p(\xi) - \left(\overline{\beta}^2 + \underline{\beta}^2\right) q(\xi)}{\left(\overline{\beta}^2 - \underline{\beta}^2\right) q(\xi)} \leq 1, \quad \forall \xi \in \Xi. \tag{14}$$

Let

$$g(\xi) = 2p(\xi) - \left(\overline{\beta}^2 + \underline{\beta}^2\right) q(\xi)$$

and

$$h(\xi) = \left(\overline{\beta}^2 - \underline{\beta}^2\right) q(\xi).$$

It hence suffices to verify that

$$\frac{g(\xi)^2}{h(\xi)^2} \leq 1, \quad \forall \xi \in \Xi \tag{15}$$

which is equivalent to showing that

$$h(\xi)^2 - g(\xi)^2 \geq 0, \quad \forall \xi \in \Xi. \tag{16}$$

Let $f(\xi) = h(\xi)^2 - g(\xi)^2$. Again f is a polynomial with real coefficients. We have reduced our problem to showing that the value of this polynomial $f(\xi)$ stays non-negative over all $\xi \in \Xi \subseteq [0, 1]$, where the interval Ξ is easily obtained from the original frequency domain $\Omega = [\omega_1, \omega_2]$.

Our approach to showing that f stays non-negative over Ξ is similar to the one set out in [1]. We typically perform the following checks:

(i) Check whether f is positive at some (arbitrarily chosen) point $\xi_1 \in \Xi$ by (interval arithmetic) evaluation of f at ξ_1 and that f has no zero over the whole interval Ξ. If so, f is non-negative over the whole interval Ξ.

(i) Check whether f is positive at both endpoints of the interval Ξ by (interval) evaluation at these endpoints and that it has exactly one zero over whole interval Ξ, not counting multiplicities. The zero it has in the interval hence is of even multiplicity and the polynomial stays non-negative over the whole interval.

(i) Check whether the interval Ξ can be split into subintervals such that one of the two aforementioned checks become satisfied.

We test whether a polynomial (with real coefficients) has no, one or more zeros over an interval, bounded subset of the reals, utilizing Sturm's theorems on the Sturm sequence of the polynomial, similarly as done in [1]. Sturm's theorem yields the number of real zeros of a real polynomial over a bounded interval, not counting multiplicities [5]. The tool we used, Sollya, includes a fast but rigorous implementation of Sturm's technique [6].

C. Numerically computing problematic frequencies

In the case when our checks verifying if a given transfer function $H(z)$ stays bounded in modulus by the two bounds $\underline{\beta}$ and $\overline{\beta}$ does not succeed, we numerically compute a list of problematic frequencies $\tilde{\omega}_i$, at which one of the bounds is violated. In contrast to the verification step which is completely rigorous in the sense that will never return a positive answer (i.e. the transfer function satisfies the given bounds) while the function actually does not, this numerical step is not fully rigorous. It may miss certain frequencies at which the bounds are violated. It is nevertheless pretty efficient with respect to speeding up the complete LTI filter verification algorithm we set out in Section IV-C, in particular concerning determining a reasonable verification margin (see below).

In our approach, we couple the verification process, described in Sections III-A and III-B, with the possibly needed step of computing problematic frequencies. These frequencies actually correspond, thru the different mappings $\omega \mapsto t \mapsto \xi$, to points ξ at which the polynomial eventually obtained, f, takes negative values. We determine these points as the (negative) extremum points of f. We therefore differentiate f and compute approximations to the zeros of f' by root isolation (still using Sturm's technique) and refinement with Newton-Raphson iterations. The tool we used, Sollya, offers all necessary basic bricks for these computations [6]. Once we obtain a list of points ξ_i at which f becomes negative, we remap these values ξ_i to a list of problematic frequencies, by following the inverse mappings $\xi \mapsto t \mapsto \omega$.

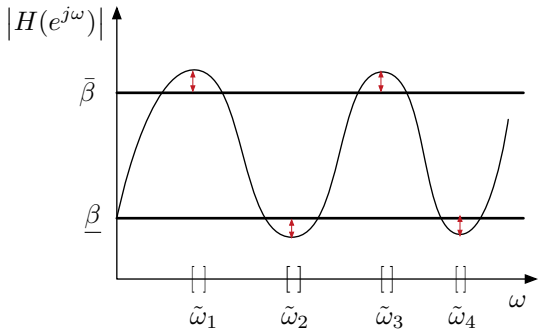


Figure 2: If needed our algorithm returns the problematic frequencies as small intervals $\tilde{\omega}_i$.

IV. Complete algorithm for a LTI realization verification

The above-described algorithm can be used to verify whether a transfer function satisfies a band specification. However, it is the quantization of filter's coefficients that can drastically impact the behavior of the *implemented* filter, leading to violation of the initial band specifications. In this section we propose an approach on verifying an implemented filter against user-given band specifications.

