Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision

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

***Abstract*—The worst-case peak gain (WCPG) of a linear filter is an important measure for the implementation of signal processing algorithms. It is used in the error propagation analysis for filters, thus a reliable evaluation with controlled precision is required. The WCPG is computed as an infinite sum and has matrix powers in each summand. We propose a direct formula for the lower bound on truncation order of the infinite sum in dependency of desired truncation error. Several multiprecision methods for complex matrix operations are developed and their error analysis performed. A multiprecision matrix powering method is presented. All methods yield a rigorous solution with an absolute error bounded by an a priori given value. The results are illustrated with numerical examples.**

Introduction

The majority of control and digital signal processing algorithms are dedicated to linear time-invariant (LTI) systems with finite or infinite impulse response. Most of them are implemented for application in embedded systems, which use finite-precision arithmetic. Unfortunately, the quantification of coefficients and further roundoff errors lead to degradation of the algorithms. Therefore, an accurate error analysis of implementation of such algorithms is required.

However, this analysis is complicated by the non-linear propagation of errors through the filter as they are amplified on each step by internal state of the system. A solution is proposed in [8], based on a property of bounded-input bounded output systems [1], [2] where the largest possible peak value of the output is determined by the use of the Worst-Case Peak Gain (WCPG) matrix. Error propagation analysis in LTI systems is directly dependent on the reliable evaluation of the WCPG.

This measure is computed with an infinite sum and has matrix powers in each summand. These problems are both known to be non-trivial. In this article we propose a detailed algorithm for the reliable evaluation of the WCPG matrix with multiple precision. This algorithm ensures that the WCPG is computed with an absolute error rigorously bounded by an a priori given value ε. For these purposes several multiprecision algorithms for complex entries with rigorous bounds were developed. This is achieved by adapting the precision of intermediate computations and correct rounding. Therefore, we present not only the error analysis of the approximations made on each step of the WCPG computation, but we also deduce the required accuracy for our kernel multiprecision algorithms such that the overall error bound is satisfied.

We analyze the error induced by truncating the infinite sum and a direct formula for the computation of a lower bound on truncation order for a desired absolute error. The truncation order algorithm involves Interval Arithmetic computations and uses Theory of Verified Inclusions.

Some preliminary definitions about LTI systems are recalled in Sec. I. Sec. II describes the global algorithm used to reliably evaluate the WCPG matrix. The truncation order and the truncation error are analyzed in Sec. III. Sec. IV is focused on the different steps used for the summation and the associated error analysis, whereas Sec. V details some basic bricks in multiple precision. Finally, numerical examples are presented in Sec. VI before conclusion.

Notation: Throughout the article matrices are in uppercase boldface (for example $\boldsymbol{A}$), vectors are in lowercase boldface (for example $\boldsymbol{v}$), scalars are in lowercase (for example α). Operators $\otimes$, and $\oplus$ denote floating-point (FP) multiplication and addition respectively, $\mathbb{F}$ the set of FP numbers. $[x]$ corresponds to an interval. $\boldsymbol{A}^*$ denotes the conjugate transpose of the matrix $\boldsymbol{A}$. All matrix absolute values and inequalities are considered to be element-by-element, for example $|\boldsymbol{A}| < |\boldsymbol{B}|$ denotes $\left|\boldsymbol{A}_{ij}\right| < \left|\boldsymbol{B}_{ij}\right|$ $\forall i, j$. In addition, $\boldsymbol{A} < \varepsilon$ denotes $\boldsymbol{A}_{ij} < \varepsilon\ \forall i, j$. $\boldsymbol{I}_n$ denotes the identity matrix of size $n \times n$ and $\rho(\boldsymbol{A})$ the spectral radius of $\boldsymbol{A}$.

In further discussions the error matrices are bounded in respect to their Frobenius norm. Let $\boldsymbol{K}$ be a square $n \times n$ matrix with $\|\boldsymbol{K}\|_2 \leqslant 1$, then for all k, $\left\|\boldsymbol{K}^k\right\|_2 \leqslant 1$ and $\left\|\boldsymbol{K}^k\right\|_F \leqslant \sqrt{n}$.

I. LTI filters and Worst-Case Peak Gain

A Linear Time Invariant (LTI) filter is a system used in signal processing, image processing, control theory, etc. It is defined by an input-output relationship in time-domain or equivalently in frequency-domain. Linear controllers, Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) are classical examples of LTI systems. We focus here only on discrete-time systems: a discrete-time LTI system (filter) is a numerical application that transforms an input signal $\{\boldsymbol{u}(k)\}_{k\geqslant 0}$ into an output signal $\{\boldsymbol{y}(k)\}_{k\geqslant 0}$ ($\boldsymbol{u}(k)$ and $\boldsymbol{y}(k)$ may be vectors or scalars), where $k \in \mathbb{N}$ is the step time.

A common input-output relationship is the state-space representation [9]. It describes the evolution of the state vector $\boldsymbol{x}(k)$ from the previous step and the input:

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

where $\boldsymbol{u}(k) \in \mathbb{R}^{q\times 1}$ is the input vector, $\boldsymbol{y}(k) \in \mathbb{R}^{p\times 1}$ the output vector, $\boldsymbol{x}(k) \in \mathbb{R}^{n\times 1}$ the state vector and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times q}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{D} \in \mathbb{R}^{p\times q}$ are the state-space matrices of the system. Unlike a mathematical function, the output at time k depends not only on in the input at time k but also on the internal state of the filter (generally determined from the previous inputs and outputs).

Proposition 1 (Bounded Input Bounded Output systems): Let $\mathcal{H}$ be a state-space system. If an input $\{\boldsymbol{u}(k)\}_{k\geqslant 0}$ is known to be bounded by $\bar{\boldsymbol{u}}$ ($\forall k \geqslant 0, \quad |\boldsymbol{u}_i(k)| \leqslant \bar{\boldsymbol{u}}_i, \quad 1 \leqslant i \leqslant q$), then the output $\{\boldsymbol{y}(k)\}_{k\geqslant 0}$ will be bounded iff the spectral radius $\rho(\boldsymbol{A})$ is strictly less than 1. This property is known as the Bounded Input Bounded Output (BIBO) stability [9].

