Linköping University Post Print

Power Series Quantization for Noisy Channels

Daniel Persson and Thomas Eriksson

N.B.: When citing this work, cite the original article.

©2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Daniel Persson and Thomas Eriksson, Power Series Quantization for Noisy Channels, 2010, IEEE TRANSACTIONS ON COMMUNICATIONS, (58), 5, 1405-1414.

<http://dx.doi.org/10.1109/TCOMM.2010.05.080688>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-56803>

Power Series Quantization for Noisy Channels

Daniel Persson and Thomas Eriksson

Abstract—A recently proposed method for transmission of correlated sources under noise-free conditions, power series quantization (PSQ), uses a separate linear or nonlinear predictor for each quantizer region, and has shown to increase performance compared to several common quantization schemes for sources with memory. In this paper, it is shown how to apply PSQ for transmission of a source with memory over a noisy channel.

A channel-optimized PSQ (COPSQ) encoder and codebook optimization algorithms are derived. The suggested scheme is shown to increase performance compared with previous state-of-the-art methods.

Index Terms—Channel-optimized quantization, noisy channels, memory-based quantization, spectrum coding.

I. INTRODUCTION

IN this paper we consider transmission of a source with memory, linear or nonlinear, over a noisy channel, which is relevant for e.g., speech, audio, and video transmission.

Efficient encoding of sources with memory may be provided by vector quantization (VQ) [1], but the exponential increase with dimension of codebook size, and thus also of storage and search complexity, renders VQ inconvenient for some applications. Another way of exploiting signal memory upon transmission is *memory-based quantizers*, that incorporate knowledge of previously quantized and transmitted data in the encoding process [2]. A memory-based VQ system always performs better than traditional VQ with the same vector dimension and codebook size, since attention is given also to inter-vector correlation. Recently, a new source coding method, power series quantization (PSQ) [3], that utilizes codebooks of arbitrary functions of previously quantized data, expressed as power series expansions, outperformed the most well-known state-of-the-art memory-based source coding methods in terms of compression for transmission over an error-free channel. This has inspired us to try to extend the same method to also work under degraded channel conditions.

Errors on a noisy channel can be mitigated by coding. So called forward error correction (FEC) [4], with the subclasses linear block codes and convolutional codes, provides protection by adding redundancy in a post-processing step after the source encoder. For infinite VQ dimension and infinite channel codeword length, it is optimal to design source

and channel coders individually [5]. However, for reasons of computational complexity and because of delay requirements, only finite VQ dimensions and channel codeword lengths are considered in practice, and in this setting, joint source-channel approaches increase performance compared to separate efforts. Because memory-based quantizers use previously quantized data for encoding, they are sensitive to transmission errors. Since the encoder in general has no knowledge of which errors occur on the channel, the encoder and decoder will be unsynchronized, and errors will propagate. In order to mitigate error propagation, care has to be taken in the design of the schemes.

We design a channel-optimized PSQ (COPSQ) strategy that overcomes the error propagation problems, and provides increased performance in comparison to state-of-the-art non-memory- and memory-based quantization for moderately to severely degraded channels.

A. Previous efforts

Traditional memory-based quantization alternatives are e.g., differential pulse-code modulation (DPCM) [2], predictive VQ (PVQ) [6], [7], vector predictive quantization [8], [9], finite-state VQ (FSVQ) [10], and safety-net quantization [11]. Recent approaches are Gaussian mixture modeling (GMM) [12], and PSQ [3]. Examples of memory-based quantization used in practice is adaptive DPCM for speech coding [13], and inter-frame prediction in video coding [14].

Contrary to DPCM and PVQ, PSQ views prediction and quantization as one single problem, where nonlinear estimator functions of previously quantized data, expressed as power series expansions, constitute the codebook. It was shown in [3] that several standard quantization methods for sources with memory: FSVQ, linear and nonlinear PVQ, vector predictive quantization, and safety-net quantization, may be described and compared in terms of PSQ. Experiments further demonstrated that PSQ performed better than VQ, FSVQ, linear PVQ, and safety-net PVQ in terms of compression, with only a small increase in memory requirement and computational complexity. It was also shown in [3] that linear PVQ outperformed FSVQ in terms of compression, number of floating point operations, and memory requirement.

Several schemes have been devised for joint source-channel coding over a noisy channel. In robust VQ (RVQ) [15], [16], which is also referred to as index assignment, quantizer optimization is first performed while ignoring the channel properties. Binary representations are thereafter assigned to the codewords in order to minimize the impact of channel errors on the decoder reconstruction. Channel-optimized quantization, where the channel is taken into account directly in the quantizer training, is a more systematic approach than

Paper approved by J. Klierer, the Editor for Iterative Methods and Cross-Layer Design of the IEEE Communications Society. Manuscript received February 5, 2009; revised August 5, 2009 and November 3, 2009.

D. Persson is with the Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden (e-mail: daniel.p@isy.liu.se).

T. Eriksson is with the Department of Signals and Systems, Chalmers University of Technology, S-412 96 Göteborg, Sweden (e-mail: thomase@chalmers.se).

This work was supported by the Swedish Foundation for Strategic Research (SSF).

Digital Object Identifier 10.1109/TCOMM.2010.05.080688

RVQ, since the source and channel coders are co-optimized. Channel-optimized quantization also provides better performance than RVQ [17], [18].

Channel-optimized VQ (COVQ) has been studied in e.g., [15], [17], [19]–[21]. Several channel-optimized memory-based schemes have also been considered. A design method for predictive trellis coding was proposed in [22], and later extended to combine trellis-coded modulation and quantization in [23]. Finite-state vector quantization for a noisy channel was proposed in [24], [25], and has recently been treated in [26]. Safety-net FSVQ, that works by applying two codebooks for quantization, one state-dependent, and one state-independent, was suggested in [11]. Predictive VQ schemes have been adapted to noisy channels by using prediction that does not fully exploit the source correlation, and by a safety-net approach where two sub-codebooks are used, one predictor-based and one without predictor [11]. A channel-optimized PVQ (COPVQ) with linear predictor and optimal encoder index search strategy was designed in [18]. Differently from [22] and [23], this scheme is not limited to usage of the same codebooks on the encoder and decoder side. It was shown in [18] that the presented method outperforms COVQ, PVQ with index assignment, and FSVQ for noisy channels [24], [25]. The COPVQ scheme in [18] is however limited by its linear predictor, and cannot handle nonlinear correlation.

B. Our contribution

We apply PSQ within a general framework for channel-optimized encoding of sources with memory. An optimal low-complexity COPSQ encoder index search strategy is proposed, and several off-line codebook optimization procedures are derived. The proposed system generalizes [18] to sources with nonlinear memory. Though our aim is to present a scheme applicable to general sources and general discrete memoryless channels, we evaluate COPSQ for coding of line spectrum frequencies (LSF) [27], and binary symmetric and additive white Gaussian noise (AWGN) channels.

