NEW L_2-DYNAMIC-RANGE-SCALING CONSTRAINTS FOR LOW PARAMETRIC SENSITIVITY REALIZATIONS

Thibault Hilaire

Vienna University of Technology, Austria
Institute of Communications and Radio-Frequency Engineering
thibault.hilaire@nt.tuwien.ac.at

ABSTRACT

This paper presents a new dynamic-range scaling for the implementation of filters/controllers in state-space form. Specific fixed-point considerations allow us to relax the classical L_2-scaling constraints while still preserving the implementation from overflows. It gives more degrees of freedom for the optimal L_2-parametric sensitivity problem. The underlying constrained problem is converted into an unconstrained problem for which a solution can be provided. This leads to realizations which are still scaled but less sensitive.

1. INTRODUCTION

The majority of signal processing (or control) systems is implemented in digital general purpose processors, DSPs[1], FPGAs[2], etc. Since these devices cannot compute with infinite precision and approximate real-number parameters with a finite binary representation, the numerical implementation of controllers (filters) leads to deterioration in characteristics and performance. This has two separate origins, corresponding to the quantization of the embedded coefficients and the roundoff errors occurring during the computations. They can be formalized as parametric errors and numerical noises, respectively. The focus of this paper are parametric errors, but one can refer to [3, 6, 10, 14] for roundoff noises.
It is also well known that these Finite Word Length (FWL) effects depend on the structure of the realization. This motivates to investigate the coefficient sensitivity minimization problem. It has been widely studied since Thiele published [16,17] and the definition of a tractable input-output sensitivity norm (the L_1/L_2-sensitivity). This work has been extended with a more natural and reasonable measure, the L_2-sensitivity ([3, 8]).
The dynamic-range-scaling constraints have been introduced in [11] and [9] to prevent overflow and underflow during the evaluation of the state-vector, and as well as the state and criteria normalization. These constraints have to be considered in the L_2-sensitivity minimization problem, for which [7] proposes an efficient quasi-Newton algorithm to solve it.

This paper investigates the L_2-dynamic-range-scaling problem by considering concrete fixed-point implementation of state-space realizations. It reveals that the classical L_2-scaling is only a sufficient condition to prevent overflows and thus it can be slightly relaxed in order to extend the degrees of freedom for the optimization process. New *relaxed-L_2*-dynamic-range-scalings are then presented with respect to the described computational scheme. Finally, the L_2-sensitivity minimization problem with relaxed L_2-scaling constraints is solved. A numerical example illustrates that the proposed constraints can offer reduced L_2-sensitivity with overflow protection.

This work has been funded in parts by the NFN SISE project (National Research Network "Signal and Information Processing in Science and Engineering").

[1] Digital Signal Processors

[2] Field Programmable Gate-Array

2. L_2-SENSITIVITY ANALYSIS

Let $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, d)$ be a stable, controllable and observable linear discrete time SISO[3] state-space system, i.e.

$$\left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{b}u(k) \\ y(k) & = & \boldsymbol{c}\boldsymbol{x}(k) + du(k) \end{array} \right. \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{c} \in \mathbb{R}^{1\times n}$ and $d \in \mathbb{R}$. $u(k)$ is the scalar input, $y(k)$ is the scalar output and $\boldsymbol{x}(k) \in \mathbb{R}^{n\times 1}$ is the state vector. Its input-output relationship is given by the scalar transfer function $h : \mathbb{C} \to \mathbb{C}$ defined by:

$$h : z \mapsto \boldsymbol{c}(z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b} + d. \tag{2}$$

The quantization of the coefficients introduces some uncertainly to $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and d leading to $\boldsymbol{A} + \boldsymbol{\Delta A}$, $\boldsymbol{b} + \boldsymbol{\Delta b}$, $\boldsymbol{c} + \boldsymbol{\Delta c}$ and $d + \Delta d$ respectively. It is of interest to consider the sensitivity of the transfer function with respect to the coefficients, based on the following definitions.

Definition 1 (Transfer function sensitivity) *Consider $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ a matrix and $f : \mathbb{R}^{m\times n} \to \mathbb{C}$ a scalar complex function, differentiable with respect to all the entries of $\boldsymbol{X}$.*
The sensitivity of f with respect to $\boldsymbol{X}$ is defined by the matrix $\boldsymbol{S}_{\boldsymbol{X}} \in \mathbb{R}^{m\times n}$:

$$\frac{\partial f}{\partial \boldsymbol{X}} \triangleq \boldsymbol{S}_{\boldsymbol{X}} \qquad with \qquad (\boldsymbol{S}_{\boldsymbol{X}})_{i,j} \triangleq \frac{\partial f}{\partial \boldsymbol{X}_{i,j}} \tag{3}$$

Definition 2 (L_p-Norm) *Let $\boldsymbol{H} : \mathbb{C} \to \mathbb{C}^{k\times l}$ be a function of the scalar complex variable z. $\|\boldsymbol{H}\|_p$ is the L_p-norm of $\boldsymbol{H}$, defined by:*

$$\|\boldsymbol{H}\|_p \triangleq \left(\frac{1}{2\pi} \int_0^{2\pi} \left\| \boldsymbol{H}\left(e^{j\omega}\right) \right\|_F^p d\omega \right)^{\frac{1}{p}} \tag{4}$$

where $\|.\|_F$ is the Froebenius norm.

