Applying Markov chains to calculate the probability of saturation in digital IIR filters

Fernando G. Almeida Neto and Vítor H. Nascimento
Electronic Systems Engineering Department
Escola Politécnica, University of São Paulo
São Paulo, Brazil
E-mails: {fganeto, vitor}@lps.usp.br

José Carlos M. Bermudez
Electrical Engineering Department
Federal University of Santa Catarina
Florianópolis, Brazil
E-mail: bermudez@eel.ufsc.br

***Abstract*— We propose a new method to model the effect of finite-precision arithmetic in infinite impulse response (IIR) digital filters. As an application, we use the proposed model to compute the probability of saturation or overflow in IIR filters implemented in fixed-point arithmetic. The transition from the current filter output to the next output is modeled as a first-order Markov chain. The Markov chain transition probability matrix is then used to evaluate the probabilities of saturation or overflow for first and second-order IIR filters.**

***Keywords*— Saturation, IIR filters, Markov chains, finite precision arithmetic.**

I. Introduction

Modeling the behavior of algorithms when implemented in finite-precision arithmetic is important for practical designs. Such models are however difficult to develop due to the highly nonlinear characteristic of the quantization operation. Infinite impulse response (IIR) filters may be considerably sensitive to finite-precision effects, given their feedback structure. This is specially true when the number of bits is small. In this case, the usual modeling of quantization noise as a uniformly-distributed random variable is not appropriate. Nevertheless, designs using short wordlengths are required in applications where low power consumption is paramount, such as cellular phones and other portable devices. In these cases, a more precise model for the quantization effect is desirable.

During its operation, the output of a filter can also exceed its allowed range, severely degrading the filter performance. In finite-precision arithmetic, such an exception may be dealt with simply by disregarding the most significant bits of the output (which we will call "overflow") or by saturating the output to its most positive or most negative value. In both cases, a large error results, which should be avoided for proper system operation. One way to avoid saturation is to scale the filter coefficients to avoid, or at least reduce, the probability of exceeding the output range [1], [2]. The approach is to determine the transfer function from the input of the filter to the input of each multiplier and then use the inverse of the p-norm of this function as a scaling factor, where p is chosen according to the signal in the input of the filter. However, this is a worst-case approach, which may lead to conservative designs (i.e., a scale factor that is smaller than necessary, leading to a lower signal-to-noise ratio). A model that incorporates the highly nonlinear overflow and saturation effects during the filter operation can be of great help for the designer. To the best of our knowledge, no precise models for predicting their probability of occurrence are available.

In this paper we propose to model the behavior of first and second-order IIR filters using a first-order Markov chain. This approach is an extension of [3] and [4], which investigated the impact of finite precision in the performance of the least mean squares (LMS) algorithm. We consider a fixed-point implementation and use no linearization in the description of the signal quantizations. We apply a Markov chain to model the transition probabilities from the current output to the possible future outputs of the filter. We take advantage of the fact that the output may assume only a finite number of values in finite-precision. These values are interpreted as states of a Markov chain. We introduce extra states in the transition matrix to calculate the probability of saturation or overflow.

This paper is organized as follows: Section II presents the nonlinear IIR filters models used here, while Section III makes a brief introduction to the Markovian concepts needed. Section IV introduces the approach to calculate the probability of saturation, while Section V shows some examples of the proposed method. Section VI concludes the paper.

II. Nonlinear effects in IIR filters

Digital IIR filters are, in general, implemented as a cascade of first and second-order filters, which are described by the difference equations (1) and (2), respectively,

$$y_1(n) = b_0u(n) + b_1u(n-1) - a(1)y_1(n-1) \tag{1}$$

and

$$\begin{aligned} y_2(n) = b_0u(n) + b_1u(n-1) + b_2u(n-2)) \\ -a_1y_2(n-1) - a_2y_2(n-2), \end{aligned} \tag{2}$$

where the filter coefficients are given by a_k, for $k = 1,2$, and b_k, for $k = 0,1,2$. Equations (1) and (2) consider coefficients and signals represented in finite precision [5]. In this paper, we consider fixed-point representation.

