



THESIS REPORT

Ph.D.

Quantization Over Discrete Noisy Channels Under Complexity Constraints

by N. Phamdo
Advisor: N. Farvardin

Ph.D. 93-7

*The Institute for Systems Research is supported by the
National Science Foundation Engineering Research Center Program (NSFD CD 8803012),
the University of Maryland, Harvard University, and Industry*

QUANTIZATION OVER DISCRETE NOISY CHANNELS
UNDER COMPLEXITY CONSTRAINTS

by

Nam Phamdo

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1993

Advisory Committee:

Professor Nariman Farvardin, Chairman/Advisor
Associate Professor Thomas Fuja
Associate Professor Steven Tretter
Associate Professor Adrian Papamarcou
Associate Professor David Mount

ABSTRACT

Title of Dissertation: QUANTIZATION OVER DISCRETE NOISY
CHANNELS UNDER COMPLEXITY
CONSTRAINTS

Nam Phamdo, Doctor of Philosophy, 1993

Dissertation directed by: Dr. Nariman Farvardin
Professor
Electrical Engineering Department

A fundamental problem in communication is the transmission of an information source across a communication channel. According to Shannon's separation principle, this problem can be separated (without loss of optimality) into two different, yet similar, problems: source coding and channel coding. This result, however, holds only when complexity and delay are not an issue. In practical situations, complexity plays a major role in many system designs. When complexity is constrained, treating these two problems jointly may prove to be more fruitful than treating them separately.

In this work, we consider two approaches to joint source-channel coding of discrete-time, continuous-amplitude sources and discrete memoryless channels when complexity is constrained.

In the first approach, we consider the analysis and design of two low-complexity vector quantizers — the tree-structured vector quantizer (TSVQ) and the multi-stage vector quantizer (MSVQ) — when used over a noisy channel. The resulting schemes are called channel-matched TSVQ and channel-matched MSVQ. These schemes are compared with (i) the ordinary TSVQ and MSVQ which are designed for the noiseless channel and (ii) a tandem source-channel coding scheme in which the source and channel codes are designed separately.

In the second approach, we assume a low-complexity quantizer (i.e., source code) is given. Because of its low complexity, the quantizer is sub-optimum and hence certain redundancy remains at its output. Our aim is to make use of this redundancy to combat channel noise. We consider two scenarios: (i) the redundancy is in the form of memory and (ii) it is in the form of a non-uniform distribution. In the second case, we propose the use of a rate-one convolutional code to convert the residual redundancy into a usable form. Comparisons are also made with a tandem source-channel coding scheme.

To Minh H. Phamdo

Acknowledgements

I would like to thank my advisor, Professor Nariman Farvardin, for his guidance and support. My time with him has been a rewarding learning experience. I thank my friends in the Communications and Signal Processing Laboratory for their help and for the stimulating hours of discussions. I would also like thanks my friends and colleagues at NTT Human Interface Laboratories for their help and support during my visit there. I am grateful to Dr. Noboru Sugamura, Dr. Takehiro Moriya and Dr. Sadaoki Furui for their guidance at various stages of my study. Finally, I like to thank my parents for their love and support.

Table of Contents

<u>Section</u>	<u>Page</u>
1 Introduction	1
1.1 Previous Work on Joint Source-Channel Coding	4
1.2 Contribution of the Dissertation	7
1.2.1 Quantizers Designed for Noisy Channels	7
1.2.2 Exploiting the Quantizer's Residual Redundancy	8
1.3 Outline of the Dissertation	10
2 TSVQ and MSVQ Designed for Noisy Channels	11
2.1 Introduction	11
2.2 Preliminaries	13
2.2.1 Structure of TSVQ and MSVQ	14
2.2.2 Channel Noise	16
2.3 Channel-Matched TSVQ and MSVQ	17
2.3.1 Problem Statement	17
2.3.2 Necessary Conditions	20
2.3.3 Design Algorithm	22
2.3.4 Channel-Matched MSVQ	23
2.4 Numerical Results	24
2.4.1 Choice of BER for the First Stage	26
2.4.2 Larger Block Sizes for the CM-MSVQ	27

2.4.3	Multiple Candidate Codebook Search	28
2.5	Tandem Source-Channel Coding	31
2.6	Channel Mismatch	35
2.7	Complexity Issues	37
2.8	Summary and Conclusions	38
3	Exploiting the Quantizer's Residual Redundancy	59
3.1	Redundancy in the Form of Memory	59
3.1.1	Introduction	59
3.1.2	Preliminaries	62
3.1.3	Statement of Problems and Solutions	64
3.1.4	Special Case: BSMS and BSC	67
3.1.5	Joint Source-Channel Coding	70
3.1.6	Conclusions	74
3.2	Redundancy in the Form of a Non-Uniform Distribution	75
3.2.1	Introduction	75
3.2.2	Motivation	79
3.2.3	Rate-One Convolutional Codes	81
3.2.4	Simulation Results	92
3.2.5	Conclusions	97
4	Summary and Future Work	120
A	Proof of Theorem 3	124
B	Proof of Theorem 4	126
C	Implementation of MAP Symbol-By-Symbol Decoder	128
	Bibliography	131

List of Figures

<u>Number</u>		<u>Page</u>
2.1	Block Diagram of the TSVQ/MSVQ Scheme Over Noisy Channels.	48
2.2	Block Diagram of a Tandem Source-Channel Coding Scheme Over a BSC and the Equivalent Channel Model.	48
2.3	SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Effective Block Length of Both Schemes = 12.	49
2.4	SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Effective Block Length of Both Schemes = 18.	50
2.5	SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Effective Block Length of Both Schemes = 24.	51
2.6	Comparisons Between Analytical Results and Simulated Results of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9.	52

2.7	SNR (in dB) Performances of the CM-TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions.	53
2.8	SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions.	54
2.9	SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions.	55
2.10	SNR (in dB) Performances of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions.	56
2.11	SNR (in dB) Performances of the CM-MSVQ and the Tandem Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions.	57
2.12	Comparisons Between CM-MSVQ and Tandem Scheme with the Same Complexity.	58
3.1	Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.9$	106
3.2	Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.95$	107
3.3	Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.99$	108
3.4	Block Diagram of Joint Source-Channel Coding System Using MAP Detection.	109

3.5	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k = 1$; $R = 1$ Bits/Sample.	110
3.6	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k = 1$; $R = 2$ Bits/Sample.	111
3.7	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k = 1$; $R = 3$ Bits/Sample.	112
3.8	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k = 2$; $R = 1$ Bits/Sample.	113
3.9	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k = 4$; $R = 1$ Bits/Sample.	114
3.10	Block Diagram of Convolutionally Encoded System.	115
3.11	Convolutional Encoder with Feedback; $K =$ Constraint Length. . .	115
3.12	SNR (in dB) Performances of Convolutionally Encoded Systems with Symbol-By-Symbol and MMSE Decoding Vs. Delay; Gen- eralized Gaussian Source with $\alpha = 0.5$; $R =$ Rate of Quantizer (Bits/Sample); $\epsilon =$ Channel Bit Error Rate; $K =$ Constraint Length of Convolutional Encoder; $2^{KR} =$ Number of States.	116

3.13	SNR (in dB) Performances of Convolutionally Encoded Systems with MAP Sequence Decoding Vs. Redundancy of Entropy-Constrained Scalar Quantizer; Generalized Gaussian Source with $\alpha = 1.0$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.	117
3.14	SNR (in dB) Performances of Convolutionally Encoded Systems with MAP Sequence Decoding Under Channel Mismatch; General- ized Gaussian Source with $\alpha = 0.5$; R = Rate of Quantizer (Bits/Sample); ϵ_d = Design BER; ϵ_a = Actual Channel BER; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.	118
3.15	Comparisons of Proposed System Versus Tandem Source-Channel Coding System; Generalized Gaussian Source with $\alpha = 0.5$; R_s = Rate of Quantizer in Tandem System; R = Overall Rate; ϵ_d = Design BER; ϵ_a = Actual Channel BER.	119

List of Tables

<u>Number</u>		<u>Page</u>
2.1	SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Memoryless Gaussian Source.	40
2.2	SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Memoryless Gaussian Source.	41
2.3	SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9.	42
2.4	SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9.	43
2.5	SNR (in dB) Performance Results of the CM-TSVQ and CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of ϵ_1	43
2.6	SNR (in dB) Performance Results of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k	44
2.7	Choices of Design BER's for the Results in Tables 2.6, 2.8 and 2.9.	44
2.8	SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of M	45

2.9	SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k and Different Values of M	46
2.10	Complexity of Two-Stage CM-MSVQ and CM-TSVQ and COVQ.	46
2.11	Complexity and Delay of the CM-MSVQ and the Tandem Scheme in Figure 2.12.	47
3.1	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$	98
3.2	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$	99
3.3	SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$	100
3.4	SNR (in dB) Performances of Joint Source-Channel Coding Scheme Using COVQ Encoding and AMMSE Decoding for AR(1) Source with $\rho = 0.9$	101
3.5	SNR (in dB) Performances of Several Scalar Quantizers Operating Over Noiseless and Noisy Channels; Generalized Gaussian Source with $\alpha = 0.5$; ϵ = Channel Bit Error Rate; OPTA = Optimum Performance Theoretically Attainable.	101
3.6	A List of Good Rate-One Convolutional Codes for the Generalized Gaussian Source with $\alpha = 0.5$; ϵ = Channel Bit Error Rate; R = Rate of Lloyd-Max Quantizer; K = Constraint Length; 2^{RK} = Number of States.	102

3.7	SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded System with MAP Sequence Decoding; Generalized Gaussian Source with $\alpha = 0.5$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.	102
3.8	SNR (in dB) Performances of Convolutionally Encoded Systems Using Three Different Decoding Techniques; Generalized Gaussian Source with $\alpha = 0.5$; K = Constraint Length; 2^{KR} = Number of States.	103
3.9	SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded Systems with MAP Sequence Decoding; Generalized Gaussian Source with $\alpha = 1.0$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.	103
3.10	A List of Good Rate-One Convolutional Codes for the Generalized Gaussian Source with $\alpha = 1.0$; ϵ = Channel Bit Error Rate; R = Rate of Lloyd-Max Quantizer; K = Constraint Length; 2^{RK} = Number of States.	104
3.11	SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded Systems with MAP Sequence Decoding and Rate- and Entropy-Constrained Scalar Quantizers; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.	104
3.12	SNR (in dB) Performances of COVQ and Convolutionally Encoded System with MAP Sequence Decoding and Rate- and Entropy-Constrained Scalar Quantizers; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; k = dimension of COVQ.	105

Chapter 1

Introduction

A fundamental problem in communication engineering is the transmission of an information source from one point to another. The two points may be separated in space or in time (as in storage) and the link between them is called the channel. In this problem, the objective is to design an encoder-decoder pair such that the reproduction at the destination resembles the original source as closely as possible. Typically, a measure of the distortion between the reproduction and the original source is given.

For this problem, an important question is the following: “Given a source, channel and distortion measure, what is the minimum distortion that can be attained?”. In 1959, Shannon [49] provided the answer to this question. Shannon showed that given a discrete, stationary and ergodic source and a single-letter distortion measure there is a rate-distortion function which can be interpreted as the information rate of the source for a given level of distortion. In an earlier work [48], he showed that given a discrete memoryless channel there is a number called the capacity which is the maximum amount of information that can be reliably transmitted over this channel. Combining these two results with a converse theorem, Shannon showed that the minimum possible distortion is that for which the rate-distortion function equals the capacity [49].

An important consequence of Shannon's work is that the above mentioned problem can theoretically be separated into two different, yet similar, problems: source coding and channel coding. In source coding, the goal is to compress the information source at the lowest possible rate for a given level of distortion or, conversely, with as little distortion as possible for a given rate. This is done by removing the source redundancy and retaining only the essential information. Let $R(d)$ be the rate-distortion function¹ of the source with respect to the given distortion measure and $\phi(\cdot)$ be its inverse, called the distortion-rate function. If d is the allowable distortion, then $R(d)$ is the minimum possible compression rate (in bits/second). Conversely, if R is the given rate, then $d = \phi(R)$ is the minimum possible average distortion. Therefore, the curve $R(d)$ provides a theoretical bound on the rate-distortion performance on any source code. Rate-distortion theory [49], [8] states that there exist source codes with rate-distortion performances lying arbitrarily close to $R(d)$ — provided that they are allowed to operate on blocks of arbitrarily large length.

Channel coding, on the other hand, involves the use of redundancy for protection of the bit stream (coming from the source code) against channel errors. If C is the channel capacity, then the maximum amount of information that can be reliably transmitted across the channel is C (in bits/second). Information theory says that there exist block channel codes which transmit arbitrarily close to C bits of information across this channel per second with arbitrarily small probability of error provided that the block length is arbitrarily large [48], [52].

For a given channel with capacity C , let $d^* = \phi(C)$ be the minimum possible average distortion, i.e., $R(d^*) = C$. Since there are source codes which compress

¹Here, we assume that the source produces symbols (or samples) at a constant rate of r_s samples/second. Similarly, the channel is assumed to be used r_c times per second. Typically, the rate-distortion function and the channel capacity are measured in bits/sample and bits/use, respectively. In this chapter, however, we assume that both are measured in bits/second. This can be done by normalizing to r_s or r_c .

the source to $R(d^*)$ bits per second with a distortion close to d^* and since there are channel codes which can reliably transmit the compressed source across the channel, it is possible to achieve a distortion close to d^* [49], [52]. The significance of this result is that the source and channel codes can be designed independently without any loss of optimality. This is important because the effect of channel errors does not play a role in the design of the source code and similarly the characteristics of the source and the distortion measure play no role in the design the channel code. The complete system, formed by concatenating the independently-designed source and channel codes together, still achieves a distortion d close to the optimum d^* . A system which is formed this way shall hereafter be referred to as a *tandem* source-channel coding system.

A major drawback of the tandem source-channel coding approach is that, in order to come close to d^* , both source and channel codes must operate on blocks of arbitrarily large lengths. In implementation, this translates to arbitrarily large complexity and delay. Even though information theory generally ignores complexity, it is a serious issue in many practical system designs. In cases where complexity and delay are constrained, it is usually not possible to achieve a distortion close to d^* using the tandem source-channel coding approach. Then the following question naturally arises: “Is there anything to be gained by a joint design of the source and channel codes and if so, how much?”. By a joint design, we mean that (i) certain characteristics of the source, the source code and the distortion measure are used to design the channel code or (ii) certain characteristics of the channel and channel code are used to design the source code. A system in which the source and channel codes are jointly designed will be referred to as a *joint* source-channel coding system.

The goal of this dissertation is to design joint source-channel codes which outperform tandem source-channel codes under the same constraints on complexity

and delay. Here, we are not claiming that, under these constraints, a joint system could achieve a distortion close to d^* while a tandem system could not. Our belief is that, except for a few special cases, neither system will be able to achieve such a distortion under these constraints. However, it is hoped that, under practical constraints on complexity and delay, a joint system is able to achieve a distortion *closer* to d^* than a tandem system.

1.1 Previous Work on Joint Source-Channel Coding

Joint source-channel coding has a long history — dating back to the early 1960's. Through the years, various joint source-channel coding techniques have been introduced for various applications. In general, these techniques can be classified into three different categories: (1) the unequal error protection approach, (2) the combined source-channel coding approach and (3) the zero-redundancy channel coding approach. These approaches are described next.

In *unequal error protection*, not all of the bits from the output of the source encoder are treated equally. The idea is that some of the bits are more sensitive to channel errors than others. Therefore, these bits should be protected more heavily than the rest. Generally, the bit stream is divided into L classes. The first class consists of the most sensitive bits and therefore is most heavily protected. Similarly, the second class consists of the second most sensitive bits and is the second most heavily protected, and so on. The last class is given the least amount of protection or, in some cases, no protection at all. As an example, in [36], two levels of protection ($L = 2$) were used in a DPCM-based image coding system. There, the most significant bits are protected by a convolutional code while the least significant bits are left unprotected. A similar study for a transform-based image coding system was made in [37]. The results in [36] and [37] indicate that

at the same overall bit rate the coded system provides a significant performance improvement over the uncoded system when the channel is noisy. Another example is the four levels of protection ($L = 4$) used in a transform-based speech coder in [50]. In this coder, the first class is protected by a rate-1/3 repetition code; the second class by a rate-7/15 BCH code; the third class by a rate-21/31 BCH code and the last class is left unprotected.

In *combined source-channel coding*², the functions of the source and channel codes are served by a single code. In designing this code, the objective is to minimize the average distortion of the overall system subject to some constraint on the code structure. Perhaps the first to formulate this problem was Fine [17] in 1964. Fine considered a digital system model general enough to include a quantization system over a discrete noisy channel. He then derived necessary conditions for optimality of this system. Kurtenbach and Wintz [30], and later Farvardin and Vaishampayan [15], considered the design of scalar (or zero-memory) quantizers for transmission over discrete memoryless channels. In both papers, the scalar quantizers are designed using an iterative algorithm in which two optimality criteria (similar to those derived by Fine) are successively solved. This technique was extended to vector quantizers in [29] and [16]. The same technique was used in the design of a shape-gain vector quantizer for noisy channel image coding applications in [45] and the design of a trellis waveform coder in [4]. In [4], it was shown that, when the channel is very noisy, the joint source-channel trellis waveform coder outperforms a tandem source-channel trellis coder, where the complexities of both systems are approximately the same.

In *zero-redundancy channel coding*, no redundancy is added to the output of the

²Some authors use the term “combined source-channel coding” synonymously with “joint source-channel coding”. However, in this dissertation, we restrict the use of the term “combined source-channel coding” to those cases where the source and channel codes are combined into a single code.

source encoder. Instead, the characteristics of the source, the source code and/or the distortion measure are used to provide protection against channel noise. For example, if there is some redundancy remaining at the output of the source encoder, then this “residual” redundancy can be used to combat channel noise [47]. The residual redundancy can be in the form of memory, in the form of a non-uniform distribution or both. In [47], it was assumed that the residual redundancy is in the form of memory. This memory was used at the decoder to reduce the effect of channel noise. Here, the characteristics of the source and the source encoder are used to combat noise. In another example, when the source code is a vector quantizer, a judicious choice of mapping between the vector quantizer output and channel input symbols may lead to a reduction in the distortion due to channel noise [10], [53], [13]. This is known as the index assignment problem. The idea here is to come up with a distance preserving mapping from the source space to the channel space. Thus, “small” channel noise results in “small” distortion. Generally, no optimal solution exists for this problem though several sub-optimal techniques had been suggested [10], [53], [13].

In this dissertation, we will take the latter two approaches (combined source-channel coding and zero-redundancy channel coding). Our study is restricted to discrete-time, continuous-amplitude sources and discrete memoryless channels. Thus, the source codes which we will consider are quantizers. Emphasis is placed on quantizers with low complexity. In Chapter 2, we present two structured (i.e., low-complexity) vector quantizers which are designed for noisy channels (category 2). We then present two methods of exploiting the quantizer’s residual redundancy for channel error protection (category 3) in Chapter 3.

1.2 Contribution of the Dissertation

1.2.1 Quantizers Designed for Noisy Channels

In the first part of the dissertation, we will consider the design of two low-complexity vector quantizers (VQ's) — namely, the tree-structured VQ (TSVQ) and the multi-stage VQ (MSVQ) — for transmission over a noisy channel. These schemes belong to the combined source-channel coding category (category 2).

A TSVQ is a VQ scheme in which the encoder computational complexity is reduced by imposing a tree structure on the codebook. Instead of quantizing the source vector using a large-size VQ, the source vector is first quantized by a small-size “primary” VQ. Depending on which of the codevectors (in the primary VQ) the source vector is closest to, a different (small-size) “secondary” VQ is then used for quantization. This process is repeated until the desired rate is obtained. Even though the encoder computational complexity is reduced using the TSVQ, the encoder memory is increased. The MSVQ, on the other hand, is a VQ scheme in which both the encoder computational complexity and memory are reduced. It is similar to the TSVQ except that it has the additional constraint that all secondary VQ's are equivalent – up to a linear translation.

In Chapter 2, the performances of the TSVQ and MSVQ when used over a noisy channel are analyzed. Necessary conditions for optimality of the secondary VQ's given the primary VQ are obtained and an algorithm for designing TSVQ and MSVQ for a given noisy channel is developed. The proposed quantizers are thus referred to as channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ). Numerical results will be given for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient 0.9 and comparisons will be made with the ordinary TSVQ and MSVQ designed for the noiseless channel. The performance of the CM-MSVQ, which is a combined source-channel code, is

compared with a tandem source-channel code which has the same block length (hence, same encoding delay) as the CM-MSVQ and consists of an optimum (full-searched) VQ followed by an optimum linear block code. It will be shown that, for the Gauss-Markov source, the combined source-channel coding scheme using CM-MSVQ is superior to the tandem source-channel coding scheme.

1.2.2 Exploiting the Quantizer's Residual Redundancy

In the second part of the dissertation, we consider exploiting the quantizer's residual redundancy for channel error protection. Two types of redundancy are considered: (i) redundancy in the form of memory and (ii) redundancy in the form of a non-uniform distribution. Since no additional redundancy is added to the quantizer output, the methods introduced here belong to the zero-redundancy channel coding category (category 3).

Exploiting the Residual Memory

Here, we assume that the source is highly correlated and that the source code is a VQ (either structured or unstructured). Due to constraints on complexity and/or delay, we are restricted to VQ's with small block lengths. Since the VQ input is correlated from block-to-block, its output will also be correlated from block-to-block. This correlation will be referred to as the *residual memory*.

To exploit the residual memory, we make the assumption that the VQ output forms a *discrete Markov chain*. This Markov chain is fed directly into the channel. At the channel output, we would like to make an estimate of what came into the channel. Specifically, we consider two *maximum a posteriori* (MAP) detectors: (i) a *sequence* MAP detector in which the objective is to determine the most probable transmitted sequence given the observed sequence and (ii) an *instantaneous* MAP detector which determines the most probable transmitted symbol at a particular

time given all the observations up to that time. The solution to the sequence detection problem results in a “Viterbi-like” implementation of the MAP detector while the solution to the instantaneous detection problem results in a recursive implementation. We will show, through simulation, that the MAP detectors can successfully exploit the VQ’s residual memory for combating channel errors. Also, for the special case of binary Markov chain and binary symmetric channel, we will derive a necessary and sufficient condition for the usefulness of the sequence MAP detector.

Exploiting the Residual Non-Uniformity

In many source coding applications, some operation on the source is first performed before encoding. Examples are transform coding and predictive coding. The purpose of these operations is to render the source memoryless which, in turn, makes it easier to encode. In transform coding, the encoder operates on the transformed coefficients and in predictive coding, it operates on the prediction residues. In either cases, the encoder input (which can be viewed as a new source) is essentially memoryless.

Thus, in this section, we assume that the given (new) source is memoryless. Furthermore, we will assume that the source code is a scalar quantizer (SQ). Since the SQ input is memoryless, its output is also memoryless. However, the source distribution is assumed to be highly non-uniform (this appears to be the case in transform [44] and subband [51] coding of images). Thus the SQ output distribution will also be non-uniform. This non-uniform distribution at the SQ output is referred to as the *residual non-uniformity*.

Due to the nature of its redundancy, the SQ output is not suitable for direct transmission over the noisy channel. That is, if the SQ output is directly transmitted over the channel, the residual non-uniformity cannot be fully exploited at

the decoder. Instead, we attempt to “match” the SQ output to the channel using a rate-one convolutional encoder. A rate-one convolutional encoder is one which produces as many bits as it accepts. It introduces no additional redundancy and hence it does not increase the bit rate. Its function is to convert the residual redundancy in the form of a non-uniform distribution into redundancy in the form of memory which is well suited for transmission over the noisy channel. In this dissertation, we will derive an approximate upper bound on the average distortion of the proposed system. We then design the convolutional encoder to minimize this bound. Simulation results will be presented and comparisons will be made with several other systems — including a tandem source-channel coding system.

1.3 Outline of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we discuss the problem of designing TSVQ and MSVQ for noisy channels. The use of the quantizer’s residual redundancy for channel error protection is presented in Chapter 3. In Section 3.1, we consider residual redundancy in the form of memory while in Section 3.2, we consider residual redundancy in the form of a non-uniform distribution. Finally, a summary of the dissertation and suggestions for future work are given in Chapter 4.

Chapter 2

TSVQ and MSVQ Designed for Noisy Channels

2.1 Introduction

One of the most important results of Shannon's pioneering work was that source coding and channel coding can be treated separately without sacrificing optimality [48], [49]. This "separation principle" has led to the development of what is known as a tandem source-channel coding scheme, in which the source and channel codes are separately designed but are then used in a cascade combination with one another. Shannon's separation principle holds provided that the source and channel codes are allowed to operate on blocks of arbitrarily large lengths [52]. In practice, however, due to certain constraints on the delay or complexity, the block lengths of the source and channel codes must be finite. In such situations, an alternative approach is to combine the source and channel codes into a single code in which the objective is to minimize the average distortion between the source and its reproduction. Such an approach is referred to as combined source-channel coding.

The study of combined source-channel coding for continuous-amplitude sources leads naturally to the study of quantizer design for noisy channels. Design algorithms for an optimum scalar quantizer operating over a noisy channel were devel-

oped in [30], [15]. These ideas were extended to vector quantizers (VQ's) in [29], [16]. The results in [16] indicate that the VQ designed for a noisy channel has better performance than its scalar counterpart – especially when the source has memory. The performance improvement increases as the block length increases. Unfortunately, increasing the block length is not always possible since the VQ is constrained by its complexity, which grows exponentially with the product of the rate and block length.

In the context of source coding alone, i.e., when the channel is assumed to be noiseless, methods of reducing VQ complexity have been studied extensively [22], [23], [27]. Some examples of low-complexity VQ's are tree-structured VQ (TSVQ), multi-stage VQ (MSVQ), product VQ and lattice VQ. For a complete description of these schemes, the interested reader is referred to [22], [20]. In this chapter, we will focus attention to the TSVQ and MSVQ. A description of TSVQ and MSVQ are given in Section 2.2.1.

In this chapter, the performances of the TSVQ and MSVQ when used over a noisy channel are analyzed. Necessary conditions for optimality are developed and an algorithm for designing TSVQ and MSVQ for a given noisy channel is presented. The resulting quantizers are called channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ). Numerical results are presented for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient 0.9 and comparisons are made with the ordinary TSVQ and MSVQ designed for the noiseless channel. The performance of the CM-MSVQ is then compared with a tandem source-channel code which has the same block length as the CM-MSVQ and consists of an optimum (full-searched) VQ followed by an optimum linear block code. It is demonstrated that, for the Gauss-Markov source, the combined source-channel coding scheme using CM-MSVQ is superior to the tandem source-channel coding scheme. Study of the CM-MSVQ under channel mismatch

shows that it has a favorable robustness property.

The rest of this chapter is organized as follows. In Section 2.2, preliminary issues concerning the structure of the TSVQ and MSVQ are presented. Section 2.3 includes a description of the CM-TSVQ and CM-MSVQ problems, the necessary conditions for optimality as well as the design algorithm. Numerical results are presented in Section 2.4. In Section 2.5, a description of the tandem source-channel coding scheme is given and a model of the channel and the channel code is established for comparison purposes. Channel mismatch issues are addressed in Section 2.6. Section 2.7 contains a brief discussion of the complexity of the CM-TSVQ and CM-MSVQ. Finally, a summary and conclusions are presented in Section 2.8.

2.2 Preliminaries

In the sequel we assume that the source to be encoded is a real-valued, stationary and ergodic process $\mathcal{X} = \{X_n\}_{n=0}^{\infty}$. A k -dimensional N -level vector quantizer (VQ) for $\mathcal{X}$ is in general described by an encoder mapping α and a decoder mapping β . The encoder mapping $\alpha : \mathbb{R}^k \mapsto \mathcal{J}_N \triangleq \{0, 1, \dots, N-1\}$ is given by

$$\alpha(\mathbf{x}) = i \quad \text{if } \mathbf{x} \in S_i, \quad i \in \mathcal{J}_N, \quad (2.1)$$

where $\mathbf{x} = (x_{nk}, x_{nk+1}, \dots, x_{nk+k-1})$ is a block of k successive samples from $\mathcal{X}$ and $\mathcal{P} \triangleq \{S_0, S_1, \dots, S_{N-1}\}$ is a partition of $\mathbb{R}^k$. The decoder mapping $\beta : \mathcal{J}_N \mapsto \mathbb{R}^k$ is described in terms of a reproduction codebook $\mathcal{C} = \{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{N-1}\}$ according to

$$\beta(i) = \mathbf{c}_i, \quad i \in \mathcal{J}_N. \quad (2.2)$$

For a given input vector $\mathbf{x}$, its reproduction is given by the quantization rule $q = \beta \circ \alpha$, where

$$q(\mathbf{x}) = \beta(\alpha(\mathbf{x})). \quad (2.3)$$

The rate of the VQ is

$$R = \frac{1}{k} \log_2 N \text{ bits/sample.} \quad (2.4)$$

With respect to a distortion measure $d(\cdot, \cdot)$, a VQ is said to be optimal if the average distortion ¹ $D = \frac{1}{k} E[d(\mathbf{X}, q(\mathbf{X}))]$ is minimized over all partitions $\mathcal{P}$ and reproduction codebooks $\mathcal{C}$. Two necessary conditions for optimality are [33],

$$S_i = \{\mathbf{x} : d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j) \forall j \in \mathcal{J}_N\} \quad \forall i \in \mathcal{J}_N, \quad (2.5)$$

and

$$\mathbf{c}_i = \arg \min_{\mathbf{y} \in \mathbb{R}^k} E[d(\mathbf{X}, \mathbf{y}) | \mathbf{X} \in S_i] \quad \forall i \in \mathcal{J}_N. \quad (2.6)$$

Based on (2.5) and (2.6), Linde, Buzo and Gray [33] developed an algorithm, known as the LBG algorithm, for designing (locally) optimal VQ's using a sequence of training data. The resulting VQ is often referred to as an LBG-VQ.

For a fixed k , the average distortion of the LBG-VQ decreases as R increases. Similarly, for a fixed R , the average distortion decreases as k increases. Unfortunately, the complexity required to implement the LBG-VQ increases exponentially with the product kR . Hence, in practice, complexity constraints prohibit the use of the LBG-VQ for large rate and/or dimension. The complexity can be reduced by using sub-optimal VQ's, such as tree-structured VQ (TSVQ), multi-stage VQ (MSVQ), product VQ or lattice VQ [22]. In this chapter, we will restrict our attention to TSVQ and MSVQ.

2.2.1 Structure of TSVQ and MSVQ

The TSVQ and MSVQ are similar in that they share a “successive approximation” characteristic. Instead of using a “full-rate” (N -level) VQ, the source vector, $\mathbf{x}$, is first approximated by a lower-rate (N_1 -level) VQ with partition $\mathcal{P}_1$, codebook

¹Here $\mathbf{X}$ is the k -dimensional random vector associated with $\mathbf{x}$ and $E[\cdot]$ denotes the expectation.