Gevers and Li [3] have proposed the L_2-sensitivity measure to evaluate the coefficient roundoff errors. It is defined by

$$M_{L_2} \triangleq \left\| \frac{\partial h}{\partial \boldsymbol{A}} \right\|_2^2 + \left\| \frac{\partial h}{\partial \boldsymbol{b}} \right\|_2^2 + \left\| \frac{\partial h}{\partial \boldsymbol{c}} \right\|_2^2 + \left\| \frac{\partial h}{\partial d} \right\|_2^2 \tag{5}$$

and can be computed by $\frac{\partial h}{\partial \boldsymbol{A}}(z) = \boldsymbol{G}^\top(z)\boldsymbol{F}^\top(z)$, $\frac{\partial h}{\partial \boldsymbol{b}}(z) = \boldsymbol{G}^\top(z)$, $\frac{\partial h}{\partial \boldsymbol{c}}(z) = \boldsymbol{F}(z)$ and $\frac{\partial h}{\partial d}(z) = 1$, with

$$\boldsymbol{F}(z) \triangleq (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b}, \qquad \boldsymbol{G}(z) \triangleq \boldsymbol{c}(z\boldsymbol{I}_n - \boldsymbol{A})^{-1}. \tag{6}$$

This measure is an extension of the more tractable but less natural L_1/L_2 sensitivity measure proposed by V. Tavşanoğlu and L. Thiele [16] ($\left\| \frac{\partial h}{\partial \boldsymbol{A}} \right\|_1^2$ instead of $\left\| \frac{\partial h}{\partial \boldsymbol{A}} \right\|_2^2$ in (5)).

[3] Single Input Single Output

Applying a coordinate transformation, defined by $\bar{\boldsymbol{x}}(k) \triangleq \boldsymbol{T}^{-1}\boldsymbol{x}(k)$ to the state-space system $(\boldsymbol{A},\boldsymbol{b},\boldsymbol{c},d)$, leads to a new equivalent realization $(\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T},\boldsymbol{T}^{-1}\boldsymbol{b},\boldsymbol{c}\boldsymbol{T},d)$.

Since these two realizations are equivalent in infinite precision but are no more equivalent in finite precision (fixed point arithmetic, floating-point arithmetic, etc.), the L_2-sensitivity then depends on $\boldsymbol{T}$, and is denoted $M_{L_2}(\boldsymbol{T})$.
In this case, it is natural to define the following problem:

Problem 1 (optimal L_2-sensitivity problem) *Considering a state-space realization $(\boldsymbol{A},\boldsymbol{b},\boldsymbol{c},d)$, the optimal L_2-sensitivity problem consists of finding the coordinate transformation $\boldsymbol{T}_{opt}$ that minimizes M_{L_2}:*

$$\boldsymbol{T}_{opt} = \underset{\boldsymbol{T}\ invertible}{\arg\min}\ M_{L_2}(\boldsymbol{T}). \tag{7}$$

[3] shows that the problem has one unique solution. Hence, for example, a gradient method can be used to solve it.

3. L_P-DYNAMIC-RANGE SCALING

The L_p-dynamic-range-scaling constraints have been introduced by Jackson in [11] and Hwang in [9]. It consists in scaling the state-variable vector such that overflows or underflows during its evaluation are prevented.

Definition 3 (L_p-scaling) *A state-space realization $(\boldsymbol{A},\boldsymbol{b},\boldsymbol{c},\ d)$ is said to be L_p-**scaled** if the L_p-norms of the transfer functions from the input to each state are set to 1,* i.e.*:*

$$\left\| \boldsymbol{e}_i^\top (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b}) \right\|_p = 1, \quad \forall 1 \leqslant i \leqslant n \tag{8}$$

where $\boldsymbol{e}_i$ is the column vector of appropriate dimension and with all elements being 0 except from the i^{th} element which is 1.

Let $\overset{\max}{u}$ denote the maximum value of the input u:

$$\overset{\max}{u} \triangleq \max_{k\in\mathbb{N}} |u(k)|. \tag{9}$$

The L_1-scaling guarantees that the dynamic of each state $\boldsymbol{x}_i$ is lower than $\overset{\max}{u}$, whereas the L_2-scaling guarantees that the variance of each state is unitary for a unit-variance centered white noise input. L_2-scaling doesn't completely prevent overflow as does L_1, but it is less conservative and more realistic, so it is widely used [15].