To apply our algorithm from Section III, we need to first recover the transfer function that corresponds to the implemented filter with quantized coefficients. Just like polynomials, a rational transfer function can be evaluated in many different ways. Numerous filter structures (filter algorithms) have been developed over time. With some exceptions (Direct Forms) filter structures do not directly use the coefficients of the transfer function but some modification of them. Deducing a transfer function from a structure differs from one structure to another [7], [8]. Moreover, if they involve approximate computation, the error analysis must be done on a case-by-case basis.

In order to unify the approach of transfer function determination for any filter structure, we propose to use the Specialized Implicit Form (SIF) [9]. In this paper we restrict ourselves to the Single-Input Single-Output (SISO) case, but an extension

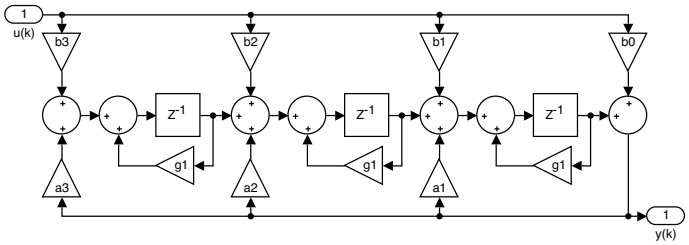


Figure 3: Simulink data-flow of a ρDFIIt form.

to the Multiple-Input Multiple-Output (MIMO) case is straightforward. In Section IV-A we present a quick overview of this representation. In Section IV-B we propose our approach on the computation of the filter's transfer function with the corresponding bound on the approximation error. Finally, in Section IV-C we propose an algorithm for the verification of any implemented filter against some band specifications.

A. Specialized Implicit Form

The Specialized Implicit Form (SIF) has been proposed as an extension of the state-space form:

$$\begin{pmatrix} \boldsymbol{J} & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{K} & \boldsymbol{I}_{n_{\boldsymbol{x}}} & \mathbf{0} \\ -\boldsymbol{l} & \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{t}(k+1) \\ \boldsymbol{x}(k+1) \\ y(k) \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \boldsymbol{M} & \boldsymbol{n} \\ \mathbf{0} & \boldsymbol{P} & \boldsymbol{q} \\ \mathbf{0} & \boldsymbol{r} & s \end{pmatrix} \begin{pmatrix} \boldsymbol{t}(k) \\ \boldsymbol{x}(k) \\ u(k) \end{pmatrix} \tag{17}$$

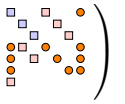
where $u(k)$ represents the input, and $y(k)$ the output, $\boldsymbol{x}(k+1)$ is the $n_{\boldsymbol{x}}$ stored states. Vector $\boldsymbol{t}(k+1)$ holds the $n_{\boldsymbol{t}}$ intermediate variables in the calculations on step k. These calculations are topologically ordered in the matrix $\boldsymbol{J}$, which is lower-triangular with 1s on the diagonal [10]. The diagonal matrix on the left side of the implicit equation (17) allows us to describe the sequence of computations within a filter. For example, $y \leftarrow \boldsymbol{m}_2(\boldsymbol{M}_1\boldsymbol{x})$ is computed as $\boldsymbol{t} \leftarrow \boldsymbol{M}_1\boldsymbol{x}$ and then $y \leftarrow \boldsymbol{m}_2\boldsymbol{t}$. So this sequence is described as

$$\begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ -\boldsymbol{m}_2 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{t} \\ y \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}_1 \\ 0 \end{pmatrix} \boldsymbol{x}. \tag{18}$$

A similar approach to (18) can be used to transform any linear signal processing algorithm or any input/output data-flow graph with delays, multiplications by constants and additions into a SIF. A systematic algorithm for the conversion of any linear data-flow graph represented in Simulink format to the SIF is presented in [10].

For example, a 3rd order linear digital filter implemented with the ρ-Direct Form II transposed (ρDFIIt) algorithm [11], [12], will be described in Simulink as the data-flow graph given in Figure 3. We can observe, that this structure uses ten non-trivial coefficients. The corresponding SIF is the following 7×7 sparse matrix $\boldsymbol{Z}$:

$$\boldsymbol{Z} \triangleq \begin{pmatrix} -\boldsymbol{J} & \boldsymbol{M} & \boldsymbol{n} \\ \boldsymbol{K} & \boldsymbol{P} & \boldsymbol{q} \\ \boldsymbol{l} & \boldsymbol{r} & s \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$



where blue and pink rectangles are '-1' and '1', and circles are non-trivial coefficients.

Once rigorous filter analysis and implementation techniques are developed for the SIF formalism, they can be applied upon any linear realization (classical structures like direct forms, ρ-forms, lattice, etc. have already been directly converted to SIF [9], [13], [14]; others can be obtained from their input-output data flow graph [10]).