A fixed-point implementation uses a fixed number of bits to represent the integer and the fractional parts of a number. A filter has thus a finite number of codes to describe quantities, which is given by $N = 2^B$, where B is the word length in bits. If the range of representable number is from -1 to $+1-\Delta$, the quantization step will be $\Delta = 2^{-B+1}$, and the result of all operations must be rounded or truncated to fit to the numerical representation. If a sum or a multiplication result exceeds the representation, another nonlinear operation is used to find a representation within the established bounds. For instance, saturation limits the exceeding quantities to the bounds of the representation, while overflow disregards the most significant bits outside the allowed range, resulting in large errors. Figure 1 shows saturation and overflow for a two's complement 2-bit signal representing the set $\{-1, -0.5\; 0\; 0.5\}$.

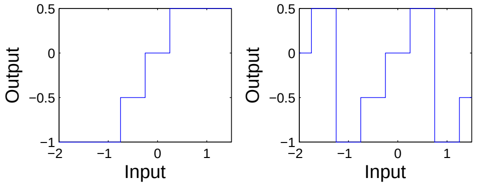

Fig. 1. Saturation (left) and overflow (right) for a 2-bit signal

Equations (3) and (4) describe the application of the nonlinear operations to the filter equations, i.e.,

$$y_1(n) = R[R[Q\{b_0 u(n)\} + Q\{b_1 u(n-1)\}] + Q\{-a_1 y_1(n-1)\}] \tag{3}$$

and

$$y_2(n) = R[R[Q\{b_0 u(n)\} + R[Q\{b_1 u(n-1)\} + Q\{b_2 u(n-2)\}]] \\ +R[Q\{-a_1 y_2(n-1)\} + Q\{-a_2 y_2(n-2)\}]], \tag{4}$$

where $R[\cdot]$ can be the saturation or the overflow operator after a sum and $Q\{\cdot\}$ represents quantization after a multiplication. Here we consider the most economical implementation, where accumulators are not available to perform multiply-accumulation operations. Figures 2 and 3 show the quantized filters in a direct form I implementation. Note that the $R[\cdot]$ operator may be applied only once in (3) and (4) if the processor has a register with guard bits for intermediate computation, such as is found in most DSP. The method presented here may be easily modified to consider all details of a specific implementation.

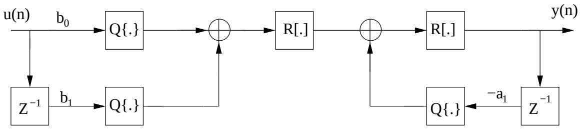

Fig. 2. Quantized first-order IIR filter implemented in direct-form I

The possible outputs of IIR filters implemented in fixed-point are contained in a finite set, and the current output clearly depends on the last outputs and on the current and last inputs, as we can observe in (3) and (4). We can take advantage of these two characteristics and use a first-order discrete Markov chain [6] to find a probabilistic description of the filter output as a function of the past output and the input. In this case, we can define each possible output of the filter as a state of a Markov chain. In the next section, we introduce some concepts of Markov chains used in this paper. To clarify the calculation of the transition matrix, we also present an example with a first-order filter.

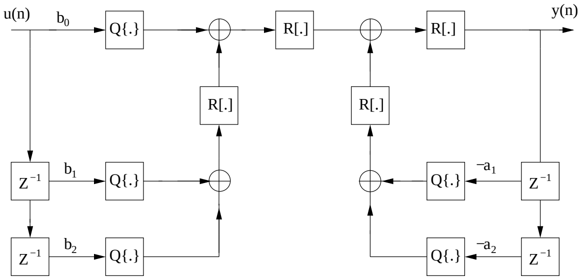

Fig. 3. Quantized second-order IIR filter implemented in direct-form I

III. Discrete-time Markov chains

A discrete-first-order Markov chain [6] is a discrete stochastic process where the probability of the next state, given the current and the past states, only depends on the current state. That means that given a *stochastic process* $\{X_n\}_{n=0}^{\infty}$,

$$P(X_n = i_n | X_{n-1} = i_{n-1}, \cdots, X_0 = i_0) = P(X_n = i_n | X_{n-1} = i_{n-1}),$$