In recent work [28], we have also applied PSQ for a fundamentally different problem, namely multiple description coding for packet loss channels. The transmission of several source vectors in the same packet in [28] is modeled in terms of channels with memory, which can be seen as an extension of the treatment of memoryless channels in this paper.

The remainder of this paper is organized as follows. The problem setting is described in Section II. Section III deals with generic channel-optimized coding of correlated sources. The COPSQ method is addressed in Section IV. Experimental results are presented in Section V, and the paper is finally concluded in Section VI.

II. PROBLEM SPECIFICATION

Figure 1 shows our system, where the mathematical nomenclature will be defined later. For clarity of the presentation of the material in this paper, we begin by stating which problem we want to solve. These listed conditions describe a relevant standard setting:

- A. The source produces discrete time, continuous amplitude, and zero mean vectors, with linear or nonlinear inter-vector correlation.

Fig. 1. Proposed system with encoder, channel, and decoder. An input vector $\mathbf{x}_n$ is quantized to an index i_n that is mapped to an index j_n by the channel, and the decoder reconstructs the source as $\tilde{\mathbf{x}}_n$. The quantities $\tilde{\mathbf{y}}_n$, $\mathbf{s}_n$, and $\mathbf{W}_n$ are used in the recursive formulation of the coder.

- B. The channel transitions at different times are independent.
- C. The source and channel are independent.
- D. One source vector is encoded and decoded at a time, and how encoding and decoding influence future quantization is not considered.
- E. The Euclidean distance measure is used for assessing performance.

All listed problem specifications will be expressed in mathematical terms and employed in the development in the next section. Whenever a problem specification is used, this will be clearly stated by a reference to the listed point in question.

III. CHANNEL-OPTIMIZED MEMORY-BASED QUANTIZATION

The source described in (A) in Section II produces discrete time, continuous amplitude, and zero mean vectors $\mathbf{x}_n \in \mathbb{R}^d$, $n = 1, \dots, N$, with linear or nonlinear correlation, see Fig. 1¹. Each vector $\mathbf{x}_n$ is quantized to an index $i_n \in \{1, \dots, M\}$ at time n , whose binary representation is sent on the channel. The channel noise transforms the index i_n to a received index $j_n \in \{1, \dots, M\}$. Vectors of all sent and received indexes until and including time n are

$$\mathbf{i}_n = [i_1, \dots, i_n], \quad (1)$$

$$\mathbf{j}_n = [j_1, \dots, j_n], \quad (2)$$

respectively. A general decoder, that reconstructs $\mathbf{x}_n$ at time n , see (D) in Section II, gives

$$\tilde{\mathbf{x}}_n = \tilde{\mathbf{x}}_n(\mathbf{j}_n) \in \mathbb{R}^d. \quad (3)$$

The encoder also treats $\mathbf{x}_n$ at time n , without consideration of future quantization, as was stated in (D). At time n , it accesses the source vectors $\mathbf{x}_1$ to $\mathbf{x}_n$, the candidate index i , and previously transmitted indices $\mathbf{i}_{n-1}$. We may thus write the distortion measure as

$$J_n(i) = E_{\mathbf{j}_n | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{i}} [D(\mathbf{x}_n, \tilde{\mathbf{x}}_n)], \quad (4)$$

and the optimal encoder as

$$i_n = \arg \min_{i \in \{1, \dots, M\}} J_n(i), \quad (5)$$

where

$$\mathbf{i} = [(\mathbf{i}_{n-1}), i]. \quad (6)$$

¹Definitions of some notation for the recursive coder formulation used in the figure will be deferred to later sections.

Observe that i is the index over which the optimization is performed in (5), while i_n is the chosen index that is the result of the same optimization. Note also that there is a corresponding relationship between (1) and (6). According to (C) in Section II, the received sequence is conditionally independent of the source output, given the encoder output

$$P(\mathbf{j}_n | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{i}) = P(\mathbf{j}_n | \mathbf{i}). \quad (7)$$

As specified in (E) in Section II, the Euclidean distance measure

$$D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^r = ((\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}))^{r/2}, \quad (8)$$

where $r = 2$, is used. We moreover consider independent channel transitions in time in (B) in Section II

$$P(\mathbf{j}_n | \mathbf{i}) = P(j_n | i) \prod_{k=1}^{n-1} P(j_k | i_k), \quad (9)$$

and using (7) to (9), we may now rewrite (4) as

$$J_n(i) = E_{j_n|i} [E_{j_{n-1}|i_{n-1}} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2]], \quad (10)$$

and (5) as

$$i_n = \arg \min_{i \in \{1, \dots, M\}} E_{j_n|i} [E_{j_{n-1}|i_{n-1}} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2]]. \quad (11)$$

We have now employed all specifications in Section II in order to obtain (11). Encoding is most often performed online, and has to be reasonably fast. Evaluating all expectations in (11) for each n would result in a coder whose complexity increases over time since the received index vector $\mathbf{j}_n$ becomes longer and longer. Our goal is to formulate (11) as a low-complexity algorithm where calculations at time $n - 1$ may be reused in a recursive manner for encoding at time n . This venture may be difficult in general, but we know that solutions exist for COVQ [17] and COPVQ, [18]. In the coming section, we will present a solution also for the COPSQ encoder.

IV. CHANNEL-OPTIMIZED POWER SERIES QUANTIZATION (COPSQ)

Before COPSQ is described, we review VQ and PSQ for a noise-free channel [3] as an introduction.

A. VQ for a noise-free channel

When the channel is noise-free, the index i_n arrives correctly to the decoder for all n . In the case of VQ, the decoder employs a codebook $\mathbf{c}_m \in \mathbb{R}^d$, $m = 1, \dots, M$, that is fixed in time, and the decoder reconstruction of $\mathbf{x}_n$ at time n is

$$\tilde{\mathbf{x}}_n(i_n) = \mathbf{c}_{i_n}. \quad (12)$$

Using the codebook and (12), the encoder in (11) can be rewritten as

$$i_n = \arg \min_{m \in \{1, \dots, M\}} \|\mathbf{x}_n - \mathbf{c}_m\|_2^2. \quad (13)$$