B. Reliable computation of the Transfer Function of a filter

The direct way to obtain the transfer function of a given SIF is to first convert it to a discrete-time state-space (dSS) representation $\mathcal{S} \triangleq (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, d)$

$$\mathcal{S} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{b}u(k) \\ y(k) &= \boldsymbol{c}\boldsymbol{x}(k) + du(k) \end{cases} \tag{19}$$

$$\boldsymbol{A} = \boldsymbol{K}\boldsymbol{J}^{-1}\boldsymbol{M} + \boldsymbol{P}, \qquad \boldsymbol{b} = \boldsymbol{K}\boldsymbol{J}^{-1}\boldsymbol{n} + \boldsymbol{q} \tag{20a}$$

$$\boldsymbol{c} = \boldsymbol{l}\boldsymbol{J}^{-1}\boldsymbol{M} + \boldsymbol{R}, \qquad d = \boldsymbol{l}\boldsymbol{J}^{-1}\boldsymbol{n} + s. \tag{20b}$$

The coefficients $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and d can be computed exactly from a SIF, since matrix $\boldsymbol{J}$ is lower-triangular with 1s on the main diagonal.

Then, applying the $\mathcal{Z}$-transform of (2) to (19) we have the classical formula for the transfer function of a state-space [3]:

$$H(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d. \tag{21}$$

By considering the eigendecomposition $\boldsymbol{V}\boldsymbol{E}\boldsymbol{V}^{-1}$ of the matrix $\boldsymbol{A}$ (with $\{\lambda_i\}_{1\leq i\leq n_{\boldsymbol{x}}}$ its associated eigenvalues, i.e. the diagonal elements of $\boldsymbol{E}$), we have

$$\begin{aligned} H(z) &= \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{V}\boldsymbol{E}\boldsymbol{V}^{-1})^{-1}\boldsymbol{b} + d && (22)\\ &= \boldsymbol{c}\boldsymbol{V}(z\boldsymbol{I} - \boldsymbol{E})^{-1}\boldsymbol{V}^{-1}\boldsymbol{b} + d && (23)\\ &= \boldsymbol{c}\boldsymbol{V}\begin{pmatrix} \frac{1}{z-\lambda_1} & & \\ & \ddots & \\ & & \frac{1}{z-\lambda_n} \end{pmatrix}\boldsymbol{V}^{-1}\boldsymbol{b} + d, && (24) \end{aligned}$$

Therefore, the transfer function can be written as a rational function $H(z) = \frac{b(z)}{a(z)}$ with

$$a(z) = \prod_{j=1}^{n_{\boldsymbol{x}}}(z - \lambda_j) \tag{25}$$

$$b(z) = \sum_{i=1}^{n_{\boldsymbol{x}}}(\boldsymbol{c}\boldsymbol{V})_i(\boldsymbol{V}^{-1}\boldsymbol{b})_i\prod_{j\neq i}(z - \lambda_j) + d\ a(z) \tag{26}$$

Then, the accuracy of the computation of the transfer function relies on the accuracy of the computation of its eigendecomposition and on the computation of the polynomial coefficients. Exhibiting a bound on the accuracy of the eigendecomposition is a non-trivial task. Moreover, as it will be shown below, we may need to iteratively increase the accuracy of the computed transfer function. Thus, we propose to use Multiple Precision Floating-Point arithmetic to compute an approximation $\widehat{H}(z)$ using formulas (25) and (26).

Suppose we have available a multiple precision eigensolver such that its error monotonically decreases when its working precision increases, and multiple precision basic bricks for matrix arithmetic. Then, increasing the precision of the computation of $\widehat{H}$ decreases its error $\left\|H - \widehat{H}\right\|$ for any transfer function norm $\|.\|$.
In order to bound this absolute error we propose the following:
Step 1: since we cannot exactly compute H (at least not for any structure), we first compute the approximation $\widehat{H}$ on the transfer function of the dSS $\mathcal{S} = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, d)$ using an eigendecomposition and Multiple Precision arithmetic (equations (25) and (26));
Step 2: we compute the system $\widehat{\mathcal{S}}$ which exactly corresponds to the approximation $\widehat{H}(z)$ using the controllable canonical state-space [3]: if $\{\widehat{b}_i\}_{0\leq i\leq n}$ and $\{\widehat{a}_i\}_{1\leq i\leq n}$ are the coefficients of the polynomial defining $\widehat{H}$, then $\widehat{\mathcal{S}}$ is given by $(\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{b}}, \widehat{\boldsymbol{c}}, \widehat{d})$ with:

$$\begin{aligned} \widehat{\boldsymbol{A}} &= \begin{pmatrix} -\widehat{a}_1 & 1 & & \\ \vdots & & \ddots & \\ \vdots & & & 1 \\ -\widehat{a}_n & 0 & \dots & 0 \end{pmatrix}, \quad \widehat{\boldsymbol{b}} = \begin{pmatrix} \widehat{b}_1 - \widehat{a}_1\widehat{b}_0 \\ \vdots \\ \vdots \\ \widehat{b}_n - \widehat{a}_n\widehat{b}_0 \end{pmatrix} \\ \widehat{\boldsymbol{c}} &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}, \qquad \widehat{d} = \widehat{b}_0 \end{aligned} \tag{27}$$