$\mathcal{C}_1$ and quantization rule $q_1 = \beta_1 \circ \alpha_1$ ($N_1 < N$). The quantization error of this stage,

$$\mathbf{e}_1 = \mathbf{x} - q_1(\mathbf{x}), \quad (2.7)$$

is then quantized by a second-stage (N_2 -level) VQ with corresponding $\mathcal{P}_2, \mathcal{C}_2$ and $q_2 = \beta_2 \circ \alpha_2$. The main difference between the TSVQ and the MSVQ is that in the MSVQ, a single VQ is used in the second stage, whereas in the TSVQ a different VQ is used for each different value of α_1 [22], [23], [27]. Thus, for the TSVQ, α_2 is a function of both² $\mathbf{e}_1$ and α_1 while for MSVQ it is only a function of $\mathbf{e}_1$. Actually, as defined in [22], [23], the second stage of the TSVQ quantizes $\mathbf{x} = \mathbf{e}_1 + q_1(\mathbf{x})$ directly. However, we formulate the problem by viewing the second-stage quantizer as operating on $\mathbf{e}_1$ rather than $\mathbf{x}$. In this way, both TSVQ and MSVQ fall under the same unified framework. The quantization error of the second stage,

$$\mathbf{e}_2 = \mathbf{e}_1 - q_2(\mathbf{e}_1) = \mathbf{x} - q_1(\mathbf{x}) - q_2(\mathbf{e}_1), \quad (2.8)$$

can be further quantized by a third-stage VQ, and so on. Thus, on the average, a better approximation of the source vector can be obtained at each stage. In general, a TSVQ or MSVQ can have S stages, with each stage having an intermediate rate $R_s = \frac{1}{k} \log_2 N_s$, a partition $\mathcal{P}_s$, a codebook $\mathcal{C}_s$ and a quantization rule $q_s = \beta_s \circ \alpha_s$ for $s = 1, 2, \dots, S$. Hereafter, we will assume that for each s , $\log_2 N_s$ is an integer. The overall rate of the quantizer is $R = R_1 + R_2 + \dots + R_S$ and the overall encoder mapping is

$$\alpha(\mathbf{x}) = \sum_{s=1}^S M_s \alpha_s(\mathbf{e}_{s-1}), \quad (2.9)$$

where $\mathbf{e}_0 = \mathbf{x}$, $M_s = \prod_{r=s+1}^S N_r$ for $s = 1, 2, \dots, S-1$, and $M_S = 1$. The overall decoder mapping is

$$\beta(i) = \sum_{s=1}^S \beta_s \left(\left\lfloor \frac{i}{M_s} \right\rfloor \bmod N_s \right), \quad (2.10)$$

²For notational simplicity, we shall not explicitly express the dependency on α_1 in this section.

and the overall quantization rule is given by

$$q(\mathbf{x}) = \beta(\alpha(\mathbf{x})) = \sum_{s=1}^S q_s(\mathbf{e}_{s-1}). \quad (2.11)$$

Equation (2.9) states that, in binary representation, the overall encoder output α is obtained by multiplexing the outputs of the intermediate encoders α_s for $s = 1, 2, \dots, S$, with α_1 taking the $\log_2 N_1$ most significant bit (MSB) positions, α_2 taking the next $\log_2 N_2$ MSB positions, and so forth. That is, in binary representation, $\alpha = [\alpha_1 \alpha_2 \dots \alpha_S]$. Likewise, equation (2.10) states that the overall decoder is the sum of the outputs of the intermediate decoders, where the inputs of the intermediate decoders are obtained by demultiplexing $\alpha = i$.

In the rate-distortion sense, the performances of the TSVQ and MSVQ are inferior to that of a globally optimum LBG-VQ. This is because the overall partition of (2.9) and the overall reproduction codebook of (2.10), in general, do not satisfy the necessary conditions of (2.5) and (2.6), respectively. For fixed k , S and $\{R_s\}_{s=1}^S$, since the TSVQ is more general and less restrictive, it is expected to have better rate-distortion performance than the MSVQ. The computational complexities (associated with the codebook search) of the TSVQ and MSVQ are equivalent, but depending on the number of stages S their computational complexities can be significantly less than the LBG-VQ. The TSVQ requires more memory (to store the codebook) than the LBG-VQ, while the MSVQ requires the least amount of memory among the three. More will be said about the complexities of these three schemes later on.

2.2.2 Channel Noise

Channel noise can have a damaging effect on the performance of VQ's. In the following, we assume that the encoder α and the decoder β are separated by a discrete memoryless channel (DMC) characterized by the random mapping

$\gamma : \mathcal{J}_N \mapsto \mathcal{J}_N$. The reconstructed vector is no longer $q(\mathbf{x}) = \beta(\alpha(\mathbf{x}))$ but is a composition of the three mappings:

$$q(\mathbf{x}) = \beta(\gamma(\alpha(\mathbf{x}))). \quad (2.12)$$

The effect of channel errors on the LBG-VQ has been studied in [29], [16]. In the next section, we study the channel-error effect on the TSVQ and MSVQ and develop a scheme to combat this undesirable effect.

2.3 Channel-Matched TSVQ and MSVQ

In this section, the performances of the TSVQ and MSVQ over noisy channels are analyzed and an algorithm is developed for designing TSVQ and MSVQ for a given noisy channel. The resulting schemes are thus called channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ), respectively. We have avoided using the term “channel-optimized” as in [16] since, by their definitions in [23], [27], the TSVQ and MSVQ are sub-optimal. The proposed algorithms do, however, result in local optimality at each stage of the design process.

Since TSVQ is the more general of the two, we will discuss it first. For now, let us consider only the two-stage case ($S = 2$). Extension to the general case ($S > 2$) will be made later.

2.3.1 Problem Statement

Consider the block diagram given in Figure 2.1. The first-stage (or primary) encoder, α_1 , is described in terms of a partition $\mathcal{P}_1 = \{S_{1,0}, S_{1,1}, \dots, S_{1,N_1-1}\}$ according to

$$\alpha_1(\mathbf{x}) = i \quad \text{if } \mathbf{x} \in S_{1,i}, \quad i \in \mathcal{J}_{N_1}. \quad (2.13)$$

The output of this encoder is transmitted over a DMC described by the random mapping $\gamma_1 : \mathcal{J}_{N_1} \mapsto \mathcal{J}_{N_1}$ and the transition matrix:

$$Q_1(j|i) = \Pr\{\gamma_1(i) = j\}, \quad i, j \in \mathcal{J}_{N_1}. \quad (2.14)$$

The primary decoder, β_1 , is given in terms of the reproduction codebook, $\mathcal{C}_1 = \{\mathbf{c}_{1,0}, \mathbf{c}_{1,1}, \dots, \mathbf{c}_{1,N_1-1}\}$ according to

$$\beta_1(j) = \mathbf{c}_{1,j}, \quad j \in \mathcal{J}_{N_1}. \quad (2.15)$$

The output of this decoder is denoted by $\mathbf{z} = \mathbf{c}_{1,j}$.

Note that the source vector, $\mathbf{x}$, is directly sent to the second-stage (or secondary) encoder. This is quite different from the noiseless-channel case, where (according to our formulation) the input to the second-stage encoder is the coding error, $\mathbf{x} - \mathbf{z}$, of the first stage. Since the channel here is noisy, the value of j and hence $\mathbf{z} = \mathbf{c}_{1,j}$ is not known at the transmitter. Later on, we will see that, for the optimum system with the squared-error distortion measure, what the secondary encoder actually does is encode the *expected* coding error between $\mathbf{x}$ and $\mathbf{Z}$. For now, let us just assume that the secondary encoder is the mapping $\alpha_2 : \mathbb{R}^k \times \mathcal{J}_{N_1} \mapsto \mathcal{J}_{N_2}$, described by

$$\alpha_2(\mathbf{x}, i) = u \quad \text{if } \mathbf{x} \in S_i^u, \quad i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}, \quad (2.16)$$

where $S_i^u \triangleq S_{1,i} \cap S_{2,u}$ and $\mathcal{P}_2 = \{S_{2,0}, S_{2,1}, \dots, S_{2,N_2-1}\}$ is another partition of $\mathbb{R}^k$. Note that for each $u \in \mathcal{J}_{N_2}$, the set $S_{2,u}$ need not be a connected set. Typically, $S_{2,u}$ is a union of N_1 separate sets (S_i^u), each of which is itself connected. Furthermore, the sets $S_{1,i}$ and S_i^u can be empty. For additional details on empty cells in channel-optimized VQ (COVQ) refer to [16]. The output of this encoder is transmitted over a second DMC described by $\gamma_2 : \mathcal{J}_{N_2} \mapsto \mathcal{J}_{N_2}$ and

$$Q_2(v|u) = \Pr\{\gamma_2(u) = v\}, \quad u, v \in \mathcal{J}_{N_2}. \quad (2.17)$$

Observe that, beside having $\mathbf{x}$ as an input to the secondary encoder, another difference between the formulation in Figure 2.1 and that of Section 2.2 is that the output of the primary and secondary encoders are transmitted over two different channels as opposed to being multiplexed together (as in (2.9)) and transmitted over a single channel. In this chapter, we assume that this single channel is a model of a binary symmetric channel (BSC) which is used $\log_2(N_1 N_2)$ times for each block of k source samples. Each channel-output bit depends only on the corresponding input bit. Thus, the combination of the multiplexer, the BSC and the demultiplexer can be modeled by two *independent* channels – the first of which accepts $\log_2 N_1$ bits and the second $\log_2 N_2$ bits. Hence, for the BSC case, the single-channel formulation of Section 2.2 and the two-channel formulation of Figure 2.1 are exactly the same. If the channel is not a BSC, the two formulations are still equivalent – except that the two channels may be dependent.

The secondary decoder, β_2 , depends on the output of both channels, j and v , and it is described by N_1 *different* codebooks, $\mathcal{C}_{2,j} = \{\mathbf{c}_{2,j,0}, \mathbf{c}_{2,j,1}, \dots, \mathbf{c}_{2,j,N_2-1}\}$, $j \in \mathcal{J}_{N_1}$, according to the formula

$$\beta_2(j, v) = \mathbf{c}_{2,j,v}, \quad j \in \mathcal{J}_{N_1}, v \in \mathcal{J}_{N_2}. \quad (2.18)$$

Here, we shall use $\mathcal{C}_2$ to refer to the collection $\{\mathcal{C}_{2,0}, \mathcal{C}_{2,1}, \dots, \mathcal{C}_{2,N_1-1}\}$. The output of β_2 is denoted by $\mathbf{w}$ and the reconstructed vector is $\tilde{\mathbf{x}} = \mathbf{z} + \mathbf{w}$.

The primary encoder and decoder are assumed to be given and are fixed. Our problem then is to minimize the average distortion $D = \frac{1}{k} E[d(\mathbf{X}, \tilde{\mathbf{X}})]$ by appropriate choices of $\mathcal{P}_2$ and $\mathcal{C}_2$.

2.3.2 Necessary Conditions

Assuming that the two channels are independent, the average distortion may be written as

$$D = \frac{1}{k} \sum_{i,u} \sum_{j,v} Q_1(j|i) Q_2(v|u) \int_{S_i^u} d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v}) f(\mathbf{x}) d\mathbf{x}, \quad (2.19)$$

where $f(\mathbf{x})$ is the k -fold probability density function (p.d.f) of the source.

Following the developments in [16] and [4], the second summation and the integral in (2.19) can be interchanged, yielding

$$D = \frac{1}{k} \sum_{i,u} \int_{S_i^u} \left\{ \sum_{j,v} Q_1(j|i) Q_2(v|u) d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v}) \right\} f(\mathbf{x}) d\mathbf{x}. \quad (2.20)$$

The term in braces is defined as the *modified* distortion measure [16], [4],

$$d'(\mathbf{x}; i, u) \triangleq \sum_{j,v} Q_1(j|i) Q_2(v|u) d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v}), \quad (2.21)$$

which can be interpreted as the expected distortion between $\mathbf{x}$ and $\tilde{\mathbf{X}}$ given that i and u are transmitted. With this modified distortion measure, it is clear that for a fixed codebook, the optimum partition, $\mathcal{P}_2^* = \{S_{2,0}^*, S_{2,1}^*, \dots, S_{2,N_2-1}^*\}$, must satisfy

$$S_{2,u}^* = \{\mathbf{x} : d'(\mathbf{x}; \alpha_1(\mathbf{x}), u) \leq d'(\mathbf{x}; \alpha_1(\mathbf{x}), u') \quad \forall u' \in \mathcal{J}_{N_2}\}, \quad u \in \mathcal{J}_{N_2}. \quad (2.22)$$

Similarly, for a fixed partition, the optimum codebook, $\mathcal{C}_2^* = \{\mathcal{C}_{2,0}^*, \mathcal{C}_{2,1}^*, \dots, \mathcal{C}_{2,N_1-1}^*\}$, where $\mathcal{C}_{2,j}^* = \{\mathbf{c}_{2,j,0}^*, \mathbf{c}_{2,j,1}^*, \dots, \mathbf{c}_{2,j,N_2-1}^*\}$ for each j in $\mathcal{J}_{N_1}$, can be derived using estimation theoretic arguments:

$$\begin{aligned} \mathbf{c}_{2,j,v}^* &= \arg \min_{\mathbf{w} \in \mathbb{R}^k} E[d(\mathbf{X}, \tilde{\mathbf{X}})|j, v] \\ &= \arg \min_{\mathbf{w} \in \mathbb{R}^k} E[d(\mathbf{X}, \mathbf{Z} + \mathbf{w})|j, v], \quad j \in \mathcal{J}_{N_1}, v \in \mathcal{J}_{N_2}. \end{aligned} \quad (2.23)$$

In the case where $d(\cdot, \cdot)$ is the squared-error distortion, i.e.,

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2, \quad (2.24)$$

the modified distortion measure of (2.21) can be simplified as follows. Let

$$\mathbf{y}_{1,i} \triangleq \sum_j Q_1(j|i) \mathbf{c}_{1,j}, \quad i \in \mathcal{J}_{N_1}, \quad (2.25)$$

$$\mathbf{y}_{2,i,u} \triangleq \sum_{j,v} Q_1(j|i) Q_2(v|u) \mathbf{c}_{2,j,v}, \quad i \in \mathcal{J}_{N_1}, \quad u \in \mathcal{J}_{N_2}, \quad (2.26)$$

and

$$\delta_{1,i} \triangleq \sum_j Q_1(j|i) \|\mathbf{c}_{1,j}\|^2, \quad i \in \mathcal{J}_{N_1}, \quad (2.27)$$

$$\delta_{2,i,u} \triangleq \sum_{j,v} Q_1(j|i) Q_2(v|u) \|\mathbf{c}_{2,j,v}\|^2, \quad i \in \mathcal{J}_{N_1}, \quad u \in \mathcal{J}_{N_2}. \quad (2.28)$$

Then the modified distortion measure can be expressed as

$$d'(\mathbf{x}; i, u) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i} + \mathbf{y}_{2,i,u} \rangle + \delta_{1,i} + 2\langle \mathbf{y}_{1,i}, \mathbf{y}_{2,i,u} \rangle + \delta_{2,i,u}, \quad (2.29)$$

or equivalently,

$$d'(\mathbf{x}; i, u) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i} \rangle + \delta_{1,i} + \delta_{2,i,u} - 2\langle \mathbf{x} - \mathbf{y}_{1,i}, \mathbf{y}_{2,i,u} \rangle. \quad (2.30)$$

The sum of the first three terms in (2.30) is defined as $d_1(\mathbf{x}; i)$ which can be viewed as the expected distortion of the first stage in encoding $\mathbf{x}$ given that $\mathbf{x}$ is in $S_{1,i}$, where the expectation is taken over all possible channel transitions. The sum of the last two terms is defined as $d_2(\mathbf{x}; i, u)$, where $-d_2(\mathbf{x}; i, u)$ can be regarded as the amount of distortion reduction associated with the second stage given that $\mathbf{x}$ is in S_i^u .

The codebook search of the secondary encoder (equation (2.22)) involves the determination of the value of u which minimizes $d'(\mathbf{x}; i, u)$ while keeping i fixed. Accordingly, it suffices to compute $d_2(\mathbf{x}; i, u)$ for each u in $\mathcal{J}_{N_2}$. The distortion d_2 corresponds exactly to the distortion measure used in the COVQ codebook search in [16], with $(\mathbf{x} - \mathbf{y}_{1,i})$ replacing $\mathbf{x}$. Hence, the secondary encoder acts just like the COVQ encoder with $(\mathbf{x} - \mathbf{y}_{1,i})$ as its input vector. Note that $\mathbf{y}_{1,i}$, as defined by (2.25), is the expected value of $\mathbf{Z}$ (the output of the primary decoder) given that

i is transmitted. Thus, $(\mathbf{x} - \mathbf{y}_{1,i})$, the vector which is encoded by the secondary encoder, is nothing but the *expected* coding error of the first stage.

Finally, for the squared-error distortion measure, it can be readily shown that the optimum codebook of (2.23) reduces to

$$\mathbf{c}_{2,j,v}^* = \frac{\sum_{i,u} Q_1(j|i)Q_2(v|u) \int_{S_1^u} (\mathbf{x} - \mathbf{c}_{1,j})f(\mathbf{x})d\mathbf{x}}{\sum_{i,u} Q_1(j|i)Q_2(v|u) \int_{S_1^u} f(\mathbf{x})d\mathbf{x}}, \quad j \in \mathcal{J}_{N_1}, v \in \mathcal{J}_{N_2}. \quad (2.31)$$

2.3.3 Design Algorithm

An algorithm for designing the channel-matched TSVQ (CM-TSVQ) for the squared-error distortion measure is presented next. This algorithm is a generalization of the iterative algorithms of [33] and [23]. A training sequence can be used for the design if each integral is replaced by the appropriate sum.

CM-TSVQ Design Algorithm:

(1) Given $\mathcal{P}_1$, $\mathcal{C}_1$ and $\epsilon_0 > 0$, set $m = 0$ and $D^{(m)} = \infty$ (m = iteration index).

Choose initial $\mathcal{P}_2^{(m)}$ and $\mathcal{C}_2^{(m)}$.

(2) Compute $\mathbf{y}_{1,i}$ and $\delta_{1,i} \forall i \in \mathcal{J}_{N_1}$ using $\mathcal{C}_1$ and equations (2.25) and (2.27).

(3) Set $m = m + 1$.

(4) Compute $\mathbf{y}_{2,i,u}$ and $\delta_{2,i,u} \forall i \in \mathcal{J}_{N_1}, \forall u \in \mathcal{J}_{N_2}$ using $\mathcal{C}_2^{(m-1)}$ and equations (2.26) and (2.28).

(5) $\forall i \in \mathcal{J}_{N_1}$:

$\{\forall \mathbf{x} \in S_{1,i} :$

$\{ \text{Set } u^* = \arg \min_{u \in \mathcal{J}_{N_2}} d'(\mathbf{x}; i, u) = \arg \min_{u \in \mathcal{J}_{N_2}} d_2(\mathbf{x}; i, u).$

Assign $\mathbf{x}$ to $S_{2,u^*}^* \}$

(6) Compute $D^{(m)}$ using $\mathcal{C}_2^{(m-1)}$, the optimum partition $\mathcal{P}_2^*$ obtained in step (5) and equation (2.20). Compute the optimum codebook $\mathcal{C}_2^*$ using $\mathcal{P}_2^*$ and equation (2.31). Set $\mathcal{C}_2^{(m)} = \mathcal{C}_2^*$ and $\mathcal{P}_2^{(m)} = \mathcal{P}_2^*$.

(7) If $(\frac{D^{(m-1)} - D^{(m)}}{D^{(m)}} > \epsilon_0)$ go to (3).

(8) Stop with $\mathcal{P}_2^{(m)}$ and $\mathcal{C}_2^{(m)}$ as the final partition and codebook.

The choice of $\mathcal{P}_1$ and $\mathcal{C}_1$ is particularly important in the CM-TSVQ design. A natural candidate for these is the optimum partition and codebook associated with the COVQ [16] designed for γ_1 and rate R_1 . We will see later that this choice is not necessarily the “best” one.

Designing CM-TSVQ for more than two stages is simple. Suppose that we are interested in designing a CM-TSVQ with three stages. This can be done by first designing one with only two stages and then viewing these two stages as a single VQ, i.e., as the first-stage VQ and then proceed as before. This procedure can be repeated until the desired number of stages is reached. We should note that this procedure does not guarantee that the overall encoder-decoder pair will be optimal - neither globally nor locally.

2.3.4 Channel-Matched MSVQ

As stated earlier, the above development is for the more general case of TSVQ. For the MSVQ, it suffices to add the constraint that all codebooks in the second stage are the same. This implies that the secondary decoder, β_2 , depends only on the output of the second channel, v , and is described by a *single* codebook, $\mathcal{C}_2 = \{\mathbf{c}_{2,0}, \mathbf{c}_{2,1}, \dots, \mathbf{c}_{2,N_2-1}\}$, according to the formula

$$\beta_2(v) = \mathbf{c}_{2,v}, \quad v \in \mathcal{J}_{N_2}. \quad (2.32)$$

In this case, the average distortion in equations (2.19) and (2.20) and the modified distortion measure in equation (2.21) are the same except that $\mathbf{c}_{2,j,v}$ is replaced by $\mathbf{c}_{2,v}$. The optimum partition is the same as in (2.22), while the optimum codebook

of (2.23) is replaced by

$$\mathbf{c}_{2,v}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^k} E[d(\mathbf{X}, \mathbf{Z} + \mathbf{w})|v], \quad v \in \mathcal{J}_{N_2}. \quad (2.33)$$

For the squared-error distortion measure, $\mathbf{y}_{2,i,u}$ and $\delta_{2,i,u}$ in equations (2.26), (2.28), (2.29) and (2.30) are replaced by

$$\mathbf{y}_{2,u} \triangleq \sum_v Q_2(v|u) \mathbf{c}_{2,v}, \quad u \in \mathcal{J}_{N_2}, \quad (2.34)$$

and

$$\delta_{2,u} \triangleq \sum_v Q_2(v|u) \|\mathbf{c}_{2,v}\|^2, \quad u \in \mathcal{J}_{N_2}. \quad (2.35)$$

Finally, the simplified optimum codebook of equation (2.31) can be replaced by

$$\mathbf{c}_{2,v}^* = \frac{\sum_{i,u} Q_2(v|u) \int_{S_i^u} (\mathbf{x} - \mathbf{y}_{1,i}) f(\mathbf{x}) d\mathbf{x}}{\sum_{i,u} Q_2(v|u) \int_{S_i^u} f(\mathbf{x}) d\mathbf{x}}, \quad v \in \mathcal{J}_{N_2}. \quad (2.36)$$

The design algorithm for the CM-MSVQ is identical to that of the CM-TSVQ, so we will not repeat it here.

2.4 Numerical Results

Performances of the CM-TSVQ and CM-MSVQ are presented next for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient $\rho = 0.9$. We consider the binary symmetric channel (BSC) with bit error rates (BER's) $\epsilon = 0.0, 0.005, 0.01, 0.05$ and 0.1 . In all cases, we assume that $N_s = N_1$ for all s . Numerical results for the CM-TSVQ and CM-MSVQ at the overall rate of 1 bit/sample are presented in Tables 2.1-2.4 for dimensions $k = 2, 4, 6, 8$, and 10. These results are given in

$$\text{SNR} = 10 \log_{10} \frac{\sigma^2}{D}, \quad \text{dB}, \quad (2.37)$$

where σ^2 is the source variance and D is the average per letter squared-error distortion. The results in these tables were obtained using 50,000 vectors outside

of the training sequence which itself consists of 50,000 vectors³. In the design, the first-stage VQ's are chosen to be the COVQ [16] designed for the given BER and the method of extension (Section 2.3.3) is used whenever there are more than two stages.

The procedure for choosing the initial codebook, $\mathcal{C}_2^{(0)}$, and partition, $\mathcal{P}_2^{(0)}$, in step (1) of the design algorithm (borrowed from [16]) is as follows. At each stage s , a codebook, designed for a noiseless channel, is obtained for the expected coding error of the first $(s - 1)$ stages. For the noiseless channel codebook we have used a simulated annealing algorithm, described in [13], to assign binary indices to the codevectors. This codebook is used as the initial codebook in the design for the channel with BER $\epsilon = 0.005$. The final codebook for $\epsilon = 0.005$ is used as the initial choice for the $\epsilon = 0.01$ design, and so on until the desired BER is obtained.

Also included in these tables are the results of the ordinary TSVQ [23] and MSVQ [27], i.e., TSVQ and MSVQ designed for the noiseless channel. The cases where $S = 1$ correspond to the full-searched COVQ or LBG-VQ.

Several interesting observations and conclusions can be made from these tables:

(i) When the channel is very noisy, i.e., $\epsilon = 0.05$ or 0.1 , the channel-matched schemes always outperform the ordinary schemes with the same set of parameters. The largest performance gain in these cases is 4.90 dB (Table 2.3, $k = 10, S = 10, \epsilon = 0.1$). When the channel is less noisy ($\epsilon = 0.005$ or 0.01), there are a few exceptions in which the ordinary schemes outperform the channel-matched schemes. For example, in Table 2.4, for $k = 4, S = 4$ and $\epsilon = 0.005$, the SNR of the CM-MSVQ is 8.50 dB while that of the ordinary MSVQ is 8.53 dB. In these cases the differences in performance are always less than or equal to 0.03 dB. We feel that

³We have found that for the TSVQ (Tables 2.1 and 2.3), the training data size was too small for the $k = 8$ and 10 cases. Thus, in these two cases, we have re-designed the codebook using larger training data consisting of 500,000 vectors for $k = 8$ and 1,000,000 vectors for $k = 10$. In both cases, the results are given for 50,000 test vectors. The same method was used to obtain the full-searched ($N_1 = 256$ and 1024) results given in Tables 2.2 and 2.4.

these exceptions are due to the sub-optimality of the design algorithm. (ii) Even though in a noiseless channel the ordinary MSVQ is inferior to the ordinary TSVQ (for fixed R, k and S), it is much more robust to channel errors than the TSVQ. This can be explained by observing that a single bit error during transmission can cause a decoding error at only a single stage of the MSVQ, while all other stages are not affected [38]; in the case of TSVQ, a single error in stage s will cause decoding errors in all stages $r \geq s$. For fixed R and k , the robustness of the ordinary MSVQ increases as S increases. (iii) For the memoryless Gaussian source with fixed R and N_1 , the performance of the channel-matched schemes does not always increase as the dimensionality increases. In fact, in many instances it decreases. On the contrary, for the Gauss-Markov source (with $\rho = 0.9$), the performance of the channel-matched schemes tends to increase monotonically with k (for fixed R and N_1). (iv) Surprisingly, for fixed R and k , the performance of the channel-matched schemes does not increase monotonically with N_1 . In many cases, the results for $N_1 = 2$ are better than for $N_1 = 4$. This again can be attributed to the structure of the CM-TSVQ and CM-MSVQ and the sub-optimality of the design algorithm. Similar observations for the MSVQ with $\epsilon = 0.0$ have been reported in [7] for the memoryless Gaussian source. (v) Finally, an interesting result in Table 2.4 for $k = 2$ and $\epsilon = 0.1$ indicates that the two-stage CM-MSVQ performs better than the COVQ. This implies that the COVQ design for this case⁴ resulted in a locally optimum solution rather than a global one.

2.4.1 Choice of BER for the First Stage

As mentioned earlier in Section 2.3.3, the choice of the first-stage VQ is important in the overall performance of the TSVQ or MSVQ. We have hinted that using a COVQ designed for the given BER in the first stage does not necessarily lead to

⁴Essentially the same result was reported in [16] for the COVQ with $R = 1, k = 2$ and $\epsilon = 0.1$.

the best results. In Table 2.5, we present several results on two-stage CM-TSVQ and CM-MSVQ in which the first-stage VQ is designed for a BER (denoted by ϵ_1) which is different from the actual BER. The second stage is always designed for the actual BER since this leads to a locally optimum system (given the first stage). It can be seen from the results in Table 2.5 that choosing ϵ_1 slightly larger than the actual BER can lead to some performance improvements. These improvements are more noticeable for very noisy channels.

In a two-stage TSVQ or MSVQ, the codevectors in the first stage are related to the source vector $\mathbf{x}$ while the second-stage codevectors are related to the error vector of the first stage. Thus, one may argue that the first stage contains more useful information than the second stage – even if the same number of bits are allocated to each stage. That is, given the first-stage codevector by itself, we have more information than if we are given only the second-stage codevector. Hence, intuitively, one would like to have more protection (against channel noise) in the first stage than in the second stage. A way of achieving this goal is to design the first stage for a slightly larger BER than the actual channel, as evidenced by the results in Table 2.5.

In the design of a CM-TSVQ or CM-MSVQ, it is left up to the designer to make a judgement as to the choice of ϵ_1 .

2.4.2 Larger Block Sizes for the CM-MSVQ

As noted earlier, one of the main advantages of the MSVQ is its low complexity – both in terms of computation and memory. Thus the MSVQ can operate at rates and/or block lengths larger than what is possible for VQ or TSVQ. Even though the TSVQ has low computational complexity, it still requires large memory. Furthermore, to design a TSVQ at large rates or block lengths requires a training sequence which can be prohibitively large. For the MSVQ, it suffices to have a

training sequence which is large enough for the codebook of each stage.

In this subsection, we will examine the performance of the CM-MSVQ as the *block length* becomes large. Table 2.6 shows the performances at $k = 12, 18$ and 24 for the Gauss-Markov source. We consider only the case where $N_1 = 64$ and we have made use of the findings of the previous subsection to choose the design BER of each stage. Our choices of the design BER's are provided in Table 2.7. We make no claim that these choices are optimum in any sense. However, we have tried several others and have found these to be among the best.

It is important to note that, for this source, the 12-dimensional CM-MSVQ is comparable to the 8-dimensional COVQ (Table 2.4; $N_1 = 256$) in SNR performance even though the CM-MSVQ has approximately half the computational complexity (128-vector codebook search compared to 256). We will have more discussion on the complexity issues in a later section.

2.4.3 Multiple Candidate Codebook Search

In the previous section, we stated that the modified distortion measure of equations (2.21) and (2.30) can be expressed as

$$d'(\mathbf{x}; i, u) = d_1(\mathbf{x}; i) + d_2(\mathbf{x}; i, u), \quad (2.38)$$

where

$$d_1(\mathbf{x}; i) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i} \rangle + \delta_{1,i}, \quad (2.39)$$

and

$$d_2(\mathbf{x}; i, u) = \begin{cases} \delta_{2,i,u} - 2\langle \mathbf{x} - \mathbf{y}_{1,i}, \mathbf{y}_{2,i,u} \rangle & \text{for the TSVQ,} \\ \delta_{2,u} - 2\langle \mathbf{x} - \mathbf{y}_{1,i}, \mathbf{y}_{2,u} \rangle & \text{for the MSVQ.} \end{cases} \quad (2.40)$$

The purpose of the primary encoder is to provide an index i to the secondary encoder (when the primary quantizer is a COVQ designed for the given BER, this value of i is that which minimizes $d_1(\mathbf{x}; i)$). For this i , the secondary encoder then chooses the u which minimizes $d'(\mathbf{x}; i, u)$, or equivalently, $d_2(\mathbf{x}; i, u)$. Note that

this procedure does not necessarily lead to an optimum index pair (i, u) , i.e., that which minimizes $d'(\mathbf{x}; i, u)$. This is a disadvantage associated with the successive approximation method.

For the two-stage MSVQ, if the encoder structure is removed while still maintaining the decoder structure, the resulting quantizer is equivalent to the *two-channel conjugate vector quantizer* (2C-CVQ) proposed by Moriya [38]. The 2C-CVQ encoder chooses the pair (i, u) which minimizes $d'(\mathbf{x}; i, u)$ by exhaustively searching the two codebooks. A generalization of the 2C-CVQ to more than two stages, called *exhaustive search residual quantizer* (ESRQ), was proposed by Barnes and Frost [7] for the noiseless channel. Locally optimum design algorithms for the 2C-CVQ and ESRQ are known [38], [7].

Even though the 2C-CVQ and ESRQ are nice, they are not always desirable because of the large amount of computation required on the encoder⁵. In this subsection we introduce a scheme whose performance and complexity is in between that of the two-stage CM-MSVQ and 2C-CVQ. This scheme, referred to as the *multiple candidate CM-MSVQ* (or, the *M*-algorithm), operates as follows. Instead of passing the single index i , the primary encoder passes a set consisting of the top M candidates, $\mathcal{I} = (i_1, i_2, \dots, i_M) \subset \mathcal{J}_{N_1}$, to the secondary encoder. The secondary encoder then chooses the pair (i_m, u) which minimizes $d'(\mathbf{x}; i_m, u)$, where $i_m \in \mathcal{I}$ and $u \in \mathcal{J}_{N_2}$. The multiple candidate CM-MSVQ is equivalent to the 2C-CVQ when $M = N_1$. Using the *M*-algorithm, the computational complexity of the secondary encoder increases by a factor approximately equal to M . The increase in memory is negligible.