B. Power series quantization (PSQ) for a noise-free channel

A power series quantizer is best described as a scheme with a separate predictor (linear or nonlinear) in each quantizer region. Since a power series expansion can describe almost

all reasonably smooth functions, it can be shown that the PSQ can describe any type of nonlinear or linear memory. The PSQ decoder uses a codebook

$$\mathbf{c}_m^{(n)} = \mathbf{A}_m \tilde{\mathbf{y}}_n, \quad m = 1, \dots, M, \quad (14)$$

where the matrices $\mathbf{A}_m \in \mathbb{R}^{d \times d_y}$ contain the power series expansion coefficients, and $\tilde{\mathbf{y}}_n \in \mathbb{R}^{d_y}$ contains previously quantized scalar samples, powers thereof, as well as multiplications of such powers. The number d_y is determined by the power series expansion order, the number of previously quantized source vectors used for quantization at the present time, as well as the dimension of these vectors. Observe that differently from in (12), the codebook is now time-dependent, which is also indicated by superscripting $\mathbf{c}_m^{(n)}$ with (n) in (14). At the decoder side, the reconstruction of $\mathbf{x}_n$ at time n is

$$\tilde{\mathbf{x}}_n = \mathbf{A}_{i_n} \tilde{\mathbf{y}}_n. \quad (15)$$

Consider an example of decoding at time n using one 2-dimensional vector memory

$$\tilde{\mathbf{x}}_{n-1} = [\tilde{x}_{n-1,1}, \tilde{x}_{n-1,2}]^T, \quad (16)$$

where $\tilde{x}_{n-1,1}$, and $\tilde{x}_{n-1,2}$ are scalar vector components. Employing a second order power series expansion, we may write

$$\begin{aligned} \tilde{\mathbf{y}}_n = & [1, \tilde{x}_{n-1,1}, \tilde{x}_{n-1,2}, (\tilde{x}_{n-1,1})^2, (\tilde{x}_{n-1,2})^2, \tilde{x}_{n-1,1}\tilde{x}_{n-1,2}]^T. \end{aligned} \quad (17)$$

Equations (15) and (17) define a recursion, and we may thus write $\tilde{\mathbf{y}}_n = \tilde{\mathbf{y}}_n(\mathbf{i}_{n-1}) = \tilde{\mathbf{y}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1})$ and $\tilde{\mathbf{x}}_n = \tilde{\mathbf{x}}_n(\mathbf{i}_n) = \tilde{\mathbf{x}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1}, i_n)$.

First order power series expansions, with one vector memory

$$\tilde{\mathbf{y}}_n = \begin{bmatrix} 1 \\ \tilde{\mathbf{x}}_{n-1} \end{bmatrix}, \quad (18)$$

is regarded in this paper for handling nonlinear correlation in the source sequence, since this previously has showed good performance in applications [3]. Using (14) and (15), the encoder in (11) can be rewritten as

$$i_n = \arg \min_{m \in \{1, \dots, M\}} \|\mathbf{x}_n - \mathbf{A}_m \tilde{\mathbf{y}}_n\|_2^2. \quad (19)$$

For a more thorough introduction to PSQ, we refer the reader to [3].

C. COPSQ coder for a noisy channel

The PSQ scheme in Section IV-B is now expanded in order to tackle transmission of a source with memory over a noisy channel, which is the general situation described in Section III. The codebook

$$\mathbf{c}_m^{(n)} = \mathbf{A}_m \tilde{\mathbf{y}}_n, \quad m = 1, \dots, M, \quad (20)$$

is still used, and the decoder reconstruction of $\mathbf{x}_n$ at time n is

$$\tilde{\mathbf{x}}_n = \mathbf{A}_{j_n} \tilde{\mathbf{y}}_n. \quad (21)$$

The decoder recursion is defined by (18) and (21) and we now have that $\tilde{\mathbf{y}}_n = \tilde{\mathbf{y}}_n(\mathbf{j}_{n-1}) = \tilde{\mathbf{y}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1})$ and $\tilde{\mathbf{x}}_n = \tilde{\mathbf{x}}_n(\mathbf{j}_n) = \tilde{\mathbf{x}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1}, j_n)$. Expression (11) may be rewritten as

$$i_n = \arg \min_{i \in \{1, \dots, M\}} \left(E_{j_n|i}[\mathbf{x}_n^T \mathbf{x}_n] - 2\mathbf{x}_n^T \underbrace{E_{j_n|i}[\tilde{\mathbf{x}}_n]}_{\mathbf{s}_n(i) \in \mathbb{R}^d} + \underbrace{E_{j_n|i}[\tilde{\mathbf{x}}_n^T \tilde{\mathbf{x}}_n]}_{r_n(i) \in \mathbb{R}} \right) \quad (22)$$

$$= \arg \min_{i \in \{1, \dots, M\}} \left(-2\mathbf{x}_n^T \mathbf{s}_n(i) + r_n(i) \right), \quad (23)$$

where the first term in (22) is the same for all choices of i_n , and is omitted in (23).

Intuitive explanation of the following development: At every time n , a vector $\mathbf{x}_n$ is encoded. We now formulate the encoder (23) so that calculations at times preceding n can be reused at time n in order to reduce computational complexity. This will be accomplished by deriving efficient ways of calculating the expectations $\mathbf{s}_n(i)$ and $r_n(i)$.

$\mathbf{s}_n(i)$: For the term $\mathbf{s}_n(i)$ it holds that

$$\mathbf{s}_n(i) = E_{j_n|i}[\tilde{\mathbf{x}}_n] \quad (24)$$

$$= E_{j_n|i}[\mathbf{A}_{j_n} \tilde{\mathbf{y}}_n] \quad (25)$$

$$= E_{j_n|i} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 \\ \tilde{\mathbf{x}}_{n-1} \end{bmatrix} \right] \quad (26)$$

$$= E_{j_n|i} \left[E_{j_{n-1}|i_{n-1}} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 \\ \tilde{\mathbf{x}}_{n-1} \end{bmatrix} \right] \right] \quad (27)$$

$$= E_{j_n|i} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 \\ E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{x}}_{n-1}] \end{bmatrix} \right] \quad (28)$$

$$= E_{j_n|i} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 \\ \mathbf{s}_{n-1}(i_{n-1}) \end{bmatrix} \right] \quad (29)$$

$$= \sum_{j_n} P(j_n|i) \mathbf{A}_{j_n} \begin{bmatrix} 1 \\ \mathbf{s}_{n-1}(i_{n-1}) \end{bmatrix}, \quad (30)$$

where we have used (21) to obtain (25), (18) to obtain (26), the fact that matrix multiplication is a linear map to obtain (28), and the definition of $\mathbf{s}_n$ in (24) to obtain (29). The equation (29) or equivalently (30) establishes a recursive relation for calculation of $\mathbf{s}_n(i)$ with low computational complexity.