Step 3: we exactly compute the state-space difference $\Delta\mathcal{S} = \mathcal{S} - \widehat{\mathcal{S}}$, which is the difference between outputs of the two state-spaces, using the classical formula [2]:

$$\boldsymbol{\Delta A} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \widehat{\boldsymbol{A}} \end{pmatrix}, \qquad \boldsymbol{\Delta b} = \begin{pmatrix} \boldsymbol{b} \\ \widehat{\boldsymbol{b}} \end{pmatrix} \tag{28a}$$

$$\boldsymbol{\Delta c} = \begin{pmatrix} \boldsymbol{c} & -\widehat{\boldsymbol{c}} \end{pmatrix}, \qquad \boldsymbol{\Delta}d = d - \widehat{d} \tag{28b}$$

Step 4: then, we can compute a bound on the approximation error $H - \widehat{H}$. By definition of the state-space $\Delta\mathcal{S}$, we have $H = \widehat{H} + \Delta H$ where ΔH is the transfer function of $\Delta\mathcal{S}$. Figure 4 illustrates the relationships between H, $\widehat{H}$, ΔH and their corresponding state-space systems.

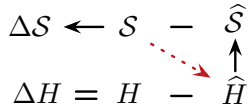


Figure 4: Dotted line: inexact transformation. Straight line: exact transformation.

Using the relations between the transfer function and impulse response norms of a filter [15], we have

$$\forall \omega \in [0, 2\pi], \quad \left|\Delta H(e^{j\omega})\right| \leq \sup_{\omega} \left|\Delta H(e^{j\omega})\right| \leq \|\Delta h\|_1$$

where $\Delta h(k)$ is the impulse response of the system (response to the filter $\Delta\mathcal{S}$ with transfer function ΔH to an impulse signal), and $\|\Delta h\|_1$ its ℓ_1-norm, i.e. $\sum_{k=0}^{\infty} |\Delta h(k)|$. As shown in [16], [17], the norm $\|\Delta h\|_1$ can also be computed using the Worst-Case Peak-Gain (WCPG) measure of $\Delta\mathcal{S}$, denoted $\langle\langle \Delta\mathcal{S} \rangle\rangle$, and defined by:

$$\langle\langle \Delta\mathcal{S} \rangle\rangle \triangleq \sum_{k=0}^{\infty} \left|\boldsymbol{\Delta c}(\boldsymbol{\Delta A})^k \boldsymbol{\Delta b}\right| + |\boldsymbol{\Delta} d| \,. \tag{29}$$

Using the result given in [17], we can compute $\langle\langle \Delta\text{dSS} \rangle\rangle$ with arbitrary precision. Finally, the approximation error $H(z) - \widehat{H}(z)$ is bounded, for any z in the unit circle:

$$\forall \omega \in [0, 2\pi], \quad \left|H(e^{j\omega}) - \widehat{H}(e^{j\omega})\right| \leq \Theta \tag{30}$$

where $\Theta = \langle\langle \Delta\mathcal{S} \rangle\rangle + \varepsilon$ is the Worst-Case Peak Gain of the system $\Delta\mathcal{S}$ computed with arbitrary small absolute error bounded by $\varepsilon > 0$.

C. Reliable verification of the implemented filter

Now, given an implemented filter, we represent it in SIF and then compute, with some precision, an approximation $\widehat{H}$ on its transfer function along with a rigorous upper bound Θ on the corresponding $\langle\langle \Delta\mathcal{S} \rangle\rangle$. We choose the precision of the computations based on an heuristic which increases the compute precision in a loop and is, hence, reasonably fast and accurate but does not provide any guarantee that the precision will eventually be enough. The result will always be reliable, though.

Then, to verify whether the exact magnitude response evaluated on the unit circle is in the interval $[\underline{\beta}; \bar{\beta}]$, it is sufficient that the approximation $\left|\widehat{H}(e^{j\omega})\right|$ for $\omega \in \Omega$ is in the interval $[\underline{\beta} + \Theta; \bar{\beta} - \Theta]$. See Figure 5 for an illustration.

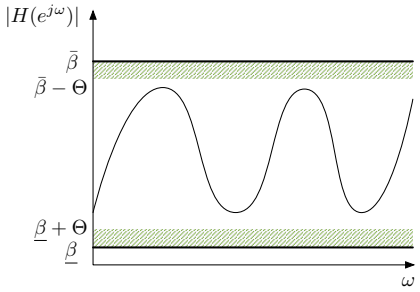
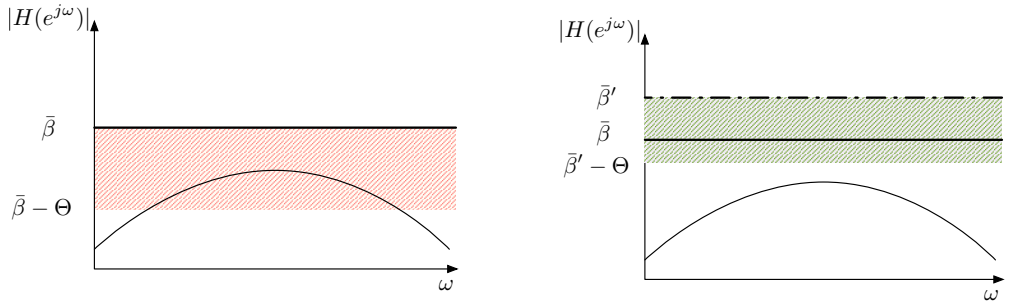


Figure 5: Verify whether $\left|\widehat{H}(e^{j\omega})\right|$ is in $[\underline{\beta} + \Theta; \bar{\beta} - \Theta]$.