Proposition 1 *The L_2-norms of the transfer functions from the input to each state are given by*

$$\left\| \boldsymbol{e}_i^\top (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b}) \right\|_2 = \sqrt{(\boldsymbol{W}_c)_{i,i}} \tag{10}$$

where $\boldsymbol{W}_c$ is the controllability Gramian of the state-space system $(\boldsymbol{A},\boldsymbol{b},\boldsymbol{c},\ d)$. This matrix is the solution of the Lyapunov equation:

$$\boldsymbol{W}_c = \boldsymbol{A}\boldsymbol{W}_c\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{b}^\top. \tag{11}$$

Proof:
A classical result on L_2-norm gives the L_2-norm of a state-space system $H := (\boldsymbol{K},\boldsymbol{L},\boldsymbol{M},\boldsymbol{N})$:

$$\|\boldsymbol{H}\|_2^2 = tr(\boldsymbol{N}\boldsymbol{N}^\top + \boldsymbol{M}\boldsymbol{W}_c\boldsymbol{M}^\top) \tag{12}$$

where $tr(.)$ is the trace operator and $\boldsymbol{W}_c$ is the controllability Gramian of the system.
This is here applied to the system $(\boldsymbol{A},\boldsymbol{b},\boldsymbol{e}_i^\top,\boldsymbol{0})$. ∎

Problem 2 (sensitivity problem with L_2-scaling constraints) *The optimal L_2-sensitivity problem with L_2-norm dynamic-range-scaling constraints can be formulated as the optimization problem 1, subject to the constraints*

$$(\boldsymbol{W}_c)_{i,i} = 1, \quad \forall 1 \leqslant i \leqslant n. \tag{13}$$

This constrained problem can be transformed into a unconstrained problem, and an optimization algorithm can be used to solve it [7].

4. FIXED-POINT IMPLEMENTATION

4.1 Fixed-point representation

In this paper, the notation (β,γ) is used for the fixed-point representation of a variable or coefficient (2's complement scheme), according to Figure 1. β is the total wordlength of the representation in bits, whereas γ is the wordlength of the fractional part (it determines the position of the binary-point). They are fixed for each variable (input, states, output) and each coefficient, and implicit (unlike the floating-point representation). β and γ will be suffixed by the variable/coefficient they refer to.

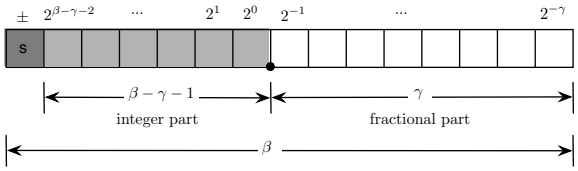

Figure 1: Fixed-point representation

To represent a value x without overflow, a fixed-point representation (β_x,γ_x) may satisfy:

$$\beta_x - \gamma_x - 1 \geqslant \lfloor \log_2 |x| \rfloor + 1 \tag{14}$$

where the $\lfloor a \rfloor$ operation rounds a to the nearest integer less or equal to a (for positive numbers $\lfloor a \rfloor$ is the integer part).

An important fixed-point issue is to find a valid fixed-point representation, such that (14) is satisfied for all values which x can assume during the execution of the algorithm.

4.2 State-overflow

Definition 4 (State-overflow) *The overflow of the state variables $(\boldsymbol{x}_i)_{1\leqslant i\leqslant n}$ can be strictly avoided if all the values $\boldsymbol{x}_i(k)$ are within the range allowed by the fixed-point representations,* i.e. $(1 \leqslant i \leqslant n)$

$$\forall k, \qquad -2^{\beta_{\boldsymbol{x}_i}-\gamma_{\boldsymbol{x}_i}-1} \leqslant \boldsymbol{x}_i(k) < 2^{\beta_{\boldsymbol{x}_i}-\gamma_{\boldsymbol{x}_i}-1}. \tag{15}$$

The overflows are avoided if the binary-point position of each state is carefully chosen, such that

$$\gamma_{\boldsymbol{x}_i} = \beta_{\boldsymbol{x}_i} - 2 - \left\lfloor \log_2 \overset{\max}{\boldsymbol{x}_i} \right\rfloor, \tag{16}$$

where $\overset{\max}{\boldsymbol{x}_i}$ is the maximum magnitude for the i^{th} state:

$$\overset{\max}{\boldsymbol{x}_i} \triangleq \max_{k\in\mathbb{N}} |\boldsymbol{x}_i(k)|. \tag{17}$$

Remark 1 At least, $\gamma_{\boldsymbol{x}_i}$ should satisfy $\gamma_{\boldsymbol{x}_i} \leqslant \beta_{\boldsymbol{x}_i} - 2 - \left\lfloor \log_2 \overset{\max}{\boldsymbol{x}_i} \right\rfloor$, but the greater $\gamma_{\boldsymbol{x}_i}$ is, the more accurate is the fixed-point format.