The performances of the multiple candidate CM-MSVQ are tabulated in Table 2.8 for the cases $k = 6, S = 2$ and $k = 10, S = 2$ for the Gauss-Markov source

⁵For the 2C-CVQ, Moriya has proposed a method for reducing the computational complexity while increasing the memory requirement of the encoder. He does this by storing a matrix of inner products of every pair of codevectors in the two codebooks [38].

with $M = 1, 2, 4$ and 8 . The choices of the design BER are given in Table 2.7. We have obtained two sets of results. In the first set, the codebooks are designed using the same algorithm described earlier. In the second set (indicated by the * in Table 2.8), the primary codebook is kept the same, but the secondary codebook was re-optimized⁶ accordingly.

First, note that in both sets the performance increases monotonically with M . Second, unlike the findings in Section 2.4.1, the most noticeable gains are found for the relatively clean channels. In this case, the biggest gain was 0.31 dB. Third, we can see that re-optimizing the secondary codebook does provide some additional gain. Finally, looking at the results for the $M = 4^*$ case, we can see that by judiciously choosing the design BER of the first stage and by using the M -algorithm we have been able to narrow the gap between the two-stage CM-MSVQ and the COVQ (Table 2.4) by approximately 40-60%. For the $k = 10, S = 2$ and $M = 4^*$ case, with the above modifications, the encoder computational complexity increases by 250% and the encoder memory increases by 50%. However, with these modifications, the encoder computational complexity and memory are still six and ten times less than that of the COVQ, respectively.

We have also obtained the results of the multiple candidate CM-MSVQ when the block length is large ($k = 12, 18$ and 24). These results are tabulated in Table 2.9. Here we have not re-optimized the codebooks. The choices of design BER's are the same as in Table 2.7. In the case where there are more than two stages, at each stage $s > 1$, the set of the top M candidates, $\mathcal{I} = \{\mathbf{i}_1, \mathbf{i}_2, \dots, \mathbf{i}_M\}$, are retained from the previous $(s - 1)$ stages, where $\mathbf{i}_m = \{i_{m,1}, i_{m,2}, \dots, i_{m,s-1}\}$ is the path map of the m -th best candidate.

The results in Table 2.9 are the best that we have been able to obtain. We

⁶The algorithm for re-optimization is the same as before except that in step (5) u^* is replaced by $u^* = \arg \min_{u \in \mathcal{J}_{N_2}} \{ \min_{\mathbf{i}_m \in \mathcal{I}} d'(\mathbf{x}; \mathbf{i}_m, u) \}$.

compare them with the predictive trellis waveform coder (PTWC) of Ayanoglu and Gray [4]. In this comparison we limit ourselves to the 128-state (constraint length=8) trellis — the best results reported in [4]. For the cases considered, the PTWC beats the proposed CM-MSVQ. However, the encoding delay associated with the trellis search in PTWC (assuming 5-6 times the constraint length) is significantly larger than the delay of the CM-MSVQ. In terms of complexity, the PTWC requires 768 floating-point operations (FLOP's) per source sample as compared to $64(1 + (S - 1)M)$ FLOP's/sample for the CM-MSVQ (the first stage requires 64 FLOP's/sample while each of the remaining stages requires $64M$ FLOP's/sample). More will be said about the complexity of the CM-TSVQ and CM-MSVQ in Section 2.7.

2.5 Tandem Source-Channel Coding

In this section, we make comparisons between the CM-MSVQ scheme (in which, in a sense, the source and channel coding operations are implicitly combined), and a traditional tandem source-channel coding scheme with the same effective block length (hence, encoding delay).

The block diagram of the tandem source-channel code is depicted in Figure 2.2. Here, we assume that the source code is an LBG-VQ with block length k_s and rate $R_s = m/k_s$ bits/sample. The source encoder thus produces m bits for every k_s source samples. The channel code is assumed to be a (q, lm) *linear block* code. Hence, the channel encoder accepts lm information bits and produces q bits for transmission ($q \geq lm$). That is, the channel encoder takes l successive m -bit codewords generated by the source encoder and produces a q -bit codeword. The overall rate of the tandem scheme is $R = q/(lk_s)$ transmitted bits per source sample and the effective block length (or delay) is $k = lk_s$ source samples. The reason we

have considered the LBG-VQ and the linear block code is because they are both among the most well-known and conceptually simple block-structured source and channel codes, respectively.

To be in harmony with the more conventional notation used in the channel coding literature, from now on we will use (n', k') to denote (q, lm) . The channel is assumed to be a BSC. For an (n', k') linear block code operating over a BSC with BER ϵ , the probability of correct decoding is upper bounded by

$$P_c = \sum_{i=0}^t \binom{n'}{i} \epsilon^i (1 - \epsilon)^{n'-i} + A_t \epsilon^{t+1} (1 - \epsilon)^{n'-t-1}, \quad (2.41)$$

where t is the greatest integer such that

$$A_t \triangleq 2^{n'-k'} - \sum_{i=0}^t \binom{n'}{i} \geq 0. \quad (2.42)$$

The above is known as the sphere-packing bound and is achievable if and only if the (n', k') code is quasi-perfect [39].

To analyze the performance of the tandem scheme, we first assume that the channel code is quasi-perfect. Secondly, we model the channel encoder, the BSC and the channel decoder as an *equivalent BSC* (represented by the dashed box in Figure 2.2). The assumption that the equivalent channel is a BSC is obviously incorrect. However, this assumption leads to a very simple analysis of the tandem scheme. The only problem is to specify the BER of the equivalent BSC. Here, we assume that, in the equivalent channel, the probability that a block of k' consecutive bits are received without error is equal to P_c . Thus the BER of this channel can be computed as

$$\epsilon' = 1 - P_c^{1/k'}. \quad (2.43)$$

With this equivalent model, the analysis for the tandem source-channel coding scheme reduces to that of the LBG-VQ operating over a BSC with BER ϵ' [29], [16].

Using the equivalent model, we have evaluated the performance of the tandem scheme for $k_s = 6$ and $l = 2, 3$ and 4 . These results (for the Gauss-Markov source with $\rho = 0.9$) are plotted in Figures 2.3, 2.4 and 2.5, respectively. For the LBG-VQ of the tandem scheme, a simulated annealing algorithm [13] was used for index assignment. In each plot, we have chosen $m = 3, 4, 5$ and 6 (corresponding to $R_s = 0.5, 0.67, 0.83$ and 1.0). In all cases, we have chosen $q = lk_s$ so that the overall rate is always 1 bit/sample. We have also plotted on these graphs the optimum performance theoretically attainable (OPTA) obtained by equating the rate-distortion function to the channel capacity [8] and the performances of the CM-MSVQ ($M = 1$) and the multiple candidate CM-MSVQ ($M = 8$) with the same k as that of the tandem scheme (thus, similar encoding delays). The CM-MSVQ results are taken directly from Table 2.9. In these graphs, the performance of the multiple candidate CM-MSVQ with $M = 8$ is always better than the tandem scheme. Looking at the results of the tandem scheme, it is clear that when the channel is relatively noise-free, all of the available bits should be allocated to source coding. As the channel becomes noisier, more and more bits should be allocated to channel coding (in the form of redundancy). In any case, even with an optimum allocation of bits between the source and channel codes, the tandem scheme is always inferior to the CM-MSVQ for the cases considered.

Note that the CM-MSVQ outperforms the tandem scheme even when $\epsilon = 0.00$. The reason for this is that the source code of the tandem scheme has a block length of $k_s = 6$ only, while the CM-MSVQ operates on blocks of lengths $k = lk_s = 12, 18$ or 24 . The argument here is that the CM-MSVQ is a low-complexity scheme which can operate at larger block sizes than the LBG-VQ. Since the source has memory, the CM-MSVQ with large block lengths beats the LBG-VQ with a small block length ($k_s = 6$). An important point to make is that the CM-MSVQ is a suboptimal combined source-channel coding scheme; whereas the tandem scheme

consists of a source code which, by itself, is optimum for its block length and a channel code which, by itself, is optimum (according to the equivalent model) for its block length. For this reason, we believe that, for the source and block sizes considered, the combined source-channel coding approach may be more desirable than the traditional tandem approach.

Before closing this section, we should make a note that our analysis of the tandem scheme was somewhat optimistic. First of all, a quasi-perfect code does not exist for all values of (n', k') [39]. In fact, for all the cases which we have considered, only when $(n', k') = (24, 12)$, which corresponds to $R_s = 0.5$ in Figure 2.5, is there a known quasi-perfect code – the extended Golay code [39]. In all other cases, the probability of correct decoding is smaller than P_c [39]. Secondly, even if a quasi-perfect code does exist for (n', k') , there is still the question about the validity of the equivalent BSC model. It is clear that the equivalent model is incorrect; however, it is unclear how the inaccuracy in the model affects the performance analysis. We conjecture that our analysis of the tandem scheme over-estimates its actual performance.

To support this conjecture, we have simulated the tandem scheme with $R_s = 0.5$ using the $(24, 12)$ extended Golay code for 300,000 test samples of the Gauss-Markov source. The simulation was performed twenty times using twenty different realizations of a noise process. In Figure 2.6, the simulation results are compared with the analytical results using the equivalent BSC model. For the simulated results, the average SNR's are shown; the error bars indicate the minimum and maximum SNR's in the twenty trials. From this figure, we can see that the two curves coincide when the channel is relatively clean, but they differ dramatically when the channel is noisy. At $\log_{10} \epsilon = -1$, they differ by more than 3 dB.

Since we do not know the optimum channel code for other values of R_s , we have not simulated those cases. However, to make fair comparisons, we have simulated

the multiple candidate CM-MSVQ and the results are also plotted in Figure 2.6. Here we have intentionally left out the error bars for the simulated CM-MSVQ scheme since these bars are so small that they only clutter up the figure. However, we note that the differences between the maximum and minimum SNR's are always less than or equal to 0.17 dB for the simulated CM-MSVQ scheme. The largest difference in SNR for the simulated tandem scheme was 0.28 dB. For the CM-MSVQ, the simulated results coincide almost exactly with the analytical results, as expected.

In summary, the equivalent model for the tandem source-channel coding scheme is overly optimistic in two senses. First, it assumes that an (n', k') quasi-perfect code exists and secondly it assumes that the actual equivalent channel is a BSC. In light of this discussion, we have every reason to believe that the actual gap between the CM-MSVQ and the tandem scheme is even larger than what is suggested by Figures 2.3-2.5.

2.6 Channel Mismatch

Up to this point, we have been working under the assumption that the channel and hence the channel BER is given. In real-life situations, the channel BER is not known *a priori* to the designer or, in some situations, it is time-varying. In this section, we will examine the performance of CM-TSVQ and CM-MSVQ under mismatch, i.e., when they are designed for a BER ϵ_d , but applied to a channel whose BER is actually ϵ_a .

The mismatch results are plotted in Figures 2.7 and 2.8 for $k = 10$ and $S = 2$. In these figures, the design BER is the same for both the first and second stages. It can be seen that the CM-TSVQ and CM-MSVQ are fairly robust to channel mismatch, though the CM-MSVQ exhibits more robustness than the CM-TSVQ.

An interesting phenomenon can be observed in these figures. For the CM-TSVQ results in Figure 2.7, at $\epsilon_a = 0.005$ ($\log_{10} \epsilon_a = -2.3$), the performance with $\epsilon_d = 0.010$ is better than that with $\epsilon_d = 0.005$. The same thing occurs for the CM-MSVQ in Figure 2.8 at $\epsilon_a = 0.005, 0.010$ and 0.050 . This can be explained by the fact that we have not chosen the design BER for the first stage correctly. In fact, after carefully examining the numbers, we found that if we had chosen it correctly (according to Table 2.7) then the best result would have occurred when $\epsilon_d = \epsilon_a$.

Next, we evaluate the performance of the CM-MSVQ (with two stages) when the first stage is designed for $\epsilon_{d,1}$ and the second stage for $\epsilon_d = \epsilon_{d,2}$. These results are given in Figure 2.9 for $k = 10$ and $S = 2$. As expected, when $\epsilon_{d,1}$ is slightly larger than $\epsilon_{d,2}$ the performance improves at $\epsilon_a = \epsilon_d$ but degrades at $\epsilon_a = 0.0$. We have also studied the performance of the M -algorithm under channel mismatch conditions and these results are plotted in Figure 2.10 again for $k = 10$ and $S = 2$. From this figure, we may conclude that the M -algorithm does not degrade the performance of the CM-MSVQ under mismatch conditions. In fact, in most cases the performance improves.

Finally, we have examined the performance of the CM-MSVQ scheme at a large block size ($k = 24$) and compared this with the tandem scheme. Figure 2.11 shows the results for the CM-MSVQ matched to $\epsilon_d = 0.050$ but tested for various values of ϵ_a . Here, the first stage of the CM-MSVQ was designed for a BER of 0.200, the second stage for 0.100 and the third and fourth stages for 0.050 (see Table 2.7). It is clear from this figure that the CM-MSVQ is a robust scheme. For a wide range of BER's ($\epsilon_a \gtrsim 0.0016$), this single CM-MSVQ scheme outperforms all tandem schemes considered. For the other BER's ($\epsilon_a \lesssim 0.0016$), its performance is only slightly inferior to the tandem scheme with $R_s = 1.0$. Furthermore, the reader should keep in mind the earlier discussion that the analysis of the tandem scheme over-estimates its actual performance. Considering this fact, the significance of

the CM-MSVQ scheme becomes more evident.

2.7 Complexity Issues

Let us now analyze the complexity of the CM-MSVQ and CM-TSVQ and compare these with the full-searched COVQ. We will consider the two-stage CM-MSVQ and CM-TSVQ. Let us assume that the first stage is a COVQ whose design BER is different from that of the second stage. Hence, the first-stage encoder requires kN_1 words for storing $\{\mathbf{y}_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$ and N_1 words for storing $\{\delta_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$, where $\{\delta_{1,i}^{(1)}, \mathbf{y}_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$ are computed according to equations (2.25) and (2.27) with ϵ_1 used in the channel transition matrix. It also requires N_1 FLOP's per source sample for encoding. The secondary codebook requires $(k+1)N_1$ words to store $\{\delta_{1,i}^{(2)}, \mathbf{y}_{1,i}^{(2)} : i \in \mathcal{J}_{N_1}\}$ (these are different from those in the first stage since they are computed using ϵ_2). In addition, it requires $(k+1)N_1N_2$ words to store $\{\delta_{2,i,u}^{(2)}, \mathbf{y}_{2,i,u}^{(2)} : i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}\}$ for the CM-TSVQ or $(k+1)N_2$ words to store $\{\delta_{2,u}^{(2)}, \mathbf{y}_{2,u}^{(2)} : u \in \mathcal{J}_{N_2}\}$ for the CM-MSVQ. The computational complexity of the second stage is N_2 FLOP's per sample. For the decoder, the CM-MSVQ requires $k(N_1 + N_2)$ words for storing $\{\mathbf{c}_{1,i} : i \in \mathcal{J}_{N_1}\}$ and $\{\mathbf{c}_{2,u} : u \in \mathcal{J}_{N_2}\}$ while the CM-TSVQ requires kN_1N_2 words for storing $\{\hat{\mathbf{c}}_{i,u} = \mathbf{c}_{1,i} + \mathbf{c}_{2,i,u} : i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}\}$. Table 2.10 summarizes the overall complexity of the two-stage CM-MSVQ and CM-TSVQ. We have also included the complexity of the COVQ with $N = N_1N_2$ codevectors. The extension of this analysis to more than two stages is obvious. It should be noted that the encoder memory and computational complexity in Table 2.10 are upper bounds on the actual complexity. It has been stated in [16] that for the COVQ when ϵ_d is large the number of non-empty encoding regions can be significantly less than the number of codevectors. The same phenomenon occurs for CM-MSVQ and CM-TSVQ. In such cases, the encoder memory and

computational requirement can be reduced accordingly. Furthermore, if the design BER's are the same for the first and second stages, then the encoder memory can be reduced as well. Note that the decoder memory and the encoder computational requirement of the CM-TSVQ and CM-MSVQ are the same as those of the ordinary TSVQ and MSVQ. The encoder memory of the channel matched schemes is slightly larger than the ordinary schemes.

Finally, examining the complexity of the tandem scheme, we see that the comparisons in Figures 2.3-2.6 are not entirely fair. In some cases, the CM-MSVQ has significantly higher complexity than the tandem scheme. To make a fair comparison, we plot in Figure 2.12 the performances of a CM-MSVQ and a tandem scheme whose complexities are approximately equal (see Table 2.11). Again, we have left out the error bars for the simulated CM-MSVQ scheme because they clutter up the graph. Here, the largest differences between the minimum and maximum SNR's for the simulated CM-MSVQ scheme and simulated tandem scheme are 0.16 and 0.28 dB, respectively. For the CM-MSVQ, both stages were designed for the same BER, $\epsilon_d = 0.050$. Also, the complexity of the channel encoder/decoder of the tandem scheme is assumed to be negligibly small and is not included in Table 2.11. Note that the CM-MSVQ beats the tandem scheme in most cases even though its delay is less than half that of the tandem scheme.

2.8 Summary and Conclusions

In this chapter, we have presented a unified approach for the design of TSVQ and MSVQ in the presence of channel noise. Numerical results indicate that this combined source-channel coding scheme yields substantial improvements over the ordinary TSVQ and MSVQ when the channel is very noisy. For the two-stage TSVQ and MSVQ, we have shown that designing the first stage for a channel

noisier than the actual channel, in general, leads to performance gains. We have also demonstrated that additional gains can be obtained using the M -algorithm. Our comparisons of the CM-MSVQ scheme with a more traditional tandem source-channel coding scheme (utilizing the LBG-VQ as the source code and a quasi-perfect block code as the channel code) have indicated superior performance of the CM-MSVQ in all cases considered. It is also shown that CM-MSVQ is robust against channel mismatch. These findings, while important from a theoretical point of view, may pave the way for the development of practical coding schemes for highly noisy channels such as those encountered in mobile radio situations.

k	N_1	S	CM-TSVQ					Ordinary TSVQ				
			BER					BER				
			0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	4.35	4.21	4.07	3.11	2.23	4.35	4.21	4.06	3.06	2.06
	4	1	4.36	4.21	4.07	3.11	2.23	4.36	4.19	4.05	3.04	2.04
4	2	4	4.37	4.11	3.89	2.80	1.89	4.37	4.11	3.87	2.35	1.12
	4	2	4.33	4.09	3.87	2.86	2.19	4.33	4.06	3.81	2.32	1.09
	16	1	4.63	4.41	4.21	3.14	2.26	4.63	4.40	4.19	2.81	1.61
6	2	6	4.38	4.06	3.85	2.80	2.27	4.38	4.06	3.77	2.06	0.78
	4	3	4.29	3.98	3.75	2.88	2.18	4.29	4.00	3.74	2.14	0.91
	8	2	4.49	4.15	3.89	2.92	2.18	4.49	4.15	3.84	2.06	0.75
	64	1	4.77	4.50	4.27	3.15	2.27	4.77	4.47	4.20	2.55	1.25
8	2	8	4.39	3.99	3.81	2.84	2.27	4.39	3.99	3.63	1.68	0.36
	4	4	4.36	3.98	3.71	2.92	2.13	4.36	3.97	3.61	1.70	0.39
	16	2	4.56	4.14	3.85	2.87	2.14	4.56	4.11	3.72	1.64	0.25
	256	1	4.88	4.52	4.27	3.15	2.27	4.88	4.47	4.10	2.10	0.70
10	2	10	4.16	3.75	3.67	2.78	2.26	4.16	3.71	3.32	1.27	-0.03
	4	5	4.12	3.89	3.56	2.83	2.13	4.12	3.70	3.32	1.34	0.03
	32	2	4.64	4.14	3.75	2.80	2.06	4.64	4.11	3.65	1.37	-0.04
	1024	1	4.97	4.55	4.27	3.14	2.26	4.97	4.30	3.74	1.12	-0.35

Table 2.1: SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Memoryless Gaussian Source; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

k	N_1	S	CM-MSVQ					Ordinary MSVQ				
			BER					BER				
			0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	4.34	4.20	4.05	3.10	2.23	4.34	4.21	4.06	3.06	2.06
	4	1	4.36	4.21	4.07	3.11	2.23	4.36	4.19	4.05	3.04	2.04
4	2	4	4.38	4.23	4.08	3.13	2.26	4.38	4.23	4.09	3.08	2.08
	4	2	4.09	3.92	3.78	2.90	2.18	4.09	3.92	3.77	2.69	1.67
	16	1	4.63	4.41	4.21	3.14	2.26	4.63	4.40	4.19	2.81	1.61
6	2	6	4.39	4.24	4.08	3.14	2.27	4.39	4.24	4.09	3.09	2.08
	4	3	4.02	3.86	3.69	2.81	2.16	4.02	3.85	3.69	2.58	1.55
	8	2	4.33	4.13	3.94	2.84	2.12	4.33	4.11	3.90	2.57	1.40
	64	1	4.77	4.50	4.27	3.15	2.27	4.77	4.47	4.20	2.55	1.25
8	2	8	4.39	4.24	4.10	3.14	2.27	4.39	4.24	4.09	3.08	2.08
	4	4	4.10	3.93	3.77	2.85	2.16	4.10	3.94	3.78	2.69	1.66
	16	2	4.41	4.12	3.88	2.80	2.10	4.41	4.13	3.86	2.24	0.96
	256	1	4.88	4.52	4.27	3.15	2.27	4.88	4.47	4.10	2.10	0.70
10	2	10	4.38	4.24	4.08	3.14	2.26	4.38	4.23	4.09	3.08	2.08
	4	5	4.01	3.87	3.71	2.80	2.12	4.01	3.85	3.69	2.60	1.57
	32	2	4.45	4.16	3.91	2.79	2.10	4.45	4.14	3.86	2.17	0.87
	1024	1	4.97	4.55	4.27	3.14	2.26	4.97	4.30	3.74	1.12	-0.35

Table 2.2: SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Memoryless Gaussian Source; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

k	N_1	S	CM-TSVQ					Ordinary TSVQ				
			BER					BER				
			0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	7.84	7.29	6.81	4.36	2.77	7.84	7.29	6.70	4.14	2.21
	4	1	7.83	7.29	6.81	4.36	2.77	7.83	7.28	6.80	4.14	2.20
4	2	4	9.98	9.03	8.24	5.63	4.49	9.98	8.89	8.02	4.15	1.78
	4	2	10.16	9.10	8.29	6.08	4.61	10.16	9.06	8.19	4.30	1.93
	16	1	10.18	9.18	8.45	6.24	4.64	10.18	9.11	8.25	4.43	2.08
6	2	6	10.67	9.43	8.57	6.71	4.90	10.67	9.32	8.30	4.06	1.61
	4	3	10.80	9.53	8.50	6.85	5.06	10.80	9.43	8.39	4.14	1.70
	8	2	10.89	9.57	8.94	6.72	5.17	10.89	9.48	8.42	4.13	1.72
	64	1	11.00	9.97	9.30	6.98	5.32	11.00	9.74	8.77	4.65	2.24
8	2	8	10.96	9.63	9.14	7.11	5.66	10.96	9.37	8.22	3.72	1.25
	4	4	11.14	9.65	9.06	7.12	5.61	11.14	9.54	8.38	3.90	1.45
	16	2	11.29	9.90	9.37	7.30	5.74	11.29	9.70	8.55	4.08	1.66
	256	1	11.39	10.31	9.74	7.51	5.83	11.39	9.95	8.87	4.55	2.12
10	2	10	11.14	9.54	9.32	7.27	5.89	11.14	9.34	8.08	3.43	0.99
	4	5	11.31	9.69	9.22	7.32	5.89	11.31	9.52	8.28	3.65	1.22
	32	2	11.51	9.86	9.56	7.53	5.98	11.51	9.75	8.51	3.91	1.48
	1024	1	11.64	10.35	9.94	7.79	6.10	11.64	9.80	8.53	3.87	1.46

Table 2.3: SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

k	N_1	S	CM-MSVQ					Ordinary MSVQ				
			BER					BER				
			0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	7.47	7.05	6.67	4.64	3.24	7.47	7.07	6.70	4.51	2.78
	4	1	7.83	7.29	6.81	4.36	2.77	7.83	7.28	6.80	4.14	2.20
4	2	4	9.14	8.50	7.96	5.37	3.82	9.14	8.53	7.99	5.14	3.11
	4	2	9.71	8.84	8.14	5.11	3.52	9.71	8.85	8.14	4.70	2.48
	16	1	10.18	9.18	8.45	6.24	4.64	10.18	9.11	8.25	4.43	2.08
6	2	6	9.50	8.83	8.27	5.67	4.12	9.50	8.83	8.25	5.25	3.16
	4	3	10.14	9.19	8.44	5.47	3.89	10.14	9.20	8.42	4.82	2.54
	8	2	10.48	9.40	8.61	6.18	4.51	10.48	9.40	8.54	4.71	2.37
	64	1	11.00	9.97	9.30	6.98	5.32	11.00	9.74	8.77	4.65	2.24
8	2	8	10.00	9.23	8.61	5.86	4.26	10.00	9.24	8.60	5.40	3.23
	4	4	10.37	9.38	8.64	5.68	4.09	10.37	9.39	8.59	4.92	2.61
	16	2	11.00	9.88	9.08	6.31	5.11	11.00	9.78	8.83	4.77	2.36
	256	1	11.39	10.31	9.74	7.51	5.83	11.39	9.95	8.87	4.55	2.12
10	2	10	10.10	9.31	8.72	5.95	4.35	10.10	9.33	8.68	5.43	3.24
	4	5	10.55	9.56	8.75	5.76	4.21	10.55	9.52	8.69	4.93	2.61
	32	2	11.22	10.05	9.31	6.96	5.34	11.22	9.88	8.87	4.67	2.25
	1024	1	11.64	10.35	9.94	7.79	6.10	11.64	9.80	8.53	3.87	1.46

Table 2.4: SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

ϵ_1	CM-TSVQ					CM-MSVQ				
	Actual BER					Actual BER				
	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
$k = 6; N_1 = 8; S = 2$										
0.000	10.89	9.55	8.92	6.68	5.14	10.48	9.41	8.56	4.84	2.61
0.005	10.91	9.57	8.89	6.69	5.17	10.43	9.40	8.58	4.96	2.76
0.010	10.90	9.59	8.94	6.74	5.13	10.39	9.41	8.61	5.07	2.88
0.050	10.63	9.55	9.01	6.72	5.04	9.95	9.35	8.84	6.18	4.27
0.100	10.34	9.59	9.05	6.90	5.17	9.79	9.23	8.75	6.32	4.51
0.200	9.67	9.11	8.77	6.83	5.21	8.88	8.50	8.17	6.39	4.91

Table 2.5: SNR (in dB) Performance Results of the CM-TSVQ and CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of ϵ_1 ; ϵ_1 = Design BER of the First Stage; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			CM-MSVQ				
			Actual BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10
12	64	2	11.39	10.21	9.71	7.41	5.85
18	64	3	11.46	10.37	9.69	7.35	5.78
24	64	4	11.47	10.40	9.80	7.36	5.81

Table 2.6: SNR (in dB) Performance Results of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k ; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			Actual BER				
k	S	ϵ_s	0.00	0.005	0.01	0.05	0.10
6	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
10	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
12	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
18	3	ϵ_1	0.000	0.050	0.100	0.200	0.300
		ϵ_2	0.000	0.010	0.050	0.100	0.200
		ϵ_3	0.000	0.005	0.010	0.050	0.100
24	4	ϵ_1	0.000	0.050	0.100	0.200	0.300
		ϵ_2	0.000	0.010	0.050	0.100	0.200
		ϵ_3	0.000	0.005	0.010	0.050	0.100
		ϵ_4	0.000	0.005	0.010	0.050	0.100

Table 2.7: Choices of Design BER's for the Results in Tables 2.6, 2.8 and 2.9; k = Dimension; S = Number of stages; ϵ_s = Design BER for Stage s .

Multiple Candidate CM-MSVQ					
Actual BER					
M	0.00	0.005	0.01	0.05	0.10
$k = 6; N_1 = 8; S = 2$					
1	10.48	9.41	8.84	6.39	4.91
2	10.58	9.48	8.94	6.48	4.93
4	10.59	9.49	8.97	6.49	4.94
8	10.79	9.49	8.98	6.50	4.94
2*	10.69	9.58	8.93	6.49	4.94
4*	10.82	9.66	9.07	6.51	4.94
8*	10.82	9.66	9.07	6.52	4.95
$k = 10; N_1 = 32; S = 2$					
1	11.22	10.11	9.56	7.27	5.64
2	11.35	10.21	9.67	7.35	5.67
4	11.38	10.25	9.70	7.37	5.67
8	11.38	10.26	9.71	7.38	5.67
2*	11.38	10.22	9.69	7.35	5.67
4*	11.44	10.28	9.73	7.39	5.69
8*	11.46	10.30	9.75	7.39	5.69

Table 2.8: SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of M ; M = Number of Candidates; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages; * = Re-optimized Secondary Codebook.

Multiple Candidate CM-MSVQ					
Actual BER					
M	0.00	0.005	0.01	0.05	0.10
$k = 12; N_1 = 64; S = 2$					
1	11.39	10.21	9.71	7.41	5.85
2	11.53	10.32	9.83	7.51	5.88
4	11.58	10.36	9.87	7.54	5.89
8	11.59	10.37	9.88	7.55	5.89
$k = 18; N_1 = 64; S = 3$					
1	11.46	10.37	9.69	7.35	5.78
2	11.68	10.56	9.90	7.49	5.87
4	11.80	10.66	10.01	7.55	5.91
8	11.84	10.70	10.06	7.57	5.92
$k = 24; N_1 = 64; S = 4$					
1	11.47	10.40	9.80	7.36	5.81
2	11.74	10.61	10.01	7.49	5.91
4	11.90	10.73	10.13	7.55	5.96
8	11.97	10.79	10.18	7.58	5.98
PTWC [4] (128 states)					
-	12.58	11.14	10.38	N/A	N/A

Table 2.9: SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k and Different Values of M ; M = Number of Candidates; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

	CM-MSVQ	CM-TSVQ	COVQ
Encoder Memory (32-bit words)	$(k+1)(2N_1 + N_2)$	$(k+1)(2N_1 + N_1N_2)$	$(k+1)(N_1N_2)$
Decoder Memory (32-bit words)	$k(N_1 + N_2)$	kN_1N_2	kN_1N_2
Encoder Computational Requirement (FLOP's/sample)	$N_1 + N_2$	$N_1 + N_2$	N_1N_2

Table 2.10: Complexity of Two-Stage CM-MSVQ and CM-TSVQ and COVQ; k = Dimension; N_i = Number of Codevectors at Stage i , $i = 1, 2$.

	CM-MSVQ	Tandem Scheme
Encoder Memory (32-bit words)	704	832
Decoder Memory (32-bit words)	640	768
Encoder Computational Requirement (FLOP's/sample)	64	64
Delay (Number of samples)	10	24

Table 2.11: Complexity and Delay of the CM-MSVQ and the Tandem Scheme in Figure 2.12.

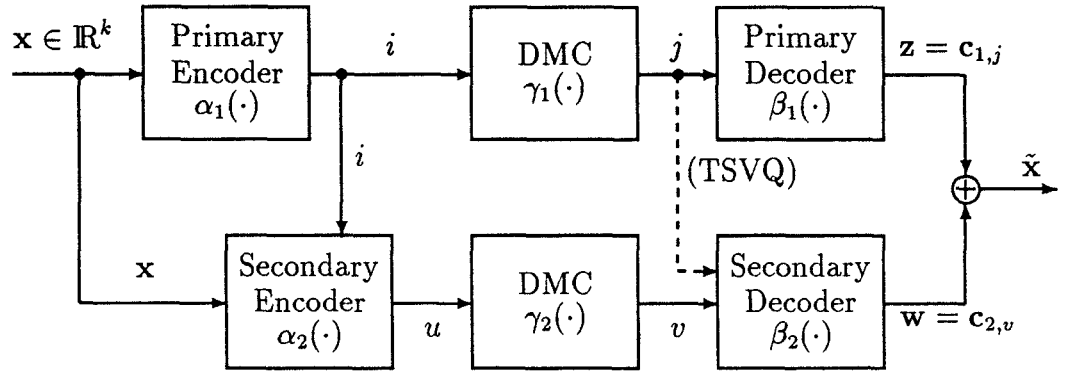


Figure 2.1: Block Diagram of the TSVQ/MSVQ Scheme Over Noisy Channels.

