Robust and Practical Analog-to-Digital Conversion With Exponential Precision

Ingrid Daubechies, *Fellow, IEEE*, and Özgür Yılmaz, *Member, IEEE*

***Abstract*—Beta-encoders with error correction were introduced by Daubechies, DeVore, Güntürk and Vaishampayan as an alternative to pulse-code modulation (PCM) for analog-to-digital conversion. An N-bit beta-encoder quantizes a real number by computing one of its N-bit truncated β-expansions where $\beta \in (1,2)$ determines the base of expansion. These encoders have (almost) optimal rate–distortion properties like PCM; furthermore, they exploit the redundancy of beta-expansions and thus they are robust with respect to quantizer imperfections. However, these encoders have the shortcoming that the decoder needs to know the value of the base of expansion β, a gain factor in the circuit used by the encoder, which is an impractical constraint. We present a method to implement beta-encoders so that they are also robust with respect to uncertainties of the value of β. The method relies upon embedding the value of β in the encoded bitstream. We show that this can be done without *a priori* knowledge of β by the transmitting party. Moreover the algorithm still works if the value of β changes (slowly) during the implementation.**

***Index Terms*—Analog-to-digital (A/D) conversion, beta-encoders, beta-expansions, quantization, sampling, sigma–delta.**

I. Introduction

ANALOG-to-digital (A/D) conversion consists of two steps: *sampling* and *quantization*. We shall assume that f, the analog signal of interest, is a bandlimited function, i.e., its Fourier transform $\hat{f}$ vanishes outside a bounded interval. In this case, the standard sampling theorem states that we can reconstruct the signal from its sample values on a sufficiently dense grid. In particular, if the support of $\hat{f}$ is contained in $[-\Omega,\Omega]$, the sequence $\{f(n\pi/\Omega)\}_{n\in\mathbb{Z}}$ determines f via

$$f(t)=\sum_n f\left(\frac{n\pi}{\Omega}\right)\ \mathrm{sinc}\ \left(t-\frac{n\pi}{\Omega}\right). \tag{1}$$

In practice, (1) is not very useful because the "sinc kernel" decays slowly, and thus the reconstruction is not local. However, we can overcome this problem by sampling on a denser grid, in which case we can replace the sinc in (1) by a kernel with much faster decay. In particular, if instead of the sample sequence $\{f(n\pi/\Omega)\}_{n\in\mathbb{Z}}$ the more closely spaced samples $\{f(n\pi/(\lambda\Omega))\}_{n\in\mathbb{Z}}$ (with $\lambda>1$) are used, (1) can be replaced by

$$f(t)=\frac{1}{\lambda}\sum_n f\left(\frac{n\pi}{\lambda\Omega}\right)\ \varphi\left(t-\frac{n\pi}{\lambda\Omega}\right) \tag{2}$$

where φ is any function such that $\hat{\varphi}(\xi)=1$ for $|\xi|\le\Omega$ and $\hat{\varphi}(\xi)=0$ for $|\xi|\ge\lambda\Omega$. If, in addition, we choose φ such that $\hat{\varphi}$ is smooth, φ will have the desired decay properties.

The second step in A/D conversion is *quantization*, which will be our focus in this paper. Quantization is the replacement of the sample values $f(n\pi/(\lambda\Omega))$ in the reconstruction formula (2) by numbers from a discrete (usually finite) set. Unlike sampling, quantization is not an invertible operation. Therefore, a *quantization scheme* is usually described by a pair of mappings, an encoder E and a decoder D. The encoder E maps functions of interest to bitstreams; the decoder D aims to invert E as "closely" as possible. In general, however, $D(Ef)\neq f$.

In this paper, we focus on quantization algorithms for functions in $S(\Omega,M)$, the class of functions in $L^2(\mathbb{R})$ with L^∞-norm bounded by M, and with Fourier transforms supported in $[-\Omega,\Omega]$. When considering functions in $S(\Omega,M)$, we shall refer to any interval of length π/Ω as a *Nyquist interval*. In what follows we shall consider, without loss of generality, only the case $\Omega=\pi$ and $M=1$, and denote $S(\pi,1)$ by S. For any quantization scheme, or, equivalently, for any encoder–decoder pair (E,D), we define the *distortion* d by

$$d(S;E,D):=\sup_{f\in S}\|f-D(Ef)\| \tag{3}$$

where $\|\cdot\|$ is the norm of interest. The performance of a quantization scheme on S is typically characterized by the relation between the distortion d and the number B of bits per Nyquist interval consumed (on average) by the encoder E associated with the quantization scheme; we call this number the *bit budget* of the quantization scheme. A typical way of comparing the performances of two quantization schemes is to compare the bit budgets they utilize to produce approximations with the same amount of distortion; if no other considerations prevail, the quantization scheme with the smaller bit budget is superior, and is preferable. Equivalently, one might compare the distortions associated with several quantization schemes for the same fixed bit budget; in this case, the scheme with the smaller distortion is favored.

One widely used quantization scheme is pulse-code modulation (PCM). Given a function f in S, an N-bit PCM algorithm simply replaces each sample value $f(n/\lambda)$ with N bits: one bit for its sign, followed by the first $N-1$ bits of the binary expansion of $|f(n/\lambda)|$. One can show that for signals in

Manuscript received September 2, 2005; revised April 6, 2006. The work of I. Daubechies was supported in part by the National Science Foundation under Grant DMS-0219233 and by the Air Force Office of Scientific Research under Grant F49620-01-0099. The work of Ö. Yılmaz was supported in part by the Natural Science and Engineering Research Council of Canada.

I. Daubechies is with the Department of Mathematics and with the Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544 USA (e-mail: ingrid@math.princeton.edu).

Ö. Yılmaz is with the Department of Mathematics, The University of British Columbia, Vancouver, BC V6T 1Z2 Canada (e-mail: oyilmaz@math.ubc.ca).

Communicated by V. A. Vaishampayan, Associate Editor At Large.

Digital Object Identifier 10.1109/TIT.2006.878220

S, this algorithm achieves a precision of order $O(2^{-N})$; i.e., for f in S, the distortion $d(f;E,D) \leq C2^{-N}$ when (E,D) is the encoder–decoder pair associated with an N-bit PCM, and when the norm $\|\cdot\|$ to measure the distortion is either the L^∞ norm on $\mathbb{R}$, or the limit, as $T \to \infty$, of the normalized L^2-norm $(2T)^{-1/2}\|\cdot\|_{L^2(-T,T)}$. This can also be rephrased in terms of the bit-budget: since the average number of bits per Nyquist interval is $B := N\lambda$, we have $d(f,E,D) \leq C_\lambda 2^{-B/\lambda}$ for $f \in S$, where $\lambda > 1$ can be chosen arbitrarily, with $C_\lambda \leq C'(\lambda - 1)^{-1/2}$ tending to ∞ as $\lambda \to 1$.

On the other hand, *sigma–delta modulation*, another commonly implemented quantization algorithm for functions in S, achieves a precision that decays like only an inverse polynomial in the bit budget B. For example, a kth-order sigma–delta scheme produces an approximation where the L^∞ approximation error (i.e., distortion) is of order $O(B^{-k})$.

Despite their clearly inferior performance in terms of distortion for a given bit budget, sigma–delta modulators are often *preferred* over PCM for A/D conversion of audio signals in practice. This must mean that other factors are more important. A plausible explanation for the prevalence of sigma–delta modulators is given by the robustness properties of these schemes, when implemented in analog circuits. Every quantization scheme for A/D conversion must contain (at least one) nonlinear "decision element" that "performs the quantization," i.e., transforms its real number input into an element of a (finite) alphabet; we shall call this nonlinear element the *quantizer*, reserving this name for this use only. These quantizers are bound to be imprecise, i.e., the location of their "toggle point(s)" is typically not known with high precision. Under these circumstances, PCM performs poorly whereas the distortion associated with a sigma–delta scheme is hardly affected at all [1]; more precisely, the error made by a PCM quantization scheme with an imprecise quantizer is bounded below by a strictly positive constant (directly related to the measure of imprecision of the quantizer), whereas the error made by a kth-order sigma–delta scheme still decays like B^{-k}, and therefore can be made arbitrarily small despite the imprecision in the quantizer.

One of several reasons why sigma–delta schemes are robust is that they quantize redundant expansions of the original function. Since redundant expansions are not unique, any error caused by an imprecise quantizer can be implicitly corrected later. This is clearly not the case for PCM, since the binary expansion of almost every real number is unique, and therefore a bit flip in the expansion cannot be corrected by changes in the consecutive bits. For a detailed discussion of robustness properties of sigma–delta schemes as well as a comparison with PCM, we refer to [2], [3], [1], [4].

It is natural to wonder whether we can have the best of both worlds. In other words, can we construct a quantization algorithm that utilizes the bit-budget efficiently, like PCM, but that is at the same time robust with respect to unavoidable imprecision errors in the (analog) circuit implementation, like sigma–delta modulators? In [1], two quantization schemes are discussed that show that the answer to this question is affirmative: a modified, filtered sigma–delta scheme, first proposed in [5], and a scheme that is introduced in [1] for the first time (as far as we know), called a "beta-encoder with error correction." These beta-encoders are PCM-like in that they quantize each sample finely and achieve exponential precision for the resulting approximation; moreover, they are robust, like sigma–delta schemes, because this fine quantization is done in a redundant way.

Although the encoders discussed in [1] are robust against quantizer imperfections, they nevertheless still have certain robustness problems, at least from a practical point of view. Both of them use other parameters (the filter parameters for the scheme of [5], the factor β used in the beta-encoder) that need to be implemented with high accuracy. In this paper, we shall discuss this issue further for beta-encoders.

The beta-encoder of [1], which we outline in Section II, computes a "truncated β-ary expansion" for each sample value, i.e., a series expansion in a base β, where $1 < \beta < 2$, and with binary coefficients. In order to reconstruct a good approximation to the original signal value from these binary coefficients, it is crucial to know the value of β, with a precision comparable to at least the one we hope the quantization scheme to achieve. However, measuring the exact value of β as well as ensuring that it stays constant over a long stretch of time is difficult, if not impossible, in an analog setting.

In this paper, we show how this obstacle can be circumvented. We present an algorithm that encodes and transmits the value of β without in fact physically measuring its value. This is done by embedding the value of β in the encoded bitstream for each sample value separately. Thus, we construct a beta-encoder that is not only robust to quantizer imperfections, but also robust with respect to β, as long as the value of β remains constant or varies sufficiently slowly that it does not change by more than the desired precision range of the quantization scheme over, say, 20 sampling intervals.