Moreover, in that case, the output is (component-wise) bounded by $\bar{\boldsymbol{y}}$ with $\bar{\boldsymbol{y}} = \boldsymbol{W}\bar{\boldsymbol{u}}$ where $\boldsymbol{W} \in \mathbb{R}^{p\times q}$ is the Worst-Case Peak Gain matrix [1] of the system, defined by

$$\boldsymbol{W} := |\boldsymbol{D}| + \sum_{k=0}^{\infty} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right| \tag{2}$$

Proof: Let $\{\boldsymbol{J}(k)\}_{k\geqslant 0}$ be the impulse response matrix of the system, i.e. $\boldsymbol{J}_{ij}(k)$ is the response on the i^{th} output to the Dirac impulse at time $k = 0$ (i.e. $\delta(0) = 1$ and $\delta(k) = 0, \quad \forall k \neq 0$) on the j^{th} input. With (1), we have

$$\boldsymbol{J}(k) = \begin{cases} \boldsymbol{D} & \text{if } k = 0 \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B} & \text{if } k > 0. \end{cases} \tag{3}$$

Thanks to the linearity and time invariance property of LTI systems [9], we get

$$\boldsymbol{y}(k) = \sum_{l=0}^{k} \boldsymbol{J}(l)\boldsymbol{u}(k-l). \tag{4}$$

Then the output is (component-wise) bounded by

$$\boldsymbol{y}(k) \leqslant \left(\sum_{l=0}^{k} |\boldsymbol{J}(l)|\right)\bar{\boldsymbol{u}}, \quad \forall k \geqslant 0. \tag{5}$$

We have equality for the i^{th} output if the input is such that $\boldsymbol{u}_j(l) = \bar{\boldsymbol{u}} \cdot \operatorname{sign}\left(\boldsymbol{J}_{ij}(k-l)\right)$, $\forall 0 \leqslant l \leqslant k$, where $\operatorname{sign}(x)$ returns ± 1 or 0 depending on the sign of x. ■

Remark 1: $\boldsymbol{W}\bar{\boldsymbol{u}}$ is the supremum of the output $\{\boldsymbol{y}\}_{k\geqslant 0}$, since it is possible to build a finite input $\{\boldsymbol{u}(k)\}_{0\leqslant k\leqslant K}$ to approach it on any given output at any given distance.

Remark 2: This proposition can be completed when considering intervals for the input, instead of bounds (corresponding to symmetric intervals). In that case, the Worst-Case Peak Gain matrix indicates by how much the radius of the input interval is amplified on the output [8] (although this is not valid for the transient phase, i.e. for the few first steps). However, even in that case, $\boldsymbol{W}\bar{\boldsymbol{u}}$ is a supremum we need to compute.

This proposition can be used to bound outputs, states or intermediate variables in the context of finite precision implementation of algorithms, and more specifically in Fixed-Point arithmetic. In [7], an extension of the state-space has been presented, in order to represent and encompass all the possible algorithms for linear filters (i.e. all the input-to-output data flows based on additions, multiplications by constant and delay, such as state-space, direct forms, ρDFIIt [18], etc.), and the same approach was applied.

First, it is used to bound all the variables involved in the algorithm, and then to determine their fixed-point representation (position of the Most Significant Bit and scaling) while preserving by construction from overflow.

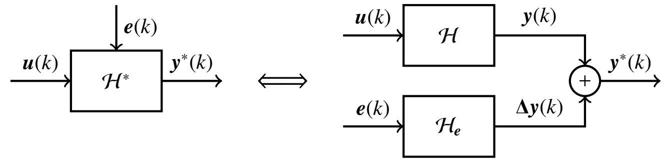

Fig. 1. The implemented filter is equivalent to the exact filter where the output is corrupted by the computational errors passing themselves through a filter.

Second, it is used to determine the impact on the output of the computational errors. Classical error analysis cannot be used in that context due to the feedback scheme of the computation (Interval Arithmetic or Affine Arithmetic do not provide tight bounds [12]).

Since the filter is linear, the implemented filter $\mathcal{H}^*$ can be seen as the exact filter $\mathcal{H}$ where the output is corrupted by the vector of errors $\boldsymbol{e}(k)$ occurring at each sum of product through a given linear filter $\mathcal{H}_e$ (see Figure 1).

A State-space representation of $\mathcal{H}_e$ can be obtained analytically [8] and Proposition 1 can be used to determine the output error $\boldsymbol{\Delta y}$ due to finite-precision arithmetic.

For all these reasons, the reliable computation of the Worst-Case Peak Gain matrix is a required step for the accurate error analysis of LTI systems in finite precision.

II. Algorithm for WCPG evaluation

Given a BIBO stable LTI filter in state-space realization (1) and ε, a desired absolute approximation error, we want to determine the Worst-Case Peak Gain matrix $\boldsymbol{W}$ of this filter, defined in (2). While computing such an approximation, various errors, such as truncation and summation errors, are made.

Instead of directly computing the infinite sum $\left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|$ for any $k \geqslant 0$, 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 FP sum $\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|$ for $0 \leqslant k \leqslant N$.

Our approach to compute the approximation $\boldsymbol{S}_N$ of $\boldsymbol{W}$ is summarized in algorithm 1 where all the operations ($\otimes$, $\oplus$, inv, abs, etc.) are FP multiple precision operations done at various precisions to be determined such that the overall error is less than ε:

$$|\boldsymbol{W} - \boldsymbol{S}_N| \leqslant \varepsilon. \tag{6}$$

The overall error analysis decomposes into 6 steps, where each one expresses the impact of a particular approximation (or truncation), and provides the accuracy requirements for the associated operations such that the result is rigorously bounded by ε. These steps are discussed in detail in Sec. III and IV:

Step 1: Let $\boldsymbol{W}_N$ be the truncated sum

$$\boldsymbol{W}_N := \sum_{k=0}^{N} \left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right| + |\boldsymbol{D}|. \tag{7}$$

We compute a truncation order N of the infinite sum $\boldsymbol{W}$ such that the truncation error is less than $\varepsilon_1 > 0$:

$$|\boldsymbol{W} - \boldsymbol{W}_N| \leqslant \varepsilon_1. \tag{8}$$

See Sec. III for more details.

Algorithm 1: 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: $\boldsymbol{S}_N \in \mathbb{F}^{p\times q}$
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: $\boldsymbol{P}_k \leftarrow \boldsymbol{T} \otimes \boldsymbol{P}_{k-1}$
Step 5: $\boldsymbol{L}_k \leftarrow \boldsymbol{C}' \otimes \boldsymbol{P}_k \otimes \boldsymbol{B}'$
Step 6: $\boldsymbol{S}_k \leftarrow \boldsymbol{S}_{k-1} \oplus \mathrm{abs}(\boldsymbol{L}_k)$
end
return $\boldsymbol{S}_N$

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 norm of $\boldsymbol{A}$ is larger than 1 and its eigenvalues are close to 1. However, if $\boldsymbol{A}$ may 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 the diagonal matrix $\boldsymbol{E}$, which is more convenient.