If the verification is not successful, we obtain a list of problematic frequencies with the algorithm from Section III. Thus, we can compute the maximum excess of the bounds, enlarge the band by this amount and repeat the process. For example, on Figure 6i we suppose that the approximation $\left|\widehat{H}(e^{j\omega})\right|$ is too close to the bound or Θ is too large. Then, we enlarge the band margin up to some new bound $\overline{\beta}'$ while decreasing Θ, and perform the verification again with the updated band.



(i) (ii)

Figure 6: If verification fails (i), we enlarge the bound (ii).

This algorithm may yield a false negative answer if the initial precision of the transfer function computation was not large enough, which we never observed on the numerical examples we ran the algorithm on.

The detailed description of the heuristics that we used can be found in the long version of the paper that will be published along with our software implementation.

V. Numerical examples

The algorithm presented in Section III was implemented using a modified Sollya tool[3] and further bound with our automatic filter generator using pythonSollya[4]. We use an implementation of the algorithm for the WCPG[5] computation in arbitrary precision [17], which is written in C, using GNU MPFR version 3.1.12, GNU MPFI version 1.5.1 and CLAPACK version 3.2.1. Experiments were done on a laptop computer with an Intel Core i5 processor running at 2.8 GHz and 16 GB of RAM. We present three different use cases of our algorithm illustrated with examples.

Example 1: Our tool can be used to certify an *already existing filter implementation*. Consider the implementation represented as the Simulink data-flow graph of Figure 7, with gains given as 8-bit constants $g_1 = 0.34765625$, $g_2 = 0.3359375$ and $g_3 = 0.08496094$.

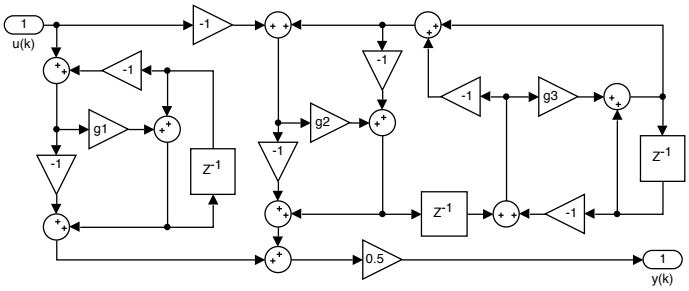


Figure 7: Implemented digital filter.

The task is to verify whether the given implementation is a lowpass filter which satisfies following normalized frequency constraints: passband $\omega_p = 0.1$ with passband amplitude between 1 dB and 3 dB, stopband starting $\omega_s = 0.3$ with minimum attenuation 20 dB.

First, we convert the Simulink graph to the SIF (for which we can compute the transfer function with arbitrary precision). Then, we apply our algorithm from Section IV-C and obtain a positive result in 1.9s. Thus, we obtain a guarantee that the given filter implementation satisfies the desired frequency response requirements. Remark Figure 7 represents a Lattice Wave Digital Filter, the coefficients of which are usually derived from the specifications using direct formulas without actually computing the filter's transfer function [18].

Example 2: Suppose that we need to choose among four different filter realizations: a Direct Form II transposed, a Direct Form II transposed with optimized ρ operator [12], a balanced State-Space [19] and a Lattice Wave Digital Filter [18]. These realizations have different number of coefficients and were designed using different approaches that are beyond the scope of our article. Their floating-point coefficients are results of various approximations and optimizations specific for each structure and can be found in the Appendix.

Our goal is to verify whether after quantization of the coefficients to different formats these realizations will satisfy following frequency response constraints: sampling frequency $F_s = 48$ kHz, passband up to 2.4 kHz with amplitude in $[-0.5\text{dB}; 0.5\text{dB}]$; stopband starting 7.2 kHz with minimum attenuation 80 dB.

The results of verification of the above realizations to 32, 16 and 8 bits are listed in Table I. We can observe that for certain filters a specification is fulfilled only if a safe margin is added; for certain cases that margin is as large as $6.71e-1$ dB. Exceptions similar to the balanced State-Space and Lattice Wave structures may also occur: with 32 bits coefficients, their transfer functions do not pass the verification but the coefficient quantization effects cancel out for 16 bits and verification "luckily" passes.

If 16-bit quantization is the target format, we see that except DFIIt, all structures verify the specifications and now the designer may concentrate on choosing the best realization according to other criteria (for example, number of coefficients or datapath delay). If 8-bit quantization is the target format, the safe margins computed with our tool can be used to redesign the realizations and repeat the verification.

On the other hand, if the realizations are impossible to redesign, we can determine what is the maximum quantization of the coefficients for which our tool gives a positive answer.

Finally, for the cases when the transfer function respects the band specifications our algorithm gives an answer quickly. Most time is spent on computing the safe margin, especially when an overflow is on the edge of the band.