In Section II, we outline the "beta-encoders with error correction," introduced in [1]. In Section III, we present our "beta-robust" approach and prove that it has the robustness properties discussed above. In Section IV, we present numerical examples and consider some variations to our approach that address other robustness concerns.

II. Review of Beta-Encoders

In this section, we summarize certain properties of beta-encoders with error correction. See [1] for more details. We start again from the observation that any function $f \in S$ can be reconstructed from its sample values $f(n/\lambda)$ via

$$f(t) = \lambda^{-1} \sum_n f\left(\frac{n}{\lambda}\right)\, \varphi\left(t - \frac{n}{\lambda}\right) \tag{4}$$

where $\lambda > 1$ and $\varphi \in C^\infty$, such that $\hat{\varphi}(\xi) = 1$ for $|\xi| \leq \pi$ and $\hat{\varphi}(\xi) = 0$ for $|\xi| \geq \lambda\pi$.

Proposition 1: Let $f \in S$, $\epsilon > 0$, and consider any quantization scheme that produces q_n such that $\sup|f(n/\lambda) - q_n| \leq \epsilon$. Then the approximation $\tilde{f}$, defined as

$$\tilde{f}(t) := \lambda^{-1} \sum_n q_n\, \varphi\left(t - \frac{n}{\lambda}\right) \tag{5}$$

satisfies

$$|f(t)-\tilde{f}(t)| \le \lambda^{-1}\sum_n \epsilon \left|\varphi\left(t-\frac{n}{\lambda}\right)\right| \le C_\varphi\,\epsilon \qquad (6)$$

where $C_\varphi := \|\varphi\|_{L^1} + \lambda^{-1}\|\varphi'\|_{L^1}$.

Proposition 1 shows, for example, that if the q_n are produced by an N-bit PCM algorithm, then the function $\tilde{f}$ defined by (5) satisfies

$$\|f-\tilde{f}\|_{L^\infty} \le C_\varphi 2^{-N+1} \qquad (7)$$

because, by construction, an N-bit PCM generates q_n such that $|f(n/\lambda) - q_n| \le 2^{-N+1}$ for all n.

Next, we will consider, instead of truncated binary expansions, truncated 1-bit β expansions of the sample values. By a 1-*bit* β *expansion* we mean an expansion with binary coefficients and base $\beta \in (1,2)$. For any real number $x \in [0,1]$, and given $1 < \beta \le 2$, there exists a sequence $(b_j)_{j\in\mathbb{N}}$, with $b_j \in \{0,1\}$ such that

$$x = \sum_{j=1}^{\infty} b_j \beta^{-j}. \qquad (8)$$

Clearly, for $\beta = 2$, (8) corresponds to the binary expansion of x; for $1 < \beta < 2$, the expansion is said to be a beta-expansion, and is not unique; in fact, for every $1 < \beta < 2$ and for almost every x in $(0,1]$, the set of different β expansions of x has the cardinality of the continuum [6].

Given $x \in [0,1]$, one way to compute a binary sequence $\{b_j\}$ that satisfies (8) is to run the iteration

$$\begin{aligned} u_j &= \beta(u_{j-1} - b_{j-1}) \\ b_j &= Q(u_j) \end{aligned} \qquad (9)$$

with $u_1 = \beta x$, and

$$Q(u) := \begin{cases} 1, & u > 1 \\ 0, & u \le 1. \end{cases} \qquad (10)$$

The nonlinear operation Q in (10) is an idealized quantizer; in practice, one has to deal with quantizers that only approximate this ideal behavior (see below). The samples $f(n/\lambda)$ of a function $f \in S$ will be real numbers in $(-1,1)$ rather than $(0,1)$; we shall here perform a small "sleight of hand" and consider the value-shifted and renormalized function $g = (f+1)/2$ instead, which does take values in $(0,1)$ only. Running (9) N times for each sample value $g(n/\lambda) = [1+f(n/\lambda)]/2$, where $f \in S$, will yield an N-bit truncated beta-expansion $\{q_n := \sum_{j=1}^{N} b_j\beta^{-j}\}$ that will satisfy

$$|g(n/\lambda) - q_n| < C\beta^{-N} \quad \text{or} \quad |f(n/\lambda) - (2q_n - 1)| < C'\beta^{-N} \qquad (11)$$

where $C' = 2C = 2/(\beta-1)$. Thus, by Proposition 1, such a quantization scheme will yield an approximation to $f \in S$ with distortion of size $O(\beta^{-N})$.

Before defining the "beta-encoders with error correction" of [1], we want to repeat the key observation that is used in [1] to construct these encoders.

Proposition 2: Let $x \in (0,1)$ and $b_1, b_2, \ldots, b_J$ be arbitrary such that $b_j \in \{0,1\}$. If

$$0 \le x - \sum_{j=1}^{J} b_j\beta^{-j} \le \frac{1}{\beta^J(\beta-1)} \qquad (12)$$

then there exists an extension $\{b_j\}_{j=J+1}^{\infty}$ such that

$$x = \sum_{j=1}^{\infty} b_j\beta^{-j}. \qquad (13)$$

This is exactly where redundancy comes into play: For $\beta = 2$, the only way the bits $b_1, \ldots, b_J$ can satisfy (12) is by being the first J bits of the binary expansion of x. However, for $1 < \beta < 2$, this is not the case; one is allowed to "undershoot" (i.e., put to 0 some of the bits $b_j = 1$ in the beta-expansion produced by (9)) as long as (12) is satisfied. This means that the beta-encoder described by (9) is "semi-robust" in the sense that it can correct quantizer errors that result from underestimating certain bit values in the expansion.

This feature has the following important consequence in practice. When the beta-encoder runs the iterative algorithm (9) to quantize a real number, the specific form (10) for the quantizer Q is difficult to implement precisely. A more realistic model is obtained by replacing Q in (9) with $Q(\cdot+\delta)$ where δ is a quantity that is not known exactly, but over the magnitude of which we have some control, e.g., $|\delta| < \epsilon$ for a known ϵ. We shall call a quantization scheme "robust" if the worst approximation error it produces can be made arbitrarily small by allowing a sufficiently large bit budget, even if $\epsilon > 0$ is kept fixed. In the form described above, beta-encoders are robust only for some range $\delta_{\text{crit}} > \delta \ge 0$ that is asymmetric with respect to 0, which is the reason why we called them semi-robust. (More precisely, later bits b_ℓ, with $\ell > j$, automatically rectify the error made by assigning a value 0 to a b_j when the "ideal" value of b_j, according to (9), should have been 1. The opposite error, in which the value 1 was erroneously assigned to a b_j that should have been 0, does not get corrected by this scheme.) In [1], a modified version of the algorithm described by the recursion in (9) is introduced, which remedies this situation, i.e., it is robust for both positive and negative small δ. The basic strategy of [1] is to replace Q in (9) with $Q_\mu := Q(\cdot - \mu)$, for some appropriate μ. This means that the quantizer prefers assigning 0 to assigning 1, in those cases when perfect reconstruction with the remaining bits is not only possible but can even be done with a comfortable margin. In this way, a range for δ can be determined, symmetric with respect to 0, as specified below in Theorem 3, so that when the scheme is implemented with $Q_\mu(\cdot-\delta)$ instead of Q_μ, one still has exponentially precise reconstruction of the original sample value. More precisely, we have the following.

Theorem 3: Let $\epsilon > 0$ and $x \in (0,1)$. Suppose that in the beta-encoding of x, the procedure (9) is followed, but the quantizer $Q_\mu(\cdot-\delta)$ is used instead of the ideal Q at each occurrence, with the values of δ possibly varying from one j to the next, but always satisfying $|\delta| < \epsilon$. Denote by $(b_j)_{j\in\mathbb{N}}$ the bit sequence

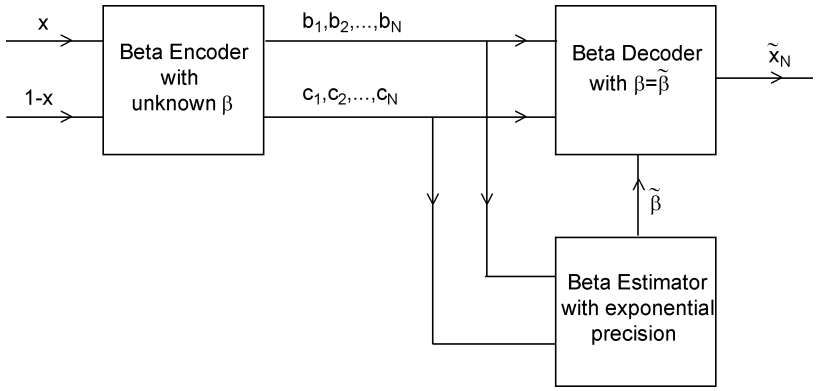

Fig. 1. The algorithm. Using the bitstreams obtained by beta-encoding both $x \in (0,1)$ and $1-x$, we estimate the value of β with exponential precision. This estimated value $\tilde{\beta}$ is then used to decode. The resulting $\tilde{x}_N$ is an exponentially precise approximation of x. Here we assume that β remains constant during the encoding of x and $1-x$, approximately on each sampling interval when quantizing the samples of a bandlimited function.

produced by applying this encoding to the number x. If $\epsilon \leq \mu$, and β satisfies

$$1 < \beta < \frac{2+\mu+\epsilon}{1+\mu+\epsilon} \tag{14}$$

then for each $N \geq 1$

$$|x - \sum_{j=1}^{N} b_j \beta^{-j}| \leq C\beta^{-N} \tag{15}$$

with $C = 1+\mu+\epsilon$.

Theorem 3 is a slightly rephrased version of Theorem 4.1 in [1]; it shows that the "beta-encoder with offset μ" has the desired robustness properties with respect to the quantizer function Q_μ. However, there is *still* one parameter in the iteration that must be known precisely to implement the "beta-encoder with offset μ" and to reconstruct the original signal from the encoded bitstream, namely, the base β of the expansion. If we were to implement the encoder in an analog setting, it seems that we would have two choices: either make sure β has a precise value, known to the decoding as well as the encoding party, or measure and transmit β before sending the encoded bits. Neither is feasible in practice: measuring the value of β physically and transmitting it separately is another potential source for errors; moreover, it would be very hard, if not impossible, to measure β with great precision. On the other hand, implementing the beta-encoder with offset μ in hardware with a precise and fixed β would be as costly as implementing a high-precision PCM. In the next section, we propose a way to circumvent this problem.