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 $\varepsilon_2 > 0$:

$$\left|\boldsymbol{W}_N - \sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|\right| \leqslant \varepsilon_2. \tag{9}$$

See Sec. IV-A.

Step 3: We compute approximations $\boldsymbol{B}'$ and $\boldsymbol{C}'$ of $\boldsymbol{V}^{-1}\boldsymbol{B}$ 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 $\varepsilon_3 > 0$:

$$\left|\sum_{k=0}^{N}\left|\boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right| - \sum_{k=0}^{N}\left|\boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}'\right|\right| \leqslant \varepsilon_3. \tag{10}$$

See Sec. IV-B.

Step 4: We compute in $\boldsymbol{P}_k$ the powers $\boldsymbol{T}^k$ of $\boldsymbol{T}$ with a certain accuracy. It is required that the error be less than $\varepsilon_4 > 0$:

$$\left|\sum_{k=0}^{N}\left|\boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}'\right| - \sum_{k=0}^{N}\left|\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'\right|\right| \leqslant \varepsilon_4. \tag{11}$$

See Sec. IV-C.

Step 5: We compute in $\boldsymbol{L}_k$ each summand $\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'$ with a error small enough such that the overall approximation error induced by this step is less than $\varepsilon_5 > 0$:

$$\left|\sum_{k=0}^{N}\left|\boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'\right| - \sum_{k=0}^{N}\left|\boldsymbol{L}_k\right|\right| \leqslant \varepsilon_5. \tag{12}$$

See Sec. IV-D.

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 $\varepsilon_6 > 0$:

$$\left|\sum_{k=0}^{N}|\boldsymbol{L}_k| - \boldsymbol{S}_N\right| \leqslant \varepsilon_6. \tag{13}$$

See Sec. IV-E.

By ensuring that each step verifies its bound ε_i, and taking $\varepsilon_i = \frac{1}{6}\varepsilon$, we get $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \varepsilon_5 + \varepsilon_6 \leqslant \varepsilon$, hence (6) will be satisfied if inequalities (8) to (13) are.

Our approach hence determines first a truncation order N and then performs summation up to that truncation error, whilst adjusting precision in the different summation steps. A competing approach would be not to start with truncation order determination but to immediately go for summation and to stop when adding more terms does not improve accuracy. However, such an approach would not allow the final error to be bounded in an a priori way. As we shall see, the multiple precision FP summation needs to know a bound on the number of terms to be summed, beforehand.

III. Truncation order and truncation error

In [1] Balakrishnan and Boyd propose "simple" lower and upper bounds on N. However, they describe their algorithm in terms of exact arithmetic, *i.e.* do not propose any error analysis. This iterative algorithm has several difficulties: first of all, matrix $\boldsymbol{A}$ exponentiation is present, which would require an error analysis such as the one proposed in this article. Secondly, on each iteration (the quantity of which may reach order as high as N) a solution of Lyapunov equations is required, for which there exist no ready to use solution with rigorous error bounds on the result. Therefore, a different approach is indispensable. In this Section we propose a direct formula for the lower bound on N along with a reliable evaluation algorithm.

The goal is to determine a lower bound on the truncation order N of the infinite sum (2) such that its tail is smaller than the given ε_1. Obviously, $\boldsymbol{W}_N$ is a lower bound on $\boldsymbol{W}$ and increases monotonically to $\boldsymbol{W}$ with increasing N. Hence the truncation error is

$$|\boldsymbol{W} - \boldsymbol{W}_N| = \sum_{k>N}\left|\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}\right|. \tag{14}$$

A. A bound on the truncation error

Many simple bounds on (14) are possible. For instance, if the eigendecomposition of $\boldsymbol{A}$ is computed

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1} \tag{15}$$

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

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{\Phi}\boldsymbol{E}^k\boldsymbol{\Psi} = \sum_{l=1}^{n}\boldsymbol{R}_l\lambda_l^k \tag{16}$$

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

$$\boldsymbol{\Phi} := \boldsymbol{C}\boldsymbol{X}, \qquad \boldsymbol{\Psi} := \boldsymbol{X}^{-1}\boldsymbol{B}, \qquad (\boldsymbol{R}_l)_{ij} := \boldsymbol{\Phi}_{il}\boldsymbol{\Psi}_{lj}. \tag{17}$$

In this setting, we obtain

$$|\boldsymbol{W} - \boldsymbol{W}_N| = \sum_{k>N} \sum_{l=1}^{n} \left|\boldsymbol{R}_l \lambda_l^k\right|. \tag{18}$$

As required by Proposition 1, all eigenvalues $\boldsymbol{\lambda}_l$ of matrix $\boldsymbol{A}$ must be strictly smaller than one in magnitude. 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 (we remind the reader that inequalities and absolute values are considered to be element by element):

$$\begin{aligned} |\boldsymbol{W} - \boldsymbol{W}_N| &\leqslant \sum_{k=N+1}^{\infty} \sum_{l=1}^{n} |\boldsymbol{R}_l| \left|\lambda_l^k\right| \leqslant \sum_{l=1}^{n} |\boldsymbol{R}_l| \frac{\left|\lambda_l^{N+1}\right|}{1 - |\lambda_l|} \\ &= \rho(\boldsymbol{A})^{N+1} \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\lambda_l|} \left(\frac{|\lambda_l|}{\rho(\boldsymbol{A})}\right)^{N+1}. \end{aligned} \tag{19}$$

Since $\frac{|\lambda_l|}{\rho(\boldsymbol{A})} \leqslant 1$ holds for all terms, we may leave out the powers. Notate

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})} \in \mathbb{R}^{p \times q}. \tag{20}$$

The tail of the infinite sum is hence bounded by

$$|\boldsymbol{W} - \boldsymbol{W}_N| \leqslant \rho(\boldsymbol{A})^{N+1} \boldsymbol{M}. \tag{21}$$

B. Deducing a lower bound on the truncation order

In order to get (21) bounded by ε_1, it is required that

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

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

$$N \geqslant \left\lceil \frac{\log \frac{\varepsilon_1}{m}}{\log \rho(\boldsymbol{A})} \right\rceil \tag{22}$$

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

However we cannot compute exact values for all quantities occuring in (22) 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.