[3]https://scm.gforge.inria.fr/anonscm/git/sollya/sollya.git

[4]https://scm.gforge.inria.fr/anonscm/git/metalibm/pythonsollya.git

[5]https://scm.gforge.inria.fr/anonscm/git/metalibm/wcpg.git

	wordlength	32	16	8
DFIIt	margin	✓	unstable	unstable
	time	12.49s	-	-
ρ DFIIt	margin	✓	✓	4.68e-3 dB
	time	13.12s	4.19s	104.01s
State-Space Balanced	margin	6.16e-10 dB	✓	6.71e-1 dB
	time	12.27s	18.18s	92.05s
Lattice Wave	margin	3.80e-10 dB	✓	1.73e-2 dB
	time	920.88s	4.58s	200.83s

Table I: Checking quantized realizations.

Example 3: Finally, our algorithm can be used to certify the result of different design methods even before structure choice or quantization. For example, we may compare the quality of different transfer function design methods in Matlab. Some of the widely used algorithms for the transfer function design are Butterworth [20], Elliptic [21], [22] and Chebyshev [23].

We propose to verify whether transfer functions (of order suggested to be minimal by Matlab) satisfy their initial frequency constraints. They always should. For a simple test, we consider the frequency specifications used in Example 2. To take into account that the computations in Matlab are done in double precision, we find to reasonable to apply a design margin 10^{-15} dB before using our algorithm from Section III. From Table II we see that only Butterworth design method instantly satisfies the specifications. Despite the initial design margin, Chebyshev and Elliptic methods still may require up to 5.65e-12 dB of margin to verify the desired specifications. On the other hand, even these margins may be acceptable depending on filter designer's needs.

Thus, our algorithm can be used to certify that the filter design method is rigorous, or give the designer a perception of the sufficient correction of the design margin.

		Butterworth	Chebyshev	Elliptic
Matlab	OK/margin	✓	5.65e-12 dB	5.65e-12 dB
	time	0.81s	11.47s	2.36s

Table II: Verification of transfer function design methods.

VI. Conclusion and perspectives

With this paper, a rigorous method to verify a transfer function of LTI digital filters against band specifications in the frequency domain has been developed. It relies on translating the problem of verifying bounds on magnitude response evaluated on a unit circle to the verification of positivity of a real polynomial. Our algorithm guarantees that no false positives occur. In the case of unsuccessful verification a list of problematic frequencies is provided, for which we compute the maximum overflow over the band specification. We propose an implementation using a combination of interval and rational arithmetic in Sollya tool.

We applied this method to develop an approach on the verification of any implemented filter. It relies on the multiple precision computation of a transfer function corresponding to the filter realization with quantized coefficients. Then, we bound the approximation error using the Worst-Case Peak Gain measure in arbitrary precision.

This approach opens various possibilities on the verification of digital filters. As it is automatic, our approach allows for easy integration into automatic filter implementation tools.

Moreover, our verification algorithm can be applied to compare different filter structures with various Fixed-Point settings. Such a comparison offers a filter designer an overview of the implementation possibilities. On top of that, the information on the safe margin can be used as a margin in design process. For instance, we can narrow down the initial band by the safe margin and re-design the filter with the new band specifications. Then, the re-computed filter with quantized coefficients should respect the initial conditions. Finally, using the list of problematic frequencies we can probably improve the rational Remez [24] algorithm, that is usually behind the transfer function design.

However, overall time-efficiency of our implementation can be improved. Passing too much time on the computation of the safe margin can be a significant drawback were the algorithm to be used during the exploration of a large design space.

The approach sets out this paper is concerned with the static quantization of the filters coefficients only. The dynamic roundoff error observed when the implemented filter runs is taken into account by other approaches [25], [26], [27]. Combining the analysis of both error sources is future work.

APPENDIX

A. Coefficients of filters from Example 2

In Example 2 from Section V we consider four realizations that realize a 9^{th} order IIR filter. Their coefficients have been subject to different optimizations specific to each structure.

Direct Form II transposed: This realization has the same coefficients as the transfer function of the filter. Its numerator $\boldsymbol{b}$ and denumerator $\boldsymbol{a}$ are:

$$\boldsymbol{b} = \begin{pmatrix} 8.09616443886441e-8 \\ 7.28654809378781e-7 \\ 2.91461913803914e-6 \\ 6.80077821613168e-6 \\ 1.02011670932711e-5 \\ 1.02011672709068e-5 \\ 6.80077808468127e-6 \\ 2.91461921175795e-6 \\ 7.28654797166328e-7 \\ 8.09616444719108e-8 \end{pmatrix}, \quad \boldsymbol{a} = \begin{pmatrix} 1.0 \\ -6.91501406218501e+0 \\ 2.14466415520778e+1 \\ -3.91222980067432e+1 \\ 4.62252163185152e+1 \\ -3.66656397046805e+1 \\ 1.95139227445332e+1 \\ -6.71659680242323e+0 \\ 1.35615062632188e+0 \\ -0.122341213054255e+0 \end{pmatrix}$$