$r_n(i)$: Introduce

$$\mathbf{W}_n(i) = E_{j_n|i}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T] \quad (31)$$

$$= E_{j_n|i}[\mathbf{A}_{j_n} \tilde{\mathbf{y}}_n (\tilde{\mathbf{y}}_n)^T \mathbf{A}_{j_n}^T] \quad (32)$$

$$= E_{j_n|i}[\mathbf{A}_{j_n} E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{y}}_n (\tilde{\mathbf{y}}_n)^T] \mathbf{A}_{j_n}^T] \quad (33)$$

$$= E_{j_n|i} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 \\ E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{x}}_{n-1}] \end{bmatrix} \cdot \begin{bmatrix} 1 & (E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{x}}_{n-1}])^T \\ E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{x}}_{n-1}] & E_{j_{n-1}|i_{n-1}}[\tilde{\mathbf{x}}_{n-1} \tilde{\mathbf{x}}_{n-1}^T] \end{bmatrix} \mathbf{A}_{j_n}^T \right] \quad (34)$$

$$= E_{j_n|i} \left[\mathbf{A}_{j_n} \begin{bmatrix} 1 & \mathbf{s}_{n-1}(i_{n-1})^T \\ \mathbf{s}_{n-1}(i_{n-1}) & \mathbf{W}_{n-1}(i_{n-1}) \end{bmatrix} \mathbf{A}_{j_n}^T \right] \quad (35)$$

$$= \sum_{j_n} P(j_n|i) \mathbf{A}_{j_n} \cdot \begin{bmatrix} 1 & \mathbf{s}_{n-1}(i_{n-1})^T \\ \mathbf{s}_{n-1}(i_{n-1}) & \mathbf{W}_{n-1}(i_{n-1}) \end{bmatrix} \mathbf{A}_{j_n}^T, \quad (36)$$

where $\mathbf{W}_n \in \mathbb{R}^{d \times d}$, and where we have used (21) to obtain (32), the linear map property of matrix multiplication to obtain (33), (18) to obtain (34), and (24) and (31) to obtain (35). For the term $r_n(i)$ it holds that

$$r_n(i) = E_{j_n|i}[\tilde{\mathbf{x}}_n^T \tilde{\mathbf{x}}_n] \quad (37)$$

$$= E_{j_n|i}[\text{tr}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T]] \quad (38)$$

$$= \text{tr}[E_{j_n|i}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T]] \quad (39)$$

$$= \text{tr}[\mathbf{W}_n(i)], \quad (40)$$

where the linear map property of the trace operation tr is used to obtain (39), and where (31) was used to obtain (40). The equations (35), or equivalently (36), and (40) establish a recursive relation which allows us to calculate $r_n(i)$ with low computational complexity. We finally summarize our findings in this section in an encoder algorithm:

1. Initialization: Set $n = 0$, $\mathbf{s}_0$ to an all zero vector, and $\mathbf{W}_0$ to an all zero matrix.
2. Set $n = n + 1$. The vector $\mathbf{x}_n$ arrives from the source to the encoder.
3. Calculate $\mathbf{s}_n(i)$ with (30), $\mathbf{W}_n(i)$ with (36), and $r_n(i)$ with (40) for all $i = 1, \dots, M$.
4. Decide i_n using (23). Also store $\mathbf{s}_n(i_n)$ and $\mathbf{W}_n(i_n)$ for usage at time $n + 1$, see Fig. 1.
5. Stop if $n = N$, or otherwise go to Step 2.

Our encoder is not sensitive to the initializations, and the initialization above works well in practice. Though we have managed to formulate the encoder in such a way that calculations at earlier times are efficiently reused, the derived encoder still has a rather large computational complexity in practice. This is because for every vector to be encoded, and every candidate index, transitions to all other codewords have to be considered. However, this complexity can be reduced by not considering transitions to all possible codewords, and by restricting power series coefficients.

By limiting the power series coefficients of order 1 to zero, COVQ is achieved as a special case of COPSQ. If the codebook power series matrices are restrained so that coefficients of order 1 at the same matrix position are equal, COPVQ is achieved. Strategies with less than M predictors, as well as safety-net PVQ, can be implemented in similar ways. Also, it should be noted that the COPSQ encoder is highly parallelizable.

D. Codebook optimization

The codebook is determined offline, and the following algorithms do thus not affect online computational complexity. Since the codebook cannot be found analytically, we resort to iterative methods.

Intuitive explanation of the following development: Three methods are devised. A sample-iterative codebook optimiza-

tion, characterized by a codebook update for every encoded vector $\mathbf{x}_n$, is derived in Section IV-D1. Thereafter, two block-iterative algorithms, i.e., algorithms that update the codebook after having coded several vectors $\mathbf{x}_n$, are considered. A Gauss-Newton algorithm updates the codebook by taking successive steps in Section IV-D2. In Section IV-D3, a generalized Lloyd algorithm (GLA)-like procedure sets the gradient to zero and solves for the power series coefficients, and circumvents in this way the need of setting a step size.

1) *Sample-iterative codebook optimization*: At each time instant n , $\mathbf{x}_n$ is first encoded and i_n is achieved. Thereafter, in order to minimize $J_n(i_n)$, cf. (10), the sample-iterative update of coefficient matrix m for use at time $n + 1$ is given by

$$\mathbf{A}_m^{\text{NEW}} = \mathbf{A}_m - \mu(n) \nabla_{\mathbf{A}_m} J_n(i_n), \quad (41)$$

where $\mu(n)$ is a scalar step size. By use of (23), (29) and (35), and considering a separate codebook for each time instant n , which means that coefficient matrices at earlier times are ignored when applying the gradient, we can rewrite (41) as

$$\begin{aligned} \mathbf{A}_m^{\text{NEW}} = & \mathbf{A}_m - \mu(n) 2P(m|i_n) \\ & \cdot \left(\mathbf{A}_m \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} - \mathbf{x}_n \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \end{bmatrix} \right). \end{aligned} \quad (42)$$

We summarize the sample-iterative algorithm:

1. Set the time index $n = 0$.
2. Set $n = n + 1$. The vector $\mathbf{x}_n$ arrives from the source to the encoder.
3. An optimal index i_n is chosen by the encoder using the power series expansion codebooks with the coefficient matrices $\mathbf{A}_m$, $m = 1, \dots, M$.
4. The coefficient matrices $\mathbf{A}_m^{\text{NEW}}$, $m = 1, \dots, M$ are calculated using (42). Set $\mathbf{A}_m = \mathbf{A}_m^{\text{NEW}}$ for $m = 1, \dots, M$.
5. Stop if $n = N$, or otherwise go to Step 2.

The N vectors can of course be reused several times.