III. The Approach

In this section, we present a way of implementing the beta-encoder with offset μ in such a way that it is robust with respect to the parameter β. In particular, we shall construct an algorithm for which we shall prove that it is possible to recover the value of $\gamma := \beta^{-1}$ from the encoded bitstream with exponential precision. An outline of our algorithm is sketched in Fig. 1.

A. Description of the Algorithm

Define $E^{\mu}_{\gamma,\delta}$ as the mapping that maps each $x \in (0,1)$ to the binary sequence $(b_j)_{j=1}^{\infty}$ generated by encoding x using a beta-encoder with $\beta = \gamma^{-1}$ and with offset μ that is implemented with an imperfect quantizer $Q_\mu(\cdot - \delta)$, as described above. Our main lemma reads then as follows.

Lemma 4: Let $x \in (0,1)$ and let $\gamma = \beta^{-1} \in (1/2,1)$ be such that (14) is satisfied, with $0 < \epsilon \leq \mu$. Suppose $|\delta| < \epsilon$, and define the sequences $b := E^{\mu}_{\gamma,\delta}(x)$ and $c := E^{\mu}_{\gamma,\delta}(1-x)$. Let N be such that $\max\{b_j, c_j : j = 1, \ldots, N\} > 0$. Then, for any $\tilde{\gamma}$ that satisfies

$$0 \leq 1 - \sum_{j=1}^{N} (b_j + c_j)\, \tilde{\gamma}^j \leq 2\, C\, \tilde{\gamma}^N \tag{16}$$

with $C = 1+\mu+\epsilon$, we have

$$|\gamma - \tilde{\gamma}| \leq C' \gamma^N \tag{17}$$

where $C' = \max\{2C, 2C/(k_0 \gamma^{(k_0-1)})\}$ with $k_0 = \frac{\log(\frac{1-\gamma}{2})}{log\gamma}$.

We shall prove this lemma in several steps below. Before proceeding, we show that knowing the value of γ with exponential precision yields exponentially precise approximations.

Theorem 5: Let $x \in (0,1)$, $\gamma \in (1/2,1)$, and $(b_j)_{j\in\mathbb{N}} \in \{0,1\}$ be such that $x = \sum_{j=1}^{\infty} b_j \gamma^j$. Suppose $\tilde{\gamma}$ is such that $|\gamma - \tilde{\gamma}| \leq C_1 \gamma^N$ for some fixed $C_1 > 0$. Define

$$N_0 := \frac{\log[(1-\gamma)/C_1]}{\log \gamma}$$

and $\eta := C_1 \gamma^{N_0+1}$. Then $\tilde{x}_N := \sum_{j=1}^{N} b_j \tilde{\gamma}^j$ satisfies the inequalities

$$|x - \tilde{x}_N| \leq \begin{cases} C_1 \gamma^N \frac{1}{1-(\gamma+\eta)^2}, & N \geq N_0 + 1 \\ C_1 \gamma^N N_0^2 (\gamma + C_1\gamma)^{N_0 - 1}, & 1 \leq N \leq N_0. \end{cases} \tag{18}$$

Proof: We want to estimate

$$|x-\tilde{x}_N| = \left|\sum_{j=1}^{N} b_j(\gamma^j - \tilde{\gamma}^j)\right| \leq \sum_{j=1}^{N} |\gamma^j - \tilde{\gamma}^j| \tag{19}$$

where we have used that $b_j \in \{0,1\}$. Define now $f_j(\tilde{\gamma}) := \tilde{\gamma}^j - \gamma^j$. Clearly, $f_j(\gamma) = 0$; moreover, the derivative satisfies $|f_j'(\tilde{\gamma})| = |j\tilde{\gamma}^{j-1}| \leq j(\gamma+\Delta)^{j-1}$ for all $\gamma - \Delta \leq \tilde{\gamma} \leq \gamma + \Delta$, where $\gamma \in (1/2, 1)$ and $\Delta > 0$. Therefore,

$$|f_j(\tilde{\gamma})| = |\tilde{\gamma}^j - \gamma^j| \leq \Delta j(\gamma+\Delta)^{j-1}. \tag{20}$$

We will now estimate the right-hand side of (19) separately for the cases when N is large and when N is small.

1) Setting $\Delta = C_1\gamma^N$, and substituting (20) in (19), we get

$$|x-\tilde{x}_N| \leq C_1\gamma^N \sum_{j=1}^{N} j(\gamma + C_1\gamma^N)^{j-1} \tag{21}$$

$$= C_1\gamma^N \frac{1-(\gamma+C_1\gamma^N)^N(1+N(1-(\gamma+C_1\gamma^N)))}{(1-(\gamma+C_1\gamma^N))^2}. \tag{22}$$

For $N \geq N_0+1$, $C_1\gamma^N \leq C_1\gamma^{N_0+1} = \eta$; by its definition η satisfies $\eta < 1-\gamma$, so that $(\gamma + C_1\gamma^N) < 1$. We then rewrite (22) as

$$\begin{aligned}|x-\tilde{x}_N| &\leq C_1\gamma^N \frac{1}{(1-(\gamma+C_1\gamma^N))^2} \\ &\leq C_1\gamma^N \frac{1}{(1-(\gamma+\eta))^2}\end{aligned}$$

which provides us with the desired bound.

2) Suppose that $1 \leq N \leq N_0$, which means $(\gamma + C_1\gamma^N) \geq 1$. Set again $\Delta = C_1\gamma^N$. For each $j = 1, \ldots, N$, we clearly have

$$\begin{aligned}|\tilde{\gamma}^j - \gamma^j| &\leq C_1\gamma^N j(\gamma + C_1\gamma^N)^{j-1} \\ &\leq C_1\gamma^N N_0(\gamma + C_1\gamma)^{N_0-1}\end{aligned} \tag{23}$$

where the second inequality holds because $j \leq N \leq N_0$ and $N \geq 1$. Substituting (23) in (19), and using that $N \leq N_0$ yields the desired estimate, i.e.,

$$|x-\tilde{x}_N| \leq \gamma^N C_1 N_0^2(\gamma + C_1\gamma)^{N_0-1}. \qquad \square$$

Remarks:

1) Combining Lemma 4 and Theorem 5, we see that one can recover the encoded number $x \in (0,1)$ from the first N bits of $E^{\mu}_{\gamma,\delta}(x)$ and $E^{\mu}_{\gamma,\delta}(1-x)$, with a distortion that decreases exponentially as N increases.
2) Given N-bit truncated bitstreams $E^{\mu}_{\gamma,\delta}(x)$ and $E^{\mu}_{\gamma,\delta}(1-x)$, one way to estimate $\tilde{\gamma}$ is doing an exhaustive search. Clearly, this is computationally intensive. Note, however, that the search will be done in the digital domain, where computational constraints are not heavy, and computation speed is high.
3) In Section IV-B we introduce a fast algorithm that can replace exhaustive search; moreover, its performance is as good as exhaustive search.

B. Proof of the Main Lemma

In what follows, we shall present several observations that lead us to a proof of Lemma 4 at the end of the section.

Proposition 6: Let $\beta \in (1,2)$, $\gamma := 1/\beta$, and $x \in (0,1)$. Suppose μ, ϵ, and δ are such that the conditions of Theorem 3 are satisfied. Let $b := E^{\mu}_{\gamma,\delta}(x)$ and $c := E^{\mu}_{\gamma,\delta}(1-x)$. Then

$$0 \leq 1 - \sum_{j=1}^{N}(b_j + c_j)\gamma^j \leq 2C\gamma^N \tag{24}$$

where $C = 1 + \mu + \epsilon$ with $|\delta| < \epsilon \leq \mu$.

Proof: By Theorem 3, we know that

$$0 \leq x - \sum_{j=1}^{N} b_j\gamma^j \leq C\gamma^N \tag{25}$$

and

$$0 \leq 1 - x - \sum_{j=1}^{N} c_j\gamma^j \leq C\gamma^N \tag{26}$$

hold. Combining (25) and (26) yields the result. $\square$

Note that $d_j := b_j + c_j \in \{0,1,2\}$. Moreover, the index of the first nonzero entry of $\{d_j\}_{j=1}^{\infty}$ cannot be arbitrarily large; more precisely we have the following.

Proposition 7: Let $k := \min\{j : j \in \mathbb{N} \text{ and } d_j \neq 0\}$. Then

$$k \leq \frac{\log(\frac{1-\gamma}{2})}{\log\gamma}. \tag{27}$$

Proof: Let k be as defined above. Clearly, with $d_j = b_j + c_j$ as above, $\sum_{j=k}^{\infty} d_j\gamma^j = 1$. On the other hand, since $d_j \in \{0,1,2\}$

$$\sum_{j=k}^{\infty} d_j\gamma^j \leq 2\sum_{j=k}^{\infty}\gamma^j = \frac{2\gamma^k}{1-\gamma}. \tag{28}$$

Therefore, we have

$$\frac{2\gamma^k}{1-\gamma} \geq 1 \tag{29}$$

which yields the desired result. $\square$

We now define

$$F_n(t) := 1 - \sum_{j=k}^{n} d_j(\gamma + t)^j \tag{30}$$

on $[-\gamma, \infty)$ for $n = k, k+1, \ldots$.

Proposition 8: F_n has the following properties:

(i) $0 \leq F_n(0) \leq 2C\gamma^n$.
(ii) F_n is monotonically decreasing. Moreover, the graph of F_n is concave for all $n \in \{2,3,\ldots\}$.

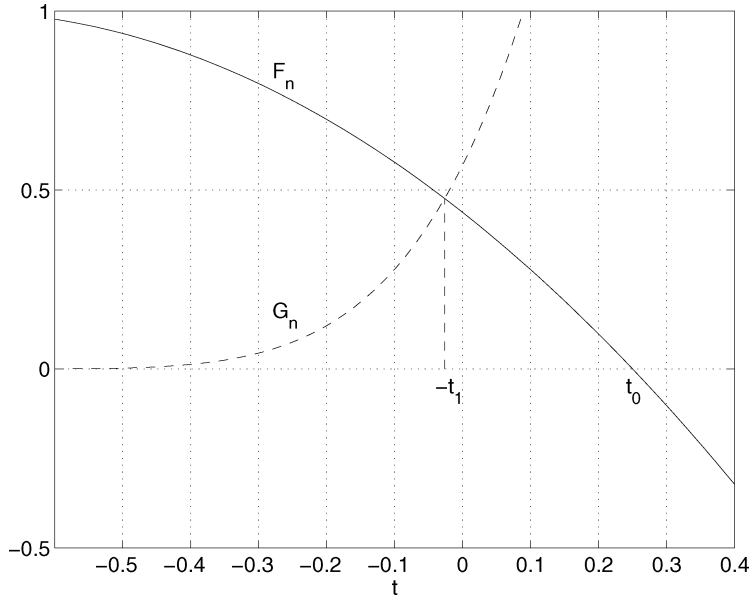

Fig. 2. A sketch of the graph of F_n along with $G_n(t) = 2C(\gamma + t)^n$. Here $n = 5$ and F_n was computed via (30) from the first n bits of the sequence $E^{\mu}_{\gamma,\delta}(x)$ with $x = 0.7, \gamma = 0.75, \mu = 0.2$, and $|\delta| < 0.1$.