However only upper bounds can be computed. A first upper bound $\overset{\text{up}}{\boldsymbol{x}}_i$ can be obtained by an L_1-norm:

$$\overset{\text{up}}{\boldsymbol{x}}_i = \left\| \boldsymbol{e}_i^\top (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b} \right\|_1 \overset{\max}{u}, \tag{18}$$

and a second one can be estimated by an L_2-norm [15]:

$$\overset{\text{up}}{\boldsymbol{x}}_i \simeq \delta \left\| \boldsymbol{e}_i^\top (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b} \right\|_2 \overset{\max}{u}. \tag{19}$$

Here, the parameter δ can be interpreted as a representation of the number of standard deviations of $\boldsymbol{x}_i$, if the input is unit-variance white centered noise ($\delta \geqslant 1$). Since the L_2-norm in (19) doesn't give a strict bound (contrary to (18)), δ can be seen as a *safety* parameter [15].
Finally, these upper bounds are used to define the binary-point positions:

$$\gamma_{\boldsymbol{x}_i} = \beta_{\boldsymbol{x}_i} - 2 - \left\lfloor \log_2 \overset{\text{up}}{\boldsymbol{x}}_i \right\rfloor. \tag{20}$$

In general, the L_1 and L_2 estimations of $\overset{\text{up}}{\boldsymbol{x}}_i$ approximately leads to the same binary-point position, with 1 or 2 bits deviation. However, since the L_2-norm is more tractable (with proposition 1) and the L_1-norm too conservative ($\overset{\max}{\boldsymbol{x}}_i \ll \overset{\text{up}}{\boldsymbol{x}}_i$), in practice (19) is used, with $\delta = 1$. After implementation, a simulation-based estimation like in [1] or [12] can also be used to verify *in situ* the peak values and the binary point positions, according to the inputs.

4.3 Computational scheme

In order to implement a realization without overflows, two equivalent choices are possible:

- set the binary-point position for each state, according to (20), to make sure that the fixed-point representation of the states avoids state-overflows;
- or define a binary-point position for each state, and apply a scaling to them in order to adapt the peak values of each state to the chosen binary-point position.

Here, we here focus on the 2nd choice, referring to dynamic-range-scaling constraints.

Let us consider in detail the fixed-point implementation of the system given in (1). It leads to $(n+1)$ scalar products to be evaluated, of the form:

$$S = \sum_{i=1}^{N} \boldsymbol{p}_i \boldsymbol{q}_i \tag{21}$$

where the $(\boldsymbol{p}_i)$ are given coefficients and $(\boldsymbol{q}_i)$ are bounded variables. To avoid bit-shift operations between each addition in the evaluation of eq. (21), the binary-point positions of each partial product of the sum should be equal.
Then, two computational schemes are possible: the *Roundoff After Multiplication* scheme, where shifts are added after each product to align the operands of the sum ($\boldsymbol{p}_i\boldsymbol{q}_i$ is implemented as $(\boldsymbol{p}'_i * \boldsymbol{q}'_i) >> d_i$) and the *Roundoff Before Multiplication* scheme, where the required shifts are reported into the coefficients ($\boldsymbol{p}_i\boldsymbol{q}_i$ is implemented as $(\boldsymbol{p}'_i >> d_i) * \boldsymbol{q}'_i$). See Figure 2.

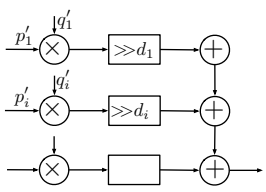

Figure 2: Scalar product with *Roundoff After Multiplication* scheme

The main idea of the scaling is to scale each variable $(\boldsymbol{q}_i)$ such that the shifts ($d_i = 0,\ \forall i$) are prevented. In fixed-point representation, the scaling only implies that all the $(\boldsymbol{q}_i)$ have a common format, and so have the $(\boldsymbol{p}_i)$. See [2, 6] for more details on implementation schemes.