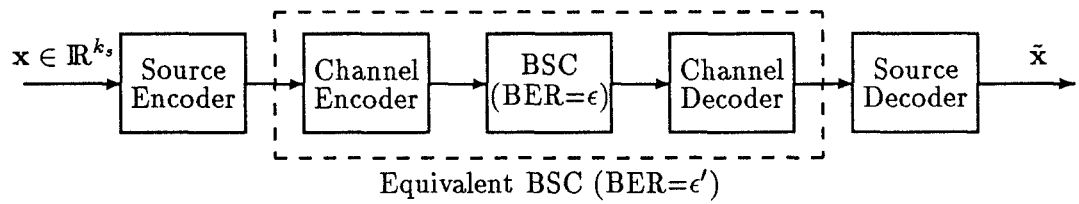


Figure 2.2: Block Diagram of a Tandem Source-Channel Coding Scheme Over a BSC and the Equivalent Channel Model.

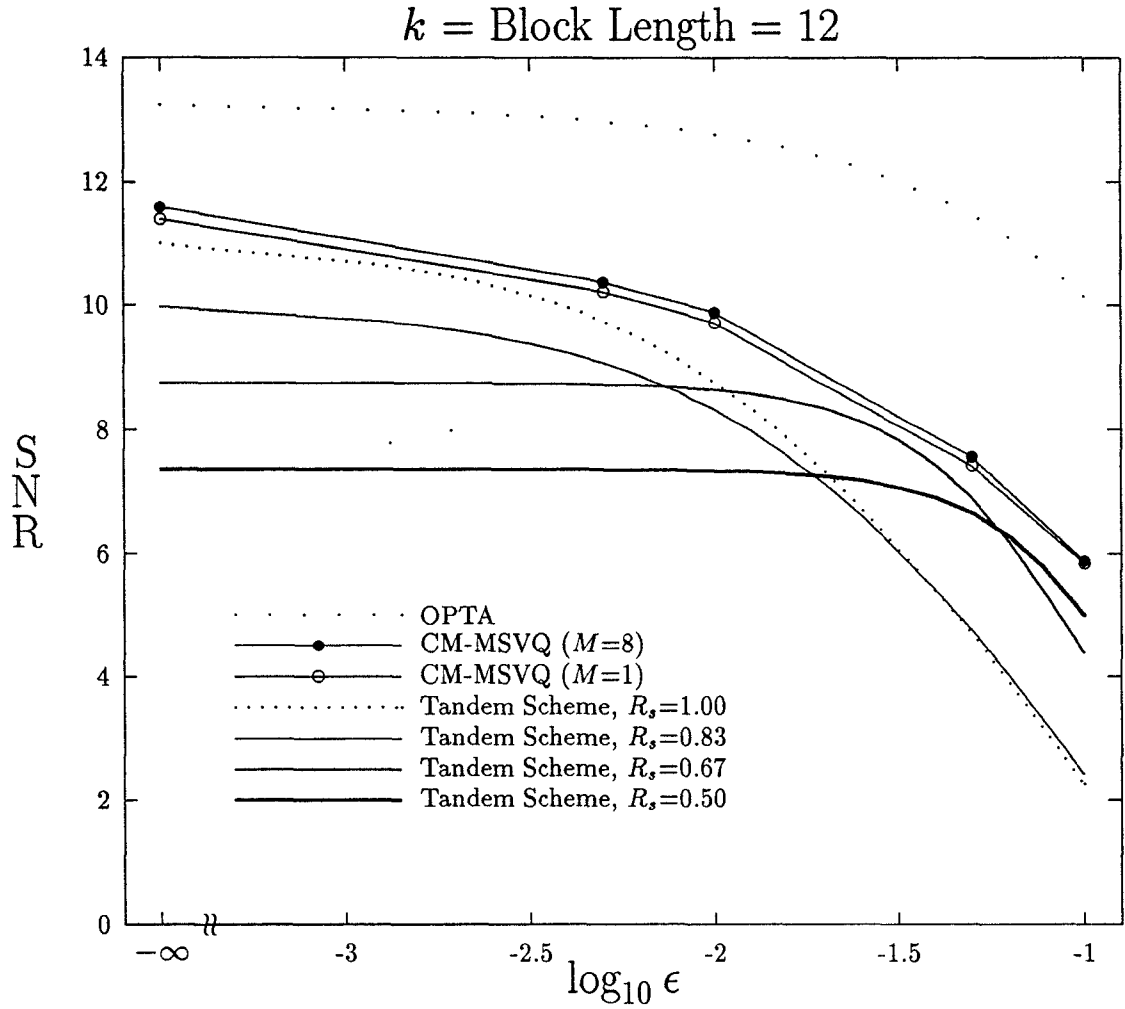


Figure 2.3: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 2; k = Effective Block Length of Both Schemes = 12.

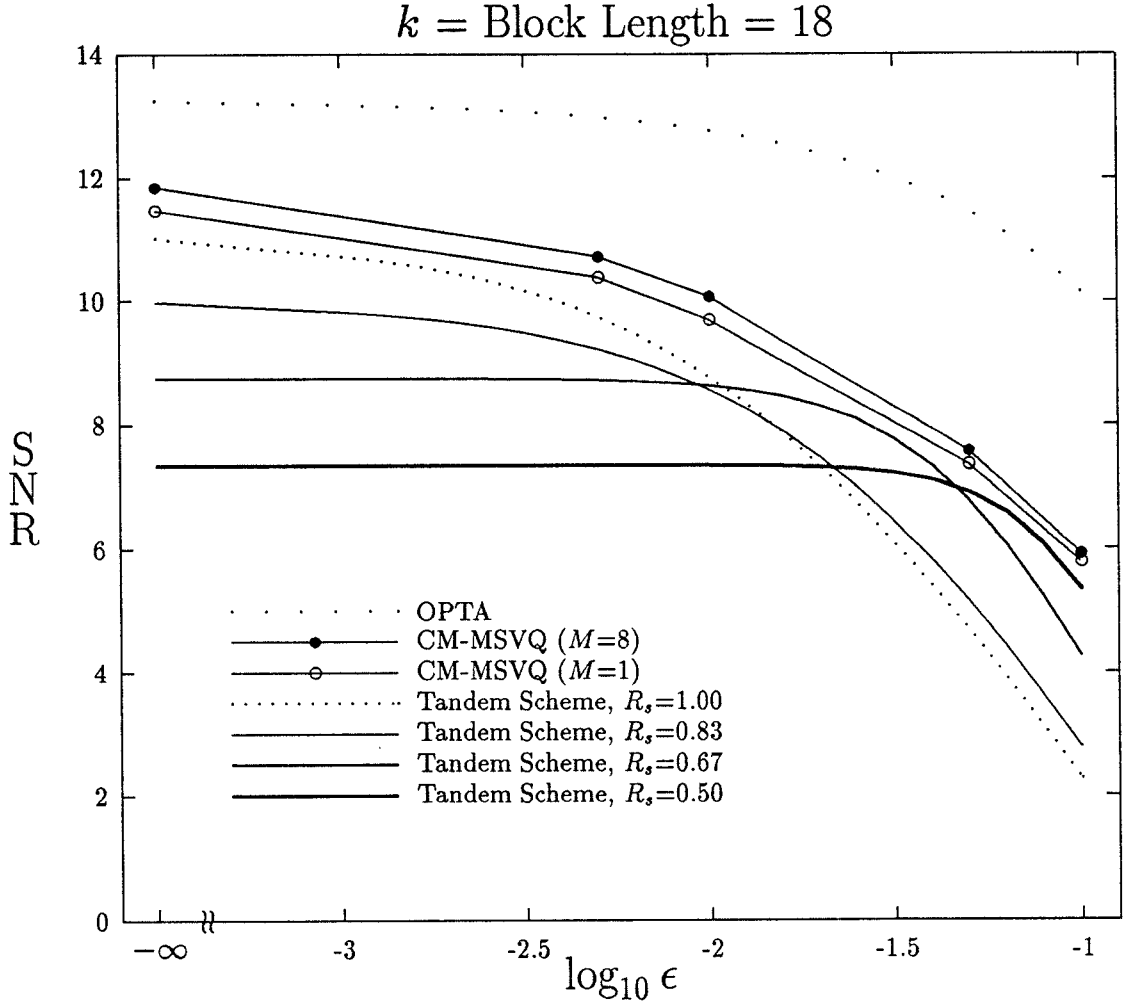


Figure 2.4: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 3; k = Effective Block Length of Both Schemes = 18.

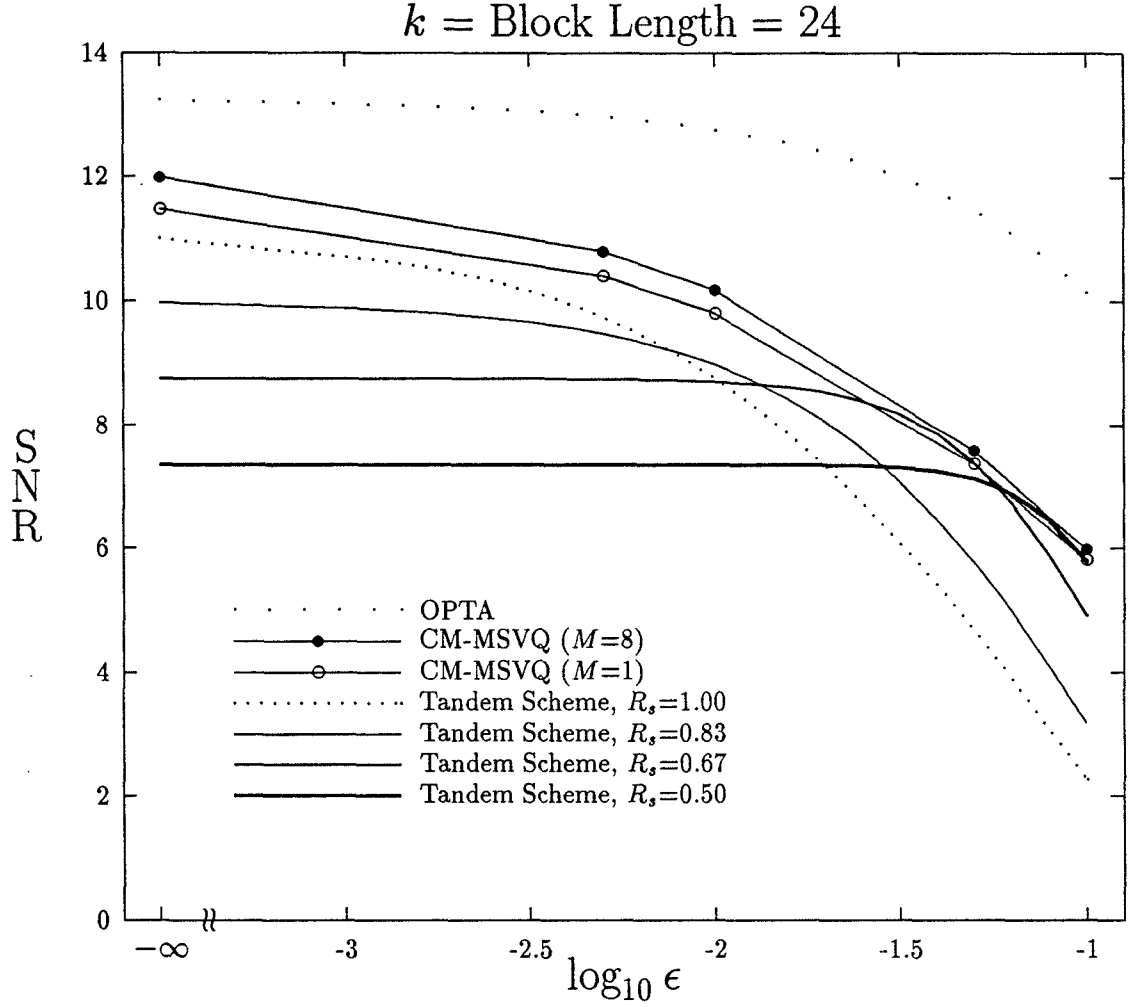


Figure 2.5: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 4; k = Effective Block Length of Both Schemes = 24.

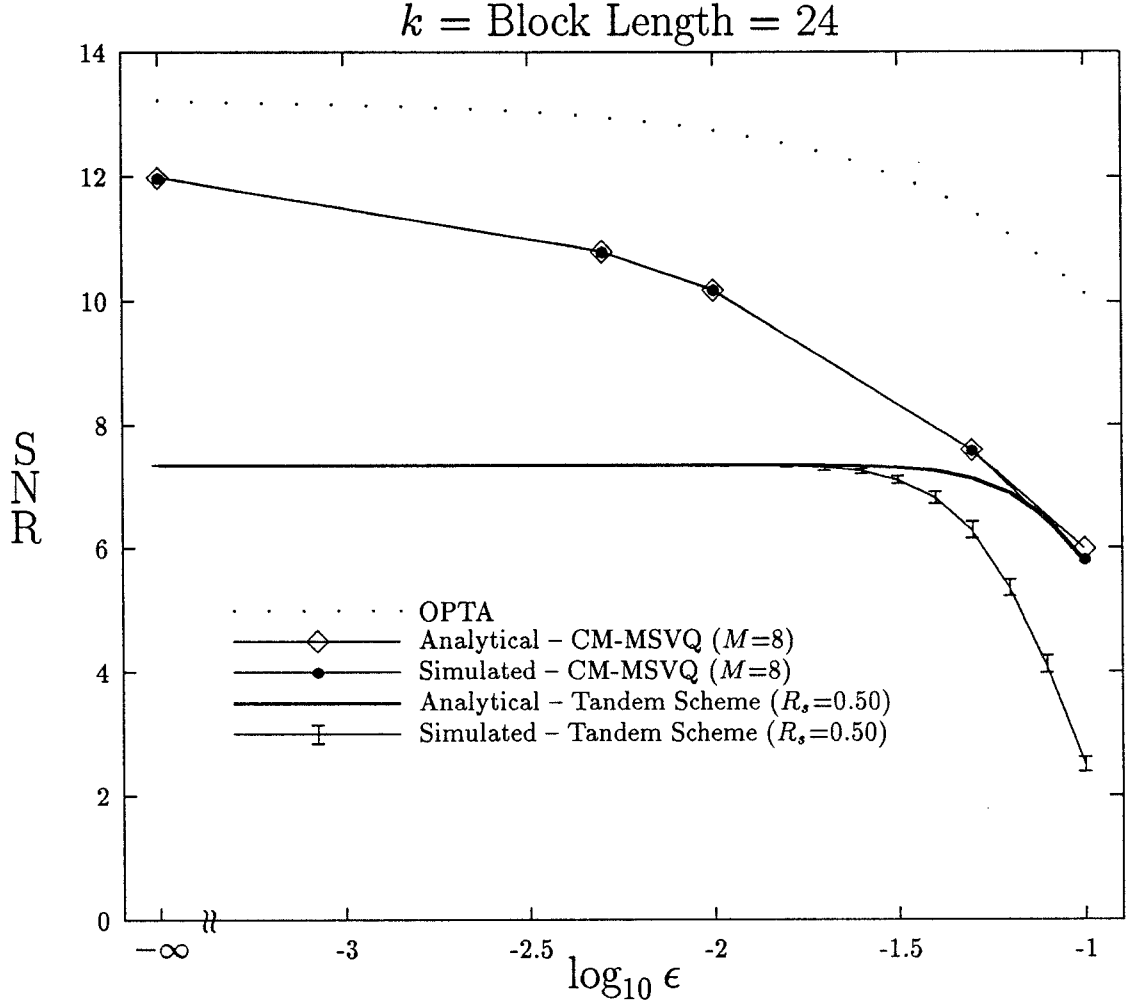


Figure 2.6: Comparisons Between Analytical Results and Simulated Results of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 4; k = Effective Block Length of Both Schemes = 24; Error Bars Indicate the Minimum and Maximum SNRs of the Simulations.

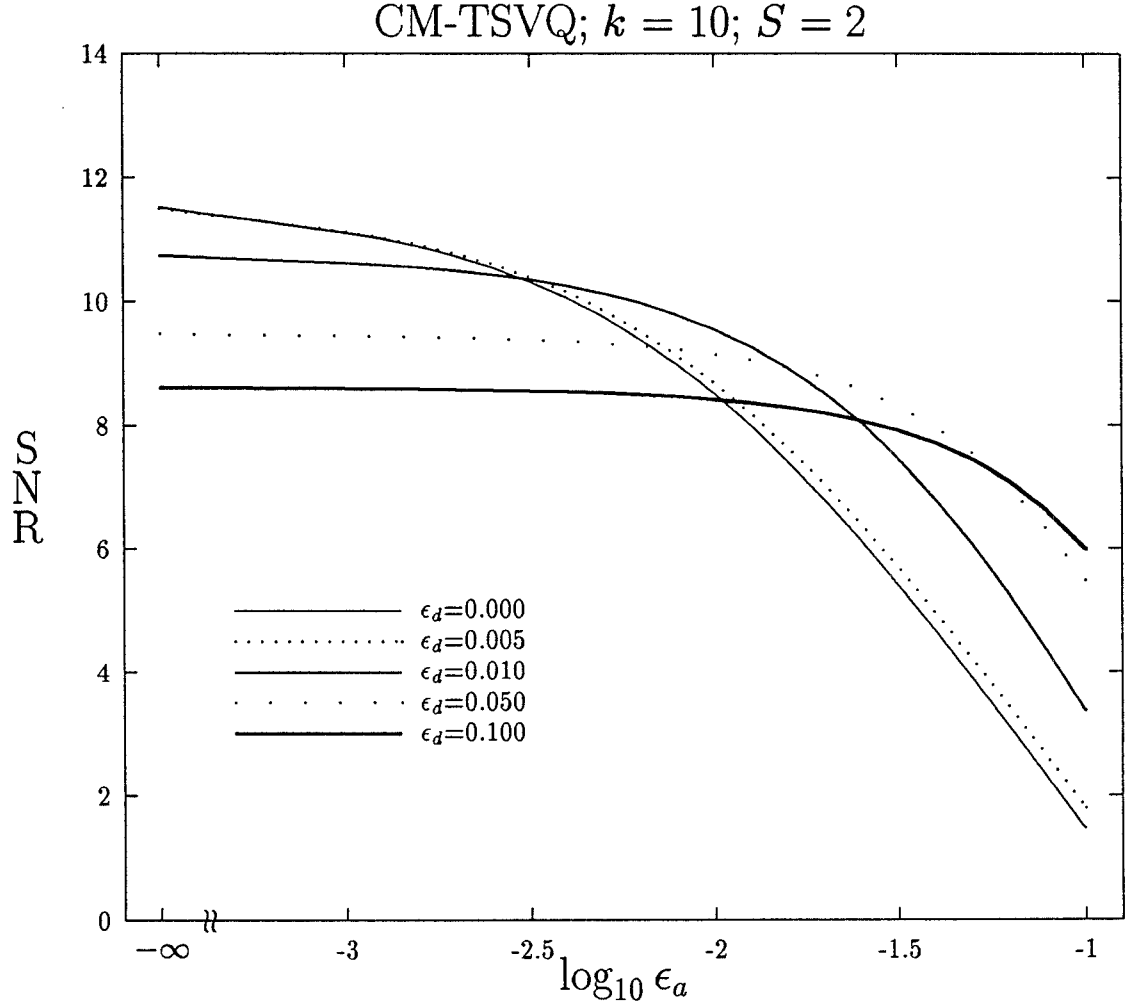


Figure 2.7: SNR (in dB) Performances of the CM-TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; ϵ_d = Design BER for the First and Second Stages; k = Dimension; S = Number of Stages.

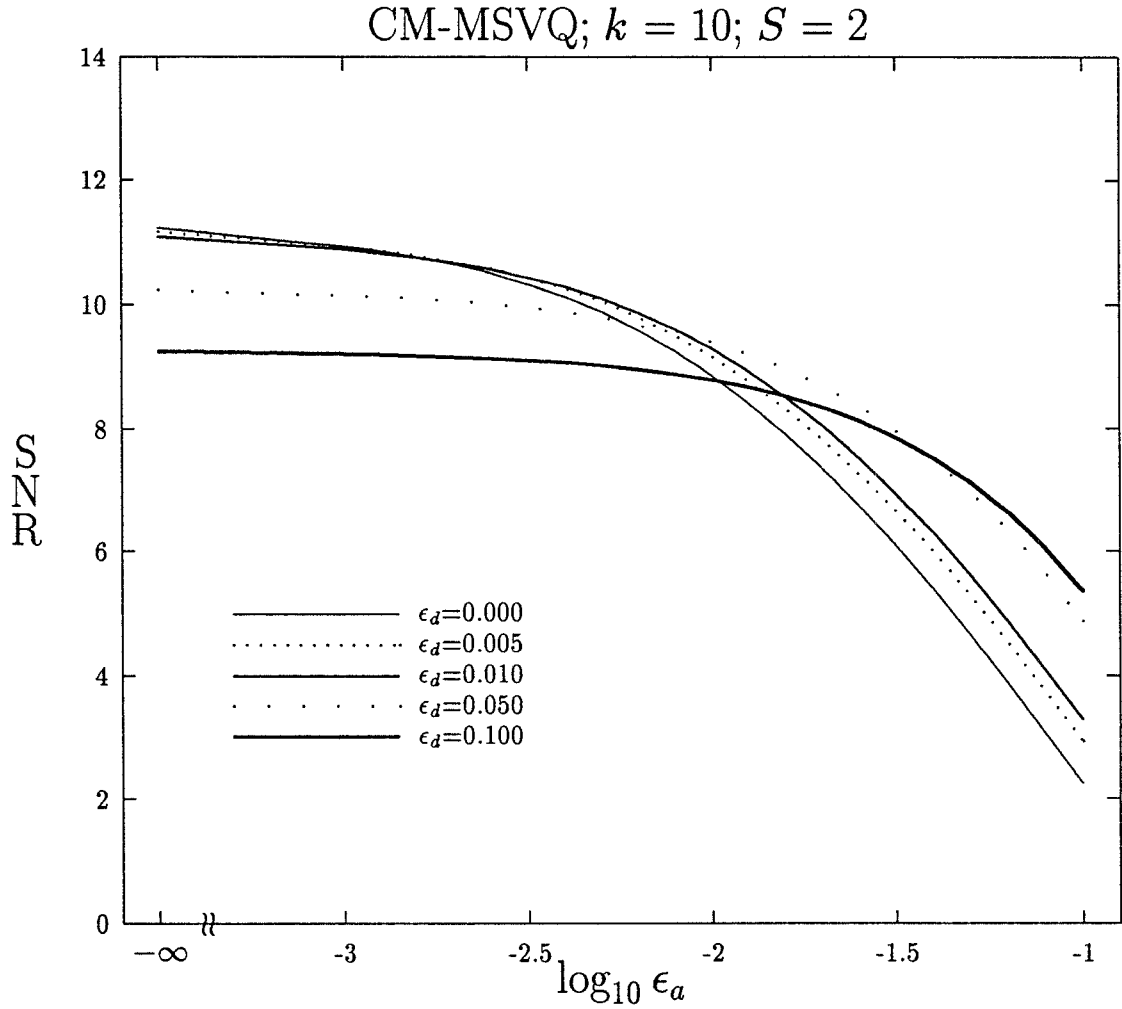


Figure 2.8: SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; ϵ_d = Design BER for the First and Second Stages; k = Dimension; S = Number of Stages.

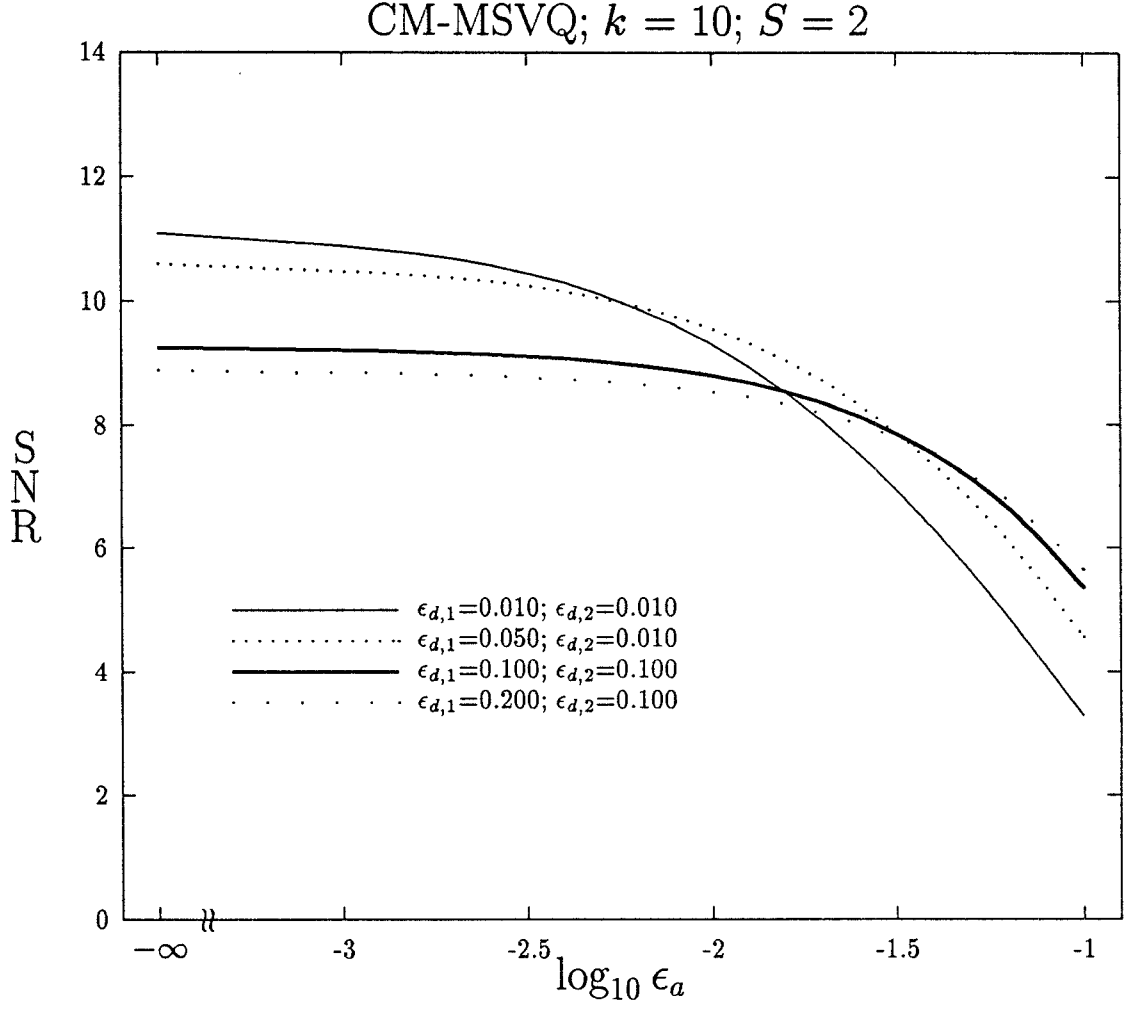


Figure 2.9: SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; $\epsilon_{d,1}$ = Design BER for the First Stage; $\epsilon_{d,2}$ = Design BER for the Second Stage; k = Dimension; S = Number of Stages.

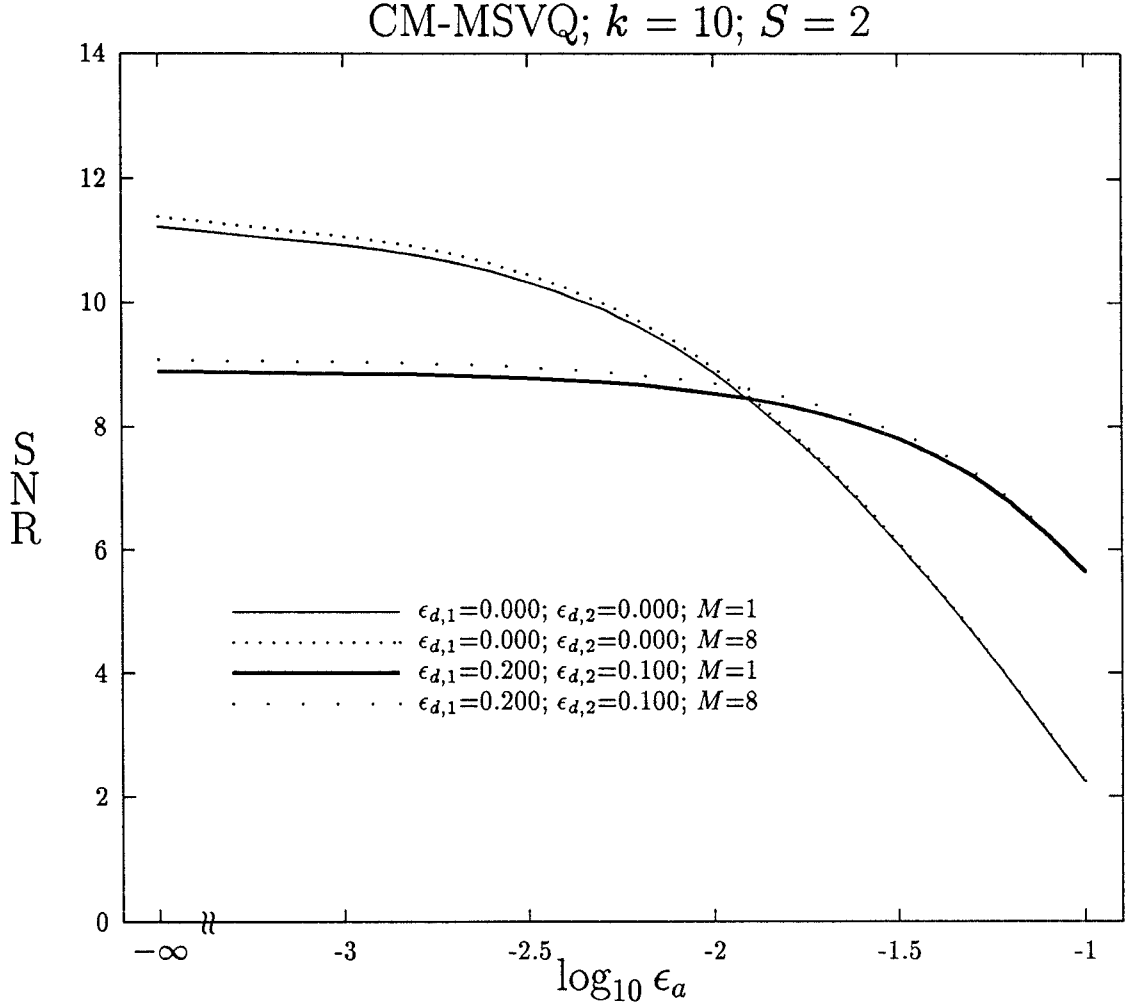


Figure 2.10: SNR (in dB) Performances of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; $\epsilon_{d,1}$ = Design BER for the First Stage; $\epsilon_{d,2}$ = Design BER for the Second Stage; M = Number of Candidates; k = Dimension; S = Number of Stages.