Proof:

(i) This is Proposition 6 restated.

(ii) First, note that

$$F_n'(t) = -\sum_{j=k}^{n} j d_j (\gamma + t)^{j-1}.$$

Thus, since $d_j \geq 0$ for all j, $F_n'(t) < 0$ for $t > -\gamma$. Moreover, a similar calculation shows that the second derivative of F_n is also negative. Therefore, the graph of F_n is concave. □

Fig. 2 shows a sketch of the graph of F_n. Define, as shown in Fig. 2, t_0 as the point at which $F_n(t_0) = 0$. Similarly, let t_1 be such that $F(-t_1) = 2C(\gamma - t_1)^n$. We will show that both t_0 and t_1 are at most of size $O(\gamma^n)$, which will lead us to our main result.

Lemma 9: Let F_n be as in (30) where $n \geq k$ is a positive integer. Then t_0, as defined above, satisfies

$$0 \leq t_0 \leq C_1 \gamma^n \tag{31}$$

with $C_1 = \frac{2C}{k\gamma^{k-1}}$ where C is as in Theorem 3 and k is as in Proposition 7.

Proof: Since F_n is deceasing and $F_n(0) \geq 0$, it follows that $t_0 \geq 0$. Moreover, $F_n(0) = F_n(0) - F_n(t_0) = |F_n'(\xi)|t_0$ for some $\xi \in (0, t_0)$, so that

$$t_0 \leq F_n(0) \left[\inf_{\xi \in (0,t_0)} |F_n'(\xi)| \right]^{-1} \leq 2C\gamma^n |F_n'(0)|^{-1}. \tag{32}$$

Finally, since $F_n'(0) = -\sum_{j=k}^{n} j d_j \gamma^{j-1}$ and since $d_j \geq 0$, we have

$$|F_n'(0)| \geq k\gamma^{k-1} \tag{33}$$

where k is as in Proposition 7. Combining (33) with (32) above yields the result. □

Lemma 10: Let F_n be as in (30) where $n \geq k$ is a positive integer. Then t_1, as defined above, satisfies

$$0 \leq t_1 \leq C_1 \gamma^n \tag{34}$$

with $C_1 = \frac{2C}{k\gamma^{k-1}}$ as in Lemma 9.

Proof: Let $G_n(t) := 2C(\gamma + t)^n$ and recall that $F_n(-t_1) = G_n(-t_1)$. Note, by Proposition 8, that $F_n(0) \leq G_n(0)$. Also, we have $F_n(-\gamma) = 1 > G_n(-\gamma) = 0$. Therefore, since F_n is decreasing and G_n is increasing on $[-\gamma, \infty)$, we have $-\gamma < -t_1 \leq 0$, i.e., the first inequality in (34).

Next, note that because $F_n(-t_1) = 2C(\gamma - t_1)^n$ and $F_n(0) \geq 0$, we have

$$2C(\gamma - t_1)^n = F_n(-t_1) \geq F_n(-t_1) - F_n(0) = |F_n'(\zeta)|t_1$$

for some $\zeta \in (-t_1, 0)$, so that

$$\begin{aligned} 2C(\gamma - t_1)^n &\geq t_1 \inf_{\zeta \in (-t_1, 0)} |F_n'(\zeta)| \\ &= t_1 |F_n'(-t_1)| = t_1 \sum_{j=k}^{n} j d_j (\gamma - t_1)^{j-1} \\ &\geq t_1 k (\gamma - t_1)^{k-1}. \end{aligned} \tag{35}$$

Now, since $\gamma - t_1 > 0$, (35) implies

$$t_1 \leq \frac{2C}{k} (\gamma - t_1)^{n-k+1}. \tag{36}$$

Finally, we conclude

$$t_1 \leq \frac{2C}{k\gamma^{k-1}} \gamma^n \tag{37}$$

since $0 < \gamma - t_1 \leq \gamma$ and $n \geq k$. □

We are now ready to prove Lemma 4.

Proof: (Proof of Lemma 4) Let x, γ, μ, ϵ, b, c, and N be as in the statement of the lemma. Let F_N be as in (30), and suppose $\tilde{\gamma} > 0$ satisfies (16), which can be rewritten as

$$0 \leq F_N(\tilde{\gamma} - \gamma) \leq 2C(\gamma + (\tilde{\gamma} - \gamma))^N. \tag{38}$$

By the monotonicity of F_N, proved in Proposition 8, this implies that

$$-t_1 \leq \tilde{\gamma} - \gamma \leq t_0 \tag{39}$$

where t_0 and t_1 are as in Lemmas 9 and 10, respectively. Thus, we have

$$|\gamma - \tilde{\gamma}| \leq C_1 \gamma^N \tag{40}$$

where $C_1 = (2C)/(k\gamma^{k-1})$

with

$$k \leq \frac{\log(\frac{1-\gamma}{2})}{\log \gamma} =: k_0$$

as in Proposition 7. Finally, since the function $g(x) := x\gamma^{x-1}$ attains its only local maximum at $x = 1/\log(\gamma^{-1}) > 1$, we conclude that $C_1 \leq \max\{2C, 2C/(k_0\gamma^{k_0-1})\} =: C'$. □

IV. Algorithms to Approximate β and Numerical Experiments

In the previous section we showed that even when β is unknown (but fixed), it is possible to recover it with exponential precision, and thus to reconstruct the samples x, from their encoding, with exponential precision as well. We did not yet address how to estimate β, or rather γ, in practice; this is the subject of the present section. The first approach we present computes γ by carrying out an exhaustive search for the value(s) $\tilde{\gamma}$ that satisfy (16). We carry out numerical experiments illustrating the discussion and touch upon some other robustness issues.

A. Exhaustive Search, Several Robustness Issues, and Numerical Experiments

The algorithm that we presented in the previous section relies on the fact that the two transmitted bitstreams are produced by quantizing both $x \in (0, 1)$ and $1 - x$ using a beta-encoder with offset μ with a fixed value of β. In order to decode this information, so as to recover x, it is clear that the decoder must evaluate γ or β (and probably re-evaluate it every so often as the encoded samples are received). In this subsection, we assume this is done by an exhaustive search for $\tilde{\gamma}_N$ that satisfy (16). We show the results of numerical experiments for $x = \pi/10$ and $\gamma = 0.75$ (unless otherwise noted). We denote by $\tilde{x}_N$ the approximation of x produced by combining the original N-bit truncated beta-expansion coefficients b_n of x with powers of the estimated value $\tilde{\gamma}_N$ of γ, as in Theorem 5. We use these numerical examples to illustrate several robustness issues.

1) Our algorithm is robust with respect to the quantizer imperfections in the sense of Theorem 3. That is, if the scheme described in (9) is implemented with $Q_\mu(\cdot - \delta)$ instead of Q, we can still estimate the value of γ with exponential precision with respect to the bit rate. In fact, our results in the previous section are for these imperfect quantizers. In Fig. 3(a) and (b), we plot $|\tilde{\gamma}_N - \gamma|$ and $|\tilde{x}_N - x|$ versus N, respectively, in the case of a precise beta-encoder ($\delta = 0$) with offset $\mu = 0.2$. Fig. 4(a) and (b), on the other hand, shows the same quantities, respectively, when the approximations are obtained via an imperfect beta-encoder with $\mu = 0.2$ and $\epsilon = 0.15$ (recall that ϵ is the bound on δ; Fig. 4 used a simulation where the δ_j did not even remain constant from one sample to the next; they were picked randomly, uniformly in $(-\epsilon, \epsilon)$).
2) We can still recover the value of γ with the same precision if instead of x and $1 - x$, the transmitted bitstreams correspond to x and $1 + \rho - x$ as long as $|\rho| \leq \epsilon_0$ for some sufficiently small ϵ_0. More precisely, if $0 \leq 2\epsilon_0 \leq 2C\gamma^N$,

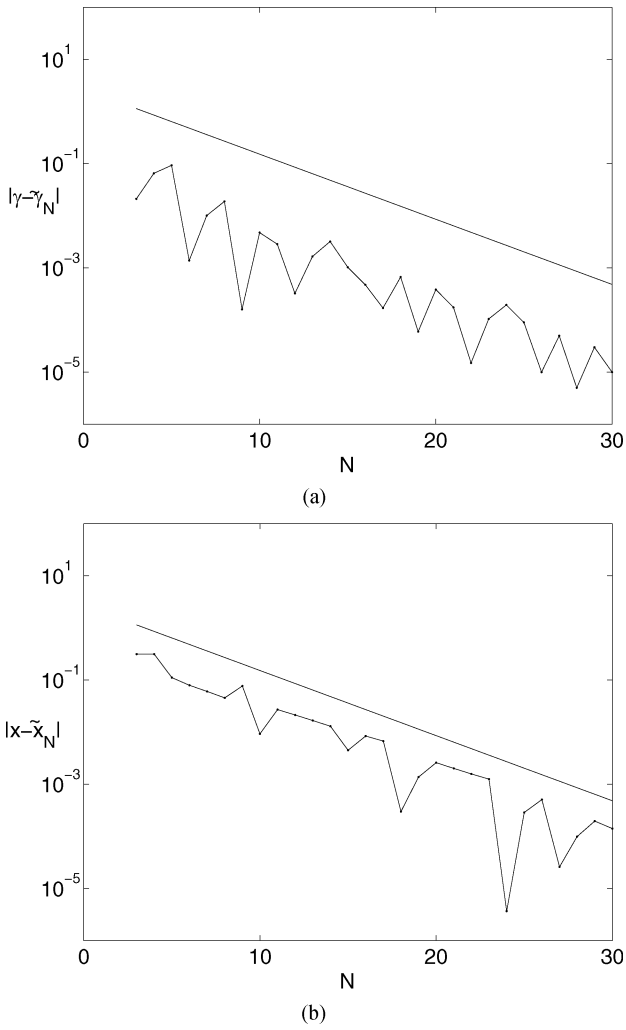

Fig. 3. The performance of the algorithm when implemented using a precise beta-encoder with offset $\mu = 0.2$. (a) Error $|\gamma - \bar{\gamma}_N|$ where $\bar{\gamma}_N$ was computed from N-bit beta-expansions of $x = \pi/10$ and $1 - x$ via an exhaustive search algorithm. (b) Approximation error $|x - \bar{x}_N|$ where $\bar{x}_N$ was computed using the estimated $\bar{\gamma}_N$ along with the N-bit beta-expansion of x.In both (a) and (b), the vertical axes are logarithmic, and the straight lines are the graph of $2C\gamma^N$ with $C = 1 + \mu$.