C. A rigorous algorithm to determine truncation order

Due to the implementation of (15) and (17) with the finite-precision arithmetic, only approximations on $\boldsymbol{\lambda}, \boldsymbol{X}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{R}_l$ can be obtained. There exist many FP libraries, such as LAPACK[1], providing functions for an eigendecomposition as needed for (15) and to solve linear systems of equations in (17). They usually deliver good and fast approximations to the solution of a given numerical problem but there is neither verification nor guarantee about the accuracy of that approximation.

For these reasons we propose to combine LAPACK FP arithmetic with Interval Arithmetic [3] enhanced with the Theory of Verified Inclusions [14], [15] ,[16], [17] in order to obtain trusted intervals on the eigensystem and, eventually, a rigorous bound on N.

[1]http://www.netlib.org/lapack/

In Interval Arithmetic real numbers are represented as sets of reals with addition, subtraction, multiplication and division defined [3]. The Theory of Verified Inclusions is a set of algorithms computing guaranteed bounds on solutions of various numerical problems, developed by S. Rump [14]. The verification process is performed by means of checking an interval fixed point and yields to a trusted interval for the solution, i.e. it is verified that the result interval contains an exact solution of given numerical problem.

It permits us to quickly obtain trusted error bounds on the truncation order without significant impact on algorithm performance, since this computation is done only once. In addition, if the spectral radius of $\boldsymbol{A}$ cannot be shown less than 1, we stop the algorithm.

Using the ideas proposed by Rump in [17], we obtain trusted intervals for the eigensystem with the following steps:

1) Using the LAPACK eigensolver, we compute FP approximations $\boldsymbol{V}$ for the eigenvectors $\boldsymbol{X}$ and $\boldsymbol{\alpha}$ for the eigenvalues $\boldsymbol{\lambda}$, along with error estimates $\boldsymbol{\varepsilon}_X$ and $\boldsymbol{\varepsilon}_\lambda$. These error estimates are such that $|\boldsymbol{\lambda} - \boldsymbol{\alpha}| \leqslant \boldsymbol{\varepsilon}_\lambda$ and $|\boldsymbol{X} - \boldsymbol{V}| \leqslant \boldsymbol{\varepsilon}_X$ should be not far from the truth.
2) We construct, verify and possibly adjust intervals for $[\boldsymbol{\lambda}] = [\boldsymbol{\alpha} - \boldsymbol{\varepsilon}_\lambda, \boldsymbol{\alpha} + \boldsymbol{\varepsilon}_\lambda]$ and $[\boldsymbol{X}] = [\boldsymbol{V} - \boldsymbol{\varepsilon}_X, \boldsymbol{V} + \boldsymbol{\varepsilon}_X]$ such that for all vectors $\boldsymbol{\lambda}' \in [\boldsymbol{\lambda}]$ there exists a matrix $\boldsymbol{X}' \in [\boldsymbol{X}]$ satisfying $\boldsymbol{A}\boldsymbol{X}' = \boldsymbol{X}' \cdot \mathrm{diag}(\boldsymbol{\lambda}')$ and such that for all matrices $\boldsymbol{X}' \in [\boldsymbol{X}]$ there exists a vector $\boldsymbol{\lambda}' \in [\boldsymbol{\lambda}]$ satisfying $\boldsymbol{A}\boldsymbol{X}' = \boldsymbol{X}' \cdot \mathrm{diag}(\boldsymbol{\lambda}')$. In this process, first intervals for the eigensystem are constructed from the error estimates $\boldsymbol{\varepsilon}_\alpha$ and $\boldsymbol{\varepsilon}_V$ as radii and the approximate solutions $\boldsymbol{V}$ and $\boldsymbol{\alpha}$ as mid-points. Further, these intervals are verified with inclusion algorithms [17]. If the verification does not succeed, the intervals are extended by some small factor and process is repeated until it succeeds or until there exists an eigenvalue interval which contains 1.

For the solution of the linear system of equations (LSE) appearing in (17), the algorithm for interval verification is based on [15] and consists of two steps:

1) Using LAPACK, compute a FP approximation $\boldsymbol{\Omega}$ on the solution of $\boldsymbol{V}\boldsymbol{\Psi} = \boldsymbol{B}$ along with an error estimate $\boldsymbol{\varepsilon}_\Psi$ such that $|\boldsymbol{\Psi} - \boldsymbol{\Omega}| \leqslant \boldsymbol{\varepsilon}_\Psi$ should be not far from the truth.
2) Construct, verify and adjust intervals $[\boldsymbol{\Psi}] = [\boldsymbol{\Omega} - \varepsilon_\Psi, \boldsymbol{\Omega} + \varepsilon_\Psi]$ such that for all matrices $\boldsymbol{X}' \in [\boldsymbol{X}]$ there exists $\boldsymbol{\Psi}' \in [\boldsymbol{\Psi}]$ such that $\boldsymbol{X}'\boldsymbol{\Psi}' = \boldsymbol{B}$ holds.

The intervals for verification are constructed in the same way as for the eigensystem solution. We require the existence of the exact solution of the linear system not for $\boldsymbol{V}\boldsymbol{\Psi} = \boldsymbol{B}$ but for $[\boldsymbol{X}]\boldsymbol{\Psi} = \boldsymbol{B}$, i.e. $[\boldsymbol{\Psi}]$ must contain the exact solution for each element of the already verified interval $[\boldsymbol{X}]$.

Finally, the intervals for (17), (20) and (22) are computed with Interval Arithmetic. Our complete algorithm to determine a reliable lower bound on N is given with algorithm 2.

IV. Summation

Once the truncation order determined, we need to provide a summation scheme reliable in FP arithmetic, i.e. such that the error of computations is bounded by an a priori given value. To

Algorithm 2: Lower bound of truncation order

Input: $\boldsymbol{A} \in \mathbb{F}^{n\times n}, \boldsymbol{B} \in \mathbb{F}^{n\times q}, \boldsymbol{C} \in \mathbb{F}^{p\times n}, \varepsilon_1 > 0$
Output: $N \in \mathbb{N}$