Applied to the state-space realization (1), this yields that all the states must have the same binary-point position as the input and the coefficients $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and d.
Besides, since they have the same fractional part γ, their quantization's errors $\Delta\boldsymbol{A}$, $\Delta\boldsymbol{b}$, $\Delta\boldsymbol{c}$ and Δd have the same magnitude $2^{-\gamma-1}$, and the L_2-sensitivity measure represents a meaningful bound on the transfer function error Δh (considering that h is shifted in $h + \Delta h$ by the coefficients' quantization):

$$\begin{aligned} \|\Delta h\|_2^2 &\leqslant \left\| \frac{\partial h}{\partial \boldsymbol{A}} \times \Delta\boldsymbol{A} \right\|_2^2 + \left\| \frac{\partial h}{\partial \boldsymbol{b}} \times \Delta\boldsymbol{b} \right\|_2^2 \\ &\quad + \left\| \frac{\partial h}{\partial \boldsymbol{c}} \times \Delta\boldsymbol{c} \right\|_2^2 + \left\| \frac{\partial h}{\partial d} \times \Delta d \right\|_2^2 \qquad (22) \\ &\leqslant 2^{-2(\gamma+1)} M_{L_2} \qquad (23) \end{aligned}$$

4.4 New L_2-scaling constraints

Taken this into consideration, the overflows will be avoided by setting the same binary-point position for the states and the input, and by applying an appropriate scaling on the states such that the constraints (16) are satisfied.
Compared to strict L_2-scaling where the states must satisfy $\overset{\max}{\boldsymbol{x}}_i = \overset{\max}{u}$, here, the constraints are relaxed (but still restrictive enough to guarantee the protection against overflow) and replaced by $\gamma_{\boldsymbol{x}_i} = \gamma_u$.

Proposition 2 (*relaxed*-L_2-scaling constraints) *Since the input and the states may have the same binary-point position, the L_2-scaling constraints* (13) *are now transformed into*

$$\frac{2^{2\boldsymbol{\alpha}_i}}{\delta^2} \leqslant (\boldsymbol{W}_c)_{i,i} < 4\frac{2^{2\boldsymbol{\alpha}_i}}{\delta^2}, \quad \forall 1 \leqslant i \leqslant n \tag{24}$$

where

$$\boldsymbol{\alpha}_i \triangleq \beta_{\boldsymbol{x}_i} - \beta_u - \mathscr{F}_2\left(\overset{\max}{u}\right) \tag{25}$$

and $\mathscr{F}_2(x)$ is defined as the fractional value of $\log_2(x)$*:*

$$\mathscr{F}_2(x) \triangleq \log_2(x) - \lfloor \log_2(x) \rfloor \tag{26}$$

Proof:
The binary-point position of the input is set to $\gamma_u = \beta_u - 2 - \left\lfloor \log_2 \overset{\max}{u} \right\rfloor$. Hence, with (20), the constraints $\gamma_u = \gamma_{\boldsymbol{x}_i}$ lead to

$$\beta_u - \left\lfloor \log_2 \overset{\max}{u} \right\rfloor = \beta_{\boldsymbol{x}_i} - \left\lfloor \log_2 \left(\delta \left\| \boldsymbol{e}_i^\top (z\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{b} \right\|_2 \overset{\max}{u} \right) \right\rfloor$$

and

$$\left\lfloor \log_2 \left(\delta \sqrt{(\boldsymbol{W}_c)_{i,i}} \right) + \mathscr{F}_2\left(\overset{\max}{u}\right) \right\rfloor = \beta_{\boldsymbol{x}_i} - \beta_u \tag{27}$$

And finally

$$2^{\boldsymbol{\alpha}_i} \leqslant \delta \sqrt{(\boldsymbol{W}_c)_{i,i}} < 2^{\boldsymbol{\alpha}_i + 1} \tag{28}$$

■

Remark 2 For microcontroller or DSP implementations (contrary to FPGA or some ASIC implementations), the wordlength of all variables is most of the time equal, *i.e.* $\beta_u = \beta_{\boldsymbol{x}_i} \quad (1 \leqslant i \leqslant n)$. δ is set to unity (as for classical L_2-scaling constraints).

In best case, $\overset{\max}{u}$ is equal to a power of 2 (2^p, $p \in \mathbb{Z}$), and the relaxed-L_2-scaling constraints (24) become:

$$1 \leqslant (\boldsymbol{W}_c)_{i,i} < 4, \quad \forall 1 \leqslant i \leqslant n, \tag{29}$$

and in worst case, $\overset{\max}{u}$ is equal to the representable value immediately lower than a power of 2 ($2^p - 2^{\beta_u - 2 - p}$), and the constraints become:

$$\frac{1}{4} \leqslant (\boldsymbol{W}_c)_{i,i} < 1, \quad \forall 1 \leqslant i \leqslant n \tag{30}$$

(in the first case, $p+1$ bits are used for the integer part of the states, whereas p bits only are used in the second case).

It is important to remark that these new constraints allow more freedom for the scaling and introduce a new degree of freedom for the search for optimal realizations. Even though not considered in this paper, moreover it could give more freedom for the minimization of the roundoff noise power.