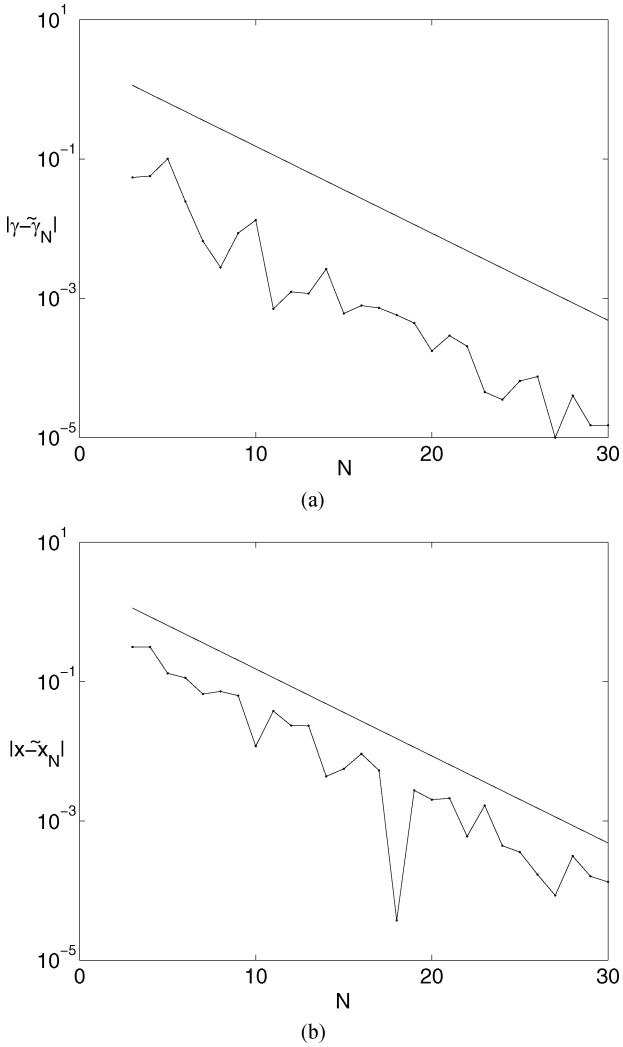

Fig. 4. The experiment of Fig. 3 repeated; however this time an imperfect beta-encoder with offset $\mu = 0.2$ and $\epsilon = 0.15$ was used. (See text.)

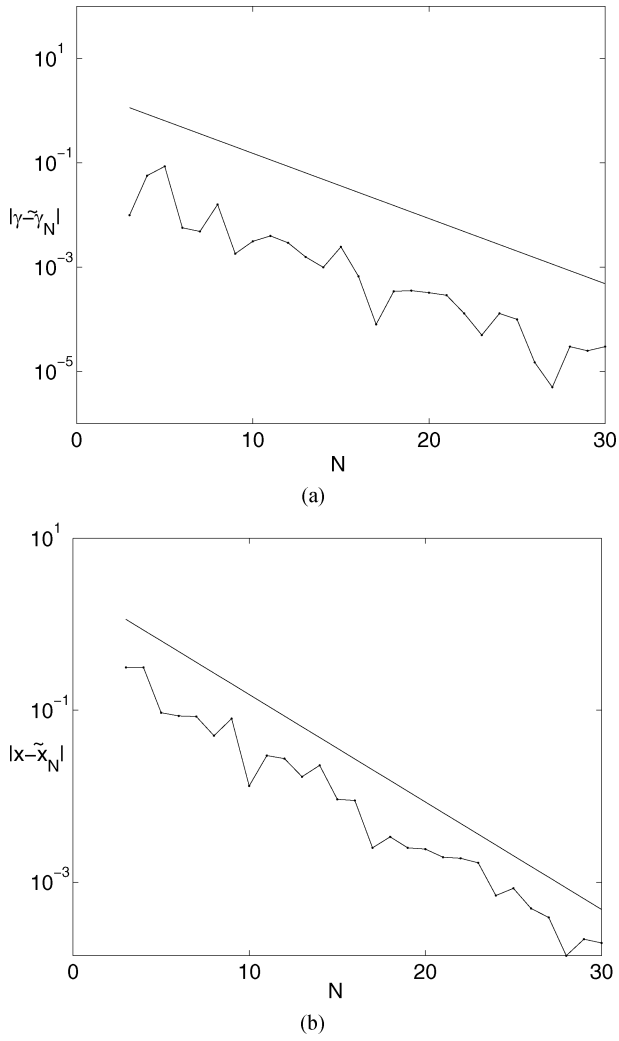

Fig. 5. The performance of the algorithm when implemented with $x = \pi/10$ and $1+\rho-x$. Here the "uncertainty" ρ was taken to be $\gamma^N/2$ (when the number of bits used to compute $\tilde{\gamma}_N$ is N).

where C is as in Theorem 3, we can replace (16) in Theorem 4 with the more stringent condition on $\tilde{\gamma}$

$$\epsilon_0 \leq 1 - \sum_{j=1}^{N} (b_j + c_j)\tilde{\gamma}^j \leq 2C\tilde{\gamma}^N - \epsilon_0 \tag{41}$$

and the theorem remains valid, i.e., it follows that $|\gamma - \tilde{\gamma}| \leq C'\gamma^N$. Fig. 5(a) and (b) shows $|\tilde{\gamma}_N - \gamma|$ and $|\tilde{x}_N - x|$ versus N, respectively, in this case with $\rho = \gamma^N/2$. This observation means that if the reference level can be kept fixed within a precision of ϵ_0, then the scheme still works up to $N_0 \sim \log \epsilon_0 / \log \gamma$.

3) If we use the pair x and $a - x$ instead of x and $1 - x$ for some $a \in (0,1)$, our approach still works, provided that $0 \leq x \leq a$ and that we *know* the value of a, at least with a precision of ϵ_0 that satisfies $0 \leq 2\epsilon_0 \leq 2C\gamma^N$, as above. In Fig. 6(a) and (b), we plot $|\tilde{\gamma}_N - \gamma|$ and $|\tilde{x}_N - x|$ versus N, respectively, when the algorithm is implemented using the pair x and $a - x$ with $a = 0.9$.
4) Our algorithm requires that the value of γ remains constant during the quantization of the numbers x and $1-x$, approximately on each sampling interval when quantizing samples of a bandlimited function. On the other hand, the algorithm still works if γ changes, however drastically, from one sampling interval to another.

As already alluded to in point 2), our approach has introduced a *new* robustness issue: typically, we cannot ensure that the reference level (1 in the case where we encode the pair x and $1-x$, $1+\rho$ in point 2)) is known with high precision, which is a new practical hurdle. More generally, in practice we would face a situation where pairs of values $(x, a-x)$ are encoded, where a is *not* known with great precision. If (as is reasonable) we can ensure that the unknown reference level a remains constant or varies very slowly only, then we still can estimate γ as follows. Suppose the beta-encoding of $x,\ a-x,\ y,\ a-y$ leads to the bit sequences $(b_j)_{j\in\mathbb{N}}, (c_j)_{j\in\mathbb{N}}, (\tilde{b}_j)_{j\in\mathbb{N}}, (\tilde{c}_j)_{j\in\mathbb{N}}$, respectively. Define $d_j = b_j + c_j$, and $\tilde{d}_j = \tilde{b}_j + \tilde{c}_j$. Then

$$\sum_{j=1}^{\infty} d_j \gamma^j = \sum_{j=1}^{\infty} \tilde{d}_j \gamma^j :$$

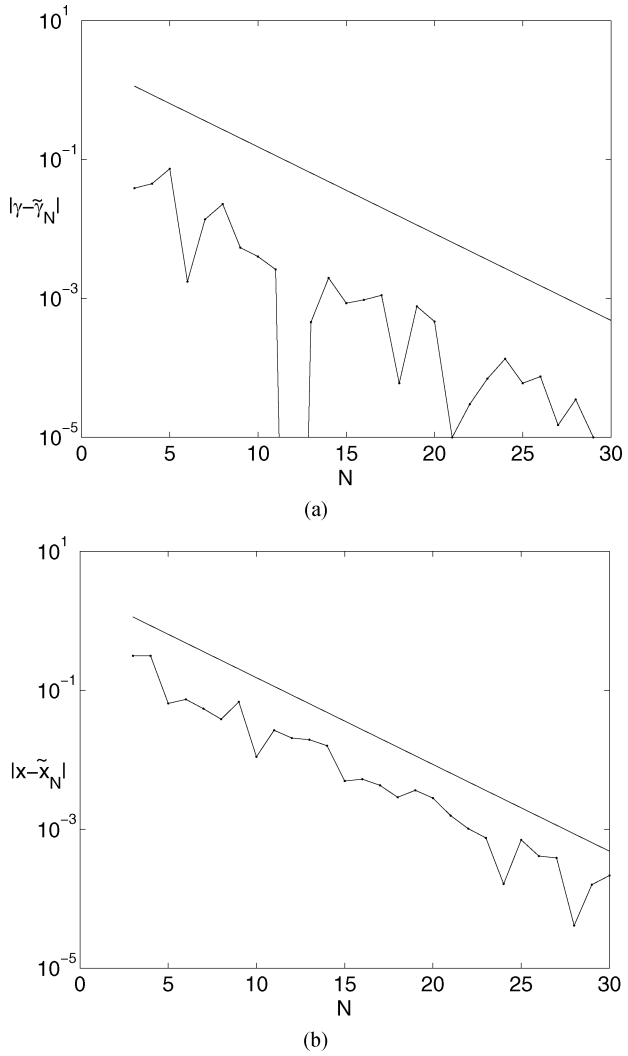

Fig. 6. The performance of the algorithm when implemented with $x = \pi/10$ and $a - x$ with $a = 0.9$. Here the value of a was *known* by the decoder.