where $P(a|b)$ is the conditional probability of *a given b* and i_n represents the possible states for X_n. The subscript n represents time instants ($n = 1, 2, \ldots$) and we consider that the states i_n belong to the finite set $\{1, 2, \ldots, N\}$. In general, the notation used in (5) is abbreviated as $P(X_n = i | X_{n-1} = j) = p_{ij}$, where p_{ij} is defined as the probability to reach state i when the current state is j. This notation is used to define an $N \times N$ matrix $\mathbb{P}$ with elements p_{ij}. $\mathbb{P}$ is called the *transition matrix* and its main characteristic, by construction, is that all the columns add to 1, since

$$\sum_{i=1}^{N} p_{ij} = \sum_{i=1}^{N} P(X_n = i | X_{n-1} = j) = 1. \tag{5}$$

Indeed, each column represents a conditional probability distribution for each state, in a specific instant. Therefore, if we consider that the transition probabilities are independent of n, we can find the transition probabilities after n instants, which corresponds to

$$p_{ij}^{(n)} = P(X_n = i | X_0 = j), \tag{6}$$

where $p_{ij}^{(n)}$ is the probability to begin in the state j and reach the state i after n steps. We can use the Chapman-Kolmogorov equation [7] to calculate $p_{ij}^{(n)}$ as

$$p_{ij}^{(n)} = \sum_{k=0}^{N} p_{kj}^{(n-1)} p_{ik}. \tag{7}$$

Equation (7) shows an iterative method to calculate $p_{ij}^{(n)}$ given the past probabilities. Writing in matrix form [6],

$$\mathbb{P}^{(n)} = \mathbb{P}^{(n-1)} \cdot \mathbb{P}^{(1)} = \mathbb{P}^{(1)} \cdot \mathbb{P}^{(n-1)} = \mathbb{P}^{n},\ \mathrm{n} = 1, 2, \ldots \tag{8}$$

where $\mathbb{P}(0) = I_{N\times N}$. We conclude that $\mathbb{P}^{(n)}$ is equivalent to $\mathbb{P}^n$. Thus, given a initial probability distribution vector π_0 for X_0, we obtain

$$\pi_n = \mathbb{P}^n \pi_0, \tag{9}$$

and we notice that the knowledge of $\mathbb{P}$ and π_0 allows us to know the distribution after n steps.

If we look to $\mathbb{P}^n$ when $n \rightarrow \infty$, we obtain the process steady-state (SS) matrix. This matrix differs from the initial matrix $\mathbb{P}$ because of the absence of transient states, which are the states that stop receiving visits after a finite number of steps. The SS matrix contains the information about the long term process, and therefore about the saturation of the output in steady-state.

Computing $\mathbb{P}$ for IIR filters is simple, as we show in the next example.

Example 1: Suppose a 2-bit filter, described by the coefficients $a_1 = 0.5$, $b_0 = 0.5$ and $b_1 = 0$. We want to calculate the 4×4 matrix $\mathbb{P}$ when there is an input $u(n)$ with uniform distribution and zero mean, (i.e., the probability of $u(n) = -0.5$, 0 and 0.5 are $1/3$ and the probability of $u(n) = -1$ is zero, and the input is white). We consider here that $R[\cdot]$ is the saturation operator and that we round up, i.e., $Q\{0.25\} = 0.5$ and $Q\{-0.25\} = 0$. Let us, for example, find the element $p_{34} = P(y(n) = 0|y(n-1) = 0.5)$. For this element, equation (3) is modified as

$$y(n) = R[Q\{0.5u(n)\} + Q\{-0.5\cdot 0.5\}],$$

since $b_1 = 0$ and $y(n-1) = 0.5$. Varying $u(n)$ for all the possible values, if we calculate $y(n)$, we notice that

$$\begin{aligned} y(n) &= R[Q\{0.5(-1)\} + Q\{-0.5(0.5)\}] = -0.5 \\ y(n) &= R[Q\{0.5(-0.5)\} + Q\{-0.5(0.5)\}] = 0 \\ y(n) &= R[Q\{0.5(0)\} + Q\{-0.5(0.5)\}] = 0 \\ y(n) &= R[Q\{0.5(0.5)\} + Q\{-0.5(0.5)\}] = 0.5 \end{aligned},$$