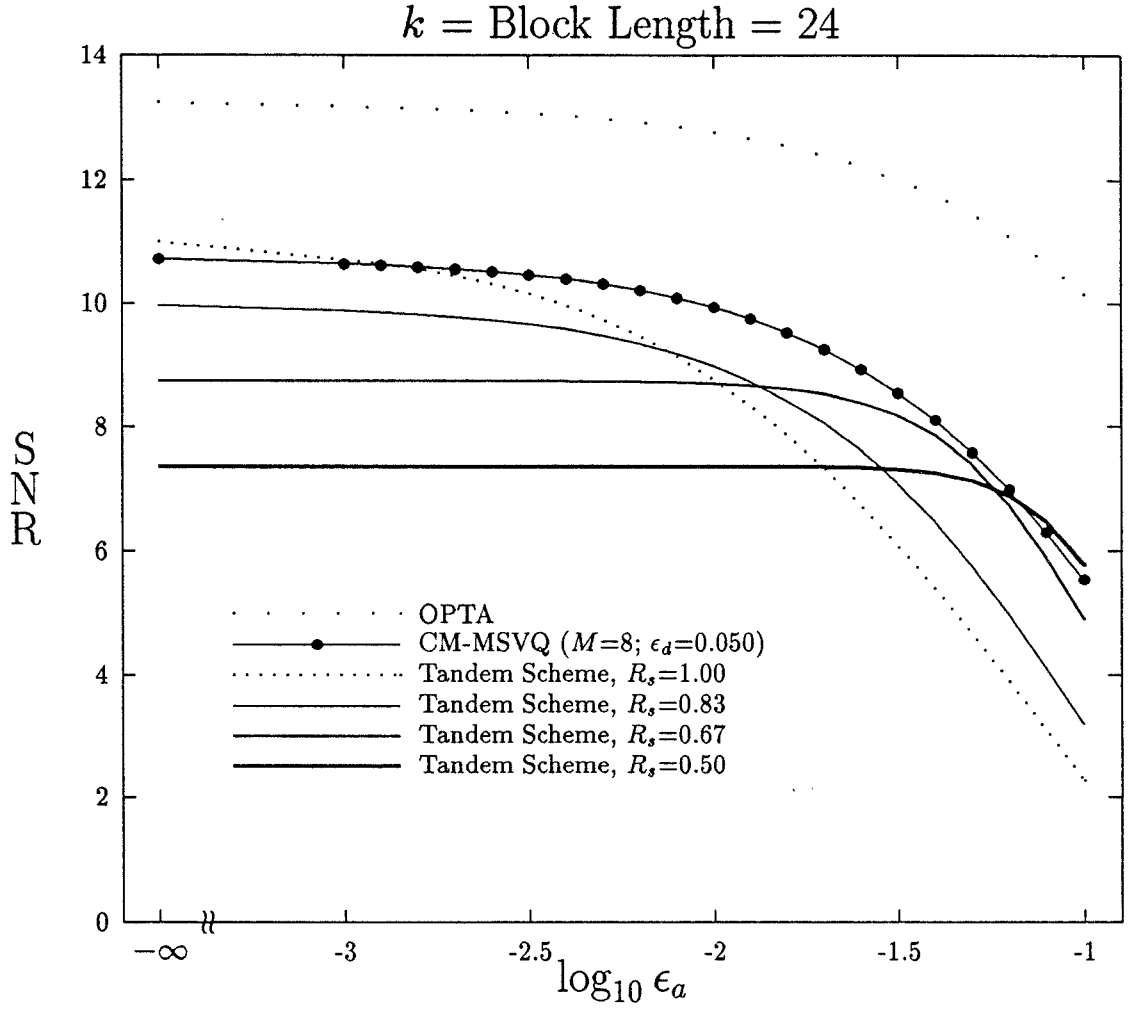


Figure 2.11: SNR (in dB) Performances of the CM-MSVQ and the Tandem Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates; ϵ_d = Design BER for the CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; ϵ_a = Actual Channel BER; k = Effective Block Length of Both Schemes = 24.

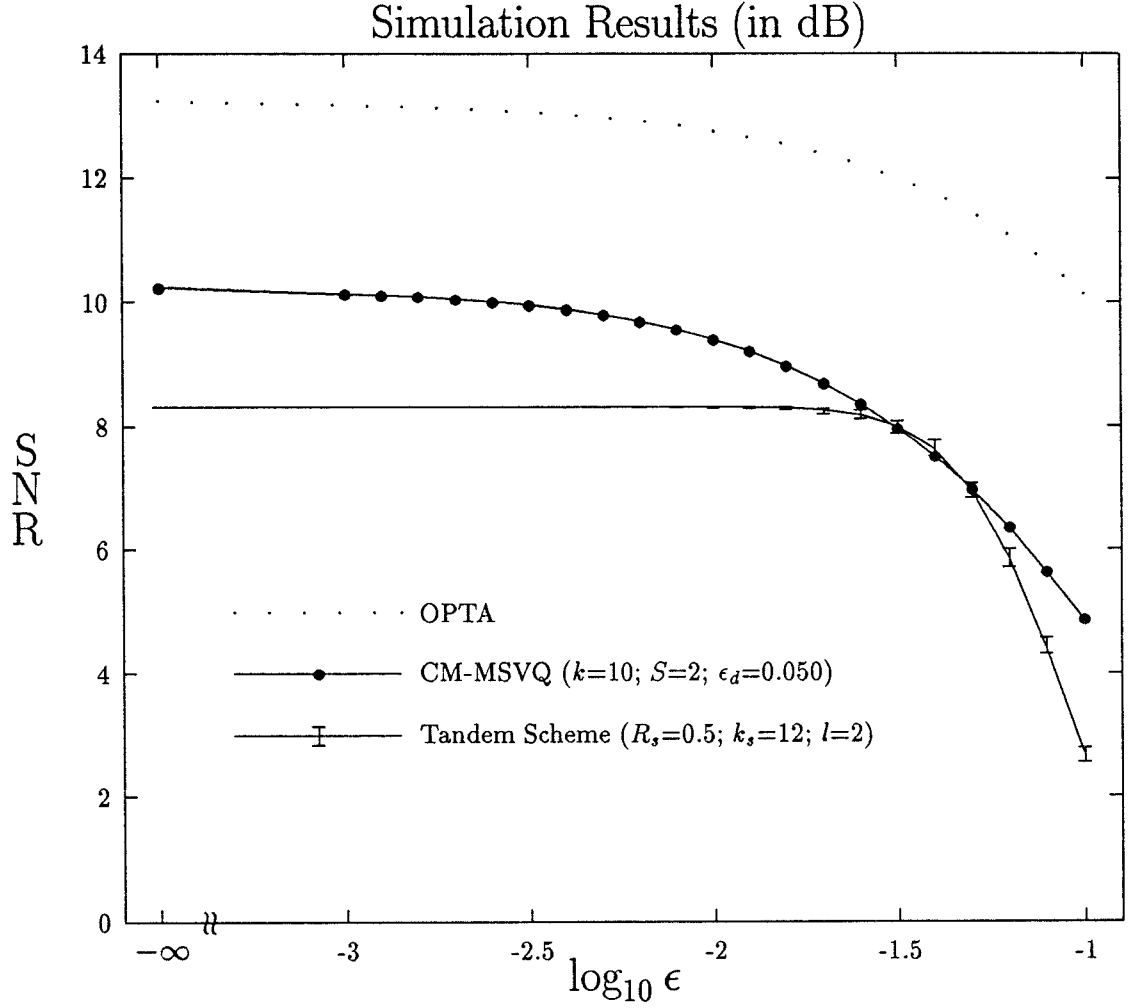


Figure 2.12: Comparisons Between CM-MSVQ and Tandem Scheme with the Same Complexity; Gauss-Markov Source with $\rho = 0.9$; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; k = Block Length of CM-MSVQ; S = Number of Stages; ϵ_d = Design BER of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme; l = Number of Source Blocks per Channel Block; Error Bars Indicate the Minimum and Maximum SNRs of the Simulations.

Chapter 3

Exploiting the Quantizer's Residual Redundancy

In this chapter, we consider a different approach to quantization over noisy channels. Here, we assume that a low-complexity quantizer had already been designed and is given to us. Since the quantizer is low in complexity, certain “residual” redundancy remains at its output. Our aim is make use of the residual redundancy to help us combat the effect of channel noise. In Section 3.1, we assume that the residual redundancy is in the form of memory. We model this memory by assuming that the quantizer output forms a discrete Markov source and we design optimal detectors for the Markov source. In Section 3.2, we assume that the residual redundancy is in the form of a non-uniform distribution. Here, we propose the use of a rate-one convolutional code to convert the residual redundancy into a more usable form.

3.1 Redundancy in the Form of Memory

3.1.1 Introduction

In his celebrated paper [48], Shannon stated that in information transmission over a noisy channel, “redundancy must be introduced in the proper way to combat the particular noise structure involved. However, any redundancy in the source will

usually help if it is utilized at the receiving point. In particular, if the source already has a certain redundancy and no attempt is made to eliminate it in matching to the channel, this redundancy will help combat noise.”

This statement, though made more than forty years ago, forms the foundation of this section. The principle assumption here is that the source to be transmitted has a certain redundancy¹ and due to certain constraints (for example, on the complexity), the transmitter makes no attempt to “match” the source to the channel. Instead, the source is transmitted directly over the channel. The problem thus is to design a receiver which fully “utilizes” the source redundancy to combat the effect of channel noise. Because of the constraint on the transmitter, the system described above inevitably suffers from a loss of optimality. Nevertheless, given this constraint, we seek to obtain the optimum receiver for a given source and channel.

First, discrete sources and channels are considered. It is hypothesized that the source is in the form of a discrete Markov chain [9] and that the channel is a discrete memoryless channel [9]. The receiver is a *maximum a posteriori* (MAP) receiver (detector) which is optimum in the sense of minimizing the probability of error [43]. Here, the redundancy between successive symbols of the Markov source is used (by the MAP detector) to provide some protection against channel errors — in a manner not unlike error correction used in convolutional coding.

The above formulation has been considered before by several authors. The most notable is the work by Drake [12], who provided the optimal instantaneous MAP decoding rule as well as bounds on the achievable probability of error. Drake also studied the special case of binary symmetric Markov source and binary sym-

¹The term “redundancy” usually has three connotations: (i) the source has memory but is uniformly distributed, (ii) the source distribution is non-uniform but it has no memory and (iii) the source has memory and a non-uniform distribution. In this section, “redundancy” implies cases (i) or (iii). More will be said on this matter in Section 3.2.1.

metric channel and gave a necessary and sufficient condition for the optimality of the singlet (“believe what you see”) decoding rule. Extending this work, Devore [11] gave necessary and sufficient conditions for the optimality of several other decoding rules. In a tutorial paper, Forney [19] stated that the MAP detection of a Markov source over a memoryless channel is the most general application of the Viterbi algorithm. He also showed that this problem is intimately related to the problem of detecting pulse-amplitude modulated (PAM) signals over channels with intersymbol interference and additive white Gaussian noise. Bahl *et al.* [5] looked at the problem of determining the *a posteriori* probabilities of a finite-state Markov source observed through a discrete memoryless channel in the context of decoding linear block and convolutional codes. More recently, Sayood and Borkenhagen [47] considered the detection of a discrete Markov source over a discrete memoryless channel in a joint source-channel DPCM image coding system. The main difference between the current section and [47] is that the detector in [47] is suboptimum, hence, it is not a MAP detector. Furthermore, we consider two variations of this problem: (i) *sequence* MAP detection which involves a large delay and (ii) *instantaneous* MAP detection which involves no delay. In [47], only sequence detection was considered. In sequence MAP detection, the objective is to determine the most probable transmitted *sequence* given an observed sequence while the objective of instantaneous MAP detection is to determine the most probable transmitted *symbol* at a particular time given all the observations up to that time.

The solutions to the above problems are applied to a joint source-channel coding system in which the source is no longer discrete but is continuous-valued. The channel remains discrete. In this case, the use of an explicit source encoder is unavoidable since the source entropy (which is infinite) is higher than the channel capacity (which is finite) [9]. The purpose of the source encoder is to reduce the source entropy to a level lower than the channel capacity by introducing

some distortion [9]. If the source encoder is optimum in the sense that it has the lowest possible rate for a given distortion criterion, then it must remove all of the source redundancy – that is, the source encoder output must have no redundancy. However, similar to [47], we assume that the source encoder (in our case, a vector quantizer (VQ) with a small block size) is suboptimum, i.e., it fails to remove all of the source redundancy. Again, this may be attributed to some complexity constraint. The redundancy which remains at the output of the source encoder is called the *residual* redundancy [47]. Here, MAP detection is applied to exploit the residual redundancy. More specifically, the VQ encoder output is modeled as a discrete Markov source and the problem reduces to that of discrete sources and channels mentioned previously. Simulation results for this system on a Gauss-Markov source are obtained and comparisons are made with Farvardin and Vaishampayan’s channel-optimized VQ [15], [16]. It is then shown that the instantaneous MAP detector can be combined with the VQ decoder to form an approximate minimum mean-squared error decoder.

The rest of this section is organized as follows. In Subsection 3.1.2, preliminary notations are established. The two MAP detection problems and their solutions are given in Subsection 3.1.3. In Subsection 3.1.4, the special case of the binary symmetric Markov source and binary symmetric channel is studied and simulation results are provided for this case. Subsection 3.1.5 includes a study of joint source-channel coding using MAP detection. Finally, conclusions are stated in Subsection 3.1.6.

3.1.2 Preliminaries

A stationary stochastic process, $\{I_n\}_{n=1}^{\infty}$, with alphabet $\mathcal{J}_N \triangleq \{0, 1, \dots, N-1\}$ of size N is said to be a discrete Markov source if $\forall n > 1$,

$$\Pr\{I_n = i_n | \mathbf{I}_1^{n-1} = \mathbf{i}_1^{n-1}\} = \Pr\{I_n = i_n | I_{n-1} = i_{n-1}\}, \quad (3.1)$$

$\forall i_k \in \mathcal{J}_N, k = 1, 2, \dots, n$, where

$$\mathbf{I}_1^{n-1} \triangleq (I_1, I_2, \dots, I_{n-1}), \quad (3.2)$$

$$\mathbf{i}_1^{n-1} \triangleq (i_1, i_2, \dots, i_{n-1}). \quad (3.3)$$

Because of stationarity, the conditional probability function, $\Pr\{I_n = i_n | I_{n-1} = i_{n-1}\}$, is independent of n . Thus, the Markov source can be completely characterized by the source transition probability matrix,

$$P(i_n | i_{n-1}) \triangleq \Pr\{I_n = i_n | I_{n-1} = i_{n-1}\}, \quad i_n, i_{n-1} \in \mathcal{J}_N, \quad (3.4)$$

and the initial source distribution,

$$P_1(i_1) \triangleq \Pr\{I_1 = i_1\}, \quad i_1 \in \mathcal{J}_N. \quad (3.5)$$

The simplest example of a discrete Markov source is the binary symmetric Markov source (BSMS) with $N = 2$, transition matrix,

$$P(i_n | i_{n-1}) = \begin{cases} p & \text{if } i_n = i_{n-1} \\ 1 - p & \text{if } i_n \neq i_{n-1} \end{cases}, \quad i_n, i_{n-1} \in \mathcal{J}_2, \quad (3.6)$$

and initial distribution $P_1(0) = P_1(1) = 0.5$. Note that for a given BSMS, we may construct a 2^m -ary Markov source by taking m successive samples of the binary source and appropriately renaming each letter of the alphabet $\mathcal{J}_{2^m}$. We will make use of this construction in a later section.

It is assumed that the Markov source, $\{I_n\}_{n=1}^\infty$, is transmitted across a discrete memoryless channel (DMC) with the same alphabet, $\mathcal{J}_N$, as the source. The received sequence (at the output of the channel) is denoted by $\{J_n\}_{n=1}^\infty$. The channel is described by the channel transition probability matrix,

$$Q(j_n | i_n) = \Pr\{J_n = j_n | I_n = i_n\}, \quad i_n, j_n \in \mathcal{J}_N. \quad (3.7)$$

The simplest example of a DMC is the binary symmetric channel (BSC) with $N = 2$ and transition matrix,

$$Q(j_n | i_n) = \begin{cases} 1 - \epsilon & \text{if } j_n = i_n \\ \epsilon & \text{if } j_n \neq i_n \end{cases}, \quad j_n, i_n \in \mathcal{J}_2. \quad (3.8)$$

Again, note that we may construct a 2^m -ary DMC through m uses of a BSC.

3.1.3 Statement of Problems and Solutions

The two problems to be considered in this section are as follows. Given N, P, P_1, Q and an observation sequence $\mathbf{J}_1^n = \mathbf{j}_1^n$, determine (i) the most probable transmitted sequence,

$$\hat{\mathbf{i}}_1^n = \arg \max_{\mathbf{i}_1^n \in \mathcal{J}_N^n} \Pr\{\mathbf{I}_1^n = \mathbf{i}_1^n | \mathbf{J}_1^n = \mathbf{j}_1^n\}, \quad (3.9)$$

and (ii) the most probable transmitted symbol at time n ,

$$\hat{i}_n = \arg \max_{i_n \in \mathcal{J}_N} \Pr\{I_n = i_n | \mathbf{J}_1^n = \mathbf{j}_1^n\}. \quad (3.10)$$

These two problems are referred to as the *sequence* MAP detection and the *instantaneous* MAP detection problems, respectively.

We now address the sequence MAP detection problem. The solution to this problem was first derived by Forney [19]. Equation (3.9) is equivalent to

$$\hat{\mathbf{i}}_1^n = \arg \max_{\mathbf{i}_1^n \in \mathcal{J}_N^n} \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n | \mathbf{I}_1^n = \mathbf{i}_1^n\} \Pr\{\mathbf{I}_1^n = \mathbf{i}_1^n\}. \quad (3.11)$$

By using the fact that the channel is memoryless and by successively applying conditional probability and the Markovian property of the source, the above can be expressed as

$$\hat{\mathbf{i}}_1^n = \arg \max_{\mathbf{i}_1^n \in \mathcal{J}_N^n} \left[\prod_{k=1}^n Q(j_k | i_k) \right] \left[\prod_{k=2}^n P(i_k | i_{k-1}) P_1(i_1) \right]. \quad (3.12)$$

Since the logarithm is monotonic, the above is equivalent to

$$\hat{\mathbf{i}}_1^n = \arg \max_{\mathbf{i}_1^n \in \mathcal{J}_N^n} \left[\sum_{k=2}^n \log[Q(j_k | i_k) P(i_k | i_{k-1})] + \log[Q(j_1 | i_1) P_1(i_1)] \right]. \quad (3.13)$$

In the above form, the sequence MAP detector can be implemented straightforwardly using the well-known Viterbi algorithm [19]. The trellis has N states. There are (at most) N branches entering and leaving each state. The path metric

of the branch leaving state i_{k-1} at time $k - 1$ and entering state i_k at time k is $\log[Q(j_k|i_k)P(i_k|i_{k-1})]$. This metric is similar to that of Sayood and Borkenhagen [47] except that they have an additional term $-\log[\Pr\{J_k = j_k|I_{k-1} = i_{k-1}\}]$ which requires more computational burden than our metric. Unlike [47], the metric given here is optimum. We point out that the objective here is to minimize the sequence error probability rather than the symbol error probability. To minimize the symbol error probability for the given (large) delay, approaches similar to [5], [1], [24] can be taken.

The delay of the sequence MAP detector is quite large ($n - 1$ samples). That is, the sequence MAP detector must wait to observe the entire sequence $\mathbf{j}_1^n$ before making a decision on the value of i_1 . In the following, we show how this delay problem² can be alleviated.

We now turn to the instantaneous MAP detection problem. The instantaneous MAP detector is different from the sequence MAP detector in that it makes a decision on i_n as soon as j_n is received. Also, the symbol error probability is minimized in this case. Equation (3.10) is equivalent to

$$\hat{i}_n = \arg \max_{i_n \in \mathcal{J}_N} \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n, I_n = i_n\}. \quad (3.14)$$

Let us denote by

$$f^{(n)}(i_n) \triangleq \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n, I_n = i_n\}, \quad (3.15)$$

the objective function³ that is to be maximized at time n . $f^{(n)}(i_n)$ can be expressed as the sum of the joint probabilities:

$$f^{(n)}(i_n) = \sum_{\mathbf{i}_1^{n-1} \in \mathcal{J}_N^{n-1}} \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n, I_n = i_n, \mathbf{I}_1^{n-1} = \mathbf{i}_1^{n-1}\},$$

²Another way of reducing the delay is to limit the trellis path memory using some symbol release rule [32]. This will be explored in a later section.

³Actually, the objective function also depends on the observations $\mathbf{j}_1^n$. However, for notational simplicity we will not explicitly express this dependency.

$$= \sum_{\mathbf{i}_1^{n-1} \in \mathcal{J}_N^{n-1}} \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n, \mathbf{I}_1^n = \mathbf{i}_1^n\}. \quad (3.16)$$

Each term in the summation can be expanded using the definition of conditional probability,

$$f^{(n)}(i_n) = \sum_{\mathbf{i}_1^{n-1} \in \mathcal{J}_N^{n-1}} \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n | \mathbf{I}_1^n = \mathbf{i}_1^n\} \Pr\{\mathbf{I}_1^n = \mathbf{i}_1^n\}. \quad (3.17)$$

Note that the two terms in (3.17) are the same as those in (3.11). We thus proceed as before, yielding

$$f^{(n)}(i_n) = \sum_{\mathbf{i}_1^{n-1} \in \mathcal{J}_N^{n-1}} \left[\prod_{k=1}^n Q(j_k | i_k) \right] \left[\prod_{k=2}^n P(i_k | i_{k-1}) P(i_1) \right]. \quad (3.18)$$

Taking out the common factor $Q(j_n | i_n)$ results in

$$\begin{aligned} f^{(n)}(i_n) &= Q(j_n | i_n) \sum_{i_{n-1} \in \mathcal{J}_N} P(i_n | i_{n-1}) \\ &\quad \times \sum_{\mathbf{i}_1^{n-2} \in \mathcal{J}_N^{n-2}} \left[\prod_{k=1}^{n-1} Q(j_k | i_k) \right] \left[\prod_{k=2}^{n-1} P(i_k | i_{k-1}) P(i_1) \right]. \end{aligned} \quad (3.19)$$

Comparing equations (3.18) and (3.19), we can see that the second summation in (3.19) is nothing but $f^{(n-1)}(i_{n-1})$, the objective function that was maximized at time $n-1$. Hence, the instantaneous MAP detector can be implemented using the following recursion,

$$\begin{aligned} f^{(1)}(i_1) &= Q(j_1 | i_1) P_1(i_1), \\ f^{(n)}(i_n) &= Q(j_n | i_n) \sum_{i_{n-1} \in \mathcal{J}_N} P(i_n | i_{n-1}) f^{(n-1)}(i_{n-1}), \quad n = 2, 3, \dots, \end{aligned} \quad (3.20)$$

where

$$\hat{i}_n = \arg \max_{i_n \in \mathcal{J}_N} f^{(n)}(i_n), \quad \text{for } n = 1, 2, \dots \quad (3.21)$$

It is noted that (3.20) is a special case of a similar equation in [5]. Also, a similar equation was given (but not derived) in Drake's thesis [12]. Drake also provided an algorithm for determining the steady-state statistics of the (normalized) cost

function [12]. Implementation of the instantaneous MAP detector using the recursion (3.20) requires $2(N^2 + N)$ words of memory (for storing $P(\cdot|\cdot)$, $Q(\cdot|\cdot)$, $f^{(n)}(\cdot)$ and $f^{(n-1)}(\cdot)$), $N^2 + N$ multiplications, $N^2 - N$ additions and $N - 1$ comparisons per unit time.

We should point out that there is an obvious gap between the sequence and the instantaneous MAP detectors. One has a very large delay while the other has no delay. It would be useful to have a detector whose delay is in between that of the instantaneous and the sequence MAP detectors. We will show how this can be accomplished in the next subsection.

3.1.4 Special Case: BSMS and BSC

In this subsection, we consider the special case of BSMS and BSC. Simulation results for the sequence and instantaneous MAP detectors for BSMSs with $p = 0.9, 0.95$ and 0.99 are plotted in Figures 3.1, 3.2 and 3.3, respectively, for various values of ϵ . The simulation was performed on 10,000 samples of the BSMS and the experiment was repeated 50 times. For the case of sequence MAP detection, the entire sequence of 10,000 samples were decoded before the symbols were released. In Figures 3.1-3.3, $\hat{\epsilon} \triangleq \Pr\{\text{error}\} = \Pr\{\hat{I}_n \neq I_n\}$ and the average values of $\hat{\epsilon}$ (over the 50 experiments) are shown with the error bars indicating the minimum and maximum values of $\hat{\epsilon}$. The curve labeled “w/o MAP” indicates the probability of error when no MAP detection is performed (hence $\hat{\epsilon} = \epsilon$).

We also present in these figures the result of the instantaneous MAP detector for a 2^m -ary Markov source (constructed from m successive samples of the BSMS) and a 2^m -ary DMC (constructed through m uses of the BSC) for $m = 4$. Here, we treat a block of m binary samples as one symbol in an alphabet of size 2^m and proceed as before. Hence, we have accomplished the task of bridging the delay

gap⁴ between the instantaneous and sequence MAP detectors. In this case, the delay is $m - 1$ samples (of the BSMS). Note that the performance curve of the instantaneous MAP (with $m=4$) lies in between that of the instantaneous MAP (with $m=1$) and the sequence MAP.

Observe that for small values of ϵ all four curves coincide with each other. In fact, we found that in these cases, $\hat{I}_n \equiv J_n$, i.e., the MAP detector does absolutely nothing. At a certain point on the graph, the “Inst. MAP ($m=4$)” and the “Seq. MAP” curves diverge from the others. At another point, the “Inst. MAP ($m=1$)” curve diverges from the “w/o MAP” curve. This phenomena can be explained by the following theorems and corollaries.

Theorem 1 *Given $p \in (\frac{1}{2}, 1), \epsilon \in (0, \frac{1}{2})$. For $n \geq 3$, $\hat{\mathbf{I}}_1^n = \mathbf{J}_1^n$ is the optimum sequence detection rule if and only if*

$$\epsilon \leq \epsilon_1 \triangleq \frac{(1-p)^2}{p^2 + (1-p)^2}. \quad (3.22)$$

Proof: We first show that if $\epsilon \leq \epsilon_1$, then $\forall \mathbf{j}_1^n \in \mathcal{J}_2^n$,

$$\Pr\{\mathbf{I}_1^n = \mathbf{j}_1^n | \mathbf{J}_1^n = \mathbf{j}_1^n\} \geq \Pr\{\mathbf{I}_1^n = \mathbf{i}_1^n | \mathbf{J}_1^n = \mathbf{j}_1^n\}, \quad \forall \mathbf{i}_1^n \in \mathcal{J}_2^n. \quad (3.23)$$

Define $d \triangleq d_H(\mathbf{i}_1^n, \mathbf{j}_1^n)$ = Hamming distance between the two sequences ($0 \leq d \leq n$).

Then

$$\alpha \triangleq \frac{\Pr\{\mathbf{I}_1^n = \mathbf{j}_1^n | \mathbf{J}_1^n = \mathbf{j}_1^n\}}{\Pr\{\mathbf{I}_1^n = \mathbf{i}_1^n | \mathbf{J}_1^n = \mathbf{j}_1^n\}} = \frac{(1-\epsilon)^n \prod_{k=2}^n P(j_k | j_{k-1}) P_1(j_1)}{\epsilon^d (1-\epsilon)^{n-d} \prod_{k=2}^n P(i_k | i_{k-1}) P_1(i_1)}. \quad (3.24)$$

The terms $P_1(j_1)$ and $P_1(i_1)$ cancel because they are both $\frac{1}{2}$. Now define,

$$\begin{aligned} \mathcal{K} &\triangleq \{2, 3, \dots, n\}, \\ \mathcal{A} &\triangleq \{k \in \mathcal{K} : i_k = j_k, i_{k-1} = j_{k-1}\}, \\ \mathcal{B} &\triangleq \{k \in \mathcal{K} : k \notin \mathcal{A}\}. \end{aligned} \quad (3.25)$$

⁴Another way of bridging the delay gap is to modify the instantaneous MAP formulation to allow a finite delay. See [1] for more details.

Hence

$$\alpha = \left(\frac{1-\epsilon}{\epsilon}\right)^d \frac{\prod_{k \in \mathcal{A}} P(j_k | j_{k-1}) \prod_{k \in \mathcal{B}} P(j_k | j_{k-1})}{\prod_{k \in \mathcal{A}} P(i_k | i_{k-1}) \prod_{k \in \mathcal{B}} P(i_k | i_{k-1})}. \quad (3.26)$$

By definition of $\mathcal{A}$, the products over $\mathcal{A}$ are equal and they cancel out,

$$\alpha = \left(\frac{1-\epsilon}{\epsilon}\right)^d \prod_{k \in \mathcal{B}} \left[\frac{P(j_k | j_{k-1})}{P(i_k | i_{k-1})} \right]. \quad (3.27)$$

Note that each term inside the product is either $\frac{p}{1-p}$, 1 or $\frac{1-p}{p}$. The smallest of these is $\frac{1-p}{p}$ (since $p > \frac{1}{2}$). Thus

$$\alpha \geq \left(\frac{1-\epsilon}{\epsilon}\right)^d \left(\frac{1-p}{p}\right)^{|\mathcal{B}|}. \quad (3.28)$$

A moment of reflection would reveal that $|\mathcal{B}| \leq 2d$ (equality is achieved if the set $\{k \in \mathcal{K} \cup \{1\} : i_k \neq j_k\}$ does not contain either any pair of consecutive integers, $\{1\}$ or $\{n\}$). Hence

$$\alpha \geq \left(\frac{1-\epsilon}{\epsilon}\right)^d \left(\frac{1-p}{p}\right)^{2d} \geq 1. \quad (3.29)$$

The last inequality follows from our assumption on ϵ ($\leq \epsilon_1$).

The converse is proved by a counterexample. Let $\mathbf{i}_1^n = (0, 0, \dots, 0)$ and $\mathbf{j}_1^n = (0, 0, \dots, 0, 1, 0, \dots, 0)$ ($d = 1$). If $\epsilon > \epsilon_1$, then

$$\alpha = \left(\frac{1-\epsilon}{\epsilon}\right) \left(\frac{1-p}{p}\right)^2 < 1. \quad (3.30)$$

Here we use the assumption that $n \geq 3$. $\square$

For instantaneous MAP detection, we have the following theorem and corollaries; the proof of the theorem is provided in [12], [11].

Theorem 2 *Given $p \in (\frac{1}{2}, 1)$, $\epsilon \in (0, \frac{1}{2})$. For $n \geq 1$, $\hat{I}_n = J_n$ is the optimum instantaneous detection rule if and only if*

$$(1 - 2\epsilon)^2 \geq 4p - 3. \quad (3.31)$$

Corollary 1 *Given $p \in (\frac{1}{2}, \frac{3}{4}]$. For $n \geq 1$, $\hat{I}_n = J_n$ is the optimum instantaneous detection rule $\forall \epsilon \in (0, \frac{1}{2})$.*

Corollary 2 *Given $p \in [\frac{3}{4}, 1), \epsilon \in (0, \frac{1}{2})$. For $n \geq 1$, $\hat{I}_n = J_n$ is the optimum instantaneous detection rule if and only if*

$$\epsilon \leq \epsilon_2 \triangleq \frac{1}{2} - \sqrt{p - \frac{3}{4}}. \quad (3.32)$$

The above theorems⁵ state that under certain conditions, the simple detection rule $\hat{I}_n = J_n$ is optimum. Hence, any detection algorithm would be useless under these conditions. For a given BSMS with parameter p , ϵ_1 is defined to be the *critical* channel BER for the sequence MAP detector. If $p > 0.75$, then ϵ_2 is defined to be the critical BER for the instantaneous MAP detector. Sequence (instantaneous) MAP detection is useful if and only if $\epsilon > \epsilon_1$ ($\epsilon > \epsilon_2$). The values of ϵ_1 and ϵ_2 are indicated in Figures 3.1-3.3. It can be seen that the simulation results in these figures agree with Theorem 1 and Corollary 2.

Although the critical BER for the “Inst. MAP ($m=4$)” curve has not been derived, it seems from Figures 3.1-3.3 that this value is close to ϵ_1 (not ϵ_2). This is due mainly to the $m - 1$ samples of delay in this case. Also, notice the change in the slope of this curve at $\epsilon = \epsilon_2$.