$\boldsymbol{\alpha}, \boldsymbol{V}, \boldsymbol{\varepsilon}_\alpha, \boldsymbol{\varepsilon}_V \leftarrow$ LAPACK eigendecomposition for $\boldsymbol{A}$;
$\boldsymbol{\Omega}, \boldsymbol{\varepsilon}_\Psi \leftarrow$ LAPACK solver for $\boldsymbol{V\Psi} = \boldsymbol{B}$;
$[\lambda], [\boldsymbol{X}] \leftarrow$ Eigensystem verification algorithm;
$[\boldsymbol{\Psi}] \leftarrow$ LSE solution verification algorithm;
$[\boldsymbol{\Phi}] \leftarrow \boldsymbol{C}[\boldsymbol{X}]$;
$[\boldsymbol{R}_l]_{i,j} \leftarrow [\boldsymbol{\Phi}_{i,l}][\boldsymbol{\Psi}_{l,j}]$;
$[\rho] \leftarrow \max\limits_i \left| [\lambda_i] \right|$;
$[\boldsymbol{M}] \leftarrow \sum\limits_{i=1}^{n} \frac{\left|[\boldsymbol{R}_i]\right|}{1-\left|[\lambda_i]\right|} \frac{\left|[\lambda_i]\right|}{[\rho]}$;
$[m] \leftarrow \min\limits_{i,j} \left| [\boldsymbol{M}]_{i,j} \right|$;
$N \leftarrow \sup\left(\left\lceil \frac{\log \frac{\varepsilon_1}{[m]}}{\log [\rho]} \right\rceil\right)$;
return N

do so we propose to perform all operations in multiple precision arithmetic whilst adapting precision dynamically where needed. Several multiple precision algorithms were therefore developed:

- multiplyAndAdd$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \delta)$ that computes $\boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{C} + \boldsymbol{\Delta}$, where the error matrix $\boldsymbol{\Delta}$ is bounded by $|\boldsymbol{\Delta}| < \delta$, for the given a priori bound δ. We shall notate $\boldsymbol{A} \otimes \boldsymbol{B}$ for the output of multiplyAndAdd when $\boldsymbol{C}$ is the zero matrix.
- sumAbs$(\boldsymbol{A}, \boldsymbol{B}, \delta)$ that computes $\boldsymbol{A} + |\boldsymbol{B}| + \boldsymbol{\Delta}$, where the error matrix $\boldsymbol{\Delta}$ is bounded by $|\boldsymbol{\Delta}| < \delta$, for the given δ. With a slight notational abuse, we shall also notate $\boldsymbol{A} \oplus \text{abs}(\boldsymbol{B})$ for sumAbs.
- inv$(\boldsymbol{V}, \delta)$ that computes the inverse $\boldsymbol{V}^{-1} + \boldsymbol{\Delta}$, where the error matrix $\boldsymbol{\Delta}$ is bounded by $|\boldsymbol{\Delta}| < \delta$, for the given δ. See Sec. V.

These computation kernels adapt the precision of their intermediate computations where needed. The algorithms we use for these basic bricks will be discussed in Sec. V.

A. Step 2: using the Eigendecomposition

1) Error propagation: As seen, in each step of the summation, a matrix power, $\boldsymbol{A}^k$, must be computed. In [6] Higham devotes an entire 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.

However, the state matrices $\boldsymbol{A}$ usually have a good structure. In real life 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 $\boldsymbol{A} = \boldsymbol{XEX}^{-1}$. Then $\boldsymbol{A}^k = \boldsymbol{XE}^k\boldsymbol{X}^{-1}$. A good choice of $\boldsymbol{X}$ and $\boldsymbol{E}$ are the eigenvector and eigenvalue matrices obtained with eigendecomposition (15). However, with LAPACK we can compute only approximations on them and we cannot control their accuracy. Therefore, we propose following method to *almost* diagonalize matrix $\boldsymbol{A}$. The method does not make any assumptions on matrix $\boldsymbol{V}$ except for it being *some* approximation on $\boldsymbol{X}$. Therefore, for simplicity of further reasoning we treat $\boldsymbol{V}$ as an exact matrix.

Using our multiprecision algorithms for matrix inverse and multiplication we may compute a complex $n \times n$ matrix $\boldsymbol{T}$:

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{AV} - \boldsymbol{\Delta}_2, \tag{23}$$

where $\boldsymbol{V} \in \mathbb{C}^{n\times n}$ is an approximation on $\boldsymbol{X}$, $\boldsymbol{\Delta}_2 \in \mathbb{C}^{n\times n}$ 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 full matrix. However it has prevailing elements on the main diagonal. Thus $\boldsymbol{T}$ is an approximation on $\boldsymbol{E}$.

We require for matrix $\boldsymbol{T}$ to satisfy $\|\boldsymbol{T}\|_2 \leqslant 1$. This condition is stronger than $\rho(\boldsymbol{A}) < 1$, and Sec. IV-A2 provides a way to test it. Naturally this condition means that there exist some margin for computational errors between the spectral radius and 1.

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

$$\boldsymbol{A}^k = \boldsymbol{V}(\boldsymbol{T} + \boldsymbol{\Delta}_2)^k\boldsymbol{V}^{-1}, \tag{24}$$

therefore

$$\boldsymbol{CA}^k\boldsymbol{B} = \boldsymbol{CVT}^k\boldsymbol{V}^{-1}\boldsymbol{B} + \boldsymbol{CV\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}. \tag{25}$$

Thus the error of computing $\boldsymbol{VT}^k\boldsymbol{V}^{-1}$ instead of $\boldsymbol{A}^k$ in (7) is bounded by

$$\left|\sum_{k=0}^{N}\left|\boldsymbol{CA}^k\boldsymbol{B}\right| - \sum_{k=0}^{N}\left|\boldsymbol{CVT}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right|\right| \leqslant \tag{26}$$

$$\sum_{k=0}^{N}\left|\boldsymbol{CA}^k\boldsymbol{B} - \boldsymbol{CVT}^k\boldsymbol{V}^{-1}\boldsymbol{B}\right| \leqslant \sum_{k=0}^{N}\left|\boldsymbol{CV\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}\right| \tag{27}$$

Here and further on each step of the algorithm we use inequalities with left side in form (27) rather than (26), i.e. we will instantly use the triangulation property $\big||a| - |b|\big| \leqslant |a - b|$ $\forall a, b$ applied element-by-element to matrices.