where the last result comes from the saturation of 1 to 0.5. Therefore, there are two possibilities to reach $y(n) = 0$ (when $u(n) = -0.5$ and $u(n) = 0$), and we should use the distribution of the input to calculate p_{34}, as

$$p_{13} = P(u(n) = -0.5) + P(u(n) = 0) = \frac{1}{3} + \frac{1}{3} = 0.667.$$

Using the same procedure to determine the other elements of $\mathbb{P}$ yields

$$\begin{array}{cc} \begin{array}{cccc} -1.0 & -0.5 & 0 & 0.5 \end{array} & \textbf{States} \\ \mathbb{P} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.667 & 0.667 \\ 1.000 & 1.000 & 0.333 & 0.333 \end{array}\right] & \begin{array}{c} -1.0 \\ -0.5 \\ 0 \\ 0.5 \end{array} \end{array}$$

where the numbers above and to the right of $\mathbb{P}$ show the values of $y(n)$ and $y(n-1)$ corresponding to each row and column.

Assume now that the current output has a distribution $\pi_n = [0\ 0.5\ 0\ 0.5]$, and we want to know the probability of $y(n+1) = 0.5$. Using (9), we obtain

$$\pi_{n+1} = \mathbb{P}\pi_n = [0\ \ 0\ \ 0.333\ \ 0.667]^T,$$

which is the distribution vector at instant $n+1$. Thus, we conclude that $P(y(n+1) = 0.5) = 0.333$. We can also find the distribution for the instant $n+2$, calculating

$$\pi_{n+2} = \mathbb{P}\pi_{n+1} = \mathbb{P}^2\pi_n = [0\ \ 0\ \ 0.667\ \ 0.333]^T.$$

Therefore, we can calculate any probability for any time instant if we have $\mathbb{P}$ and an initial distribution vector for $y(0)$. In the next section, we include extra states to describe overflow or saturation. For convenience, we refer to these extra states as “saturation states”.

IV. Modelling saturation

In section III, $\mathbb{P}$ was described with states related to the output of the filter, considering that the output is limited to a range (e.g., the range of the past example is the set $\{-1,\ -0.5,\ 0,\ 0.5\}$), and using a nonlinear saturation to guarantee this limitation. This means that when the output exceeds the range, this value is limited to the bounds of the range (e.g., for the past example if the filter calculates an output of 1, the saturation limits the output to 0.5). In this case, although $\mathbb{P}$ takes saturation into account, we cannot distinguish saturated from nonsaturated outputs. However, we can introduce more states to model the saturation and obtain the exact probability of saturation in a filter. For this purpose, we add two states in the transition matrix, corresponding to the saturation to the positive and negative limits of the output. The matrix obtained will be $(N+2)\times(N+2)$ if we are using a first-order filter and $(N+2)^2\times(N+2)^2$ when a second-order filter is analyzed. (Although the matrices are large, we can apply sparse matrix computation to reduce the calculations, since the matrices have a large number of zero elements [1].)

Consider again, the first example. To observe the saturation, we define two extra states: -1_s and 0.5_s. The state -1_s corresponds to an output -1, but that is reached through saturation. In the same way, the state 0.5_s corresponds to an output 0.5 obtained by saturation. Let us find $P(y(n) = e|y(n-1) = -1)$, when $e\in\{-1_s\ -1\ -0.5\ 0\ 0.5\ 0.5_s\}$, we notice that

$$\begin{aligned} y(n) &= Q\{0.5(-1)\} + Q\{-0.5(-1)\} = 0 \\ y(n) &= Q\{0.5(-0.5)\} + Q\{-0.5(-1)\} = 0.5 \\ y(n) &= Q\{0.5(0)\} + Q\{-0.5(-1)\} = 0.5 \\ y(n) &= Q\{0.5(0.5)\} + Q\{-0.5(-1)\} = 1 = 0.5_s. \end{aligned}$$

Therefore, we conclude that the element of the transition matrix

$$p_{42} = P(x(n) = -0.5) + P(x(n) = 0) + P(x(n) = 0.5) = 1$$

includes one part related to the output saturation (when $x(n) = 0.5$). To include the two saturation states, we use an expanded matrix $\mathbb{P}$, where the column $P(y(n)|y(n-1) = -1)$, for $y(n) \in \{-1_s\ -1\ -0.5\ 0\ 0.5\ 0.5_s\}$, corresponds to

$$P(y(n)|y(n-1) = -1) = [0\ \ 0\ \ 0\ \ 0\ \ 0.667\ \ 0.333]^T.$$

[1] In fact, the use of sparse matrices is more efficient when we calculate the transition matrix for a second-order filter, since the matrix dimension is larger and zero elements appear more frequently.