5. OPTIMAL L_2-SENSITIVITY REALIZATION WITH RELAXED L_2-NORM DYNAMIC-RANGE-SCALING CONSTRAINTS

Then, these relaxed constraints can be applied to a new sensitivity problem:

Problem 3 (*relaxed* sensitivity problem) *The optimal L_2-sensitivity problem with* relaxed *L_2-norm dynamic-range-scaling constraints can be expressed in the form of the constrained problem 2 subject to constraints in* (24)*.*

This constrained problem can be solved by scaling the system after each transformation in order to ensure that the constraints are met:

Proposition 3 (*a posteriori* relaxed scaling) *Considering a state-space realization, it is possible to* a posteriori *scale it with a diagonal transformation matrix $\boldsymbol{T}$ given by*

$$\boldsymbol{T}_{i,i} = \delta \sqrt{(\boldsymbol{W}_c)_{i,i}} 2^{-\mathscr{F}_2(\delta\sqrt{(\boldsymbol{W}_c)_{i,i}}) - \boldsymbol{\alpha}_i}, \tag{31}$$

such that the constraints (24) *are satisfied. Moreover, it is possible to build all the transformation matrices that meet the constraints* (24)*: Let us consider an invertible matrix $\boldsymbol{U} \in \mathbb{R}^{n \times n}$, then the transformation matrix $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{V}$ with $\boldsymbol{V}$ diagonal such that:*

$$\boldsymbol{V}_{i,i} = \delta \sqrt{\left(\boldsymbol{U}^{-1}\boldsymbol{W}_c\boldsymbol{U}^{-\top}\right)_{i,i}} 2^{-\mathscr{F}_2(\delta\sqrt{(\boldsymbol{U}^{-1}\boldsymbol{W}_c\boldsymbol{U}^{-\top})_{i,i}}) - \boldsymbol{\alpha}_i} \tag{32}$$

produces the relaxed-L_2-scaling.
Proof:
$\mathscr{F}_2$ acts as a modulo operator. For $x \in \mathbb{R}$, $\bar{x} \triangleq 2^{\mathscr{F}_2(x)+a}$ is such that $2^a \leqslant \bar{x} < 2^{a+1}$.
Since the constraints (24) are equal to

$$2^{\boldsymbol{\alpha}_i} \leqslant \delta \sqrt{(\boldsymbol{W}_c)_{i,i}} < 2^{\boldsymbol{\alpha}_i + 1} \tag{33}$$

and $\boldsymbol{T}$ transforms $(\boldsymbol{W}_c)_{i,i}$ into $\boldsymbol{T}_{i,i}^{-2}(\boldsymbol{W}_c)_{i,i}$, then $\boldsymbol{T}_{i,i}$ has to be of the form:

$$\delta \boldsymbol{T}_{i,i}^{-1} \sqrt{(\boldsymbol{W}_c)_{i,i}} = 2^{\mathscr{F}_2(\delta\sqrt{(\boldsymbol{W}_c)_{i,i}}) + \boldsymbol{\alpha}_i} \tag{34}$$

■

Thus, the optimization problem is given by

$$\boldsymbol{U}_{opt} = \underset{\substack{\boldsymbol{U} \text{ invertible} \\ \boldsymbol{V} \text{ defined by (32)}}}{\arg\min} \; M_{L_2}(\boldsymbol{U}\boldsymbol{V}). \tag{35}$$

This was implemented in the FWR toolbox[4] for Matlab, with `fminsearch`, and `fminunc` functions, and they both give same results with similar numbers of iterations.
Of course, the use of the matrice $\boldsymbol{V}$, that is merely used to eliminate the constraints and solve an unconstrained minimization problem, increases the degree of non-linearity for the objective function to minimize. However, this seems not to be a problem, since in our tests, the optimal realizations found seem to be global optima.

[4] sources available at `http://fwrtoolbox.gforge.inria.fr/`

6. EXAMPLE

Let us consider the following digital filter, given by its transfer function[5]:

$$H(z) = \frac{4.297e-3z^4 - 8.595e-3z^2 + 4.297e-3}{z^4 - 3.803z^3 + 5.550z^2 - 3.678z + 0.9355} \tag{36}$$

and its multiple equivalent (in infinite precision) realizations:

- $\mathscr{R}_1$ is the Direct Form II given by (37).
- $\mathscr{R}_2$ is the optimal L_2-scaled realization (solution of problem 2). The numerical values are given by (38).
- $\mathscr{R}_3$ is the optimal relaxed-L_2-scaled realization (problem 3), with $\overset{\max}{u}$ a power of 2, and $\delta = 1$. It is obtained with proposition 3 . The numerical values are given by (39).