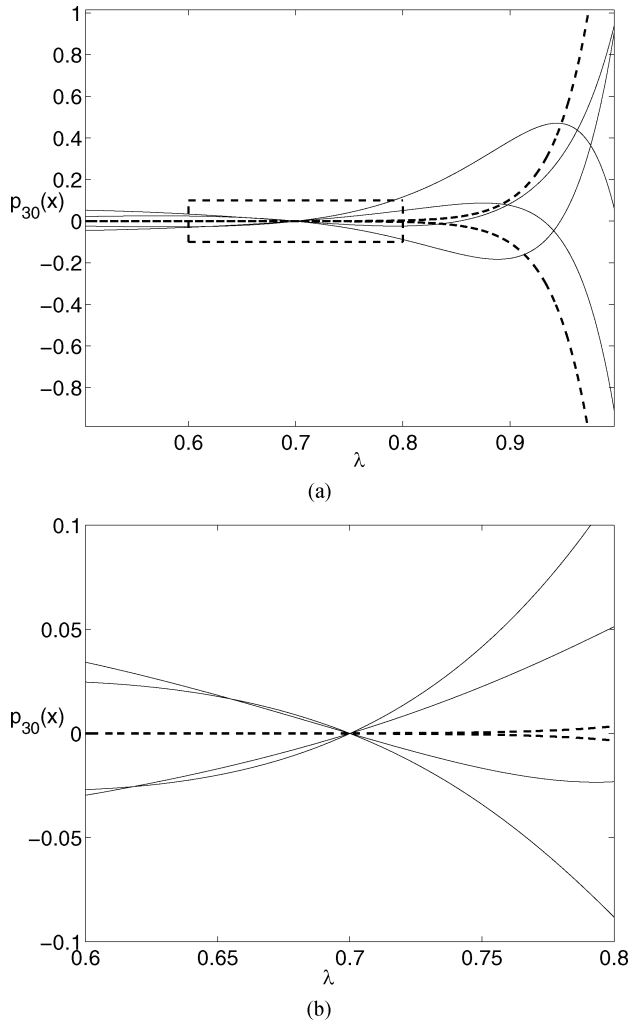

Fig. 7. (a) The graph of $p_{30}(\lambda)$ for several (x, y) pairs. The dashed curves are graphs of $\pm 2C\lambda^n$, corresponding to the constraint of (42). We observe that (42) is satisfied for values $\tilde{\gamma}$ close to 1 as well as close to the true γ. (b) A zoomed-in version of (a) to $\lambda \in [0.6, 0.8]$. In this interval and for these (x, y) pairs, (42) is satisfied in only a small neighborhood of the true γ value ($\gamma = 0.700067$ in this case).

if we put $k = min\{j \in \mathbb{N} : d_j \neq \tilde{d}_j\}$, then

$$d_k - \tilde{d}_k = \sum_{j=1}^{\infty} (\tilde{d}_{k+j} - d_{k+j})\gamma^j;$$

clearly, this puts some constraint on the possible values of γ. However, because the $\boldsymbol{d}_j = \tilde{d}_j - d_j \in \{-2, -1, 0, 1, 2\}$ can now take negative values as well, the arguments used in the proofs in Section III no longer hold. We have therefore no guarantee that we will obtain an estimate for γ with precision of size $O(\gamma^N)$ if we know the first N entries of the sequences b, c, $\tilde{b}$, $\tilde{c}$; similarly, we cannot put an upper bound, *a priori*, on $k = \min\{j \in \mathbb{N} : d_j \neq \tilde{d}_j\}$. More precisely, arguments similar to those in Proposition 6 show that

$$-2C\gamma^N \leq \sum_{j=k}^{N} \boldsymbol{d}_j \gamma^{j-k} \leq 2C\gamma^N$$

with C as in Proposition 6. This suggests we replace the constraint (16) of Lemma 4 with

$$-2C\tilde{\gamma}^N \leq \sum_{j=k}^{N} \boldsymbol{d}_j \tilde{\gamma}^{j-k} \leq 2C\tilde{\gamma}^N \tag{42}$$

and expect any $\tilde{\gamma}$ that satisfies (42) will be an exponentially precise estimate of γ. But since in this case the polynomial $p_N(\lambda) := \sum_{j=k}^{N} \boldsymbol{d}_j \lambda^{j-k}$ is not strictly decreasing, we face two major difficulties.

a) The constraint (42) may be satisfied for values $\tilde{\gamma}$ close to 1, regardless of what the value of γ is because the constraint gets quite weak when $\tilde{\gamma}$ is close to 1, and $p_N(\tilde{\gamma})$ *can* be small enough to satisfy this weak constraint. Fig. 7(a) shows samples of such p_N, obtained by encoding $x, a - x, y, a - y$ as described above with $\gamma = 0.700067$ for

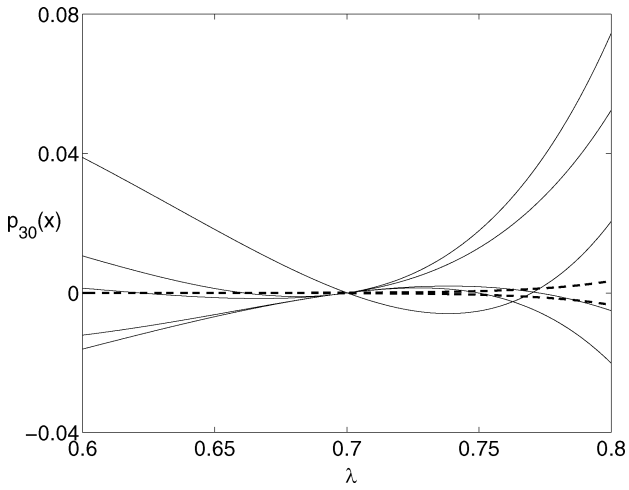

Fig. 8. The graph of $p_{30}(\lambda)$ for several other (x, y) pairs. The dashed curves are graphs of $\pm 2C\lambda^3 0$, corresponding to the constraint of (42). We observe that for these pairs, p_{30} has multiple roots in $[0.6, 0.8]$ in small neighborhoods of which (42) is satisfied.

several x and y that cause the above mentioned difficulty. One could overcome the problem outlined above by restricting the values of γ to a narrower interval that is known *a priori*. Indeed, if we were given that $\gamma \in [0.6, 0.8]$, then, as observed in Fig. 7(b), the constraint (42) is satisfied only in a small neighborhood of the true value of γ. Another way of dealing with this problem is to aim directly for the root(s) of the polynomial p_N in $(1/2, 1)$ (instead of searching for some $\tilde{\gamma}$ that satisfies (42)). This will be discussed further in Section IV-B.

b) Another problem arises because the polynomial $p_N(\lambda) := \sum_{j=k}^{N} \boldsymbol{d}_j \lambda^{j-k}$ *can* have several roots in the interval $(0.5, 1)$ (as well as in the narrower interval which we know, *a priori*, to contain γ). This would in turn mean that (42) is satisfied in sufficiently small neighborhoods of each of these roots (note that p_N is a polynomial, and thus smooth). Fig. 8 shows examples of such p_N that are obtained by encoding x, $a-x$, y, $a-y$ as described above for several x and y. (Note that the examples in Figs. 7 and 8, although they may seem contrived, stem from actual coding simulations—these problems *can* occur in practice.) In this case, clearly, computing the root(s) of p_N in $(1/2, 1)$ will not help as we have several roots. Moreover, these roots may be very close to each other so that even if we know *a priori* that γ lies in some restricted interval, this might not suffice to exclude all but one root. On the other hand, we know that one of the roots of p_N is always approximately at the right γ value independently of what (x, y) pair is used (this follows from (42) because p_N is smooth), and it is reasonable to expect the other root(s) to be located at different points in the interval $(0.5, 1)$ depending on the (x, y) pair that was used to obtain p_N. This motivates us to compute the polynomials $p_{N,i}$ corresponding to several (x_i, y_i) pairs as above, and consider the polynomial $P_{N,I} := \sum_{i=1}^{I} \epsilon_i p_{N,i}$ where we choose $\epsilon_i \in \{-1, 1\}$ in a way that would guarantee (or increase the probability)

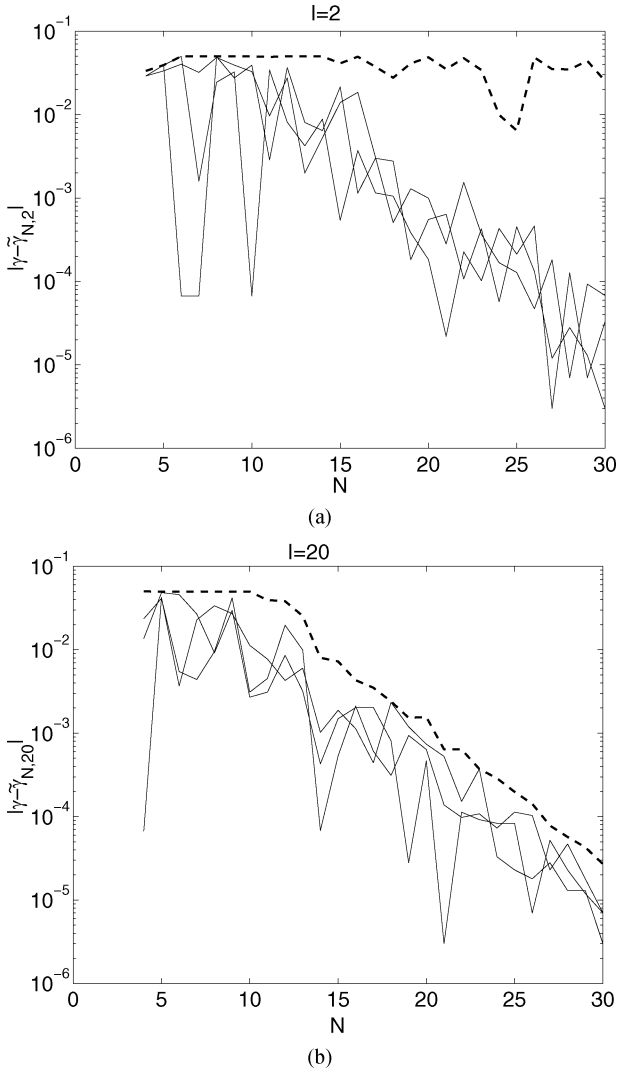

Fig. 9. The plot of $|\gamma - \bar{\gamma}_{N,I}|$ where $\gamma = 0.700067$ and $\bar{\gamma}_{N,I}$ was estimated as described in point b) of Section IV-A. In (a) $I = 2$, and in (b) $I = 20$. In both cases, the experiment was repeated 100 times with different $\{(x_i, y_i)\}_{i=1}^{I}$; each graph shows the results for three typical experiments; dashed curves show the worst case error among the 100 experiments.

that $P_{N,I}$ has only one root in the interval $(0.5, 1)$. In this case, we replace the constraint (42) with

$$-2CI\tilde{\gamma}^N \le P_{N,I}(\tilde{\gamma}) \le 2CI\tilde{\gamma}^N \tag{43}$$

and search for $\tilde{\gamma}$ values that satisfy (43) with the hope that (43) is satisfied only for $\tilde{\gamma}$ that approximates γ.