In order to determine the accuracy of the computations on this step such that (27) 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:

$$\|\boldsymbol{\Xi}_k\|_2 \leqslant \|\boldsymbol{\Xi}_{k-1}\|_2 + \|\boldsymbol{\Delta}_2\|_2\,(\|\boldsymbol{\Xi}_{k-1}\|_2 + 1) \tag{28}$$

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

$$\|\boldsymbol{\Xi}_k\|_2 \leqslant \|\boldsymbol{\Xi}_{k-1}\|_2 + 2\,\|\boldsymbol{\Delta}_2\|_2 \tag{29}$$

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:

$$\|\boldsymbol{\Xi}_k\|_F \leqslant 2\sqrt{n}(k+1)\,\|\boldsymbol{\Delta}_2\|_F\,. \tag{30}$$

Substituting this bound to (27) and folding the sum, we obtain

$$\sum_{i=0}^{N}\left|\boldsymbol{CV\Xi}_k\boldsymbol{V}^{-1}\boldsymbol{B}\right| \leqslant \beta\,\|\boldsymbol{\Delta}_2\|_F\,\|\boldsymbol{CV}\|_F\,\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F, \tag{31}$$

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

$$\|\boldsymbol{\Delta}_2\|_F \leqslant \frac{1}{\beta} \frac{\varepsilon_2}{\|\boldsymbol{CV}\|_F \left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F}. \tag{32}$$

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.

2) Checking $\|\boldsymbol{T}\|_2 \leqslant 1$: 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 [5], 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). Denote $\boldsymbol{Y} := \boldsymbol{T}^*\boldsymbol{T} - \boldsymbol{F}^*\boldsymbol{F} = \boldsymbol{F}^*\boldsymbol{G} + \boldsymbol{G}^*\boldsymbol{F} + \boldsymbol{G}^*\boldsymbol{G}$. Then

$$\begin{aligned}\sum_{j\neq i} \left|(\boldsymbol{T}^*\boldsymbol{T})_{ij}\right| &= \sum_{j\neq i} \left|\boldsymbol{Y}_{ij}\right| \\ &\leqslant (n-1)\,\|\boldsymbol{Y}\|_F \\ &\leqslant (n-1)\left(2\,\|\boldsymbol{F}\|_F\,\|\boldsymbol{G}\|_F + \|\boldsymbol{G}\|_F^2\right) \\ &\leqslant (n-1)\left(2\sqrt{n} + \|\boldsymbol{G}\|_F\right)\|\boldsymbol{G}\|_F. \end{aligned} \tag{33}$$

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 $(\boldsymbol{Y})_{ii}$ of $\boldsymbol{Y}$ are equal to the diagonal elements $(\boldsymbol{G}^*\boldsymbol{G})_{ii}$ of $\boldsymbol{G}^*\boldsymbol{G}$. They can hence be bounded as follows:

$$|(\boldsymbol{Y})_{ii}| = |(\boldsymbol{G}^*\boldsymbol{G})_{ii}| \leqslant \|\boldsymbol{G}\|_F^2. \tag{34}$$

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

$$\forall i, \quad |(\boldsymbol{F}^*\boldsymbol{F})_{ii}| \leqslant 1 - \|\boldsymbol{G}\|_F^2 - \gamma, \tag{35}$$

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

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

After computing $\boldsymbol{T}$ out of $\boldsymbol{V}$ and $\boldsymbol{A}$ according to (23), the condition on $\boldsymbol{T}$ should be tested in order to determine if $\|\boldsymbol{T}\|_2 \leqslant 1$. This test failing means that $\boldsymbol{V}$ is not a sufficient approximate of $\boldsymbol{X}$ or that the error $\boldsymbol{\Delta}_2$ done computing (23) 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 real-world examples we tested, never saw it fail.

B. Step 3: computing $\boldsymbol{CV}$ and $\boldsymbol{V}^{-1}\boldsymbol{B}$

We compute approximations on 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 impact of these approximations is less than ε_3.

Notate $\boldsymbol{C}' := \boldsymbol{CV} + \boldsymbol{\Delta}_{3_C}$ 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 Frobenius norm, we can bound the error in the approximation of $\boldsymbol{CV}$ and $\boldsymbol{V}^{-1}\boldsymbol{B}$ by $\boldsymbol{C}'$ and $\boldsymbol{B}'$ as follows:

$$\begin{aligned}&\sum_{k=0}^{N} \left|\boldsymbol{CVT}^k\boldsymbol{V}^{-1}\boldsymbol{B} - \boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}'\right| \leqslant \\ &\sum_{k=0}^{N} \left\|\boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{B}' + \boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B} + \boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B}\right\|_F. \end{aligned} \tag{36}$$

Since $\|\boldsymbol{T}\|_2 < 1$ holds we have (using Frobenius norm properties)

$$\begin{aligned}&\left\|\boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{B}' + \boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B} + \boldsymbol{\Delta}_{3_C}\boldsymbol{T}^k\boldsymbol{\Delta}_{3_B}\right\|_F \leqslant \\ &\sqrt{n}\left(\left\|\boldsymbol{\Delta}_{3_C}\right\|_F\left(\left\|\boldsymbol{B}'\right\|_F + \left\|\boldsymbol{\Delta}_{3_B}\right\|_F\right) + \left\|\boldsymbol{C}'\right\|_F\left\|\boldsymbol{\Delta}_{3_B}\right\|_F\right). \end{aligned} \tag{37}$$

This bound represents the impact of our approximations for each $k = 0 \dots N$. If (37) 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:

$$\left\|\boldsymbol{\Delta}_{3_C}\right\|_F \leqslant \frac{1}{3\sqrt{n}} \cdot \frac{1}{N+1}\frac{\varepsilon_3}{\|\boldsymbol{C}'\|_F} \tag{38}$$

$$\left\|\boldsymbol{\Delta}_{3_B}\right\|_F \leqslant \frac{1}{3\sqrt{n}} \cdot \frac{1}{N+1}\frac{\varepsilon_3}{\|\boldsymbol{B}'\|_F}. \tag{39}$$

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.

C. Step 4: powering $\boldsymbol{T}$

Given a square complex matrix $\boldsymbol{T}$ with prevailing main diagonal we need to compute its k^{th} power. Notate

$$\boldsymbol{P}_k := \boldsymbol{T}^k - \boldsymbol{\Pi}_k, \tag{40}$$

where $\boldsymbol{\Pi}_k \in \mathbb{C}^{n\times n}$ represents element-by-element the error on the matrix powers, including error propagation from the first to the last power. Using the same simplification as in (26) and (27) we get the error of computing the approximations $\boldsymbol{P}_k$ rather than the exact powers bounded by

$$\sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{T}^k\boldsymbol{B}' - \boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}'\right| \leqslant \sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{\Pi}_k\boldsymbol{B}'\right|. \tag{41}$$

Thus a bound on a norm of $\boldsymbol{\Pi}_k$, say $\|\boldsymbol{\Pi}_k\|_F$, is required.