If we calculate the full expanded matrix $\mathbb{P}$, we obtain

$$\mathbb{P} = \begin{array}{cccccc|c} -1.0_s & -1.0 & -0.5 & 0 & 0.5 & 0.5_s & \textbf{States} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -1.0_s \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0.667 & 0.667 & 0.667 & 0 \\ 0.667 & 0.667 & 0.667 & 0.333 & 0.333 & 0.333 & 0.5 \\ 0.333 & 0.333 & 0.333 & 0 & 0 & 0 & 0.5_s \end{array}$$

The rows corresponding to -1_s and 0.5_s show the conditional probabilities of saturation. We notice from the transition matrix that the columns related to -1_s and -1 have the same distribution. This happens because when we have state -1_s or -1, the output is -1. Therefore, $P(y(n)|y(n-1) = -1_s) = P(y(n)|y(n-1) = -1)$, and the columns must be equal. The same argument is valid for the states 0.5_s and 0.5. (It is important to note that the columns corresponding to -0.5 and 0, in general, do not need to be equal to the columns of the states -1 and 0.5, as we observe in this example.)

V. Examples

In order to calculate the probability of saturation with the proposed method, we wrote two programs in *Matlab*. The programs calculate the transition matrix based on the probability distribution of the inputs, and they describe first and second order IIR filters, as presented in equations (3) and (4). For simplicity, we present only one example, assuming a 2-bit first order filter We use saturation after sums. The program inputs are the filter coefficients, the input word length B and the distribution of $u(n)$. We use sparse matrix calculation to reduce the number of operations.

A. *Probability of saturation*

Consider the filter $a_1 = 0.75$, $b_0 = 1$ and $b_1 = 0$, where the coefficients have a 3-bit description, while we still have a 2-bit input and output (this choice is made only to keep the example simple). Using the input distribution $[0\ 1/3\ 1/3\ 1/3]$, the transition matrix is given by

$$\mathbb{P} = \begin{array}{cccccc|c} -1.0_s & -1.0 & -0.5 & 0 & 0.5 & 0.5_s & \textbf{States} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -1.0_s \\ 0 & 0 & 0 & 0 & 0.333 & 0.333 & -1.0 \\ 0 & 0 & 0 & 0.333 & 0.333 & 0.333 & -0.5 \\ 0.333 & 0.333 & 0.333 & 0.333 & 0.333 & 0.333 & 0 \\ 0.333 & 0.333 & 0.333 & 0.333 & 0 & 0 & 0.5 \\ 0.333 & 0.333 & 0.333 & 0 & 0 & 0 & 0.5_s \end{array},$$

and the SS matrix is

$$\mathbb{P}^\infty = \begin{array}{cccccc|c} -1.0_s & -1.0 & -0.5 & 0 & 0.5 & 0.5_s & \textbf{States} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -1.0_s \\ 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & -1.0 \\ 0.222 & 0.222 & 0.222 & 0.222 & 0.222 & 0.222 & -0.5 \\ 0.333 & 0.333 & 0.333 & 0.333 & 0.333 & 0.333 & 0 \\ 0.222 & 0.222 & 0.222 & 0.222 & 0.222 & 0.222 & 0.5 \\ 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.111 & 0.5_s \end{array}.$$