In Fig. 9(a) and (b), we show the plots of $|\gamma - \tilde{\gamma}_{N,I}|$ versus N with $I = 2$ and $I = 20$, respectively. In these numerical experiments, (x_i, y_i) pairs were randomly chosen, $a = 0.9$ was used in the encoding only (i.e., the value of a was unknown to the decoder), and $\gamma = 0.700067$. Moreover, we chose ϵ_i such that the leading coefficients of polynomials $p_{N,i}$ have the same sign, and $\tilde{\gamma}_{N,I}$ was taken to be the median of all the values that satisfy (42). This experiment was done for $I = 2$, $I = 5$, $I = 10$, and $I = 20$, 100 times in each case with different sets of (x_i, y_i) pairs. Fig. 10 shows the

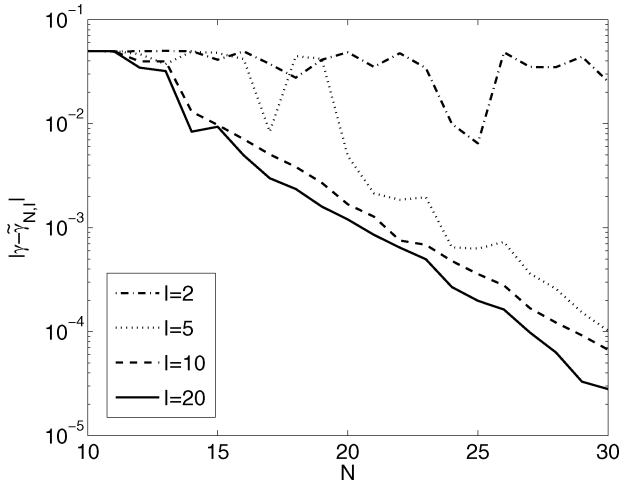

Fig. 10. The worst case error $|\gamma - \tilde{\gamma}_{N,I}|$ for $I = 2, 5, 10, 20$, among 100 numerical experiments conducted in each case.

worst case approximation error, $\max_\ell |\gamma - \tilde{\gamma}^{(\ell)}_{N,I}|$, versus N in each case ($I = 2, 3, 5, 10, 20$). Here $\tilde{\gamma}^{(\ell)}_{N,I}$ is the approximation obtained as above from the ℓth experiment ($\ell = 1, \ldots, 100$). We observe that the worst case error can be very large when I is small, i.e., when we use a small number of (x, y) pairs (although in individual examples, one can have very good estimates as seen in Fig. 9(a)). On the other hand, using a large number of (x, y) pairs makes the estimates significantly more reliable: we observe that the worst case error decreases exponentially fast as N increases.

Remark: Note that robustness remark 4) earlier has to be adapted in this case: γ would no longer be allowed to vary from one sampling interval to the next, since we would most likely obtain our different pairs from consecutive samples. Some change of γ would still be allowed without hurting the algorithm if it were sufficiently slow.

A different approach to determining γ approximately from encoded sequences (b_n) and (c_n) is proposed in the next subsection.

B. An Alternative to Exhaustive Search

In this section, we present a fast algorithm to estimate γ using the sequences $b := E^\mu_{\gamma,\delta}(x)$ and $c := E^\mu_{\gamma,\delta}(1-x)$ where $x \in (0, 1)$ and $E^\mu_{\gamma,\delta}$ is defined as before. First, we define $d := b + c$ and rewrite (16), the constraint of the main lemma, as

$$0 \le P_N(\tilde{\gamma}) \le 2C\tilde{\gamma}^N \tag{44}$$

where $P_N(\lambda) := 1 - \sum_{j=1}^N d_j \lambda^j$ and $N \ge k = \min\{j : j \in \mathbb{N} \text{ and } d_j \neq 0\}$. We then have the following.

Proposition 11: Let k be as above. For $N \ge k$, P_N has exactly one root γ_N in $[\gamma, 1]$. Moreover, γ_N satisfies $|\gamma - \gamma_N| \le C'\gamma^N$ where C' is as defined in Lemma 4.

Proof: First, note that $P_k(\lambda) = 1 - d_k\lambda^k$ has a root at $\gamma_k := (1/d_k)^{1/k}$ which satisfies $0 < \gamma \le \gamma_k \le 1$. (Recall that, by Proposition 7, k cannot be arbitrarily large.) Moreover, for $N \ge k$, $P_N(\lambda) \le P_k(\lambda)$ for all $\lambda > 0$ (since each d_j is nonnegative). Thus, $P_N(\gamma_k) \le 0$ for all $N \ge k$.

On the other hand, for any positive integer N, we have

$$P_N(\gamma) \ge 1 - \sum_{j=1}^{\infty} d_j \gamma^j = 0.$$

Since P_N is continuous on $[\gamma, \gamma_k]$, by the intermediate value theorem P_N has a root in $[\gamma, \gamma_k] \subseteq [\gamma, 1]$. Also, since P_N is strictly decreasing for $N \ge k$, this root is the only root of P_N in $[\gamma, 1]$; we denote it by γ_N. Finally, Lemma 4 implies that $|\gamma - \gamma_N| \le C'\gamma^N$. □

Proposition 11 shows that if we can approximate γ_N, the root of P_N in $[\gamma, 1]$, by $\tilde{\gamma}_N$ with a precision of $O(\gamma^N)$, then $|\gamma - \tilde{\gamma}_N|$ will also be of order γ^N. Since P_N is a "nice" polynomial in that it is strictly decreasing on $[0, 1]$ with a strictly decreasing derivative, we can compute γ_N with an arbitrary precision using Newton's method. As a matter of fact, we get the following.

Proposition 12: Let k be as in Proposition 7, and suppose $N > k$. Choose $\gamma < \gamma_N^{(0)} < 1$, and compute $(\gamma_N^{(l)})_{l=0}^L$ via

$$\gamma_N^{(l+1)} = \gamma_N^{(l)} - \frac{P_N(\gamma_N^{(l)})}{P'_N(\gamma_N^{(l)})}. \tag{45}$$

Then

$$\gamma_N^{(l+1)} - \gamma_N \le (\gamma_N^{(l)} - \gamma_N)(1 - C) \tag{46}$$

with

$$C := \frac{k}{k+2}\gamma^{k-1}(1 - \gamma_N^{(0)})^2.$$

Remark: We have implicitly assumed that, even though γ is not known, it is bounded above, away from 1, by a *known* upper bound. Note that this is a very reasonable assumption in any practical setting. (If $d_k = 2$, we can of course take $\gamma_N^{(0)} = \gamma_k$; if $d_k = 1$, $\gamma_k = 1$ however.)

Proof: For $\lambda > \gamma_N$, we have $P_N(\lambda) < 0$, $P'_N(\lambda) < 0$, and $P''_N(\lambda) < 0$, so that

$$\begin{aligned} 0 = P_N(\gamma_N) &= P_N(\lambda) + P'_N(\lambda)(\gamma_N - \lambda) + \frac{1}{2}P''_N(\xi)(\gamma_N - \lambda)^2 \\ &\le P_N(\lambda) + P'_N(\lambda)(\gamma_N - \lambda) \end{aligned} \tag{47}$$

which implies

$$\frac{|P_N(\lambda)|}{|P'_N(\lambda)|} \le \lambda - \gamma_N$$

for all $\lambda > \gamma_N$. An easy induction argument then shows that the $\gamma_N^{(l)}$, computed iteratively from the starting point $\gamma_N^{(0)} > \gamma_N$ satisfy, for all l

$$\gamma_N^{(l)} - \gamma_N \ge \left|\frac{P_N(\gamma_N^{(l)})}{P'_N(\gamma_N^{(l)})}\right| = \frac{P_N(\gamma_N^{(l)})}{P'_N(\gamma_N^{(l)})} = \gamma_N^{(l)} - \gamma_N^{(l+1)} \tag{48}$$

so that

$$0 \le \gamma_N^{(l+1)} - \gamma_N \le \gamma_N^{(l)} - \gamma_N. \tag{49}$$

On the other hand we have the following.

1) We have

$$\begin{aligned} P_N(\gamma_N^{(l)}) &= P_N(\gamma_N) + P_N'(\gamma_N)(\gamma_N^{(l)} - \gamma_N) \\ &\quad + \frac{1}{2}P_N''(\xi)(\gamma_N^{(l)} - \gamma_N)^2 \\ &\le P_N'(\gamma_N)(\gamma_N^{(l)} - \gamma_N) \end{aligned} \tag{50}$$

which, as $P_N'(\gamma_N) < 0$, implies

$$\left| \frac{P_N(\gamma_N^{(l)})}{P_N'(\gamma_N)} \right| \ge \gamma_N^{(l)} - \gamma_N. \tag{51}$$

2) Note that $P_N'(\lambda) = -\sum_{j=k}^{N} j d_j \lambda^{j-1}$. Then we have the following.