2) *Gauss-Newton codebook optimization*: The Gauss-Newton algorithm is block-iterative, i.e., it takes several vectors into account for a single update. Therefore, we introduce the mean of (10)

$$\bar{J}_n = E_{j_n, i_n, \mathbf{x}_n} [D(\mathbf{x}_n, \tilde{\mathbf{x}}_n)] = E_{i_n, \mathbf{x}_n} [J_n(i_n)]. \quad (43)$$

The derivation, that is given in Appendix A, leads to the update equations

$$\begin{aligned} \mathbf{A}_m^{\text{NEW}} = & \mathbf{A}_m - \mu \sum_{n=1}^N P(m|i_n) \\ & \cdot \left(\mathbf{A}_m \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} - \mathbf{x}_n \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \end{bmatrix} \right) \\ & \cdot \left(\sum_{n=1}^N P(m|i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} \right)^{-1}. \end{aligned} \quad (44)$$

We summarize the Gauss-Newton algorithm:

1. Set the iteration counter $t = 0$.
2. Set $t = t + 1$. Encode all vectors $n = 1, \dots, N$, and calculate $\mathbf{A}_m^{\text{NEW}}$ for $m = 1, \dots, M$ using (44). Set $\mathbf{A}_m = \mathbf{A}_m^{\text{NEW}}$ for $m = 1, \dots, M$.
3. If t is less than a predetermined threshold, go to Step 2, or otherwise stop.

3) *GLA-like codebook optimization*: Setting the gradient $\nabla_{\mathbf{A}_m} \bar{J}_n = 0$, ignoring that $\tilde{\mathbf{y}}_n$ depends on $\mathbf{A}_m$ for simplicity, cf. the development in [3], and using the linear map property of matrix multiplication, the centroid condition for a GLA-like algorithm

$$\begin{aligned} \mathbf{A}_m^{\text{NEW}} = & E_{i_n, \mathbf{x}_n} \left[P(m|i_n) \mathbf{x}_n \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \end{bmatrix} \right] \\ & \cdot \left(E_{i_n, \mathbf{x}_n} \left[P(m|i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} \right] \right)^{-1} \end{aligned} \quad (45)$$

$$\begin{aligned} = & \sum_{n=1}^N P(m|i_n) \mathbf{x}_n \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \end{bmatrix} \\ & \cdot \left(\sum_{n=1}^N P(m|i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} \right)^{-1}, \end{aligned} \quad (46)$$

is obtained, where we assume stationarity and ergodicity in order to replace the expectations by arithmetic means in (46). Stationarity and ergodicity assumptions are not always fulfilled, but have been applied with success in many source coding strategies, see e.g., [17], [18]. The nearest neighbor condition corresponds to the actual encoder (23). We summarize the GLA-like algorithm:

1. Set the iteration counter $t = 0$.
2. Set $t = t + 1$. Encode all vectors $n = 1, \dots, N$, and calculate $\mathbf{A}_m^{\text{NEW}}$ for $m = 1, \dots, M$ using (46). Set $\mathbf{A}_m = \mathbf{A}_m^{\text{NEW}}$ for $m = 1, \dots, M$.
3. If t is less than a predetermined threshold, go to Step 2, or otherwise stop.

The risk (43) is not convex in $\mathbf{A}_m$, and the Gauss-Newton and GLA-type codebook optimizations are involved with some approximations. Therefore, convergence of these algorithms can not be guaranteed. The sample-iterative algorithm focuses on one vector at a time, and does thus not guarantee successive minimization of any measure either.

E. Performance predictions

In order to support the experimental evaluation of COPSQ, two types of performance predictions are envisaged:

- **High rate-capacity prediction**: Approximatively optimal high rate performance for memory-based quantization over a noise-free channel was derived in [12] for a general r , cf. (8). Evaluation of the high rate-expression

in [12] for the channel capacity yields

$$\bar{J}_n \approx C(r, d) 2^{-\frac{rC_c}{d}} \cdot E_{\mathbf{x}_{n-1}, \mathbf{x}_n} \left[\frac{(f(\mathbf{x}_n | \mathbf{x}_{n-1}))^{-\frac{r}{d+r}}}{E_{\mathbf{x}_n | \mathbf{x}_{n-1}} [(f(\mathbf{x}_n | \mathbf{x}_{n-1}))^{-\frac{r}{d+r}}]^{-\frac{r}{d}}} \right], \quad (47)$$

where d is the dimension of $\mathbf{x}_n$, C_c is the channel capacity in bits associated with the channel uses employed for transmitting the vector $\mathbf{x}_n$,

$$C(r, d) = \frac{d}{d+r} V_d^{-r/d}, \quad (48)$$

$$V_d = \frac{\pi^{d/2}}{\frac{d}{2} \Gamma(\frac{d}{2})}, \quad (49)$$

and $f(\mathbf{x}_n, \mathbf{x}_{n-1})$ is the source distribution. Equation (47) does not rely on a strict mathematical development, and should be considered as a rule of thumb. For $\mathbf{x}_n$ and $\mathbf{x}_{n-1}$ being jointly Gaussian with covariance $\mathbf{C}_{\mathbf{x}_n, \mathbf{x}_{n-1}}$

$$\bar{J}_n \approx C(r, d) 2^{-\frac{rC_c}{d}} \left(\frac{r+d}{d} \right)^{\frac{d+r}{2}} \cdot (2\pi)^{r/2} \left(\frac{\det(\mathbf{C}_{\mathbf{x}_n, \mathbf{x}_{n-1}})}{\det(\mathbf{C}_{\mathbf{x}_{n-1}})} \right)^{\frac{r}{2d}}. \quad (50)$$

No analytical expression for $\bar{J}_n$ exists for general distributions. Therefore, assuming stationarity and ergodicity, $f(\mathbf{x}_n | \mathbf{x}_{n-1})$ is replaced by a model, and the expectations with arithmetic means, in order to evaluate (47) for general sources.

- **Rate-distortion-capacity prediction:** We assume a Gaussian model for the linear prediction error spectrum, and evaluate the rate-distortion function at capacity C_c through water-filling [29].

It is worth pointing out that while the Gaussian assumptions above are made for simplicity, our COPSQ scheme can tackle arbitrary sources.

V. EXPERIMENTS

COPSQ is now employed for transmission of LSF coefficients over different channels.

A. Prerequisites

The simulation details are as follows:

- **Database:** In all experiments, the TIMIT database (lowpass-filtered and downsampled to 8 kHz) was used. A tenth-order linear predictive coding (LPC) analysis using the auto-correlation method is performed every 20 ms using a 25-ms Hamming window. A fixed 10-Hz bandwidth expansion is applied to each pole of the LPC coefficient vector, and the LPC vectors are transformed to the LSF representation. The vectors are split into three parts prior to quantization, with dimensions 3, 3, and 4 respectively.
- **Benchmarking:** COPSQ is compared to separate source and channel coding consisting of DPCM with bit allocation by means of the zonal sampling algorithm, trained codebook for every vector dimension, and Reed-Solomon

error correction based on a finite field with 32 elements. Comparisons to the joint source-channel coding schemes COVQ [17], COPVQ [18], and to the performance predictions in Section IV-E, are also presented. For the Monte Carlo evaluation of (47), a Gaussian mixture model (GMM) with 20 component densities is employed. The expectation-maximization (EM) algorithm is used to estimate model parameters.