Since we need all the powers of $\boldsymbol{T}$ from 1 to N, we use an iterative scheme to compute them. It is then evident, that we may write the recurrence

$$\boldsymbol{P}_k = \boldsymbol{T}\boldsymbol{P}_{k-1} + \boldsymbol{\Gamma}_k, \tag{42}$$

where $\boldsymbol{\Gamma}_k \in \mathbb{C}^{n\times n}$ is the error matrix representing the error of the matrix multiplication at step k.

With $\boldsymbol{P}_0 = \boldsymbol{I}$, $\boldsymbol{P}_1 = \boldsymbol{T}$ and using (42) we obtain

$$\boldsymbol{P}_k = \boldsymbol{T}^k + \sum_{l=2}^{k} \boldsymbol{T}^{k-l}\boldsymbol{\Gamma}_l. \tag{43}$$

Using the condition $\|\boldsymbol{T}\|_2 \leqslant 1$ and properties of the Frobenius norm we get

$$\|\boldsymbol{\Pi}_k\|_F \leqslant \left\|\sum_{l=2}^{k} \boldsymbol{T}^{k-l}\boldsymbol{\Gamma}_l\right\|_F \leqslant \sqrt{n}\sum_{l=2}^{k} \|\boldsymbol{\Gamma}_l\|_F. \tag{44}$$

Therefore the impact of approximation of the matrix powers is bounded by

$$\sum_{k=0}^{N} \left|\boldsymbol{C}'\boldsymbol{\Pi}_k\boldsymbol{B}'\right| \leqslant \sqrt{n}(N+1)\sum_{l=2}^{N} \left\|\boldsymbol{C}'\right\|_F \|\boldsymbol{\Gamma}_l\|_F \left\|\boldsymbol{B}'\right\|_F. \quad (45)$$

Obviously, if the error of matrix multiplication $\boldsymbol{\Gamma}_l$ satisfies

$$\|\boldsymbol{\Gamma}_l\|_F \leqslant \frac{1}{\sqrt{n}} \cdot \frac{1}{N-1} \cdot \frac{1}{N+1} \cdot \frac{\varepsilon_4}{\|\boldsymbol{C}'\|_F \|\boldsymbol{B}'\|_F} \quad (46)$$

for $l = 2 \dots N$, then we have (45) to be less than ε_4. Hence using (46) we may deduce the required accuracy of matrix multiplications on each step in dependency of ε_4.

D. Step 5: computing $\boldsymbol{L}_k$

Once the matrices $\boldsymbol{C}', \boldsymbol{B}'$ and $\boldsymbol{P}_k$ are pre-computed and the error of their computation is bounded, we must evaluate their product. Let $\boldsymbol{L}_k$ be the approximate product of these three matrices at step k:

$$\boldsymbol{L}_k := \boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}' + \boldsymbol{\Upsilon}_k, \quad (47)$$

where $\boldsymbol{\Upsilon}_k \in \mathbb{C}^{p\times q}$ is the matrix of element-by-element errors for the two matrix multiplications. Then the error of computations induced by this step is bounded by $\sum_{k=0}^{N} |\boldsymbol{\Upsilon}_k|$.

If we want the overall error of approximation on this step to be less than ε_5 then we can easily deduce the required accuracy of each of those multiplications on every iteration of summation algorithm.

E. Step 6: final summation

Finally the absolute value of the $\boldsymbol{L}_k$ must be taken and the result accumulated in the sum. We remind the reader that if all previous computations were exact, the matrix $\boldsymbol{L}_k$ would be a real matrix and the absolute-value-operation would have been an exact sign manipulation. However, as the computations were in finite-precision arithmetic, $\boldsymbol{L}_k$ is complex with a small imaginary part, which is naturally caused by the errors of computations and must not be neglected. Therefore the element-by-element absolute value of the matrix must be computed.

Since we perform $N+1$ accumulations of absolute values in the result sum $\boldsymbol{S}_N$, it is evident that bounding the error of each such computation by $\frac{1}{N+1}\varepsilon_6$ is sufficient.

Therefore, using this bound for each invokation of our basic brick algorithm sumAbs we guarantee bound (13).

V. Basic bricks

In Sec. IV, we postulated the existence of three basic FP algorithms, multiplyAndAdd, sumAbs and inv, computing, respectively, the product-sum, the sum in absolute value and the inverse of matrices. Each of these operators was required to satisfy an absolute error bound $|\boldsymbol{\Delta}| < \delta$ to be ensured by the matrix of errors $\boldsymbol{\Delta}$ 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 FP arithmetic is used. Any such algorithm, when returning its result, must round into that fixed-precision FP format. Hence, when the output grows sufficiently large, the unit-in-the-last-place of that format and hence that final rounding error in fixed-precision FP arithmetic will grow larger than a set absolute error bound.

In multiple precision FP arithmetic, such as offered by software packages like MPFR[2] [4], it is however possible to have algorithms determine themselves the output precision of the FP variables they return their results in. Hence an absolute error bound as the one we require can be guaranteed. In contrast to classical FP arithmetic, such as Higham analyzes, there is no longer any clear, overall *computing precision*, though. Variables just bear the precision that had been determined for them by the previous compute step.

This preliminary clarification made, description of our three basic bricks multiplyAndAdd, sumAbs and inv is easy.

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

For 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 FP summation with an absolute error bound can be performed with a modified, software-simulated Kulisch accumulator [10], 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 [13]:

$$\begin{aligned} \boldsymbol{U}_0 &\leftarrow \text{some seed inverse matrix for } \boldsymbol{V}^{-1} \\ \boldsymbol{R}_k &\leftarrow \boldsymbol{V}\boldsymbol{U}_k - \boldsymbol{I}_n \\ \boldsymbol{U}_{k+1} &\leftarrow \boldsymbol{U}_k - \boldsymbol{U}_k\boldsymbol{R} \end{aligned} \quad (48)$$

where the iterated matrices $\boldsymbol{U}_k$ converge to $\boldsymbol{V}^{-1}$ provided the multiplyAndAdd operations computing $\boldsymbol{R}_k$ and $\boldsymbol{U}_{k+1}$ are performed with enough accuracy, i.e. small enough δ and the seed matrix satisfies some additional properties. In order to ensure these properties with an explicit check, an operator to compute the Frobenius norm of a matrix with a given a priori absolute error bound δ is required. Implementing such a Frobenius norm operator again boils down to summation, as above. See the extended version of this paper, available at http://hal.upmc.fr/hal-01083879 for details.