3.1.5 Joint Source-Channel Coding

We now apply the solutions of the MAP detection problems to a joint source-channel coding system. The system to be considered is depicted in Figure 3.4. The source code is a VQ with blocklength k and rate $R = \frac{1}{k} \log_2 N$ bits/sample (or bits/dimension), where N is the size of the VQ codebook. The VQ is memoryless in the sense that it operates on a block of k source samples independent of every other samples. The source, $\{\mathbf{X}_n\}_{n=1}^{\infty}$, is assumed to be correlated from block to

⁵Theorem 1 can be extended to a class of non-binary Markov sources. This class consists of discrete Markov sources with alphabet size $N = 2^m$, p = the probability of returning to the same state and $q = (1 - p)/(N - 1)$ = the probability of going to any other state ($p > q$). The channel is obtained through m uses of a BSC. In this case, the critical BER is $\epsilon_1 = (1 - p)^2 / [(N - 1)^2 p^2 + (1 - p)^2]$. The proof of this is similar to that of Theorem 1 and is omitted.

block and, therefore, the VQ encoder output, $\{I_n\}_{n=1}^\infty$, is also correlated. This correlation is called the *residual* redundancy [47]. We model the residual redundancy by assuming that the sequence, $\{I_n\}_{n=1}^\infty$, forms a discrete Markov source. As before, $\{I_n\}_{n=1}^\infty$ is transmitted across a DMC. At the receiver, either sequence or instantaneous MAP detection is performed. The MAP detector output, $\{\hat{I}_n\}_{n=1}^\infty$, is then provided to the VQ decoder to obtain the reconstructed vector, $\tilde{\mathbf{x}}_n = \mathbf{c}_{\hat{I}_n}$, where $\mathcal{C} = \{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{N-1}\}$ is the VQ codebook. The system in Figure 3.4 is similar to that of Sayood and Borkenhagen [47] except that in [47], they used DPCM rather than VQ for the source code.

Though the purpose of the VQ encoder is to remove all of the source redundancy, it fails to do so mainly because of the limitation on the complexity and hence on the blocklength k [20]. Since, by assumption, the source is highly correlated and since k is small, the VQ encoder output is also correlated from one block to the next. This inherent redundancy in the data is exploited by the MAP detector to combat channel errors. The VQ encoder can be thought of as a combined source-channel encoder. It acts like a source encoder because it reduces the source entropy. It also functions like a channel encoder since its output contains redundancy. Thus, in Figure 3.4, there is no explicit channel encoder (though the MAP detector can be thought of as a channel decoder). Note that the system in Figure 3.4 does not require an increase in bandwidth like those of traditional channel coding techniques.

Simulation results for a first-order Gauss-Markov source, i.e., an autoregressive (AR(1)) source, with correlation coefficient $\rho = 0.9$ are given in Tables 3.1-3.3. These results are also plotted in Figures 3.5-3.9. The Linde, Buzo and Gray (LBG) algorithm [33] was used to design the VQ. The transition probability matrix, $P(\cdot|\cdot)$, was estimated using a long training sequence (1.2 million source samples). The initial distribution, $P_1(\cdot)$, was chosen to be the equilibrium distribution [9]. The

simulation was performed for 50,000 vectors⁶ from outside the training sequence and the experiment was repeated 50 times using 50 different realizations of the channel. The average squared-error distortion (averaged over the 50 trials) are shown (in dB). The variation between the minimum and maximum average distortion is quite small and is not shown in the figures. For $k = 1$ and $R = 1.0$ (Figure 3.5), the estimated transition matrix for $\{I_n\}_{n=1}^{\infty}$ is close to that of a BSMS with $p = 0.856$; the corresponding values of ϵ_1 and ϵ_2 are shown in Figure 3.5. Note the dip in the performance curve at the critical BERs. For comparison, we also present the results of Farvardin and Vaishampayan’s channel-optimized VQ (COVQ) [15], [16]⁷. In many cases, both the sequence and instantaneous MAP detection systems beat the COVQ by wide margins.

Next, we consider a method of improving the performance of the system in Figure 3.4. Observe that, similar to hard-decision convolutional decoding [32], there is an irreversible loss of information due to the “hard decision” made by the MAP detector. For example, the instantaneous MAP detector provides the most probable index $\hat{i}_n$ to the VQ decoder; but it does not tell the decoder how much confidence it has that this choice is correct. This information is embedded in the objective function $f^{(n)}(i_n)$. The remedy, therefore, is to combine the MAP detector and the VQ decoder into a single decoder. This decoder can be thought of as a combined source-channel decoder. We consider only the instantaneous case; the sequence decoding problem is left open.

From classical estimation theory [43], the *minimum mean-squared error* (MMSE)

⁶Due to the large complexity in the case of sequence MAP detection, we have limited the trellis path memory to $K = 50N$ vectors. The symbol release rule is as follows. The best path at time instant n is traced backward K time units. The symbol on this path is then released. We found that there was no loss of optimality for this large value of K .

⁷The results taken from [15] are for a memoryless Gaussian source. However, since $k = 1$, we assume that the same results hold for a Gauss-Markov source.

estimate of $\mathbf{X}_n$ given $\mathbf{J}_1^n = \mathbf{j}_1^n$ is given by

$$\hat{\mathbf{x}}_n = E[\mathbf{X}_n | \mathbf{J}_1^n = \mathbf{j}_1^n]. \quad (3.33)$$

Using conditional expectation, (3.33) is equivalent to

$$\hat{\mathbf{x}}_n = \sum_{i_n \in \mathcal{J}_N} E[\mathbf{X}_n | I_n = i_n, \mathbf{J}_1^n = \mathbf{j}_1^n] \Pr\{I_n = i_n | \mathbf{J}_1^n = \mathbf{j}_1^n\}. \quad (3.34)$$

Note that the second factor in the above summation is, up to a scale factor, equal to $f^{(n)}(i_n)$. Also, note that $\mathbf{X}_n$ depends directly on I_n while its dependence on $\mathbf{J}_1^{n-1}$ is only through $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{n-1})$ and through $\mathbf{I}_1^{n-1}$. Roughly speaking, $\mathbf{X}_n$ depends on I_n more than it depends on $\mathbf{J}_1^{n-1}$. Hence, we approximate the first factor in the summation in (3.34) by $E[\mathbf{X}_n | I_n = i_n] = \bar{\mathbf{c}}_{i_n}$, where $\bar{\mathbf{c}}_{i_n}$ is the centroid of encoding region i_n (for LBG-VQ [33], $\bar{\mathbf{c}}_{i_n} = \mathbf{c}_{i_n}$ is a codevector). Following this, we form an approximate MMSE (AMMSE) decoder as follows:

$$\tilde{\mathbf{x}}_n = \frac{1}{\beta_n} \sum_{i_n \in \mathcal{J}_N} \bar{\mathbf{c}}_{i_n} f^{(n)}(i_n), \quad (3.35)$$

where

$$\beta_n = \sum_{i_n \in \mathcal{J}_N} f^{(n)}(i_n) = \Pr\{\mathbf{J}_1^n = \mathbf{j}_1^n\}, \quad (3.36)$$

is the scale factor. Implementation of the AMMSE decoder requires only an additional $kN + N$ multiply-add per vector over the instantaneous MAP system in Figure 3.4.

Simulation results for the instantaneous AMMSE decoder are given in Tables 3.1-3.3 and also in Figures 3.5-3.9. Note that, in most cases, the result for the instantaneous AMMSE decoder is close to that of the sequence MAP decoder. However, when the channel is very noisy, the AMMSE decoder is considerably better. Of course, the delay of the instantaneous AMMSE decoder is much less than that of the sequence MAP decoder.

Finally, we combine the ideas presented in this section with that of COVQ [16]. Since these two ideas are, loosely speaking, independent, combining them should

lead to a system that is better than one which uses only one of those two ideas. The system considered consists of a COVQ encoder and an AMMSE decoder. Simulation results for this system are given in Table 3.4 for a few selected cases. Here, ϵ_d is the design BER of the COVQ encoder and ϵ_a is the actual channel BER (the AMMSE decoder was designed for ϵ_a). It can be seen that for all positive ϵ_d considered the proposed system beats the COVQ. However, the best result (also plotted in Figures 3.7 and 3.9) does not occur for $\epsilon_d = \epsilon_a$ but for an ϵ_d smaller than ϵ_a . One possible explanation for this is the following. Suppose that instead of using the AMMSE decoder, we use the system in Figure 3.4 with COVQ replacing VQ. Since the effective channel (between I_n and $\hat{I}_n$) will be less noisy than the actual channel (between I_n and J_n), the COVQ should be designed for a channel with a lower level of noise than the actual channel. Of course, a possible improvement to the present system is to jointly design the COVQ encoder and the AMMSE decoder using some training sequence approach. We have not tried this in the present work.

3.1.6 Conclusions

We have presented two formulations for MAP detection of discrete Markov sources over discrete memoryless channels. The first, sequence MAP detection, involves large delay and results in a “Viterbi-like” implementation. The second, instantaneous MAP detection, requires no delay and results in a recursive implementation. For the special case of the binary symmetric Markov source and binary symmetric channel, we have shown that the two MAP detectors are useful for only a certain range of channel bit error rates. The MAP detection problem was applied to a joint source-channel coding system in which the source code is a small-block-size VQ which does not remove all of the source redundancy. The MAP detectors were applied to utilize the residual redundancy at the output of the VQ encoder.

Improvements of the joint source-channel coding system were proposed in the form of instantaneous AMMSE decoding and combined COVQ encoding with AMMSE decoding. For the Gauss-Markov source with correlation coefficient 0.9, the proposed system beats the COVQ with the same (small) blocklength (hence same delay).

3.2 Redundancy in the Form of a Non-Uniform Distribution

3.2.1 Introduction

Source and channel coding techniques are widely employed in many communication systems. In source coding, the objective is to retain the essential information of the source by removing its redundancy. Channel coding, on the other hand, introduces redundancy in order to provide protection against channel noise. When designing channel codes, it is common to assume that the source encoder has removed all of the source redundancy, i.e., the output of the source encoder (input to the channel encoder) has no redundancy. In some practical situations, however, the source encoder fails to entirely eliminate the source redundancy. In these cases, the redundancy remaining at the output of the source encoder — the “residual redundancy” [47] — may be used to combat channel noise [47], [40].

The residual redundancy can take on one of several different forms. Let $\mathcal{U} = \{U_n\}_{n=1}^{\infty}$ be the output of the source encoder. Assume that $\mathcal{U}$ is stationary and ergodic with alphabet size M . Let

$$\begin{aligned} H_1(\mathcal{U}) &\triangleq H(U_1), \\ H_{\infty}(\mathcal{U}) &\triangleq \lim_{n \rightarrow \infty} H(U_n | U_{n-1}, U_{n-2}, \dots, U_1), \end{aligned}$$

where $H(U_1)$ is the entropy of U_1 and $H(U_n | U_{n-1}, U_{n-2}, \dots, U_1)$ is the conditional

entropy of U_n given $(U_{n-1}, U_{n-2}, \dots, U_1)$. Also, let

$$\begin{aligned}\rho_D &\triangleq \log M - H_1(\mathcal{U}), \\ \rho_M &\triangleq H_1(\mathcal{U}) - H_\infty(\mathcal{U}), \\ \rho_T &\triangleq \rho_D + \rho_M = \log M - H_\infty(\mathcal{U}),\end{aligned}$$

be the redundancy due to the distribution, the redundancy due to the memory and the total redundancy, respectively. If $\rho_T = 0$, there is no residual redundancy. If $\rho_M > 0$, we say that there is residual redundancy in the form of memory. If $\rho_D > 0$, we say that there is residual redundancy in the form of a non-uniform distribution. If $\rho_M > 0$ and $\rho_D = 0$, we say that the residual redundancy is strictly in the form of memory. Likewise, if $\rho_D > 0$ and $\rho_M = 0$, we say that the residual redundancy is strictly in the form of a non-uniform distribution.

Residual redundancy in the form of memory had been considered in [47] and in Section 3.1. There, $\mathcal{U}$ was assumed to be a discrete Markov chain which is directly transmitted over a discrete memoryless channel. A maximum a posteriori (MAP) decoder was introduced to exploit the redundancy in $\mathcal{U}$. In the special case where $\mathcal{U}$ is binary with redundancy strictly in the form of memory and the channel is a binary symmetric channel (BSC) with crossover probability ϵ , it was shown in Section 3.1 that sequence MAP detection reduces the sequence error probability if and only if $\rho_M > 1 - H_b(\sqrt{\epsilon}/(\sqrt{\epsilon} + \sqrt{1 - \epsilon}))$, where $H_b(\cdot)$ is the binary entropy function. Furthermore, if the source code is a vector quantizer with a small block size and the source is correlated from block to block, simulation results showed that MAP detection also reduces the average distortion.

In this section, we will consider the case where the residual redundancy is strictly in the form of a non-uniform distribution. We assume that the source is a discrete-time, continuous-valued, memoryless process with a non-uniform distribution and the source code is a Lloyd-Max quantizer (LMQ) [34], [35]. Since the

quantizer operates on each source sample independently from all other samples and since the source samples themselves are independent, the quantizer output are also independent ($\rho_M = 0$). For any source and any scalar quantizer, the quantizer output distribution may be uniform. However, for a LMQ with a non-uniform input, one can argue that, at high rates, the output distribution is also non-uniform. Let Δ_i be the length of the i -th quantization interval and p_i be the probability that the source falls in this interval. It was shown in [26] that, for a LMQ operating at a high rate, $p_i \Delta_i^2$ is constant. For a smooth source probability density function (p.d.f.), $f(\cdot)$, at high rates, $p_i \approx f(q_i) \Delta_i$, where q_i is the i -th quantization level [26]. If the output distribution is uniform (p_i is constant), then Δ_i and $f(q_i)$ are also constant, implying that the source distribution is uniform. Therefore, if the source distribution is non-uniform, then at high rates the LMQ output distribution is also non-uniform ($\rho_D > 0$). Since the effects of channel noise are most damaging at high bit rates [26], we will only consider these cases (rates of three to four bits per sample). For the sources considered in this section, the LMQ output distributions are always non-uniform.

The system to be considered is depicted in Figure 3.10. Here, instead of transmitting the quantizer output directly over the channel, we first encode it by a rate-one convolutional encoder. A rate-one convolutional encoder is one which produces as many bits as it accepts. The purpose of the convolutional encoder is not to introduce additional redundancy but to convert the residual redundancy into a more usable form. When the channel is memoryless and symmetric (as is assumed here), redundancy at the channel input can be better utilized at the decoder if it is strictly in the form of memory rather than strictly in the form of a non-uniform distribution. This is because, for a symmetric memoryless channel, a uniform input distribution maximizes the mutual information between the channel input and output. As stated by Blahut: “if the channel treats all input symbols

the same, then all input symbols should be used equally often” [9]. If the channel input distribution is to be uniform, any redundancy at the channel input should be strictly in the form of memory. Therefore, the function of the convolutional encoder is to convert residual redundancy strictly in the form of a non-uniform distribution into redundancy strictly in the form of memory.

A major focus of this section is the design of the rate-one convolutional encoder. There are several approaches to doing this: (i) design the convolutional encoder so that the redundancy at the channel input is strictly in the form of memory; (ii) design the convolutional encoder to minimize the probability of decoding error (at the output of the MAP decoder) or (iii) design the convolutional encoder to minimize the average distortion of the system in Figure 3.10.

In this section, we will consider the third approach. An upper bound on the average distortion of the proposed system is derived. An approximation of this upper bound is computable and the convolutional encoder is chosen as that which minimizes this approximate upper bound. Simulation results are presented and comparisons are made with the LMQ and the channel-optimized scalar quantizer (COSQ) [30], [15]. Improvements as much as 11.9 dB in signal-to-noise ratio (SNR) over the LMQ and 4.6 dB over the COSQ are found. Additional improvements can be obtained by using a rate- and entropy-constrained scalar quantizer in the proposed system instead of the LMQ. A simulation study of channel mismatch shows that the proposed scheme is robust over a small range of channel bit error rate. Finally, comparisons with a tandem source-channel coding scheme indicate superior performance of the proposed scheme.

The rest of this section is organized as follows. In Subsection 3.2.2, a motivational example is given. In Subsection 3.2.3, we present rate-one convolutional codes and derive an approximate upper bound on the average distortion of the proposed system. Simulation results are presented in Subsection 3.2.4 followed by

conclusions in Subsection 3.2.5.

3.2.2 Motivation

Consider, as an example, two LMQ's [34], [35] of rates $R = 1$ and 4 bits/sample (b/s) designed for a generalized Gaussian source with p.d.f.,

$$f(x) = \left[\frac{\alpha \eta(\alpha, \beta)}{2\Gamma(1/\alpha)} \right] \exp\{-[\eta(\alpha, \beta)|x|]^\alpha\}, \quad (3.37)$$

where $\eta(\alpha, \beta) \triangleq \beta^{-1} \left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)} \right]^{1/2}$, $\alpha > 0$ is the exponential rate of decay and β^2 is the variance of the distribution [15]. We assume that a natural binary code [26] is used to assign binary codewords to the quantization levels of the LMQ. It is assumed that these codewords are transmitted over a BSC with bit error rate (BER) ϵ and, at the decoder, the received codewords are mapped back to quantization levels. Table 3.5 shows the SNR performances (in dB) for $\alpha = 0.5$ and $\epsilon = 0.0$ (noiseless channel) and $\epsilon = 0.005$. For the LMQ with a natural binary code and with $R = 4$ b/s, the performance degradation is 12.20 dB when going from a noiseless channel to a noisy channel with $\epsilon = 0.005$. For $R = 1$ b/s, the degradation is significantly less.

The column designated LMQ-FBC shows the results when a folded binary code [26], [46] is used in place of the natural binary code. The fourth column corresponds to a COSQ. The last column refers to the optimum performance theoretically attainable (OPTA) which is obtained by equating the source rate-distortion function to the channel capacity, where both are measured in bits per source sample. OPTA determines the best possible performance predicted by information theory [9], [8]. It is clear from Table 3.5 that both LMQ-FBC and COSQ perform better than LMQ-NBC when the channel is noisy. However, the degradation of both systems due to channel noise are still quite severe as compared to the drop in OPTA. Consider, for example, the case where $R = 4$ b/s. The performance of COSQ drops by

4.90 dB when ϵ goes from 0.0 to 0.005, while the drop in OPTA is only 0.84 dB.

To improve on the performance of COSQ, one may consider a channel-optimized vector quantizer (COVQ) [29], [16] which is a multi-dimensional extension of COSQ. However, since we are interested in high-rate quantizers where the channel noise problem is most severe, COVQ is viable only for small block sizes. This is due to the COVQ complexity which grows exponentially with the product of the rate and block size. Low-complexity vector quantizers, like tree-structured vector quantizer and multi-stage vector quantizer, which are designed for noisy channels [41] are possible alternatives. However, in this section, we will restrict attention to scalar quantizers though appropriate comparisons will be made with COVQ at a later stage.

Let us consider the symmetric LMQ designed for the generalized Gaussian distribution with parameter $\alpha = 0.5$ at rate 4 b/s. Let $\{p_i\}_{i=0}^{15}$ be the probability distribution of the output of this quantizer. The LMQ output entropy is given by

$$H = - \sum_{i=0}^{15} p_i \log_2 p_i \quad \text{b/s.} \quad (3.38)$$

We have computed this value to be $H = 2.89$ b/s. The capacity of the BSC (assuming it is used four times per source sample) is

$$C = 4[1 + \epsilon \log_2 \epsilon + (1 - \epsilon) \log_2 (1 - \epsilon)] \quad \text{b/s,} \quad (3.39)$$

which corresponds to $C = 3.82$ b/s for $\epsilon = 0.005$. From these simple calculations, we conclude that, for $\alpha = 0.5$, $R = 4$ and $\epsilon = 0.005$,

$$H < C. \quad (3.40)$$

A well known result of information theory says that when $H < C$, it is possible to reliably (i.e., with arbitrarily small probability of error) transmit the LMQ output over the noisy channel [9]. Therefore, it is possible, through proper coding, to

achieve a performance arbitrarily close to the performance of the LMQ operating over a noiseless channel — even if the channel is noisy with $\epsilon = 0.005$. In other words, provided that an LMQ is used and $\alpha = 0.5$, it is possible to have (almost) no performance degradation when ϵ goes from 0.0 to 0.005.

The most straightforward approach to achieving the above situation is to use a suitable noiseless source code (e.g., Huffman code) to compress the LMQ output to a rate approximately equal to H and then use an appropriate channel code (e.g., convolutional code) of rate $\approx H/4$ to “reliably” transmit the compressed information over the channel. The problem with this approach is that most practical noiseless source codes, like the Huffman code, use variable-length codewords. Thus, such a system suffers from the problems inherent in variable-length coding such as buffer under/overflow, error propagation and loss of synchronization.

We propose a fixed-rate system which, in principle, performs the same function as the Huffman code/convolutional code combination. This system is a combined source-channel, rate-one convolutional code, which is described next.

3.2.3 Rate-One Convolutional Codes

Even though the LMQ output has some redundancy in the form of its non-uniform distribution, this type of redundancy is not suitable for providing protection against channel noise.

In this section, we describe a rate-one convolutional code whose function is to “match” the LMQ output to the BSC. The theoretical justification for choosing convolutional code for this task can be found in the works by Koshelev [28] and Hellman [25]. Koshelev investigated the computational complexity of sequential decoders of convolutional codes used in joint source-channel coding of discrete non-equiprobable sources [28]. Hellman proved that there exist convolutional codes of rate $R_c < C/H$ “that can be reliably decoded when used for joint source-channel

coding” provided that MAP decoding is employed [25].

Note that, as a consequence of the results in [25], it is possible to consider convolutional codes of rates even higher than one (assuming that (3.40) holds). However, for simplicity, we will restrict attention to codes of rate one. In what follows, we give an example of how a convolutional code might be used to “match” binary non-equiprobable sources to a BSC.

Example

Let $\{U_n\}_{n=1}^{\infty}$ be an independent and identically distributed (i.i.d.) binary source with $\Pr\{U_n = 0\} = p \in (0.5, 1)$. Suppose this source is to be transmitted over a BSC with BER $\epsilon \in (0, 0.5)$. The channel is used once for every source sample. We consider the following two scenarios: (i) $\{U_n\}$ is directly transmitted over the channel and (ii) $\{U_n\}$ is encoded by a rate-one convolutional encoder, transmitted over the channel and then decoded using a MAP decoder.

(i) Here we assume that $\{U_n\}$ is directly transmitted over the channel and we receive the sequence $\{V_n\}_{n=1}^{\infty}$. Based upon a block of N observations, $\mathbf{v} \triangleq (v_1, v_2, \dots, v_N)$, we make the MAP estimate

$$\begin{aligned}\hat{\mathbf{u}} &= \arg \max_{\mathbf{u}} \Pr\{\mathbf{U} = \mathbf{u} | \mathbf{V} = \mathbf{v}\} \\ &= \arg \max_{\mathbf{u}} \Pr\{\mathbf{V} = \mathbf{v} | \mathbf{U} = \mathbf{u}\} \Pr\{\mathbf{U} = \mathbf{u}\}.\end{aligned}\tag{3.41}$$

Since the source and channel are memoryless,

$$\hat{\mathbf{u}} = \arg \max_{\mathbf{u}} \prod_{n=1}^N \Pr\{V_n = v_n | U_n = u_n\} \Pr\{U_n = u_n\}.\tag{3.42}$$

The above product is maximized if each individual term is maximized. Thus $\hat{\mathbf{u}} = (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_N)$, where for each $n \in \{1, 2, \dots, N\}$,

$$\hat{u}_n = \arg \max_{u_n} \Pr\{V_n = v_n | U_n = u_n\} \Pr\{U_n = u_n\}.\tag{3.43}$$

If $v_n = 0$, we choose $\hat{u}_n = 0$ if $p(1 - \epsilon) > (1 - p)\epsilon$. Since $p \in (0.5, 1)$ and $\epsilon \in (0, 0.5)$, this inequality always holds; so one always decodes a “0” if a “0” is received. On the other hand, if $v_n = 1$, we choose $\hat{u}_n = 0$ if $p\epsilon > (1 - p)(1 - \epsilon)$ or, equivalently, if $\epsilon > (1 - p)$. Otherwise (if $\epsilon \leq (1 - p)$), we choose $\hat{u}_n = v_n = 1$.

In summary, if $\epsilon > (1 - p)$, then $\hat{\mathbf{u}} = (0, 0, \dots, 0)$. Otherwise, $\hat{\mathbf{u}} = \mathbf{v}$. The probability of decoding error for this case is

$$\Pr\{\hat{U}_n \neq U_n\} = \min(\epsilon, 1 - p). \quad (3.44)$$

Note that, in this case, the source distribution (which is non-uniform) is not “well-matched” to the channel.

(ii) In the second case, through the use of a rate-one convolutional encoder (with feedback), we form the sequence

$$W_n = U_n \oplus W_{n-1}, \quad n = 1, 2, \dots, \quad (3.45)$$

where $\oplus$ is the binary exclusive-or operation and W_0 is the initial state of the convolutional encoder. Note that $\{W_n\}$ is a discrete Markov chain and for a given W_0 , there is a one-to-one relationship between the sequences $\{U_n\}_{n=1}^\infty$ and $\{W_n\}_{n=1}^\infty$.

Given p and $q_0 \triangleq \Pr\{W_0 = 0\}$, the quantity $q_n \triangleq \Pr\{W_n = 0\}$ is easily shown to be

$$q_n = \frac{1}{2} + (q_0 - \frac{1}{2})(2p - 1)^n, \quad (3.46)$$

implying that $\lim_{n \rightarrow \infty} q_n = 1/2$. That is, for large n , the distribution of W_n approaches that which maximizes the mutual information between the channel input and output. Loosely speaking, $\{W_n\}$ is asymptotically “well-matched” to the channel. Note that, for $q_0 = \frac{1}{2}$, $q_n = \frac{1}{2}$ for all $n \geq 1$ and therefore, $\{W_n\}$ is stationary.

Evaluating the probability of error for all values of $p \in (0.5, 1)$ and $\epsilon \in (0, 0.5)$ is not simple in this case. Instead, we consider a specific example with $p = 0.95$,

$\epsilon = 0.04$ and $N = 20$. A typical source sequence will look like

$$\mathbf{u} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0). \quad (3.47)$$

Assuming that $W_0 = 0$, this corresponds to

$$\mathbf{w} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1). \quad (3.48)$$

Assuming that MAP decoding is employed, single bit errors — except those occurring in components 9, 10, 11, 12 or 20 — will be corrected. A single error occurring in positions 10, 11 or 20 will not be corrected, while a single error occurring in positions 9 or 12 will be decoded incorrectly with probability $1/2$. In this case, the probability of correct block decoding is bounded by

$$\Pr\{\hat{\mathbf{U}} = \mathbf{U}\} \geq (1 - \epsilon)^{20} + 16\epsilon(1 - \epsilon)^{19} = 0.737. \quad (3.49)$$

This implies that the block probability of error is bounded by $\Pr\{\hat{\mathbf{U}} \neq \mathbf{U}\} \leq 0.263$. Comparing this bound to the block probability of error for case (i)

$$\Pr\{\hat{\mathbf{U}} \neq \mathbf{U}\} = 1 - (1 - \epsilon)^{20} = 0.558, \quad (3.50)$$

we see that, for this specific example, the block probability of error can be reduced using a trivial rate-one convolutional code.

System Description

In the following, we will consider the system depicted in Figure 3.10. The source, $\{X_n\}$, is assumed to be an i.i.d. process with p.d.f. $f(x)$. The p.d.f. is assumed to be symmetric with a sharp peak. The source is encoded by a symmetric LMQ of rate R b/s. The LMQ encoder is described by a set of $M - 1$ quantization thresholds, $\{T_1, T_2, \dots, T_{M-1}\}$, where $M = 2^R$. The output of the LMQ encoder is a sequence of binary R -tuples (R is an integer). Assume that the natural binary

code is used, let $\mathcal{J} \triangleq \{0, 1, \dots, M-1\}$ and $\gamma : \mathcal{J} \mapsto \{0, 1\}^R$ be the binary representation of the integers in $\mathcal{J}$. The encoder is then described by

$$U_n = \gamma(i) \quad \text{if } X_n \in (T_i, T_{i+1}], \quad i = 0, 1, \dots, M-1, \quad (3.51)$$

where it is assumed that $T_0 = -\infty, T_M = \infty$, and $T_i < T_{i+1}$ for $i = 0, 1, \dots, M-1$. Since the LMQ is assumed to be symmetric, $T_i = -T_{M-i}$ for $i \in \mathcal{J}$. The output of the LMQ encoder is an index, $U_n \in \{0, 1\}^R$, representing the quantization level for X_n . $\{U_n\}$ is passed through a rate-one convolutional encoder which produces $\{W_n\}$, which, in turn, is fed into a BSC, the output of which is denoted by $\{V_n\}$. At the receiver, we make a MAP estimate, $\hat{\mathbf{U}} = \{\hat{U}_n\}_{n=1}^N$, of the sequence $\mathbf{U} = \{U_n\}_{n=1}^N$ based upon the observation $\mathbf{V} = \{V_n\}_{n=1}^N$. Then, $\hat{\mathbf{U}}$ is fed to the LMQ decoder; the output of the decoder is an estimate, $\hat{X}_n$, of X_n given by

$$\hat{X}_n = q_{\gamma^{-1}(\hat{U}_n)}, \quad (3.52)$$

where q_i is the i -th reconstruction level of the quantizer ($i \in \mathcal{J}$). We will use the block notation: $\mathbf{U} = g(\mathbf{X})$, where $\mathbf{X} = (X_1, X_2, \dots, X_N)$, $\mathbf{W} = h(\mathbf{U})$, $\hat{\mathbf{U}} = \eta(\mathbf{V})$ and $\hat{\mathbf{X}} = (\hat{X}_1, \hat{X}_2, \dots, \hat{X}_N) = \delta(\hat{\mathbf{U}})$.

Note that the sequence $\{W_n\}_{n=1}^\infty$ is a discrete hidden Markov chain. Therefore, the MAP decoder in Figure 3.10 can be implemented using a slight modification of the Viterbi algorithm (see [19], [40]).

Distortion Upper Bound

Traditionally, convolutional codes are designed to minimize probability of error (or, equivalently, maximize free distance). Our aim, however, is to design codes which minimize the squared-error distortion,

$$D = E\left[\frac{1}{N} \sum_{n=1}^N (X_n - \hat{X}_n)^2\right] = \frac{1}{N} \sum_{n=1}^N E[(X_n - \hat{X}_n)^2]. \quad (3.53)$$

Let $\tilde{X}_n \triangleq q_{\gamma^{-1}(U_n)}$, i.e., $\tilde{X}_n$ is the reconstruction sample given that $\hat{U}_n = U_n$. Then

$$\begin{aligned} E[(X_n - \hat{X}_n)^2] &= E[(X_n - \tilde{X}_n + \tilde{X}_n - \hat{X}_n)^2], \\ &= E[(X_n - \tilde{X}_n)^2] + E[(\tilde{X}_n - \hat{X}_n)^2] \\ &\quad + 2E[(X_n - \tilde{X}_n)(\tilde{X}_n - \hat{X}_n)]. \end{aligned} \quad (3.54)$$

Since the LMQ satisfies the centroid condition, the last term in (3.54) is zero [26].

Hence

$$D = E\left[\frac{1}{N} \sum_{n=1}^N (X_n - \tilde{X}_n)^2\right] + E\left[\frac{1}{N} \sum_{n=1}^N (\tilde{X}_n - \hat{X}_n)^2\right]. \quad (3.55)$$

The first term in the above, denoted by D_s , is the average distortion induced by the source coding (quantization) operation while the second term, denoted by D_c , is the distortion introduced by channel noise.

Our goal is to choose a convolutional code which minimizes the average distortion D . Since D_s is independent of the choice of the code, this is equivalent to choosing a code which minimizes D_c . Evaluating D_c is not straightforward. Instead, we compute an approximate upper bound on D_c and choose that code which minimizes this approximate upper bound. The hope here is that a solution which minimizes the upper bound also minimizes D_c , or, at least, provides a “good” code.