- **Training:** In the training, 700 000 vectors were used, and in the evaluation, a separate set of 250 000 vectors were used, except where stated otherwise. In the case of COPSQ for a binary symmetric channel, only transitions of a Hamming distance of 1 were considered in the offline codebook training phase in order to speed up the process, and better performance is expected if this approximation is omitted. For the case of QAM, all possible transitions were considered in the training. In the sample-iterative codebook optimization algorithm, the step size $\mu(n)$ was set to decrease linearly to zero with n , and the start step size is chosen to 0.05 and 0.01 for power series coefficients of order 0 and 1 respectively. Using the N vectors in the database, the power series expansion coefficients of order 0 are initialized by small random numbers, while the remaining coefficients are initialized to $\sum_{n=2}^N \mathbf{x}_n \mathbf{x}_{n-1}^T (\sum_{n=2}^N \mathbf{x}_{n-1} \mathbf{x}_{n-1}^T)^{-1}$.
- **Performance measures:** We evaluate the results in signal-to-noise ratio (SNR) since our proposed algorithm could be employed for a generic source, and since our encoder and codebook optimization procedures aim to maximize SNR by minimizing (10) and (43). Since LSF quantization serves as the particular example application in this paper, the schemes are also evaluated in terms of spectral distortion, which is a well established measure of LPC coding quality [30]. It is here calculated in the full 0-4 kHz range.

B. Results

Since convergence of our codebook optimization schemes can not be guaranteed, cf. the discussion in Section IV-D, we conducted some preliminary experimental investigations of the codebook training. These showed that the sample-iterative algorithm was effective for codebook optimization. It was also noted that the Gauss-Newton and GLA-type algorithms need to be initiated by the sample-iterative update, which was also the case in [18], but that they thereafter improve performance in every iteration.

Figure 2 shows a comparison of the sample-iterative algorithm, the Gauss-Newton algorithm, and GLA for transmission over a binary symmetric channel with a gross bitrate of 18 bits per LSF vector. For fairness of comparison, 200 000 vectors were used with every algorithm. The Gauss-Newton algorithm and GLA are initiated by the sample-iterative algorithm with 50 000 vectors, and 3 codebook iterations are thereafter run with 50 000 vectors per update. The sample-iterative algorithm was run with 200 000 vectors. Our Gauss-Newton algorithm uses $\mu = 0.2$. The three algorithms perform relatively similarly for this problem, though the sample-iterative algorithm gives a slim performance increase. In the following experiments, we have used the sample-iterative algorithm.

Fig. 2. Comparison of the sample-iterative, Gauss-Newton, and GLA-like algorithms, in terms of spectral distortion versus BER, for a gross bitrate of 18 bits per LSF vector, when transmitting over a binary symmetric channel.

Fig. 3. Spectral distortion versus BER for COPSQ and DPCM-RS-schemes with different channel coding rates, when transmitting over a binary symmetric channel. The gross bitrate per LSF vector is 24 bits.

COPSQ is compared to separate source and channel coding, i.e., to DPCM and Reed-Solomon error correction, when transmitting with a bitrate of 24 bits per LSF vector, over a binary symmetric channel, and with 16-QAM over an AWGN channel, in Fig. 3 and Fig. 4 respectively. For the case of QAM in Fig. 4, we define E_t as the energy per *transmitted* bit, and N_0 as the variance of the scalar zero-mean complex Gaussian noise variables. COPSQ gives best performance for all bit error rates.

Figure 5 presents comparisons between performance predictions and simulations in terms of SNR versus bit error rate (BER), for transmission over a binary symmetric channel with a bitrate of 24 bits per LSF vector. The same experiment, conducted when transmitting with 16-QAM over an AWGN channel, is presented in Fig. 6. Figure 7 further presents comparisons between performance predictions and simulations in terms of SNR versus bitrate per LSF vector, when trans-

Fig. 4. Spectral distortion versus E_t/N_0 for COPSQ and DPCM-RS-schemes with different channel coding rates, when transmitting with 16-QAM over an AWGN channel. The gross bitrate per LSF vector is 24 bits.

Fig. 5. SNR versus BER for simulations and performance predictions, for a bitrate of 24 bits per LSF vector, when transmitting over a binary symmetric channel. The performance predictions are a high rate-capacity (HR) approximation with a GMM model (47), a HR approximation with a Gaussian model (50), as well as a rate-distortion-capacity (RD) estimate with a Gaussian model [29].

mitting over a binary symmetric channel with BER=0.02. The improvements supplied by COPSQ in comparison to COPVQ and COVQ are more than 1 and 5 bits respectively, independently of bitrate. Our performance predictions are rather loose, and suggest that further improvements may be made by improved training algorithms, and longer codes. The performance gap between the high rate-capacity performance predictions using Gaussian and GMM models is similar to that between COPVQ and COPSQ. Both these gaps show the performance gain achieved by taking nonlinear memory into account.

Figure 8 presents comparisons of COVQ, COPVQ, and COPSQ simulations in terms of spectral distortion versus BER, when transmitting over a binary symmetric channel with

Fig. 6. SNR versus E_t/N_0 for simulations and performance predictions, for a bitrate of 24 bits per LSF vector, when transmitting with 16-QAM over an AWGN channel. The performance predictions are a high rate-capacity (HR) approximation with a GMM model (47), a HR approximation with a Gaussian model (50), as well as a rate-distortion-capacity (RD) estimate with a Gaussian model [29].

Fig. 7. SNR versus bitrate per LSF vector for simulations and performance predictions, when transmitting over a binary symmetric channel with BER=0.02. The performance predictions are a high rate-capacity (HR) approximation with a GMM model (47), a HR approximation with a Gaussian model (50), as well as a rate-distortion-capacity (RD) estimate with a Gaussian model [29].

a bitrate of 24 bits per LSF vector. The same experiment, conducted when transmitting with 16-QAM over an AWGN channel, is presented in Fig. 9. Figure 10 presents comparisons of COVQ, COPVQ, and COPSQ simulations in terms of spectral distortion versus gross bitrate per LSF vector, when transmitting over a binary symmetric channel with BER=0.02. The memory-based COPVQ scheme always performs better than the non-memory-based COVQ method, and COPSQ handles nonlinear dependencies that are overlooked by the

Fig. 8. Spectral distortion versus BER for COPSQ, COPVQ, and COVQ, for a bitrate of 24 bits per LSF vector, when transmitting over a binary symmetric channel.