We conclude from the SS matrix that there are outputs which exceed the superior limit of the representation, with a probability of 0.111, no matter what is the initial condition. We can scale the coefficient b_0 to avoid saturation — in this simple example, this means guaranteeing that $|y(n)| \leq 0.5$. If we use the traditional approach of the p-norm (which we indicate by $\|\cdot\|_p$), as presented in [1], [2], we must calculate the transfer function from the input to the output of the filter, i.e.,

$$H(z) = \frac{1}{1+0.75z^{-1}},$$

to calculate the scaling factor as

$$\lambda \leq \frac{0.5}{\|h(n)\|_p \|u(n)\|_q}, \text{ for } \frac{1}{p} + \frac{1}{q} = 1, \tag{10}$$

where $h(n)$ is the filter's impulse response. This approach is based on Hölder's inequality [8],

$$|y(n)| = \sum_{k=0}^{\infty} |h(k)u(n-k)| \leq \|h(n)\|_p \|u(n)\|_q, \text{ for } \frac{1}{p} + \frac{1}{q} = 1.$$

In this example, as $u(n)$ has unlimited energy, we should use $q = \infty$ and $p = 1$. If we calculate the 1-norm for $h(n)$ and the infinity norm for $x(n)$, we obtain

$$\|h(n)\|_1 = 1 + \sum_{n=1}^{\infty} |0.75^n| = 4$$

and

$$\|x(n)\|_\infty = \max|x(n)| = 0.5.$$

Using these in (10), we find that $\lambda \leq 0.25$. However, if we iteratively apply our method, we find $\lambda_{\text{opt}} = 0.375$ (for coefficients with 3 bits). The new transition matrix is given by

$$\mathbb{P} = \begin{array}{cccccc|c} -1.0_s & -1.0 & -0.5 & 0 & 0.5 & 0.5_s & \textbf{States} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -1.0_s \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.0 \\ 0 & 0 & 0 & 0 & 1.000 & 1.000 & -0.5 \\ 0 & 0 & 0 & 1.000 & 0 & 0 & 0 \\ 1.000 & 1.000 & 1.000 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5_s \end{array}.$$

while the SS matrix is

$$\mathbb{P}^\infty = \begin{array}{cccccc|c} -1.0_s & -1.0 & -0.5 & 0 & 0.5 & 0.5_s & \textbf{States} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -1.0_s \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.0 \\ 1.000 & 1.000 & 1.000 & 0 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 1.000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.000 & 1.000 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5_s \end{array}.$$

We conclude that,using the transition matrix, one may iteratively search for the largest scaling factor that avoids saturation or overflow, avoiding conservative designs based on worst-case considerations.

B. Quantization noise

We now use the model proposed here to compute the mean and variance of the filter output, and compare with predictions based on the linearized approach, in which quantization errors are modelled as noise with uniform distribution. In the example presented, there is one error related to quantization, after the multiplication by a_1 (since $b_0 = 1$, there is no quantization error). We can model this signal $e(n)$ with a uniform distribution, zero mean and variance equal to [2]

$$\sigma_e^2 = \frac{\Delta^2}{12}.$$

Assume that the initial condition is $y(n-1) = 0$ with probability 1 and that the input is an independent sequence, with distribution as before.

$$\begin{aligned} \mathrm{E}\{y(n)^2\} = a_1^2\,\mathrm{E}\{y(n-1)^2\} + b_0^2\,\mathrm{E}\{u(n)^2\} + \\ 2a_1 b_0\,\mathrm{E}\{y(n-1)x(n)\} + \mathrm{E}\{e(n)^2\}. \end{aligned} \tag{11}$$

From the independence of the input sequence, it follows that $e(n)$ is independent of $y(n-1)$, so $2a_1 b_0 E\{y(n-1)u(n)\} = 0$. We calculated the mean and the variance of the output for the filter in the last example. Since $u(n)$ and $e(n)$ have zero mean, the output mean is also zero. The variance was calculated with (11) and is presented in figure 4.