VI. Numerical examples

The algorithms discussed above were implemented in C, using GNU MPFR version 3.1.12, GNU MPFI[3] version 1.5.1 and CLAPACK[4] version 3.2.1. Our implementation was tested on several real-life and random examples:

[2] http://www.mpfr.org/
[3] https://gforge.inria.fr/projects/mpfi/
[4] http://www.netlib.org/clapack/

	Example 1			Example 2			Example 3			Example 4		
sizes n, p and q	$n = 10$,	$p = 11$,	$q = 1$	$n = 12$,	$p = 1$,	$q = 25$	$n = 60$,	$p = 28$,	$q = 14$	$n = 3$,	$p = 1$,	$q = 4$
$1 - \rho(\boldsymbol{A})$		1.39×10^{-2}			8.65×10^{-3}			1.46×10^{-2}			2^{-60}	
$\max(\boldsymbol{S}_N)$		3.88×10^{1}			5.50×10^{9}			2.64×10^{2}			-	
$\min(\boldsymbol{S}_N)$		1.29×10^{0}			1.0×10^{0}			1.82×10^{1}			-	
ε	2^{-5}	2^{-53}	2^{-600}	2^{-5}	2^{-53}	2^{-600}	2^{-5}	2^{-53}	2^{-600}		2^{-5}	
N	220	2153	29182	308	4141	47811	510	1749	27485		-	
Inversion iterations	0	2	4	2	3	5	1	2	4		-	
overall max precision (bits)	212	293	1401	254	355	1459	232	306	1416		-	
Overall execution time (sec)	0.11	1.53	60.06	0.85	11.54	473.20	45.62	177.90	9376.86		0.00...	

TABLE I

Numerical results for 3 real-world and 1 constructed example

- The first example comes from Control Theory: the LTI system is extracted from an active controller of vehicle longitudinal oscillation [11], and WCPG matrix is used to determine the fixed-point arithmetic scaling of state and output.
- The second is a 12th-order Butterworth filter, described in ρ-Direct Form II transposed [18] (a particular algorithm, with low complexity and high robustness to quantization and computational errors), where the errors-to-output LTI system $\mathcal{H}_e$ is considered (see Figure 1).
- The third one is a large random BIBO stable filter (obtained from the `drss` command of Matlab), with 60 states, 14 inputs and 28 outputs.
- The last one is built with a companion matrix $\boldsymbol{A}$ with spectral radius equal to $1 - 2^{-60}$.

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

The numerical results detailed in Table I show that our algorithm for Worst-Case Peak Gain matrix evaluation with a priori error bound exhibits reasonable performance on typical examples. Even when the a priori error bound is pushed to compute WCPG results with an accuracy way beyond double precision, the algorithm succeeds in computing a result, even though execution time grows pretty high.

Our algorithm includes checks testing that certain properties of matrices are verified, in particular that $\rho(\boldsymbol{A}) < 1$ and $\|\boldsymbol{T}\|_2 \leqslant 1$. Our Example 4, not taken from a real-word application but constructed on purpose, shows that the algorithm correctly detects that the conditions are not fulfilled for that example.

VII. Conclusions

With this paper, a reliable, rigorous multiprecision method to compute the Worst-Case Peak Gain matrix has been developed. It relies on Theory of Verified Inclusion, eigenvalue decomposition to perform matrix powering, some multiple-precision arithmetic basic bricks developed to satisfy absolute error bounds and a detailed step-by-step error analysis.

A `C` program has been developed and now can be used as a tool for the implementation error analysis of LTI systems, and then the design of reliable finite precision digital algorithms for signal processing and control.

However, some efforts are still required to overcome double precision eigenvalue decomposition in LAPACK (specially for close-to-instability LTI systems) by using a multiple precision eigensolver. Additionnally, as the proofs on the error bounds are pretty complicated, they should be formalized in a Formal Proof Checker, such as Coq or HolLight.

VIII. Acknowledgments

The authors are grateful to M. Mezzarobba and S. Graillat for interesting discussions on topics related to this work. This work was partly supported by the *Agence nationale de la recherche* grant ANR-13-INSE-0007-02 MetaLibm.

References

[1] V. Balakrishnan and S. Boyd. On computing the worst-case peak gain of linear systems. *Systems & Control Letters*, 19:265–269, 1992.

[2] J. Carletta, R. Veillette, F. Krach, and Zhengwei F. Determining appropriate precisions for signals in fixed-point iir filters. In *Design Automation Conference, 2003. Proceedings*, pages 656–661, 2003.

[3] H. Dawood. *Theories of Interval Arithmetic: Mathematical Foundations and Applications*. LAP Lambert Academic Publishing, 2011.

[4] 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*, 33(2):13:1–13:15, 2007.

[5] S. Gershgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Bull. Acad. Sci. URSS*, 1931(6):749–754, 1931.

[6] N. J. Higham. *Accuracy and stability of numerical algorithms (2. ed.)*. SIAM, 2002.

[7] T. Hilaire, P. Chevrel, and J.F. Whidborne. A unifying framework for finite wordlength realizations. *IEEE Trans. on Circuits and Systems*, 8(54):1765–1774, 2007.

[8] T. Hilaire and B. Lopez. Reliable implementation of linear filters with fixed-point arithmetic. In *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, 2013.

[9] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[10] U. Kulisch and V. Snyder. The exact dot product as basic tool for long interval arithmetic. *Computing*, 91(3):307–313, March 2011.

[11] D. Lefebvre, P. Chevrel, and S. Richard. An H_∞ based control design methodology dedicated to the active control of longitudinal oscillations. *IEEE Trans. on Control Systems Technology*, 11(6):948–956, 2003.

[12] 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*, 26(11):1923–1933, 2007.

[13] V. Pan and J. Reif. Efficient parallel solution of linear systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 143–152. ACM, 1985.

[14] S. M. Rump. New results on verified inclusions. In *Accurate Scientific Computations, Symposium, 1985, Proceedings*, pages 31–69, 1985.

[15] S. M. Rump. Solution of linear systems with verified accuracy. *Applied numerical mathematics*, 3(3):233–241, 1987.

[16] S. M. Rump. Reliability in computing: The role of interval methods in scientific computing. chapter Algorithms for Verified Inclusions — Theory and Practice, pages 109–126. Academic Press, 1988.

[17] S. M. Rump. Guaranteed inclusions for the complex generalized eigenproblem. *Computing*, 42(2-3):225–238, September 1989.

[18] Z. Zhao and G. Li. Roundoff noise analysis of two efficient digital filter structures. *IEEE Trans. on Signal Processing*, 54(2):790–795, 2006.