Fig. 9. Spectral distortion versus E_t/N_0 for COPSQ, COPVQ, and COVQ, for a bitrate of 24 bits per LSF vector, and transmission with 16-QAM over an AWGN channel.

linear prediction-based COPVQ. We thus infer that by exploiting the memory better, the performance is improved. The improvements supplied by COPSQ in comparison to COPVQ and COVQ are, as in the case of the SNR comparison, more than 1 and 5 bits respectively, independently of bitrate. By comparison of Figures 5 to 10, we conclude that encoder and codebook optimization algorithms that improve SNR, also yield performance increments in terms of spectral distortion.

Figure 11 illustrates COPSQ robustness when transmitting over a binary symmetric channel. The different curves show performance of COPSQ optimized for different channel bit error ratios. COPSQ optimized for bit error probabilities larger than or equal to 0.01 show good robustness, and quantizers trained for noise-free conditions perform rather poorly on noisy channels. Optimization for a specific BER gives the best performance for the same BER in the evaluations.

Fig. 10. Spectral distortion versus bitrate per LSF vector for COPSQ, COPVQ, and COVQ, when transmitting over a binary symmetric channel with BER=0.02.

Fig. 11. Illustration of robustness. The different curves show performance of COPSQ optimized for a binary symmetric channel with bit error ratios 0.0, 0.001, 0.01, 0.02, 0.05, and 0.1, and a bitrate of 21 bits per LSF vector.

VI. CONCLUSION

The problem of transmission of a source with memory over a noisy channel is studied. A newly suggested quantization method PSQ has outperformed several previous state-of-the-art algorithms for encoding of correlated sources under noise-free conditions. We investigate PSQ, for exploiting memory in the signal, in conjunction with a channel optimization strategy for combating a noisy channel.

In the case of channel transitions that are independent in time, Euclidean distance measure, and first order COPSQ with one vector memory, we derive a low complexity recursive encoder, as well as three codebook optimization algorithms, namely a sample-iterative algorithm, a Gauss-Newton algorithm, and a GLA-like algorithm.

In experiments it is seen how COPSQ outperforms previously proposed state-of-the-art schemes, and COPSQ robustness is confirmed. We see, however, that systems optimized

for error-free conditions do not exhibit robustness. High rate-capacity and rate-distortion-capacity performance predictions are compared to the simulations.

It is completely straight-forward to generalize the development in this paper to usage of a weighted Euclidean distortion measure. Extensions to higher order power series expansions, to several vector memory, and to channels with memory, are possible developments.

APPENDIX A

DERIVATION OF THE GAUSS-NEWTON CODEBOOK UPDATE EQUATIONS

Consider one row $\mathbf{A}_{m,l}$ of the matrix $\mathbf{A}_m$ at the time. Newton's method [31] gives

$$(\mathbf{A}_{m,l}^{\text{NEW}})^T = \mathbf{A}_{m,l}^T - \mu (\mathbf{H}_{\mathbf{A}_{m,l}^T} \bar{\mathbf{J}}_n)^{-1} \nabla_{\mathbf{A}_{m,l}^T} \bar{\mathbf{J}}_n \quad (51)$$

$$= \mathbf{A}_{m,l}^T - \mu \left(E_{i_n, \mathbf{x}_n} [\mathbf{H}_{\mathbf{A}_{m,l}^T} \mathbf{J}_n] \right)^{-1} E_{i_n, \mathbf{x}_n} [\nabla_{\mathbf{A}_{m,l}^T} \mathbf{J}_n], \quad (52)$$

where $\nabla_{\mathbf{A}_{m,l}^T}$ and $\mathbf{H}_{\mathbf{A}_{m,l}^T}$ are the gradient and Hessian operators respectively. Now

$$E_{i_n, \mathbf{x}_n} [\nabla_{\mathbf{A}_{m,l}^T} \mathbf{J}_n] = E_{i_n, \mathbf{x}_n} [E_{j_n | i_n} [\nabla_{\mathbf{A}_{m,l}^T} (\mathbf{x}_{n,l} - \mathbf{A}_{j_n,l} \tilde{\mathbf{y}}_n)^2]] \quad (53)$$

$$\approx -2 E_{i_n, \mathbf{x}_n} [P(m | i_n) E_{j_{n-1} | i_{n-1}} [(\mathbf{x}_{n,l} - \mathbf{A}_{m,l} \tilde{\mathbf{y}}_n) \tilde{\mathbf{y}}_n]], \quad (54)$$

where (10) and (21) are used to obtain (53), and (54) is obtained by ignoring that $\tilde{\mathbf{y}}_n$ depends on $\mathbf{A}_{m,l}$ for simplicity, cf. the development in [3]. Further

$$E_{i_n, \mathbf{x}_n} [\mathbf{H}_{\mathbf{A}_{m,l}^T} \mathbf{J}_n] = E_{i_n, \mathbf{x}_n} [E_{j_n | i_n} [\mathbf{H}_{\mathbf{A}_{m,l}^T} (\mathbf{x}_{n,l} - \mathbf{A}_{j_n,l} \tilde{\mathbf{y}}_n)^2]] \quad (55)$$

$$\approx 2 E_{i_n, \mathbf{x}_n} [E_{j_n | i_n} [\nabla_{\mathbf{A}_{m,l}^T} (\mathbf{x}_{n,l} - \mathbf{A}_{j_n,l} \tilde{\mathbf{y}}_n) \cdot (\nabla_{\mathbf{A}_{m,l}^T} (\mathbf{x}_{n,l} - \mathbf{A}_{j_n,l} \tilde{\mathbf{y}}_n))^T]] \quad (56)$$

$$\approx 2 E_{i_n, \mathbf{x}_n} [P(m | i_n) E_{j_{n-1} | i_{n-1}} [\tilde{\mathbf{y}}_n \tilde{\mathbf{y}}_n^T]] \quad (57)$$

$$= 2 E_{i_n, \mathbf{x}_n} \left[P(m | i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} \right], \quad (58)$$

where (10) and (21) are used to obtain (55), the approximation that the second derivatives are relatively small, which is the Gauss-Newton approximation [31], is employed to achieve (56), and (57) is obtained by ignoring that $\tilde{\mathbf{y}}_n$ depends on $\mathbf{A}_{m,l}$ for simplicity, cf. again the development in [3]. Combining (52), (54), and (58), we have

$$\begin{aligned} \mathbf{A}_{m,l}^{\text{NEW},T} &\approx \mathbf{A}_{m,l}^T - \mu \left(E_{i_n, \mathbf{x}_n} \left[P(m | i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1} \end{bmatrix} \right] \right)^{-1} \\ &\quad \cdot E_{i_n, \mathbf{x}_n} [P(m | i_n) E_{j_{n-1} | i_{n-1}} [(\mathbf{A}_{m,l} \tilde{\mathbf{y}}_n - \mathbf{x}_{n,l}) \tilde{\mathbf{y}}_n]]. \end{aligned} \quad (59)$$