We calculated the exact mean $\mu(n)$ of the output with our approach, for the same initial condition, using the extended transition matrix, i.e.,

$$\begin{aligned} &\pi(1) = \mathbb{P}[0\;0\;0\;1\;0\;0\;0]^T \\ &\mu(1) = \pi^T(1)\left[-1\;-1\;-0.5\;0\;0.5\;0.5\right]^T \\ &\pi(2) = \mathbb{P}^2[0\;0\;0\;1\;0\;0\;0]^T \\ &\mu(2) = \pi^T(2)\left[-1\;-1\;-0.5\;0\;0.5\;0.5\right]^T \\ &\vdots \\ &\pi(n) = \mathbb{P}^n[0\;0\;0\;1\;0\;0\;0]^T \\ &\mu(n) = \pi^T(n)\left[-1\;-1\;-0.5\;0\;0.5\;0.5\right]^T \end{aligned} , \tag{12}$$

where $\mu(k)$ is the mean for iteration k. Similarly, for the variance, we calculated

$$\sigma_y^2(n) = \pi^T(n) \begin{bmatrix} (-1-\mu(n))^2 \\ (-1-\mu(n))^2 \\ (-0.5-\mu(n))^2 \\ (0-\mu(n))^2 \\ (0.5-\mu(n))^2 \\ (0.5-\mu(n))^2 \end{bmatrix} \tag{13}$$

for k from 0 to 25. Figure 5 shows the results.

We note from figures 4 and 5 that for this example, the linearized approach to analyze the quantization effects in digital filters produces significantly different results than the more precise approach proposed in this paper. This difference is expected to be large for filters with short wordlengths, and to diminish as the wordlength increases.

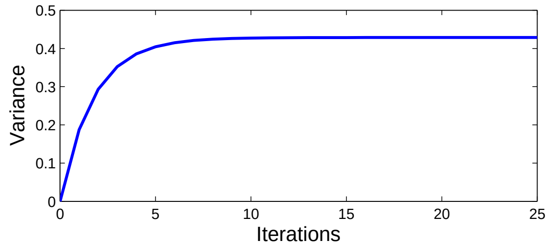

Fig. 4. Output variance for the linearized approach

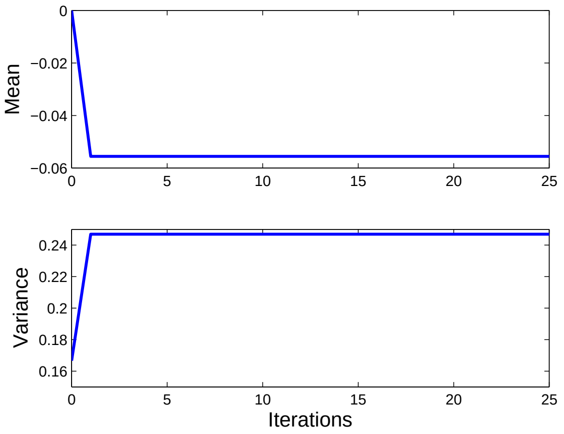

Fig. 5. Mean and variance of the output

VI. Conclusions

In this paper, we used Markov chains to describe the behavior of first and second-order IIR filters. The model allows a more precise prediction of the effect of quantization errors in digital filters implemented with short wordlengths. We calculated the transition matrix for the Markov chain with the addition of two states to represent saturation of the output. The saturation states allow one to find the best scaling factor to avoid saturation. The use of the new model was exemplified with a simple first-order filter.

References

[1] A. Antoniou, *Digital signal processing: signals, systems, and filters*. McGraw-Hill, New York, 2006.

[2] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, *Processamento Digital de Sinais: Projeto e Análise de Sistemas*. São Paulo: Bookman, 2004.

[3] Y. Montenegro Maluenda, J. C. M. Bermudez, and V. H. Nascimento, “Modeling finite precision LMS behavior using Markov chains,” in *Proc., ICASSP 2006*, vol. III, Toulouse, France, pp. 97–100.

[4] Y. R. Montenegro Maluenda, J. C. M. Bermudez, and V. H. Nascimento, “Propriedades do algoritmo LMS operando em precisão finita,” in *Anais do XXII Simpósio Brasileiro de Telecomunicações*, Campinas, SP, 2005, pp. 1–7.

[5] A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, 2nd ed. McGraw-Hill, 1993.

[6] J. R. Norris, *Markov Chains*. Cambridge University Press, 1998.

[7] D. Bertsekas and J. Tsitsiklis, *Introduction to probability*. Athena Scientific Belmont, Massachusetts, 2002.

[8] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, USA: SIAM, 2000.