ρ-operator Direct Form II transposed: This realization has the same number of coefficients as the Direct Form II transposed but uses a modified delay operator. The corresponding Specialized Implicit Form matrix $\boldsymbol{Z}$ is:

$$\boldsymbol{Z} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8.09616444e-8 \\ 6.91501406e+0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.28654799e-7 \\ -2.14466416e+1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2.91461920e-6 \\ 3.91222980e+1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 6.80077813e-6 \\ -4.62252163e+1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1.02011672e-5 \\ 3.66656397e+1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1.02011672e-5 \\ -1.95139227e+1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6.80077813e-6 \\ 6.71659680e+0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2.91461920e-6 \\ -1.35615063e+0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 7.28654799e-7 \\ 1.22341213e-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8.09616444e-8 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{31}$$

Balanced state-space: This structure has the largest amount of coefficients that are grouped into four matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$. Because of the large size, we present the coefficients column by column. For example, $M(:, j_1 : j_2)$ denotes columns of matrix M from column index j_1 to column index j_2 included.

$$\boldsymbol{A}(:, 1:3) = \begin{pmatrix} 9.84412534559901e-1 & -1.54788386642703e-1 & -2.06985457392825e-4 \\ 1.54788386642702e-1 & 9.44983562621590e-1 & -2.00998437246727e-1 \\ -2.06985457387746e-4 & 2.00998437246727e-1 & 8.91451247669984e-1 \\ -2.54616481478342e-2 & 2.56192963965194e-2 & -2.19615788882919e-1 \\ -6.32800101413183e-3 & 3.28943488585008e-2 & -4.70663147422409e-2 \\ -3.67081410584489e-3 & 1.01785432069421e-2 & -2.70763794266093e-2 \\ 9.98078553858957e-4 & -3.32572318297586e-3 & 7.35371936546159e-1 \\ -2.15657523955162e-4 & 6.88163327586583e-4 & -1.58919858004927e-3 \\ 2.76696615105348e-5 & -8.89436596970331e-5 & 2.03894253656991e-4 \end{pmatrix}$$

$$\boldsymbol{A}(:, 4:6) = \begin{pmatrix} 2.54616481478347e-2 & -6.32800101413297e-3 & 3.67081410584512e-3 \\ 2.56192963965197e-2 & -3.28943488585005e-2 & 1.01785432069422e-2 \\ 2.19615788882918e-1 & -4.70663147422419e-2 & 2.70763794266107e-2 \\ 8.28330994157642e-1 & 2.22520998809128e-1 & -5.05922546655891e-2 \\ -2.22520998809128e-1 & 7.65978810562563e-1 & 2.16914762328571e-1 \\ -5.05922546655855e-2 & -2.16914762328573e-1 & 7.08063302951966e-1 \\ 1.80048266606006e-2 & 4.36139632080078e-2 & 2.0207191573245e-1 \\ -3.65628640739942e-3 & -1.01333758827186e-2 & -3.20997699114317e-2 \\ 4.74107807436983e-4 & 1.2848654396977e-3 & 4.34923676309711e-3 \end{pmatrix}$$

$$\boldsymbol{A}(:, 7:9) = \begin{pmatrix} 9.98078553860207e-4 & 2.15657523951793e-4 & 2.76696614924035e-5 \\ 3.32572318297719e-3 & 6.88163327582342e-4 & 8.89436597243942e-5 \\ 7.35371936546042e-3 & 1.58919858005338e-3 & 2.03894253732018e-4 \\ -1.80048266605967e-2 & -3.65628640741023e-3 & -4.74107807447374e-4 \\ 4.36139632080109e-2 & 1.01333758827138e-2 & 1.28486543968479e-3 \\ -2.02071915732453e-1 & -3.20997699114184e-2 & -4.34923676306156e-3 \\ 6.5167003556539e-1 & -1.75022739029844e-1 & -1.86034616397461e-2 \\ 1.75022739029834e-1 & 5.96527629913795e-1 & -1.29541395028042e-1 \\ -1.86034616397113e-2 & 1.2954139502792e-1 & 5.43595944182089e-1 \end{pmatrix}$$

$$\boldsymbol{B} = \begin{pmatrix} -9.50221004521112e-2 \\ 2.15342515945504e-1 \\ -2.7805645514003e-1 \\ -2.34961139460641e-1 \\ -1.34013062083995e-1 \\ -5.45793498027322e-2 \\ 1.63381306938678e-2 \\ -3.44881517830594e-3 \\ 4.44234222423183e-4 \end{pmatrix} \quad \boldsymbol{C}^\top = \begin{pmatrix} -9.50221004521177e-2 \\ -2.15342515945504e-1 \\ -2.7805645514003e-1 \\ 2.34961139460642e-1 \\ -1.34013062083995e-1 \\ 5.45793498027317e-2 \\ 1.6338130693868e-2 \\ 3.44881517830474e-3 \\ 4.44234222395272e-4 \end{pmatrix} \quad \boldsymbol{D} = 8.09616443747663e-8$$