Through writing (59) for all rows, and using the linear map property of matrix multiplication, we achieve

$$\mathbf{A}_m^{\text{NEW}} = \mathbf{A}_m - \mu E_{i_n, \mathbf{x}_n} \left[P(m | i_n) \right]$$

$$\cdot \left(\mathbf{A}_m \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1}^T \end{bmatrix} - \mathbf{x}_n \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \end{bmatrix} \right) \cdot \left(\mathbf{E}_{\mathbf{i}_n, \mathbf{x}_n} \left[P(m|i_n) \begin{bmatrix} 1 & \mathbf{s}_{n-1}^T \\ \mathbf{s}_{n-1} & \mathbf{W}_{n-1}^T \end{bmatrix} \right] \right)^{-1} \cdot \quad (60)$$

By assuming stationarity and ergodicity and replacing the expectations with arithmetic means, the matrix update (44) in Section IV-D2 is obtained. While not always fulfilled, stationarity and ergodicity assumptions have been applied with success in many source coding strategies, see e.g., [17], [18].

REFERENCES

- [1] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
- [2] N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [3] T. Eriksson and F. Nordén, "Memory-based vector quantization of LSF parameters by a power series approximation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 4, pp. 1146-1155, May 2007.
- [4] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction," IETF RFC 2733, Dec. 1999.
- [5] T. Berger, *Rate-Distortion Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [6] P.-C. Chang and R. Gray, "Gradient algorithms for designing predictive vector quantizers," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 4, pp. 679-690, Aug. 1986.
- [7] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. Commun.*, vol. 38, no. 9, pp. 1285-1287, Sep. 1990.
- [8] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 685-696, July 1985.
- [9] Y. Shoham, "Vector predictive quantization of the spectral parameters for low rate speech coding," in *Proc. ICASSP*, vol. 12, Apr. 1987, pp. 2181-2184.
- [10] J. Foster, R. Gray, and M. Dunham, "Finite-state vector quantization for waveform coding," *IEEE Trans. Inf. Theory*, vol. 31, no. 3, pp. 348-359, May 1985.
- [11] T. Eriksson, J. Linden, and J. Skoglund, "Interframe LSF quantization for noisy channels," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 5, pp. 495-509, Sep. 1999.
- [12] J. Samuelsson and P. Hedelin, "Recursive coding of spectrum parameters," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 5, pp. 492-503, July 2001.
- [13] "ITU-T recommendation G.726 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (adpcm)," ITU-T, 1990.
- [14] A. Tamhankar and K. Rao, "An overview of H.264/MPEG-4 part 10," in *Proc. 4th EURASIP Conf. Video/Image Processing Multimedia Commun.*, July 2003, pp. 1-51, vol. 1.
- [15] P. Hedelin, P. Knagenhjelm, and M. Skoglund, "Theory for transmission of vector quantization data," in *Speech Coding and Synthesis*, W. Kleijn and K. Paliwal, Eds. Amsterdam, The Netherlands: Elsevier, 1995, pp. 347-396.
- [16] P. Knagenhjelm and E. Agrell, "The Hadamard transform—a tool for index assignment," *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1139-1151, July 1996.
- [17] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Trans. Inf. Theory*, vol. 36, no. 4, pp. 799-809, July 1990.
- [18] J. Lindén, "Channel optimized predictive vector quantization," *IEEE Trans. Speech Audio Processing*, vol. 8, no. 4, pp. 370-384, July 2000.
- [19] H. Kumazawa, M. Kasahara, and T. Namekawa, "A construction of vector quantizers for noisy channels," *Electron. Eng. Jpn.*, vol. 67-B, no. 4, pp. 39-47, 1984.
- [20] K. Zeger and A. Gersho, "Vector quantizer design for memoryless noisy channels," in *Proc. IEEE Int. Conf. Commun.*, June 1988, pp. 1593-1597, vol. 3.
- [21] N. Farvardin and V. Vaishampayan, "On the performance and complexity of channel-optimized vector quantizers," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 155-160, Jan. 1991.
- [22] E. Ayanoglu and R. Gray, "The design of joint source and channel trellis waveform coders," *IEEE Trans. Inf. Theory*, vol. 33, no. 6, pp. 855-865, Nov. 1987.
- [23] M. Wang and T. Fischer, "Trellis-coded quantization designed for noisy channels," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1792-1802, Nov. 1994.
- [24] Y. Hussain and N. Farvardin, "Finite-state vector quantization over noisy channels and its application to LSP parameters," in *Proc. ICASSP*, Mar. 1992, pp. 133-136, vol. 2.
- [25] Y. Hussain, "Design and performance evaluation of a class of finite-state vector quantizers," Ph.D. dissertation, University of Maryland, College Park, MD, 1992.
- [26] P. Yahampath and M. Pawlak, "On finite-state vector quantization for noisy channels," *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2125-2133, Dec. 2004.
- [27] F. Itakura, "Line spectrum representation of linear predictive coefficients," *J. Acoust. Soc. Amer.*, vol. 57, no. 1, p. S35, 1975.
- [28] D. Persson and T. Eriksson, "Memory-based multiple description coding," *IEEE Trans. Commun.*, 2009.
- [29] T. Cover and J. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons, 1991.
- [30] K. K. Paliwal and W. B. Kleijn, "Quantization of LPC parameters," in *Speech Coding and Synthesis*, W. Kleijn and K. Paliwal, Eds. Amsterdam, The Netherlands: Elsevier, 1995, pp. 433-466.
- [31] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

Daniel Persson was born in Halmstad, Sweden, in 1977. He graduated from Ecole Polytechnique, Paris, France, in 2002, received the M.Sc. degree in engineering physics from Chalmers University of Technology, Göteborg, Sweden, in 2002, and the Ph.D. degree in electrical engineering from Chalmers University of Technology in 2009. He is currently a postdoctoral researcher in the Communication Systems Division, Linköping University, Sweden. His research interests are joint source-channel coding and MIMO detection.

Thomas Eriksson was born in Skövde, Sweden, on April 7, 1964. He received the M.Sc. degree in electrical engineering and the Ph.D. degree in information theory from Chalmers University of Technology, Göteborg, Sweden, in 1990 and 1996, respectively.

He was at AT&T Labs-Research from 1997 to 1998, and in 1998 and 1999, he was working on a joint research project with the Royal Institute of Technology and Ericsson Radio Systems AB. Since 1999, he has been an Associate Professor at Chalmers University of Technology, and his research interests include vector quantization, speaker recognition, and system modeling of nonideal hardware.