Let $\mathcal{Q} \triangleq \{q_0, q_1, \dots, q_{M-1}\}$ be the set of reconstruction levels. Define

$$p_i \triangleq \Pr\{\tilde{X}_n = q_i\} = \Pr\{X_n \in (T_i, T_{i+1}]\}, \quad i \in \mathcal{J}. \quad (3.56)$$

By symmetry, we have $q_i = -q_{M-1-i}$ and $p_i = p_{M-1-i}$ for $i \in \mathcal{J}$. Since the source distribution is assumed to be sharp-peaked and since we are considering LMQ, reconstruction levels closer to the origin are more probable than those on the outskirts. We thus assume that for $i = 0, 1, \dots, M/2 - 2$, $p_i < p_{i+1}$.

Let $\tilde{\mathbf{X}} = (\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_N)$. Note that there is a one-to-one correspondence between $\tilde{\mathbf{X}} \in \mathbb{R}^N$ and $\mathbf{U} \in \{0, 1\}^{RN}$. We are interested in upper bounds for D_c given the “type” of $\tilde{\mathbf{X}}$ and the number of bit errors.

Suppose $\tilde{\mathbf{X}} \in \{q_{M/2-1}, q_{M/2}\}^N \triangleq \mathcal{T}_0$, i.e., each element of $\tilde{\mathbf{X}}$ is either $q_{M/2-1}$ or $q_{M/2}$. Such a sequence is called a *type-0* sequence and

$$\Pr\{\tilde{\mathbf{X}} \in \mathcal{T}_0\} = 2^N p_{M/2-1}^N \triangleq Q_0. \quad (3.57)$$

Type-0 sequences are the most probable. If $\mathbf{z} \in \mathcal{T}_0$, then $\Pr\{\tilde{\mathbf{X}} = \mathbf{z}\} = p_{M/2-1}^N \triangleq P_0$. Now define a *type-1* sequence as one in which exactly $N-1$ elements are either $q_{M/2-1}$ or $q_{M/2}$ and the other element is either $q_{M/2-2}$ or $q_{M/2+1}$. The set of all type-1 sequences is denoted by $\mathcal{T}_1$ and

$$\Pr\{\tilde{\mathbf{X}} \in \mathcal{T}_1\} = N 2^N p_{M/2-1}^{N-1} p_{M/2-2} \triangleq Q_1. \quad (3.58)$$

Type-1 sequences are the second most probable. If $\mathbf{z} \in \mathcal{T}_1$, then $\Pr\{\tilde{\mathbf{X}} = \mathbf{z}\} = p_{M/2-1}^{N-1} p_{M/2-2} \triangleq P_1$. Continuing this way, we get a sequence of types $\{\mathcal{T}_i\}_{i=0}^L$ and two sequences of probabilities, $\{Q_i\}_{i=0}^L$ and $\{P_i\}_{i=0}^L$ where $\cup_{i=0}^L \mathcal{T}_i = \mathcal{Q}^N$, $\sum_{i=0}^L Q_i = 1$ and $P_i \geq P_{i+1}$ for $i = 0, 1, \dots, L-1$.

Let $\mathbf{W}$ and $\mathbf{V}$ be the transmitted and received bit sequences, respectively, and let $d_H(\mathbf{W}, \mathbf{V})$ be the number of errors occurred during transmission. Define

$$D_{i,j} \triangleq E \left[\frac{1}{N} \|\tilde{\mathbf{X}} - \hat{\mathbf{X}}\|^2 \mid \tilde{\mathbf{X}} \in \mathcal{T}_i, d_H(\mathbf{W}, \mathbf{V}) = j \right], \quad (3.59)$$

$i = 0, 1, \dots, L$, $j = 0, 1, \dots, RN$. Here, the natural binary code and the rate-one convolutional encoder provide a one-to-one relationship between $\tilde{\mathbf{X}}$ and $\mathbf{W}$ ($\mathbf{W} = h(g(\tilde{\mathbf{X}}))$). D_c can be computed according to

$$D_c = \sum_{i=0}^L \sum_{j=0}^{RN} D_{i,j} \Pr\{\tilde{\mathbf{X}} \in \mathcal{T}_i, d_H(\mathbf{W}, \mathbf{V}) = j\}, \quad (3.60)$$

where

$$\Pr\{\tilde{\mathbf{X}} \in \mathcal{T}_i, d_H(\mathbf{W}, \mathbf{V}) = j\} = Q_i \binom{RN}{j} \epsilon^j (1 - \epsilon)^{RN-j}, \quad (3.61)$$

assuming that the source and channel are independent and ϵ is the channel BER.

In order to derive an upper bound on $D_{i,j}$, we first describe how the MAP detector operates. Define the *a posteriori probability* (APP) of a sequence $\mathbf{u} \in \{0,1\}^{RN}$ given that a sequence $\mathbf{v}$ is received as

$$APP(\mathbf{u}|\mathbf{v}) \triangleq \Pr\{\mathbf{U} = \mathbf{u}|\mathbf{V} = \mathbf{v}\}. \quad (3.62)$$

Then the output of the MAP detector can be related to the input by

$$\hat{\mathbf{u}} = \eta(\mathbf{v}) = \arg \max_{\mathbf{u} \in \{0,1\}^{RN}} APP(\mathbf{u}|\mathbf{v}). \quad (3.63)$$

Let $\hat{\mathbf{x}} = \delta(\hat{\mathbf{u}}) = (q_{\gamma^{-1}(\hat{u}_1)}, q_{\gamma^{-1}(\hat{u}_2)}, \dots, q_{\gamma^{-1}(\hat{u}_N)})$. Then, $D_{i,j}$ can be upper bounded by

$$D_{i,j} \leq D_{i,j}^u \triangleq \max_{\substack{\tilde{\mathbf{x}} \in \mathcal{T}_i \\ \mathbf{v} : d_H(h(g(\tilde{\mathbf{x}})), \mathbf{v}) = j}} \frac{1}{N} \|\tilde{\mathbf{x}} - \delta(\eta(\mathbf{v}))\|^2. \quad (3.64)$$

Hence, the conditional expectation in (3.59) is bounded by the conditional maximization. If the above upper bound can be determined, an upper bound on D_c can be computed by

$$D_c^u \triangleq \sum_{i=0}^L \sum_{j=0}^{RN} D_{i,j}^u Q_i \binom{RN}{j} \epsilon^j (1-\epsilon)^{RN-j}, \quad (3.65)$$

which leads to an upper bound on D given by $D^u \triangleq D_s + D_c^u$. Note that the maximization in (3.64) can be written as

$$D_{i,j}^u = \max_{\tilde{\mathbf{x}} \in \mathcal{T}_i} D_j^u(\tilde{\mathbf{x}}), \quad (3.66)$$

where

$$D_j^u(\tilde{\mathbf{x}}) = \max_{\mathbf{v} : d_H(h(g(\tilde{\mathbf{x}})), \mathbf{v}) = j} \frac{1}{N} \|\tilde{\mathbf{x}} - \delta(\eta(\mathbf{v}))\|^2. \quad (3.67)$$

In the following, we show that when $i = 0$ the maximization in (3.66) is unnecessary.

Theorem 3 $D_j^u(\tilde{\mathbf{x}})$ is constant on $\mathcal{T}_0$, i.e.,

$$D_j^u(\tilde{\mathbf{x}}_1) = D_j^u(\tilde{\mathbf{x}}_2) \quad \forall \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \in \mathcal{T}_0, \forall j = 0, 1, \dots, RN. \quad (3.68)$$

The proof of the theorem is provided in Appendix A. In the following, we provide an algorithmic approach for computing an approximation of $D_j^u(\tilde{\mathbf{x}})$ denoted as $D_j^*(\tilde{\mathbf{x}})$. The algorithm is based on the idea that if we transmit the sequence $h(\mathbf{u})$, make j errors and receive $h(\mathbf{z})$ then we decode the sequence $\mathbf{z}$ if $APP(\mathbf{z}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z}))$. We then show that $D_j^*(\tilde{\mathbf{x}})$ upper bounds $D_j^u(\tilde{\mathbf{x}})$.

Algorithm:

(1) Set $D_l^*(\tilde{\mathbf{x}}) = 0$ for $l = 0, 1, \dots, RN$. Set $l = 0$.

(2) $l = l + 1$.

(3) Let $\mathbf{u} = g(\tilde{\mathbf{x}})$, define

$$\mathcal{Z}^l \triangleq \left\{ \mathbf{z} \in \{0, 1\}^{RN} : d_H(h(\mathbf{u}), h(\mathbf{z})) = l, APP(\mathbf{z}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z})) \right\}, \quad (3.69)$$

and set

$$D_l^*(\tilde{\mathbf{x}}) = \max_{\mathbf{z} \in \mathcal{Z}^l} \frac{1}{N} \|\tilde{\mathbf{x}} - \delta(\mathbf{z})\|^2. \quad (3.70)$$

(4) $\forall \mathbf{z} \in \mathcal{Z}^l$, let j^* be the minimum j for which

$$\Pr\{\mathbf{U} = \mathbf{z}\} \epsilon^{l-j} (1 - \epsilon)^{RN-(l-j)} \geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^j (1 - \epsilon)^{RN-j}, \quad (3.71)$$

and set

$$D_{j^*}^*(\tilde{\mathbf{x}}) = \max(D_{j^*}^*(\tilde{\mathbf{x}}), \frac{1}{N} \|\tilde{\mathbf{x}} - \delta(\mathbf{z})\|^2). \quad (3.72)$$

(5) If $l < RN$, go to step (2).

(6) For $l = 0, 1, \dots, RN$, let $D_l^*(\tilde{\mathbf{x}}) = \max(D_0^*(\tilde{\mathbf{x}}), D_1^*(\tilde{\mathbf{x}}), \dots, D_l^*(\tilde{\mathbf{x}}))$.

(7) Stop.

Note that in step (3) we assume that if we receive $h(\mathbf{z})$ and $APP(\mathbf{z}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z}))$ then we decode $\mathbf{z}$. In step (4), we note that only j^* errors need to

occur for $APP(\mathbf{z}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z}))$. Finally, in step (6), we force $D_j^*(\tilde{\mathbf{x}})$ to be monotonically non-decreasing. The following theorem establishes that $D_j^*(\tilde{\mathbf{x}})$ upper bounds $D_j^u(\tilde{\mathbf{x}})$.

Theorem 4 $D_j^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$ for $j = 0, 1, \dots, RN$.

The proof of this theorem is given in Appendix B. The above provides a yet looser upper bound on the conditional average distortion. This new upper bound is computable. $D_j^*(\tilde{\mathbf{x}})$ in step (3) can be computed using a small modification of the algorithm described in [6], [31]. Note that the proof of Theorem 3 can be modified to show that $D_j^*(\tilde{\mathbf{x}})$ is also constant on $\mathcal{T}_0$.

Approximations

Even though $D_j^*(\tilde{\mathbf{x}})$ is not constant on $\mathcal{T}_i$ for $i > 0$, we have observed that, in most cases, it does not vary much on $\mathcal{T}_i$. We thus approximate the maximum of $D_j^*(\tilde{\mathbf{x}})$ over the set $\mathcal{T}_i$ by $D_{i,j}^*$ for some randomly chosen $\tilde{\mathbf{x}} \in \mathcal{T}_i$. Also, we note that $D_{i,j}$ is bounded (by the maximum distance between any two elements in $\mathcal{Q}$) while $\Pr\{\tilde{\mathbf{X}} \in \mathcal{T}_i, d_H(\mathbf{W}, \mathbf{V}) = j\}$ vanishes for large i and j . Therefore, we approximate the upper bound on D_c (see (3.65)) by

$$D_c^* = \sum_{i=0}^{i_{max}} \sum_{j=0}^{j_{max}} D_{i,j}^* Q_i \left(\binom{RN}{j} \epsilon^j (1-\epsilon)^{RN-j} \right), \quad (3.73)$$

where $D_{i,j}^* = D_j^*(\tilde{\mathbf{x}})$ for some randomly chosen $\tilde{\mathbf{x}} \in \mathcal{T}_i$ and $i_{max} < L$, $j_{max} < RN$ are chosen such that $\Pr\{\tilde{\mathbf{X}} \in \cup_{i=i_{max}+1}^L \mathcal{T}_i\}$ and $\Pr\{d_H(\mathbf{W}, \mathbf{V}) > j_{max}\}$ are sufficiently small⁸.

Finally, we perform the above calculation for only small values of N ($N \leq 10$) and we assume that a trailer of length N_T is provided at the decoder. That is,

⁸As an example, we chose i_{max} so that $\Pr\{\tilde{\mathbf{X}} \in \cup_{i=i_{max}+1}^L \mathcal{T}_i\} \approx 0.02$ and j_{max} so that $\Pr\{d_H(\mathbf{W}, \mathbf{V}) > j_{max}\} \approx 0.0002$. These values vary depending how close an approximation we want to make and how much complexity we can tolerate.

we assume that $(V_1, V_2, \dots, V_{N+N_T})$ and $(U_{N+1}, U_{N+2}, \dots, U_{N+N_T})$ are given even though we are only interested in decoding $(U_1, U_2, \dots, U_N)$.

Search for Good Codes

The above development provides an (approximate) upper bound on the average distortion for a given convolutional encoder. To find that encoder which maximizes the (approximate) upper bound, the following procedure is adopted. First, an encoder is chosen at random by randomly selecting the tap coefficients. The upper bound for this code is computed. A new encoder is chosen by randomly changing one of the tap coefficients. If this results in a smaller upper bound, it is recorded as “the best code found thus far”. The encoder is again randomly changed and the procedure is repeated. We stop the algorithm if no improvement is found after T changes.

A list of good codes found by the above procedure is provided in Table 3.6. These codes were found after searching through 200-400 codes (we set $T = 100$). Here, we consider two types of encoders: minimal encoders and encoders with feedback. For minimal encoders, the tap coefficients are listed where we assume that the least significant bits are fed into the encoder first. In the case of feedback, we used the structure of Figure 3.11 [19], where all additions and multiplications are over $GF(2^R)$. We used the primitive polynomials $p(X) = 1 + X + X^3$ and $p(X) = 1 + X + X^4$ to construct $GF(2^3)$ and $GF(2^4)$, respectively [32]. The tap coefficients of the forward loop, $\phi(z)$, are given in the first row while the tap coefficients of the feedback loop, $\varphi(z)$, are given in the second row. For both minimal and feedback encoders, the coefficients are given in octal for $R = 3$ and hexadecimal for $R = 4$.

3.2.4 Simulation Results

Simulations of the system in Figure 3.10 for a Generalized Gaussian source with $\alpha = 0.5$ have been performed and the results are given in Table 3.7. The simulations were performed on 1000 source samples and each simulation was repeated 100 times. The average distortion, averaged over the 100 trials, is given as SNR in dB. Here, we used the convolutional encoders listed in Table 3.6. Performances of LMQ-NBC and COSQ and the theoretical bound, OPTA, are also given in Table 3.7. The LMQ in the proposed system is constrained to be symmetric while LMQ-NBC and COSQ are not. Hence, the proposed system is inferior to LMQ-NBC and COSQ by 0.05 to 0.1 dB when the channel is noiseless. However, for noisy channels, in almost all cases the convolutionally encoded system beats the COSQ.

Other Decoding Techniques

We note that all of the delay of the system in Figure 3.10 is in the MAP decoder. The MAP decoder must wait for the entire sequence to arrive before it makes a decision on which sequence was transmitted. In applications such as two-way communications, such large delay is not allowed. A straightforward method of reducing the delay is to allow the Viterbi decoder to release symbols early [32]. This method, however, is sub-optimum. In the following, we will consider two optimal decoders.

We first consider optimum (MAP) symbol-by-symbol decoding for a given delay τ . Suppose we are interested in decoding u_n given that we have observed $\mathbf{v}_1^{n+\tau} = (v_1, v_2, \dots, v_{n+\tau})$. The minimum probability of error decoding rule is given by:

$$\hat{u}_n = \arg \max_{u_n} \Pr\{U_n = u_n | \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}\}. \quad (3.74)$$

Note that the above minimizes the symbol error probability whereas the MAP

Viterbi decoder minimizes the sequence (or block) error probability. Implementation of this decoder is described in Appendix C.

Note that similar to hard-decision convolutional decoding, there is an irreversible loss of information between the MAP decoder and the LMQ decoder. To remedy this problem, we combine these two decoders into a single minimum mean-squared error (MMSE) decoder. The MMSE estimate of $\mathbf{X}_n$ given that $\mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}$ is

$$\hat{\mathbf{x}}_n = E[\mathbf{X}_n | \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}], \quad (3.75)$$

which can be expanded to

$$\hat{\mathbf{x}}_n = \sum_{u_n} E[\mathbf{X}_n | U_n = u_n, \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}] \Pr\{U_n = u_n | \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}\}. \quad (3.76)$$

Since the source is memoryless, the expectation in (3.76) is just $q_{\gamma^{-1}(u_n)}$. The second factor in the above summation can be obtained from the symbol-by-symbol decoder. Simulation results using MAP symbol-by-symbol and MMSE decoding with $\tau = 20$ samples are given in Table 3.8. In all cases, MMSE decoding is superior to MAP symbol-by-symbol decoding. In Figure 3.12, we plot the SNR performance as a function of the delay for the case $\alpha = 0.5$, $R = 3$, $\epsilon = 0.005$ and constraint length $K = 2$.

Based on the results obtained thus far, we have concluded that when the system is expected to perform well, i.e., when ϵ is small, constraint length is large ($K = 2$) and/or when delay is large, encoders with feedback are superior to minimal encoders. On the other hand, when the system is expected to perform poorly, minimal encoders are superior. The reason for this is due to the error-propagating nature of the feedback encoders. Hellman [25] had suggested that error-propagating codes can be useful when there is natural redundancy in the source. He provided an example of a rate-one error-propagating code in which a single channel error causes the message (written in English) to appear meaningless. He showed a simple de-

coding strategy which correct the error to yield a sensible message. Under ideal situations (when we expect the system to work well), most errors will be corrected by the decoder since, otherwise, the decoded message will appear meaningless. When the situation is far from ideal, the code propagates more errors than the decoder is capable of correcting and it would be preferable to choose a minimal encoder or no encoder at all.

Entropy-Constrained Scalar Quantizer

We have repeated the simulations for a Generalized Gaussian source with $\alpha = 1.0$, i.e., a Laplacian source. The results are given in Table 3.9. The codes that have been found for $\alpha = 1.0$ are given in Table 3.10. In this case, the optimal LMQ is symmetric. We note that the results of the convolutionally encoded system for $\alpha = 1.0$ are not as good as those for $\alpha = 0.5$. In fact, for many cases, the SNR performances of the convolutionally encoded system are even worse than that of LMQ-NBC.

The poor performance of the convolutionally encoded system for the Laplacian source ($\alpha = 1.0$) can be attributed to the fact that the distribution with $\alpha = 1.0$ is more uniform than that with $\alpha = 0.5$ and hence the amount of redundancy at the output of the LMQ is less when $\alpha = 1.0$ than when $\alpha = 0.5$. It is possible to increase this redundancy by changing the quantization levels of the quantizer. This may reduce the SNR performance when there is no channel noise. However, the increase in redundancy will lead to additional protection when there is noise.

Here, instead of LMQ, we consider rate- and entropy-constrained symmetric scalar quantizers. These are quantizers in which both the number of levels and the output entropy are constrained. We used a generalized Lloyd algorithm, described in [14], to design these quantizers. The redundancy of a quantizer is defined to be the difference between its rate and its entropy. In Figure 3.13, we plot the SNR

performances of the convolutionally encoded system for $\alpha = 1.0$ using rate- and entropy-constrained scalar quantizers and MAP Viterbi decoding as a function of the quantizer redundancy. Here, we used the feedback codes listed in Table 3.6 since these were found to be “good” when there is high redundancy. One can see that, when the channel is noiseless, the performance decreases as redundancy increases. When the channel is noisy, the performance initially increases, but then it gradually decreases. Basically, as redundancy increases, quantization accuracy is sacrificed for more protection against channel noise. The points at which the performance curves peak is the optimum trade off point. These peak performances are given in Table 3.11 where we also show the peak performances for other sets of parameters. In all cases, we used the feedback codes listed in Table 3.6. We note that it is possible to improve on the performances listed in Table 3.11 by redesigning the convolutional codes using the new quantizers. However, we have not attempted to do this.

Channel Mismatch

Up to this point, we have been working under the assumption that the channel BER is known. In most practical situations, the BER is not known *a priori* or it is time-varying. In this subsection, we will examine the performance of the convolutionally encoded system under channel mismatch, i.e., when the system is designed for a BER ϵ_d , but applied to a channel whose BER is actually ϵ_a . We consider the case $\alpha = 0.5, R = 4, K = 2$. The simulation results for this case is plotted in Figure 3.14, where we have chosen the entropy-constrained scalar quantizers which yielded the peak performance reported in Table 3.11. We also provided in this figure the performance curves for COSQ matched to the given channel and LMQ-NBC.

Comparisons with COVQ

In Table 3.12, we compare the performance of the proposed system against that of a COVQ [16] with block size k . Due to constraints on complexity, we have limited the comparisons to $k = 1, 2$ and 3. Results for $k = 1$ are obtained from [15]. Note that, for $R = 3$ and $k = 2$ or 3, COVQ beats the proposed system both in noiseless and noisy channels. For $R = 4$, we have only obtained COVQ results for $k = 1$ and $k = 2$. In this case, the proposed system outperforms COVQ when the channel is noisy. Finally, we note that in the COVQ system, most of the complexity are in the encoder while in the proposed system most of the complexity are in the decoder.

Comparisons with Tandem Source-Channel Coding

In this subsection, we compare the proposed system, which, in a sense, is a joint source-channel coding scheme, against a more traditional tandem source-channel coding scheme where the source and channel codes are designed separately. The tandem scheme to be considered consists of a LMQ of rate R_s b/s, a convolutional encoder of rate $R_c = R_s/R$, where R is the overall rate in b/s, and a maximum likelihood (ML) decoder implemented using the Viterbi algorithm [19], [32]. For a fixed R , a large value of R_s implies that a large percentage of the transmitted bits are used for source coding and the remaining few are used for channel error protection.

In Figure 3.15, we compare the SNR performance of the proposed scheme with that of the tandem scheme for $\alpha = 0.5$. Here, the overall rate of both systems is $R = 4$ b/s and the convolutional encoders of both systems consist of 256 states (thus, the complexities of both systems are approximately the same). For the tandem scheme, we consider the cases $R_s = 3$ b/s which corresponds to a rate-3/4

convolutional code and $R_s = 2$ b/s which corresponds to a rate-1/2 convolutional code. Here, we used the rate-3/4 and rate-1/2 convolutional codes listed in [32]. For the proposed system, the (rate- and entropy-constrained) scalar quantizer and the MAP decoder are designed for $\epsilon_d = 0.01$ and $\epsilon_d = 0.05$. Note that the proposed scheme with $\epsilon_d = 0.01$ outperforms the tandem scheme with $R_s = 3$ b/s while the proposed scheme with $\epsilon_d = 0.05$ beats the tandem scheme with $R_s = 2$ b/s⁹.

3.2.5 Conclusions

We have proposed the use of a rate-one convolutional code to exploit the residual redundancy of the LMQ to combat channel errors. We derived an approximate upper bound on the average distortion of the proposed system and searched for those codes which minimize the approximate upper bound. Even though its performance is still far from OPTA, the proposed system provides significant improvements in SNR over LMQ-NBC, COSQ and the tandem scheme considered. Future research effort might be concentrated on finding better rate-one convolutional codes. Finally, we note that the proposed system is suitable for applications where the encoder complexity is limited.

⁹We note, however, that in the simulation results presented in Figure 3.15, the MAP decoder of the proposed scheme is implemented using floating point arithmetic while the ML decoder of the tandem scheme is implemented using fixed point arithmetic.

$\log_{10} \epsilon$	$k = 1; R = 1.0$				
	VQ	VQ+ Inst. MAP	VQ+ Seq. MAP	COVQ [16]	VQ+ Inst. AMMSE
$-\infty$	4.38	4.38	4.38	4.40	4.38
-3.0	4.35	4.35	4.35	—	4.37
-2.9	4.35	4.35	4.35	—	4.37
-2.8	4.34	4.34	4.34	—	4.36
-2.7	4.32	4.32	4.32	—	4.36
-2.6	4.31	4.31	4.31	—	4.35
-2.5	4.29	4.29	4.29	—	4.34
-2.4	4.27	4.27	4.27	—	4.33
-2.3	4.24	4.24	4.24	4.25	4.32
-2.2	4.20	4.20	4.20	—	4.30
-2.1	4.15	4.15	4.15	—	4.27
-2.0	4.09	4.09	4.09	4.11	4.25
-1.9	4.02	4.02	4.02	—	4.21
-1.8	3.93	3.93	3.93	—	4.16
-1.7	3.82	3.82	3.82	—	4.11
-1.6	3.69	4.68	3.68	—	4.04
-1.5	3.52	3.52	3.97	—	3.96
-1.4	3.32	3.32	3.90	—	3.86
-1.3	3.09	3.08	3.82	3.15	3.73
-1.2	2.80	2.80	3.70	—	3.59
-1.1	2.47	2.48	3.53	—	3.42
-1.0	2.09	2.09	3.29	2.27	3.20
-0.9	1.65	1.65	2.96	—	2.94
-0.8	1.16	1.16	2.92	—	2.61
-0.7	0.61	1.48	2.44	—	2.20
-0.6	-0.00	0.93	1.96	—	1.69
-0.5	-0.67	0.05	1.07	—	1.06

Table 3.1: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

$\log_{10} \epsilon$	$k = 1; R = 2.0$					$k = 1; R = 3.0$				
	VQ	VQ+ Inst. MAP	VQ+ Seq. MAP	COVQ [16]	VQ+ Inst. AMMSE	VQ	VQ+ Inst. MAP	VQ+ Seq. MAP	COVQ [16]	VQ+ Inst. AMMSE
$-\infty$	9.27	9.27	9.27	9.30	9.27	14.53	14.53	14.53	14.62	14.53
-3.0	9.09	9.11	9.19	—	9.18	13.75	14.08	14.30	—	14.22
-2.9	9.05	9.07	9.17	—	9.16	13.57	13.97	14.25	—	14.15
-2.8	8.99	9.02	9.14	—	9.13	13.36	13.87	14.18	—	14.07
-2.7	8.92	8.95	9.11	—	9.10	13.09	13.77	14.11	—	13.98
-2.6	8.84	8.87	9.07	—	9.07	12.79	13.60	14.04	—	13.86
-2.5	8.73	8.78	9.02	—	9.03	12.44	13.41	13.95	—	13.72
-2.4	8.60	8.77	8.96	—	8.98	12.03	13.18	13.83	—	13.56
-2.3	8.44	8.67	8.92	8.52	8.92	11.56	12.91	13.69	12.04	13.38
-2.2	8.25	8.54	8.84	—	8.85	11.04	12.57	13.51	—	13.15
-2.1	8.03	8.51	8.77	—	8.77	10.46	12.20	13.29	—	12.89
-2.0	7.75	8.36	8.67	7.88	8.67	9.83	11.86	13.03	10.50	12.60
-1.9	7.44	8.22	8.56	—	8.54	9.14	11.49	12.72	—	12.25
-1.8	7.07	8.02	8.41	—	8.39	8.41	10.96	12.43	—	11.86
-1.7	6.64	7.77	8.23	—	8.21	7.64	10.38	12.07	—	11.43
-1.6	6.16	7.48	8.01	—	8.00	6.84	9.95	11.63	—	10.93
-1.5	5.63	7.11	7.73	—	7.73	6.00	9.39	11.14	—	10.40
-1.4	5.04	6.69	7.39	—	7.42	5.14	8.68	10.67	—	9.80
-1.3	4.40	6.19	6.97	5.20	7.05	4.26	7.90	10.03	6.47	9.13
-1.2	3.71	5.61	6.56	—	6.61	3.37	7.07	9.29	—	8.40
-1.1	2.98	4.95	6.26	—	6.11	2.47	6.18	8.40	—	7.60
-1.0	2.22	4.24	5.75	3.63	5.54	1.56	5.31	7.34	4.67	6.73
-0.9	1.43	3.43	5.02	—	4.87	0.65	4.35	6.23	—	5.79
-0.8	0.61	2.57	4.09	—	4.12	-0.25	3.30	5.04	—	4.79
-0.7	-0.21	1.96	2.95	—	3.30	-1.15	2.26	3.50	—	3.73
-0.6	-1.05	1.18	1.70	—	2.40	-2.02	1.23	1.40	—	2.64
-0.5	-1.88	0.27	-0.01	—	1.44	-2.87	0.26	-2.21	—	1.56

Table 3.2: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

$\log_{10} \epsilon$	$k = 2; R = 1.0$					$k = 4; R = 1.0$				
	VQ	VQ+ Inst. MAP	VQ+ Seq. MAP	COVQ [16]	VQ+ Inst. AMMSE	VQ	VQ+ Inst. MAP	VQ+ Seq. MAP	COVQ [16]	VQ+ Inst. AMMSE
$-\infty$	7.92	7.92	7.92	7.87	7.92	10.19	10.19	10.19	10.18	10.19
-3.0	7.80	7.86	7.87	-	7.88	9.94	10.01	10.06	-	10.04
-2.9	7.77	7.84	7.86	-	7.86	9.88	9.96	10.03	-	10.00
-2.8	7.74	7.82	7.84	-	7.85	9.81	9.91	10.00	-	9.96
-2.7	7.69	7.79	7.82	-	7.83	9.71	9.83	9.95	-	9.90
-2.6	7.63	7.76	7.81	-	7.81	9.60	9.76	9.90	-	9.83
-2.5	7.56	7.72	7.78	-	7.78	9.46	9.66	9.83	-	9.76
-2.4	7.48	7.67	7.75	-	7.75	9.29	9.54	9.75	-	9.66
-2.3	7.37	7.61	7.70	7.31	7.71	9.08	9.40	9.64	9.15	9.54
-2.2	7.23	7.53	7.65	-	7.66	8.84	9.22	9.51	-	9.39
-2.1	7.07	7.43	7.58	-	7.59	8.55	9.01	9.37	-	9.22
-2.0	6.88	7.30	7.49	6.83	7.51	8.21	8.76	9.18	8.37	9.01
-1.9	6.65	7.15	7.39	-	7.42	7.82	8.46	8.97	-	8.76
-1.8	6.37	6.97	7.25	-	7.30	7.37	8.11	8.72	-	8.48
-1.7	6.05	6.74	7.19	-	7.16	6.87	7.70	8.43	-	8.14
-1.6	5.67	6.47	7.06	-	6.99	6.31	7.24	8.08	-	7.77
-1.5	5.24	6.14	6.86	-	6.77	5.69	6.72	7.66	-	7.34
-1.4	4.75	5.92	6.62	-	6.52	5.02	6.14	7.18	-	6.85
-1.3	4.21	5.65	6.31	4.37	6.21	4.31	5.51	6.62	6.23	6.31
-1.2	3.61	5.17	5.95	-	5.85	3.56	4.82	6.00	-	5.72
-1.1	2.97	4.63	5.50	-	5.42	2.77	4.06	5.29	-	5.08
-1.0	2.27	4.01	5.04	2.76	4.91	1.95	3.27	4.49	4.65	4.39
-0.9	1.54	3.30	4.44	-	4.30	1.11	2.42	3.55	-	3.66
-0.8	0.77	2.50	3.82	-	3.60	0.25	1.56	2.49	-	2.90
-0.7	-0.03	1.67	2.82	-	2.83	-0.62	0.69	1.28	-	2.14
-0.6	-0.85	0.79	1.42	-	1.98	-1.49	-0.15	-0.16	-	1.41
-0.5	-1.68	-0.13	-0.23	-	1.11	-2.37	-0.78	-2.34	-	0.74

Table 3.3: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

$\log_{10} \epsilon_a$	$\log_{10} \epsilon_d$	$k = 1; R = 3.0$		$k = 4; R = 1.0$	
		COVQ+ AMMSE	COVQ [15]	COVQ+ AMMSE	COVQ [16]
-1.3	$-\infty$	9.14	—	6.31	—
	-1.9	9.62	—	6.87	—
	-1.8	9.62	—	6.90	—
	-1.7	9.63	—	6.96	—
	-1.6	9.63	—	6.86	—
	-1.5	9.61	—	6.87	—
	-1.4	9.58	—	6.82	—
	-1.3	9.54	6.47	6.52	6.23
-1.0	$-\infty$	6.73	—	4.39	—
	-1.6	7.45	—	5.30	—
	-1.5	7.47	—	5.33	—
	-1.4	7.48	—	5.35	—
	-1.3	7.49	—	5.23	—
	-1.2	7.12	—	5.18	—
	-1.1	7.03	—	5.11	—
	-1.0	6.23	4.67	5.02	4.65

Table 3.4: SNR (in dB) Performances of Joint Source-Channel Coding Scheme Using COVQ Encoding and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; k = Dimension; R = Rate (Bits/Sample); ϵ_d = Design Bit Error Rate of COVQ Encoder; ϵ_a = Actual Channel Bit Error Rate.