Lattice Wave Digital Filter: This realization has the smallest number of coefficients that is equal to the order of the filter. This structure consists of two parallel branches (top and bottom) that realize all-pass filters.
Coefficients $\boldsymbol{\gamma_t}$ of the top branch of the realization:

$$\boldsymbol{\gamma_t} = (0.309545756846,\ 0.426986744076,\ 0.0648857726338,\ 0.115929866936,\ 0.0648857724342) \tag{32}$$

Coefficients $\boldsymbol{\gamma_b}$ of the bottom branch of the realization:

$$\boldsymbol{\gamma_b} = (0.499598692531,\ 0.0648857723853,\ 0.30101467827,\ 0.0648857724933) \tag{33}$$

REFERENCES

[1] S. Chevillard, J. Harrison, M. Joldes, and C. Lauter, "Efficient and accurate computation of upper bounds of approximation errors," *Theoretical Computer Science*, vol. 412, no. 16, pp. 1523 – 1543, 2011.
[2] T. Kailath, *Linear Systems*. Prentice-Hall, 1980.
[3] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. NJ, USA: Prentice Hall Press, 2009.
[4] B. W. Char, K. O. Geddes, and G. H. Gonnet, "Gcdheu: Heuristic polynomial gcd algorithm based on integer gcd computation," *Journal of Symbolic Computation*, vol. 7, no. 1, pp. 31 – 48, 1989.
[5] M.-F. Roy, *Basic algorithms in real algebraic geometry and their complexity: from Sturm's theorem to the existential theory of reals*, ser. Expositions in Mathematics. de Gruyter, 1996, vol. 23, in F. Broglia (Ed.), Lectures in Real Geometry.
[6] S. Chevillard, C. Lauter, and M. Joldes, *Users' manual for the Sollya tool, Release 6.0*, LIP, LIP6, LORIA, CNRS.
[7] A. Varga, "Computation of transfer-function matrices of generalized state-space models," *Int Journal of Control*, vol. 50, no. 6, 1989.
[8] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.
[9] T. Hilaire, P. Chevrel, and J. Whidborne, "A unifying framework for finite wordlength realizations," *IEEE Trans. on Circuits and Systems*, vol. 8, no. 54, pp. 1765–1774, 2007.
[10] T. Hilaire, A. Volkova, and M. Ravoson, "Reliable fixed-point implementation of linear data-flows," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, 2016.
[11] Z. Zhao and G. Li, "Roundoff noise analysis of two efficient digital filter structures," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 790–795, February 2006.
[12] G. Li and Z. Zhao, "On the generalized DFIIt structure and its state-space realization in digital filter implementation," *IEEE Trans. on Circuits and Systems*, vol. 51, no. 4, pp. 769–778, April 2004.
[13] T. Hilaire and P. Chevrel, "Sensitivity-based pole and input-output errors of linear filters as indicators of the implementation deterioration in fixed-point context," *EURASIP Journal on Advances in Signal Processing*, January 2011.
[14] A. Volkova and T. Hilaire, "Fixed-point implementation of lattice wave digital filter: Comparison and error analysis," in *23rd European Signal Processing Conference (EUSIPCO)*, Aug 2015, pp. 1118–1122.
[15] S. P. Boyd and J. Doyle, "Comparison of peak and rms gains for discrete-time systems," *Syst. Control Lett.*, vol. 9, no. 1, pp. 1–6, June 1987.
[16] V. Balakrishnan and S. Boyd, "On computing the worst-case peak gain of linear systems," *Systems & Control Letters*, vol. 19, 1992.
[17] A. Volkova, T. Hilaire, and C. Lauter, "Reliable evaluation of the worst-case peak gain matrix in multiple precision," in *Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on*, June 2015, pp. 96–103.
[18] L. Gazsi, "Explicit formulas for lattice wave digital filters," *IEEE Trans. Circuits & Systems*, vol. 32, no. 1, 1985.
[19] M. Gevers and G. Li, *Parametrizations in Control, Estimation and Filtering Probems*. Springer-Verlag, 1993.
[20] S. Butterworth, "On the theory of filter amplifiers." *Wireless Engineer*, vol. 7, pp. 536–541, 1930.
[21] E. I. Zolotarev, "Application of elliptic functions to problems about functions with least and greatest deviation from zero," *Zap. Imp. Akad. Nauk. St. Petersburg*, vol. 30, no. 5, 1877, (in Russian).
[22] W. Cauer, "Synthesis of linear communication networks," *McGraw-Hill, New York*, 1958.
[23] R. W. Daniels, *Approximation Methods for Electronic Filter Design.* New York: McGraw-Hill., 1974.
[24] I. W. Selesnick, M. Lang, and C. S. Burrus, "Magnitude squared design of recursive filters with the chebyshev norm using a constrained rational remez algorithm," in *Proceedings of IEEE 6th Digital Signal Processing Workshop*, Oct 1994, pp. 23–26.
[25] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 10, pp. 3197–3208, Nov 2008.
[26] A. Volkova, T. Hilaire, and C. Lauter, "Determining fixed-point formats for a digital filter implementation using the worst-case peak gain measure," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, Nov 2015, pp. 737–741.
[27] 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. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, vol. 4, 2000, pp. 57–60 vol.4.