The balanced realization (known to be close to the unscaled M_{L_2}-optimal realization) was chosen as the starting point for the optimization. The numerical results are presented in the appendix and the pseudo-code associated to the realization $\mathscr{R}_3$ is presented in algorithm 1 (16 bits are considered for the input, output, states and coefficients, 32 bits for the accumulator (no guard bits), and $\overset{\max}{u} = 16$).

The following table gives the M_{L_2} sensitivities of these different realizations:

realization	M_{L_2} sensitivity
$\mathscr{R}_1$	$1.690e+09$
$\mathscr{R}_2$	5247.9
$\mathscr{R}_3$	5222.1

In this example, the relaxed L_2-scaled realization $\mathscr{R}_3$ achieves lower sensitivity than the strict L_2-scaled optimal realization $\mathscr{R}_2$ while protecting implementation from overflows.
But it is not always the case : if we consider the example in [7], the optimal relaxed-L_2-scaled realization satisfies $(\boldsymbol{W}_c)_{i,i} = 1$ and is then also a strict L_2-scaled realization. This depends on the diagonal terms of the controllability Gramians of the (non scaled) optimal realization.
It is also interesting to notice that a good estimation of $\overset{\max}{u}$ (if it is not a power of 2) can allow to achieve lower sensitivity by moving the constraints (it could also be the case for the example in [7]).

7. CONCLUSION

This paper has presented the L_2-sensitivity minimization problem and the associated L_2-scaling constraints. These constraints that prevent from overflows have been considered with concrete fixed-point implementation schemes. Novel L_2-dynamic-range constraints have been exhibited.

Even if the goal of this paper is not a detailed optimization algorithm like in [7], a proposition to solve the constrained optimization problem has been exhibited and applied on a numerical example.

These relaxed constraints could also be very important for some other realizations, like δ-operator state-space, the ρ-Direct Form II transposed [13] and also classical filtering structures. For these realizations where a parameter Δ should be used to achieve the L_2-scaling, a relaxed-L_2-scaling permits to fix this parameter as a power of 2, in order to decrease the amount of computations.
To apply this work to other classical structures, it will be soon extended to the Specialized Implicit Framework [5] that allows to encompass existing structures in an implicit state-space form.

REFERENCES

[1] P. Belanovic and M. Rupp. Automated floating-point to fixed-point conversion with the fixify environment. *Rapid System Prototyping, 2005. (RSP 2005). The 16th IEEE International Workshop on*, pages 172–178, June 2005.

[5] Due to a lack of space, only 4 digits are given, but more may be required to completely define the system.

[2] G. Constantinides, P. Cheung, and W. Luk. *Synthesis And Optimization Of DSP Algorithms*. Kluwer Academic Publishers, 2004.

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

[4] T. Hilaire and P. Chevrel. On the compact formulation of the derivation of a transfer matrix with respect to another matrix. Technical Report RR-6760, INRIA, 2008.

[5] T. Hilaire, P. Chevrel, and J. Whidborne. A unifying framework for finite wordlength realizations. *IEEE Trans. on Circuits and Systems*, 8(54), August 2007.

[6] T. Hilaire, D. Ménard, and O. Sentieys. Bit accurate roundoff noise analysis of fixed-point linear controllers. In *Proc.* IEEE International Symposium on Computer-Aided Control System Design *(CACSD'08)*, September 2008.

[7] T. Hinamoto, H. Ohnishi, and W.-S. Lu. Minimization of l2 sensitivity of one- and two dimensional state-space digital filters subject to l2-dynamic-range-scaling constraints. *IEEE Trans. on Circuits and Systems-II*, 52(10):641–645, October 2005.

[8] T. Hinamoto and Y. Sugie. L2-sensitivity analysis and minimization of 2-d separable-denominator state-space digital filters. *Signal Processing, IEEE Transactions on*, 50(12):3107–3114, Dec 2002.

[9] S. Hwang. Dynamic range constraint in state-space digital filtering. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume 23, pages 591–593, 1975.

[10] S. Hwang. Minimum uncorrelated unit noise in state-space digital filtering. *IEEE Trans. on Acoust., Speech, and Signal Processing*, 25(4):273–281, August 1977.

[11] L. Jackson. Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form. *Audio and Electroacoustics, IEEE Transactions on*, 18(2):107–122, June 1970.

[12] S. Kim, K. Kum, and W. Sung. Fixed-point optimization utility for C and C++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems*, 45(11):1455–1464, November 1998.

[13] 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*, 51(4):769–778, April 2004.

[14] C. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. In *IEEE Transactions on Circuits and Systems*, volume CAS-23, September 1976.

[15] K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation of Digital Controllers*. Number ISBN 0-471-24186-5. John Wiley & Sons, 1999.