(a) As $d_k \in \{1, 2\}$ (see Proposition 7) and $\gamma_N \ge \gamma$, we get

$$|P_N'(\gamma_N)| \ge k\gamma^{k-1}. \tag{52}$$

(b) Noting that $d_j \in \{0, 1, 2\}$ for all j, we obtain

$$\begin{aligned} |P_N'(\gamma_N^{(l)})| &\le 2\sum_{j=k}^{N} j\lambda^{j-1} \le 2\sum_{j=k}^{\infty} j\lambda^{j-1} \\ &= 2\frac{d}{d\lambda}(\lambda^k \sum_{j=0}^{\infty} \lambda^j) \\ &= 2\lambda^{k-1}\frac{k(1-\lambda)+1}{(1-\lambda)^2} \end{aligned}$$

for all $1 > \lambda \ge \gamma_N^{(l)}$. In particular, as $1 > \gamma_N^{(0)} \ge \gamma_N^{(l)} > \frac{1}{2}$, we get

$$|P_N'(\gamma_N^{(l)})| \le 2\frac{\frac{1}{2}k+1}{(1-\gamma_N^{(0)})^2}. \tag{53}$$

Combining (51), (52), and (53), we conclude

$$\left| \frac{P_N(\gamma_N^{(l)})}{P_N'(\gamma_N^{(l)})} \right| = \frac{P_N(\gamma_N^{(l)})}{P_N'(\gamma_N^{(l)})} \ge C(\gamma_N^{(l)} - \gamma_N) \tag{54}$$

where $C := \frac{k}{k+2}\gamma^{k-1}(1-\gamma_N^{(0)})^2$. Then (45) implies

$$\begin{aligned} \gamma_N^{(l+1)} - \gamma_N &= \gamma_N^{(l)} - \gamma_N - \frac{P_N(\gamma_N^{(l)})}{P_N'(\gamma_N^{(l)})} \\ &\le (\gamma_N^{(l)} - \gamma_N)(1 - C). \end{aligned} \tag{55}$$

□

Corollary 13: Let $\gamma_N, \gamma_N^{(l)}$, and C be as in Proposition 12. Then

$$\gamma_N^{(l)} - \gamma_N \le (1-C)^l(\gamma_N^{(0)} - \gamma_N). \tag{56}$$

Therefore, for

$$l \ge \frac{1}{\log(1-C)}(N\log\gamma + \log(\gamma_N^{(0)} - \gamma_N)^{-1})$$

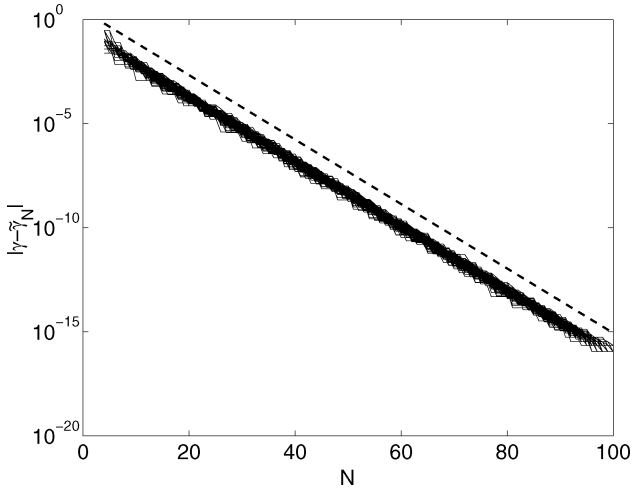

Fig. 11. 100 experiments, conducted with randomly chosen x and plotted together. Here $\gamma = 0.700067$, and the dashed line is the theoretical upper bound. $\tilde{\gamma}_N$ was computed by estimating the root of P_N via Newton's method with 10 steps.

we have

$$0 \le \gamma_N^{(l)} - \gamma_N \le \gamma^N. \tag{57}$$

Remark: Corollary 13 shows that Newton's method will compute an approximation with the desired precision after $O(N)$ iterations. In practice, however, we observe that the convergence is much faster. Fig. 11 summarizes the outcome of a numerical experiment we conducted: We chose $x \in (0, 1)$ randomly, and computed $\tilde{\gamma}_N$ by approximating the root of the polynomial P_N via Newton's method with 10 iterations. We repeated this for 100 different x values. In Fig. 11, we plot $|\gamma - \tilde{\gamma}_N|$ versus N. As one observes from the figure, the estimates are satisfactory. Moreover, the computation is much faster compared to exhaustive search.

C. The Case of Unknown a

Finally, we return to the case when the value of a is unknown to the decoder. We define the polynomial $P_{N,I}$ as in point b) of Section IV-A, and try to approximate γ by finding a root in $(0.5, 1)$. Recall that our goal in defining $P_{N,I}$ was the expectation that it has a single root in $[0.5, 1]$, and this root is located at approximately the right value. Clearly, this is not guaranteed, however, our numerical experiments suggest that $P_{N,I}$ indeed satisfies this expectation when I is large, e.g., 20 in our numerical examples outlined in point b) of Section IV-A. Also, note that $P_{N,I}$ satisfies the constraint (43) at any of its roots. So, computing its root(s) shall not give us any estimate of γ that is worse off than the estimate that we would have obtained via exhaustive search using the constraint (43).

Motivated by the discussion above, we repeat the numerical experiment of point b) of Section IV-A, only this time we estimate the value of γ by computing the root of $P_{N,I}$ via Newton's method. In Fig. 12(a) and (b), we show the plots of $|\gamma - \tilde{\gamma}_{N,I}|$ versus N with $I = 2$ and $I = 20$, respectively. In these numerical experiments $\{(x_i, y_i)\}_{i=1}^{I}$ were randomly chosen, $a = 0.9$ was used in the encoding only (i.e., the value of a was unknown

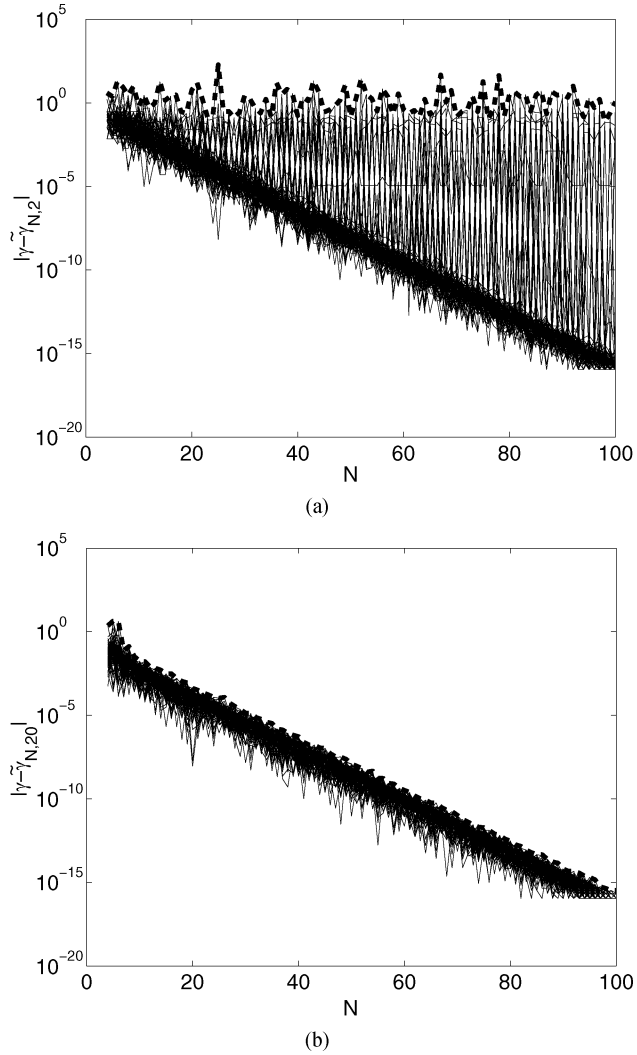

Fig. 12. 100 experiments in the case of unknown a with $\gamma = 0.700067$. $\tilde{\gamma}_{N,I}$ was computed by estimating the root of $P_{N,I}$ to which a 10 step Newton's method algorithm converges. In (a) we set $I = 2$, and in (b) $I = 20$. The dashed curves in both (a) and (b) are the worst case errors among 100 experiments for each case.

to the decoder), $\gamma = 0.700067$ (again only used in the encoding). Moreover, we chose ϵ_i such that the leading coefficient of each polynomial $p_{N,i}$ have the same sign. We computed $\tilde{\gamma}_{N,I}$ by estimating the root of $P_{N,I}$ via a 10-step iteration Newton's method with the starting point $\gamma_0 = 0.8$. This experiment was done for $I = 2$, $I = 5$, $I = 10$, and $I = 20$, 100 times in

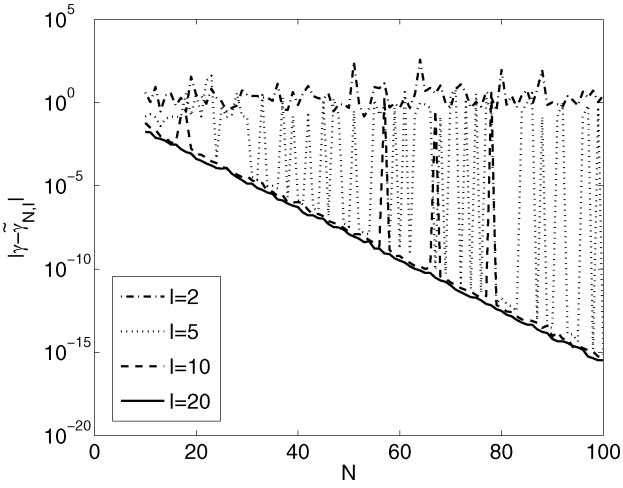

Fig. 13. The worst case error among 100 experiments with unknown a in the cases $I = 2$ and $I = 20$.

each case with different sets of $\{(x_i, y_i)\}_{i=1}^{I}$. Fig. 13 shows the worst case approximation error, $\max_k |\gamma - \tilde{\gamma}_{N,I}^k|$, versus N in each case ($I = 2, 3, 5, 10, 20$). Here, $\tilde{\gamma}_{N,I}^k$ is the approximation obtained, as above, from the kth experiment ($k = 1, \ldots, 100$). We observe that the worst case error can be very large when I is small, i.e., when we use a small number of (x, y) pairs (although in individual examples, one can have very good estimates). On the other hand, using a large number of (x, y) pairs makes the estimates significantly more reliable so that the worst case error decreases exponentially fast as N increases.

REFERENCES

[1] I. Daubechies, R. DeVore, C. Güntürk, and V. Vaishampayan, "A/D conversion with imperfect quantizers," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 874–885, Mar. 2006.

[2] I. Daubechies and R. DeVore, "Approximating a bandlimited function using very coarsely quantized data: A family of stable sigma-delta modulators of arbitrary order," *Ann. Math.*, vol. 158, no. 2, pp. 679–710, Sep. 2003.

[3] C. Güntürk, J. Lagarias, and V. Vaishampayan, "On the robustness of single loop sigma-delta modulation," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 1, pp. 63–79, Jan. 2001.

[4] Ö. Yılmaz, "Stability analysis for several sigma-delta methods of coarse quantization of bandlimited functions," *Constructive Approx.*, vol. 18, pp. 599–623, 2002.

[5] C. S. Güntürk, "One-bit sigma-delta quantization with exponential accuracy," *Commun. Pure Applied Math.*, vol. 56, no. 11, pp. 1608–1630, 2003.

[6] N. Sidorov, "Almost every number has a continuum of beta-expansions," *Amer. Math. Monthly*, vol. 110, pp. 838–842, 2003.