[16] V. Tavşanoğlu and L. Thiele. Optimal design of state-space digital filters by simultaneous minimization of sensibility and roundoff noise. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume CAS-31, October 1984.

[17] L. Thiele. Design of sensitivity and round-off noise optimal state-space discrete systems. *Int. J. Circuit Theory Appl.*, 12:39–46, 1984.

A. NUMERICAL RESULTS

$$\boldsymbol{A}_1 = \begin{pmatrix} 3.8031 & -1.3875 & 0.9196 & -0.4678 \\ 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix}, \quad \boldsymbol{b}_1 = \begin{pmatrix} 0.1250 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$
$$\boldsymbol{c}_1 = \begin{pmatrix} 0.1307 & -0.0649 & 0.0316 & 0.0011 \end{pmatrix}, \quad d_1 = 4.2974e-3 \tag{37}$$

$$\boldsymbol{A}_2 = \begin{pmatrix} 0.9386 & -0.2477 & -0.0427 & 0.0077 \\ 0.2525 & 0.9675 & -0.0010 & -0.0003 \\ 0.0228 & -0.0031 & 0.9301 & -0.2524 \\ 0.0039 & -3.071e-05 & 0.2517 & 0.9669 \end{pmatrix}, \quad \boldsymbol{b}_2 = \begin{pmatrix} -0.0652 \\ -0.0193 \\ 0.3130 \\ 0.0471 \end{pmatrix},$$
$$\boldsymbol{c}_2 = \begin{pmatrix} -1.1151 & 0.2002 & -0.1789 & 0.0729 \end{pmatrix}, \quad d_2 = 4.2974e-3 \tag{38}$$

$$\boldsymbol{A}_3 = \begin{pmatrix} 0.9300 & -0.2504 & 0.0290 & -0.0030 \\ 0.2528 & 0.9671 & 0.0027 & -0.0059 \\ -0.0483 & 0.0091 & 0.9385 & -0.2466 \\ 0.0001 & 0.0056 & 0.2527 & 0.9674 \end{pmatrix}, \quad \boldsymbol{b}_3 = \begin{pmatrix} 0.6289 \\ 0.0665 \\ -0.1326 \\ -0.0440 \end{pmatrix},$$
$$\boldsymbol{c}_3 = \begin{pmatrix} -0.0872 & 0.0335 & -0.5567 & 0.1104 \end{pmatrix}, \quad d_3 = 4.2974e-3 \tag{39}$$

Input: u: 16 bits integer
Output: y: 16 bits integer
Data: xn, xnp: array [1..4] of 16 bits integers
Data: Acc: 32 bits integer
begin
  `// Intermediate variables`
  $Acc \leftarrow (xn(1) * 31209)$;
  $Acc \leftarrow Acc + (xn(2) * -8303)$;
  $Acc \leftarrow Acc + (xn(3) * -600)$;
  $Acc \leftarrow Acc + (xn(4) * 78)$;
  $Acc \leftarrow Acc + (u * -806)$;
  $xnp(1) \leftarrow Acc >> 15$;
  $Acc \leftarrow (xn(1) * 8304)$;
  $Acc \leftarrow Acc + (xn(2) * 31690)$;
  $Acc \leftarrow Acc + (xn(3) * -77)$;
  $Acc \leftarrow Acc + (xn(4) * 10)$;
  $Acc \leftarrow Acc + (u * -104)$;
  $xnp(2) \leftarrow Acc >> 15$;
  $Acc \leftarrow (xn(1) * 2398)$;
  $Acc \leftarrow Acc + (xn(2) * -315)$;
  $Acc \leftarrow Acc + (xn(3) * 30009)$;
  $Acc \leftarrow Acc + (xn(4) * -8149)$;
  $Acc \leftarrow Acc + (u * 1523)$;
  $xnp(3) \leftarrow Acc >> 15$;
  $Acc \leftarrow (xn(1) * 313)$;
  $Acc \leftarrow Acc + (xn(2) * -41)$;
  $Acc \leftarrow Acc + (xn(3) * 8147)$;
  $Acc \leftarrow Acc + (xn(4) * 31711)$;
  $Acc \leftarrow Acc + (u * 198)$;
  $xnp(4) \leftarrow Acc >> 15$;
  `// Outputs`
  $Acc \leftarrow (xn(1) * -25782)$;
  $Acc \leftarrow Acc + (xn(2) * 3381)$;
  $Acc \leftarrow Acc + (xn(3) * -12182)$;
  $Acc \leftarrow Acc + (xn(4) * 1575)$;
  $Acc \leftarrow Acc + (u * 18)$;
  $y \leftarrow Acc >> 15$;
  `// Permutations`
  $xn \leftarrow xnp$;
end

Algorithm 1: Numerical fixed-point algorithm of realization $\mathscr{R}_3$