ϵ	LMQ- NBC [15]	LMQ- FBC [15]	COSQ [15]	OPTA
$R = 1$ b/s				
0.000	1.88	1.88	1.88	9.27
0.005	1.77	1.77	1.78	8.97
$R = 4$ b/s				
0.000	15.74	15.74	15.74	27.79
0.005	3.49	8.98	10.79	26.95

Table 3.5: SNR (in dB) Performances of Several Scalar Quantizers Operating Over Noiseless and Noisy Channels; Generalized Gaussian Source with $\alpha = 0.5$; ϵ = Channel Bit Error Rate; OPTA = Optimum Performance Theoretically Attainable.

R	ϵ	Minimal		Feedback	
		$K = 1$	$K = 2$	$K = 1$	$K = 2$
3	0.005	4 2	4 4 1	4 7	7 6 2
		2 1	2 2 2	3	1 6
		1 5	1 3 5		
	0.010	4 1	4 6 5	4 7	7 6 2
		2 4	2 0 4	3	1 6
		1 3	1 3 3		
4	0.005	8 1	8 F 8	A D	A 8 8
		4 F	4 4 D	2	8 7
		2 3	2 F 2		
		1 7	1 3 E		
	0.010	8 5	8 8 2	A D	A 8 8
		4 3	4 A 1	2	8 7
		2 B	2 5 D		
		1 2	1 B 8		

Table 3.6: A List of Good Rate-One Convolutional Codes for the Generalized Gaussian Source with $\alpha = 0.5$; ϵ = Channel Bit Error Rate; R = Rate of Lloyd-Max Quantizer; K = Constraint Length; 2^{RK} = Number of States.

				Convolutional Encoder				OPTA
				Minimal		Feedback		
R	ϵ	LMQ-NBC [15]	COSQ [15]	$K=1$	$K=2$	$K=1$	$K=2$	
3	0.000	10.45	10.45	10.35	10.35	10.35	10.35	21.74
	0.005	5.19	8.35	8.94	9.60	9.12	10.00	20.97
	0.010	2.90	7.23	7.75	8.44	7.21	8.71	20.34
4	0.000	15.74	15.74	15.69	15.69	15.69	15.69	27.79
	0.005	3.49	10.79	11.16	14.56	12.96	15.36	26.95
	0.010	0.64	9.09	9.48	11.60	9.40	12.46	26.05

Table 3.7: SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded System with MAP Sequence Decoding; Generalized Gaussian Source with $\alpha = 0.5$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

		MAP Viterbi Delay= ∞				MAP Symbol-By-Symbol Delay=20				MMSE Delay=20			
		Minimal		Feedback		Minimal		Feedback		Minimal		Feedback	
R	ϵ	$K=1$	$K=2$	$K=1$	$K=2$	$K=1$	$K=2$	$K=1$	$K=2$	$K=1$	$K=2$	$K=1$	$K=2$
3	0.005	8.94	9.60	9.12	10.00	9.12	9.64	8.96	9.90	9.38	9.92	9.33	10.10
	0.010	7.75	8.44	7.21	8.71	7.85	8.76	7.59	8.59	8.33	9.18	8.22	9.03
4	0.005	11.16	14.56	12.96	15.36	11.68	14.10	12.37	15.22	12.32	14.72	13.23	15.33
	0.010	9.48	11.60	9.40	12.46	9.70	11.66	9.17	11.85	10.43	12.37	10.37	12.75

Table 3.8: SNR (in dB) Performances of Convolutionally Encoded Systems Using Three Different Decoding Techniques; Generalized Gaussian Source with $\alpha = 0.5$; K = Constraint Length; 2^{KR} = Number of States.

				Convolutional Encoder				OPTA
				Minimal		Feedback		
R	ϵ	LMQ-NBC [15]	COSQ [15]	$K=1$	$K=2$	$K=1$	$K=2$	
3	0.000	12.64	12.64	12.64	12.64	12.64	12.64	18.68
	0.005	9.30	10.49	8.64	9.50	8.11	9.05	17.87
	0.010	7.44	9.17	6.16	6.16	5.56	5.34	17.23
4	0.000	18.13	18.13	18.13	18.13	18.13	18.13	24.70
	0.005	9.78	12.76	9.64	10.72	8.57	11.02	23.65
	0.010	7.11	11.03	5.59	5.04	5.26	5.49	22.77

Table 3.9: SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded Systems with MAP Sequence Decoding; Generalized Gaussian Source with $\alpha = 1.0$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

R	ϵ	Minimal		Feedback	
		$K = 1$	$K = 2$	$K = 1$	$K = 2$
3	0.005	4 1	4 1 0	4 1	7 2 4
		2 3	2 0 5	3	1 5
		1 6	1 0 2		
	0.010	4 1	4 2 3	4 1	7 2 4
		2 3	2 6 3	3	1 5
		1 4	1 3 1		
4	0.005	8 5	8 F 8	F 9	A 8 8
		4 E	4 4 D	F	1 7
		2 D	2 F 2		
		1 1	1 3 E		
	0.010	8 1	8 7 C	F 9	A 8 8
		4 6	4 0 D	F	1 7
		2 3	2 F 9		
		1 A	1 B E		

Table 3.10: A List of Good Rate-One Convolutional Codes for the Generalized Gaussian Source with $\alpha = 1.0$; ϵ = Channel Bit Error Rate; R = Rate of Lloyd-Max Quantizer; K = Constraint Length; 2^{RK} = Number of States.

α	R	ϵ	LMQ NBC [15]	COSQ [15]	Feedback K=2	OPTA
0.5	3	0.005	5.19	8.35	10.08	20.97
		0.010	2.90	7.23	9.37	20.34
	4	0.005	3.49	10.79	15.39	26.95
		0.010	0.64	9.09	13.91	26.05
1.0	3	0.005	9.30	10.49	10.72	17.87
		0.010	7.44	9.17	9.45	17.23
	4	0.005	9.78	12.76	15.76	23.65
		0.010	7.11	11.03	13.58	22.77

Table 3.11: SNR (in dB) Performances of LMQ-NBC, COSQ and Convolutionally Encoded Systems with MAP Sequence Decoding and Rate- and Entropy-Constrained Scalar Quantizers; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

α	R	ϵ	COVQ			Proposed Scheme
			$k = 1$	$k = 2$	$k = 3$	
0.5	3	0.000	10.45	14.19	15.99	10.35
		0.005	8.35	11.05	11.90	10.08
		0.010	7.23	9.73	10.77	9.37
	4	0.000	15.74	19.96	—	15.69
		0.005	10.79	13.85	—	15.39
		0.010	9.09	12.17	—	13.91
1.0	3	0.000	12.64	14.26	15.16	12.64
		0.005	10.49	11.61	11.83	10.72
		0.010	9.17	10.42	10.81	9.45
	4	0.000	18.13	20.04	—	18.13
		0.005	12.76	14.42	—	15.76
		0.010	11.03	12.88	—	13.58

Table 3.12: SNR (in dB) Performances of COVQ and Convolutionally Encoded System with MAP Sequence Decoding and Rate- and Entropy-Constrained Scalar Quantizers; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; k = dimension of COVQ.

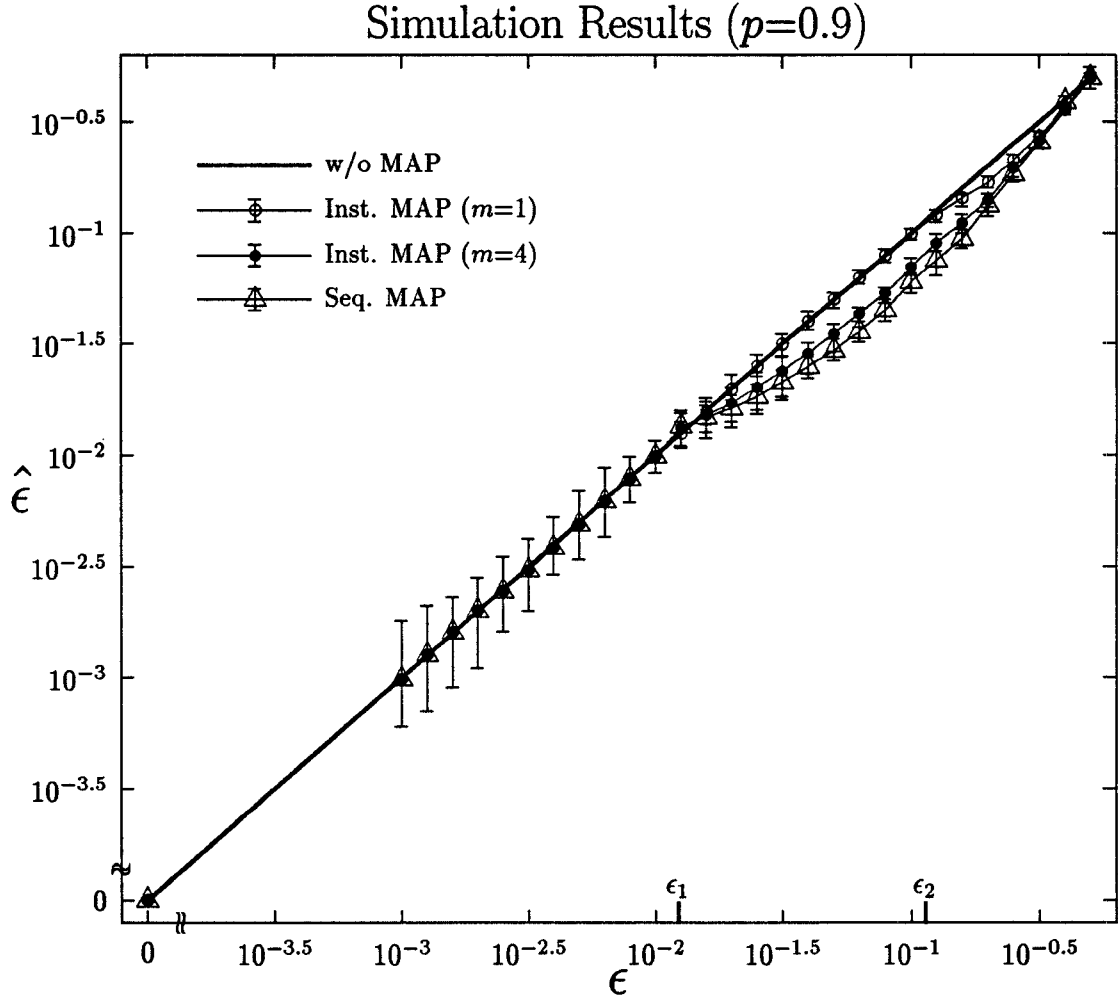


Figure 3.1: Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.9$; ϵ =Channel Bit Error Rate; $\hat{\epsilon} = \Pr\{\text{error}\}$; m =Block Size of Instantaneous MAP Detector; Error Bars Indicate the Minimum and Maximum Values of $\hat{\epsilon}$ in the Simulation.

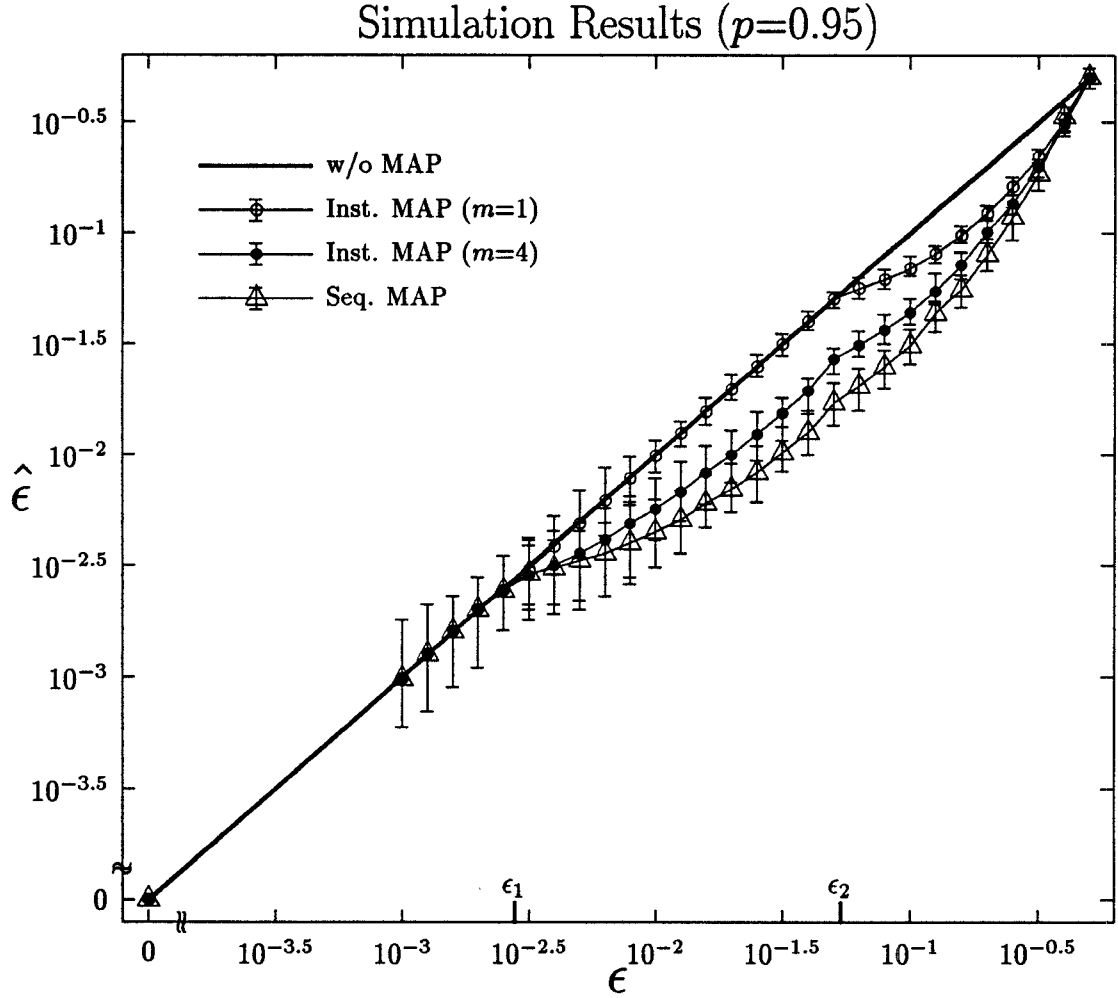


Figure 3.2: Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.95$; ϵ =Channel Bit Error Rate; $\hat{\epsilon} = \Pr\{\text{error}\}$; m =Block Size of Instantaneous MAP Detector; Error Bars Indicate the Minimum and Maximum Values of $\hat{\epsilon}$ in the Simulation.

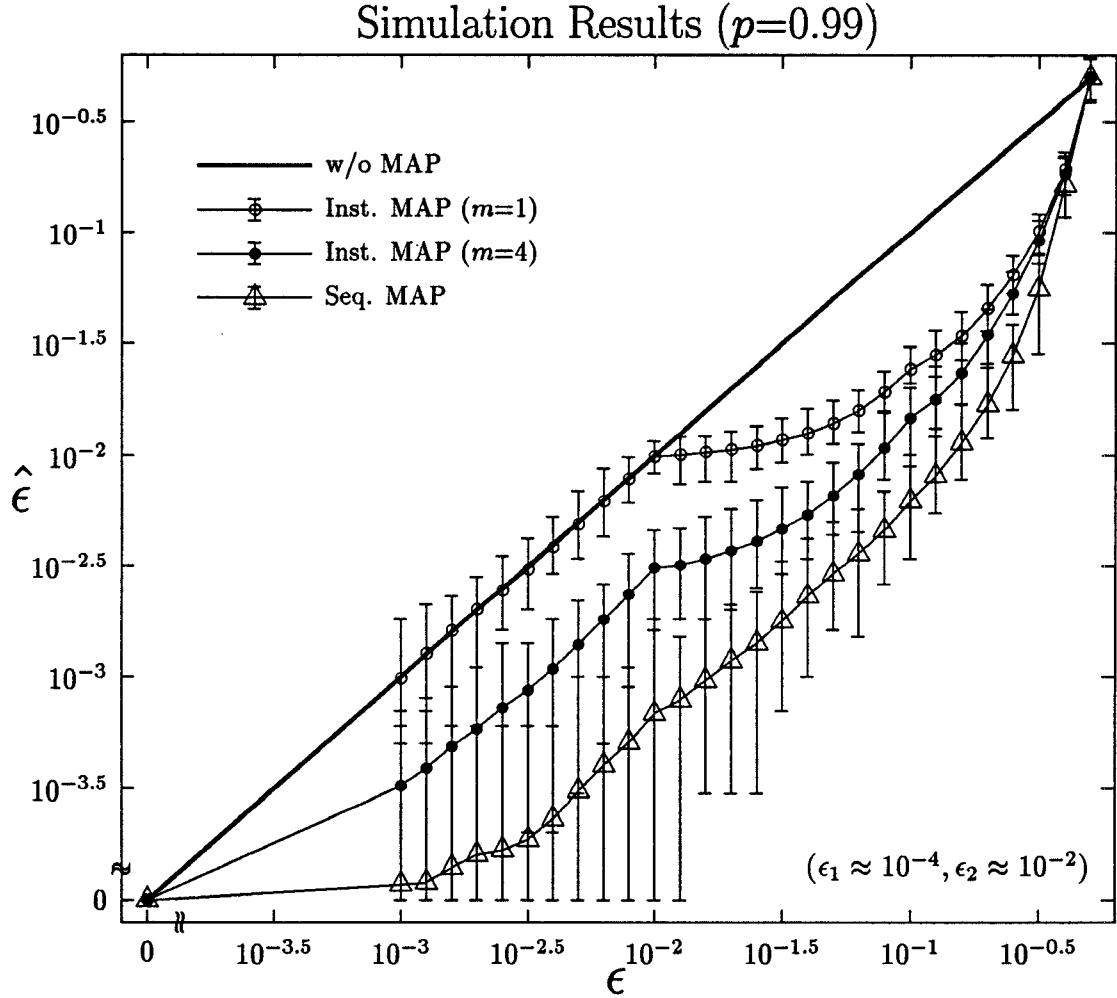


Figure 3.3: Performances of Instantaneous and Sequence MAP Detectors for Binary Markov Source with $p=0.99$; ϵ =Channel Bit Error Rate; $\hat{\epsilon} = \Pr\{\text{error}\}$; m =Block Size of Instantaneous MAP Detector; Error Bars Indicate the Minimum and Maximum Values of $\hat{\epsilon}$ in the Simulation.

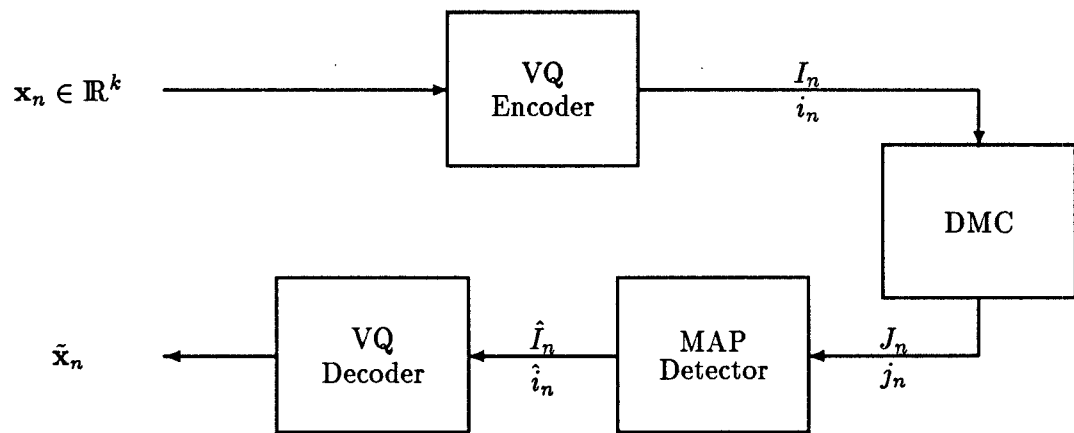


Figure 3.4: Block Diagram of Joint Source-Channel Coding System Using MAP Detection.

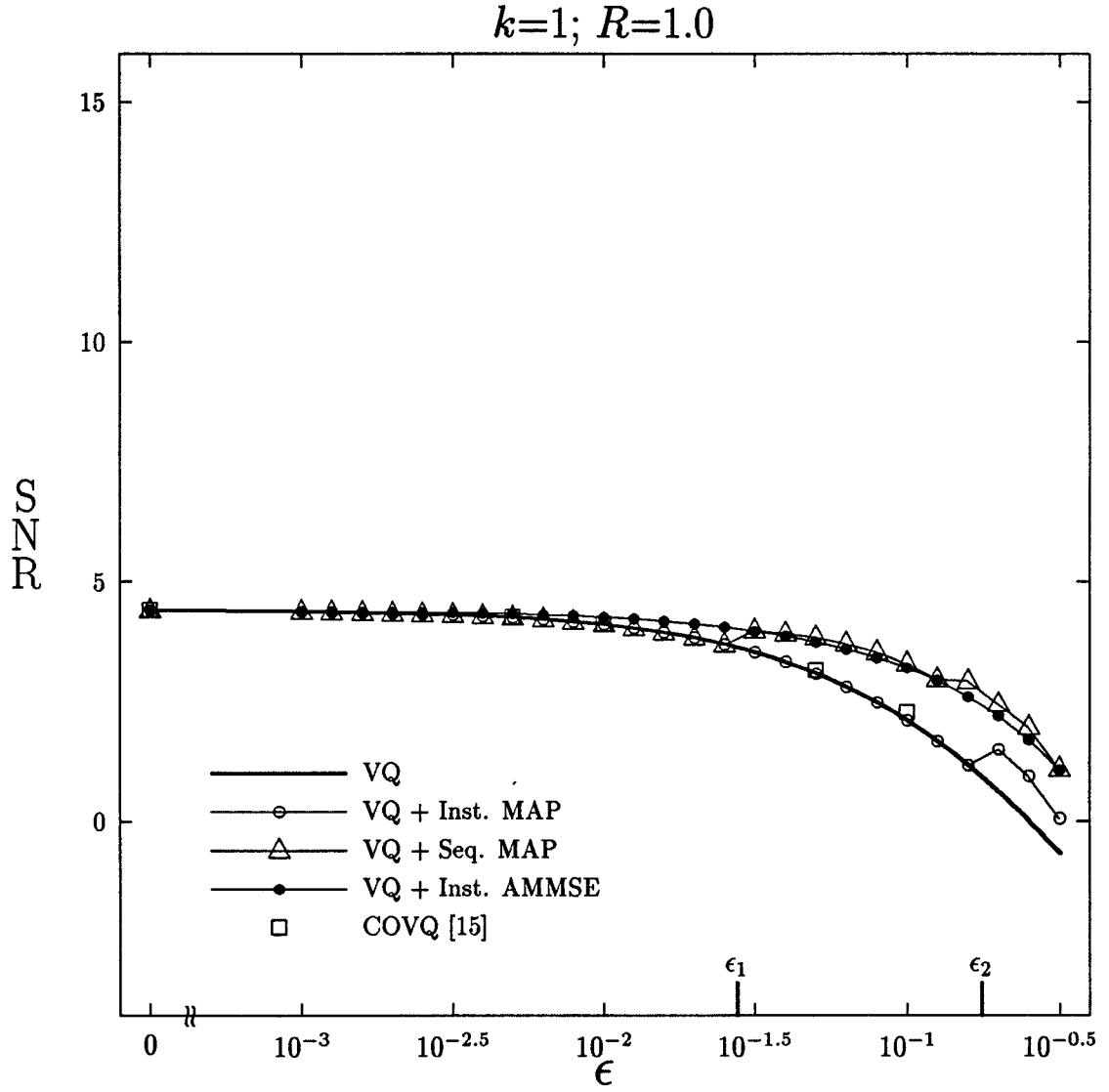


Figure 3.5: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

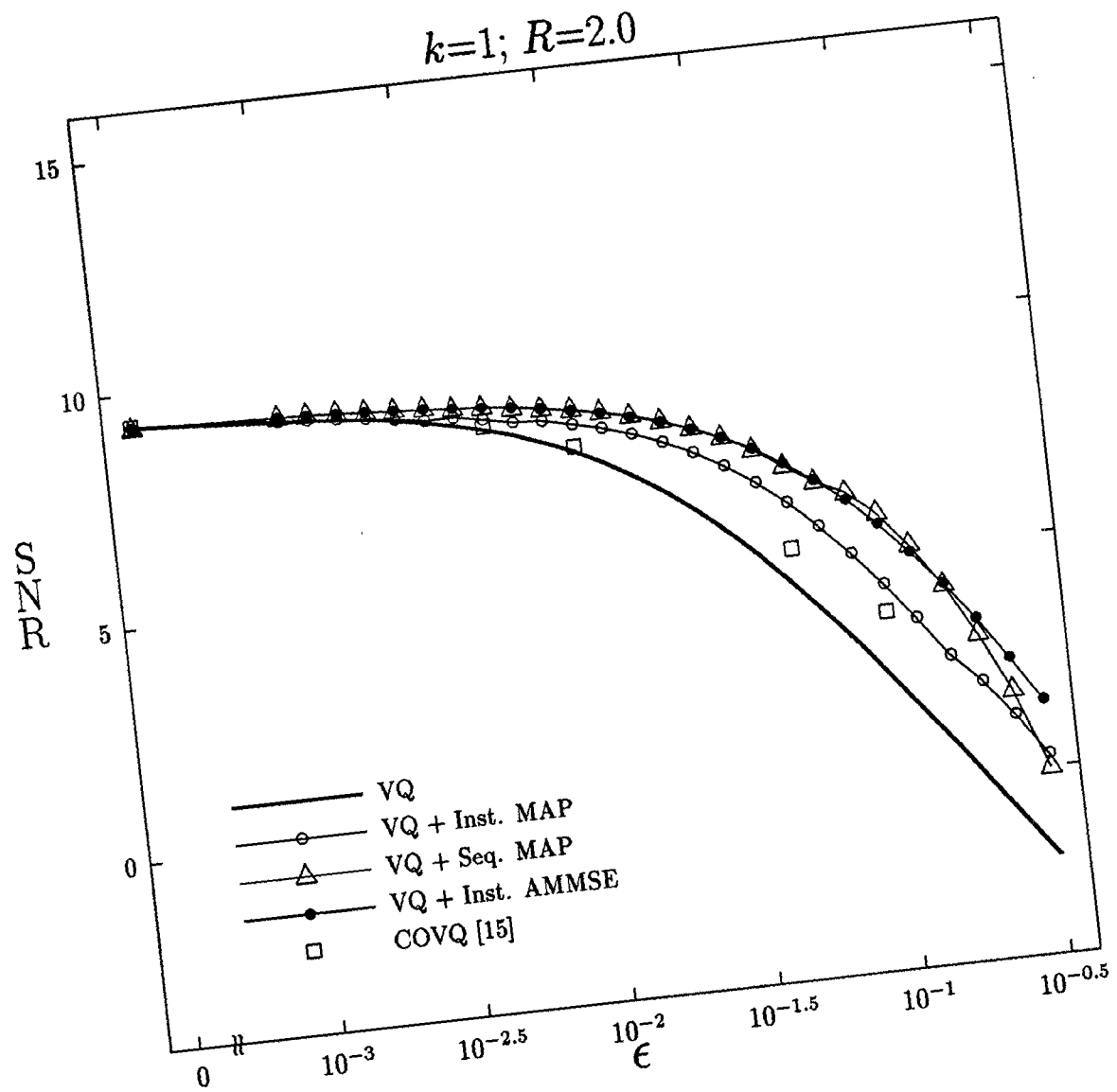


Figure 3.6: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

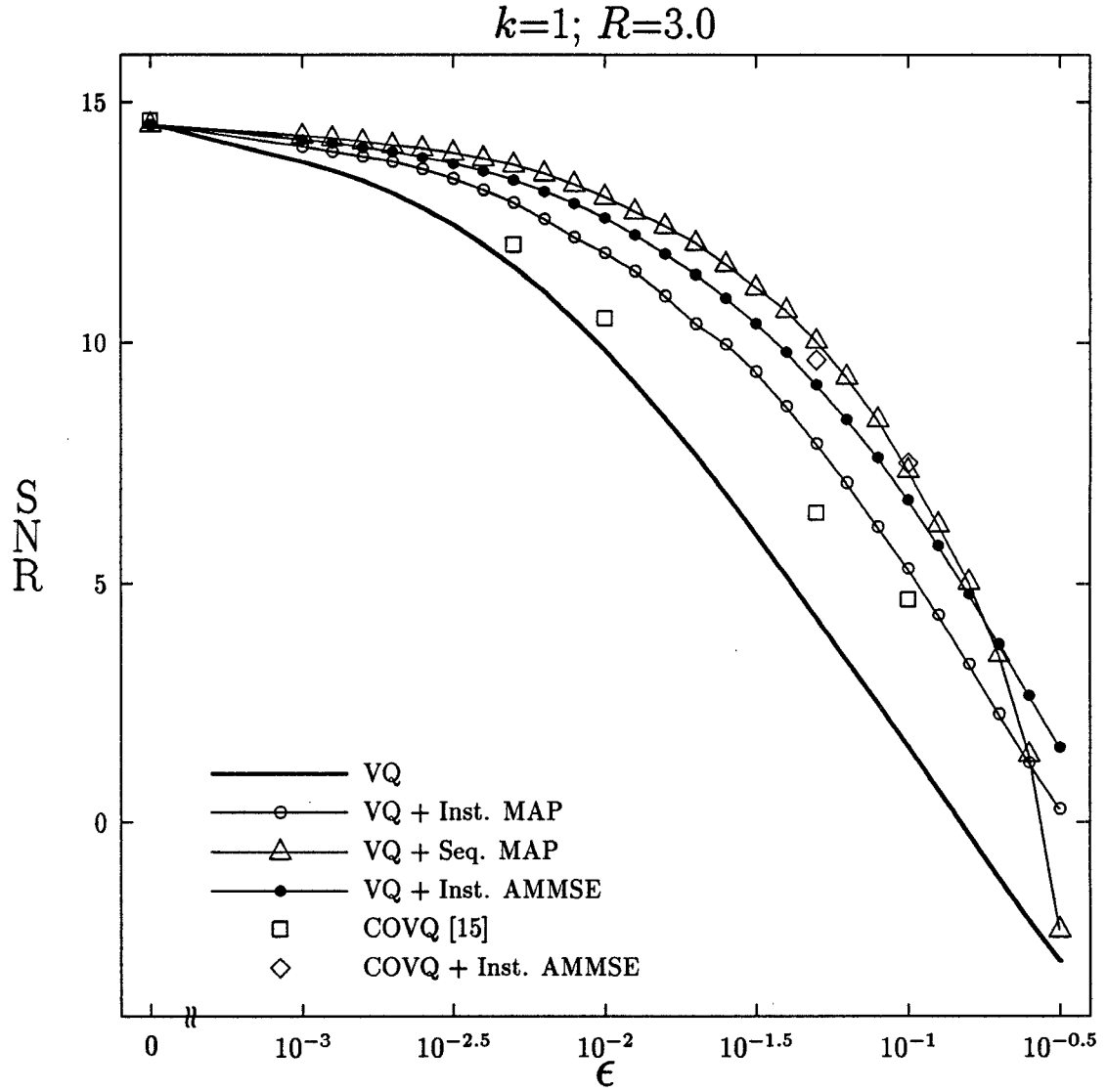


Figure 3.7: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

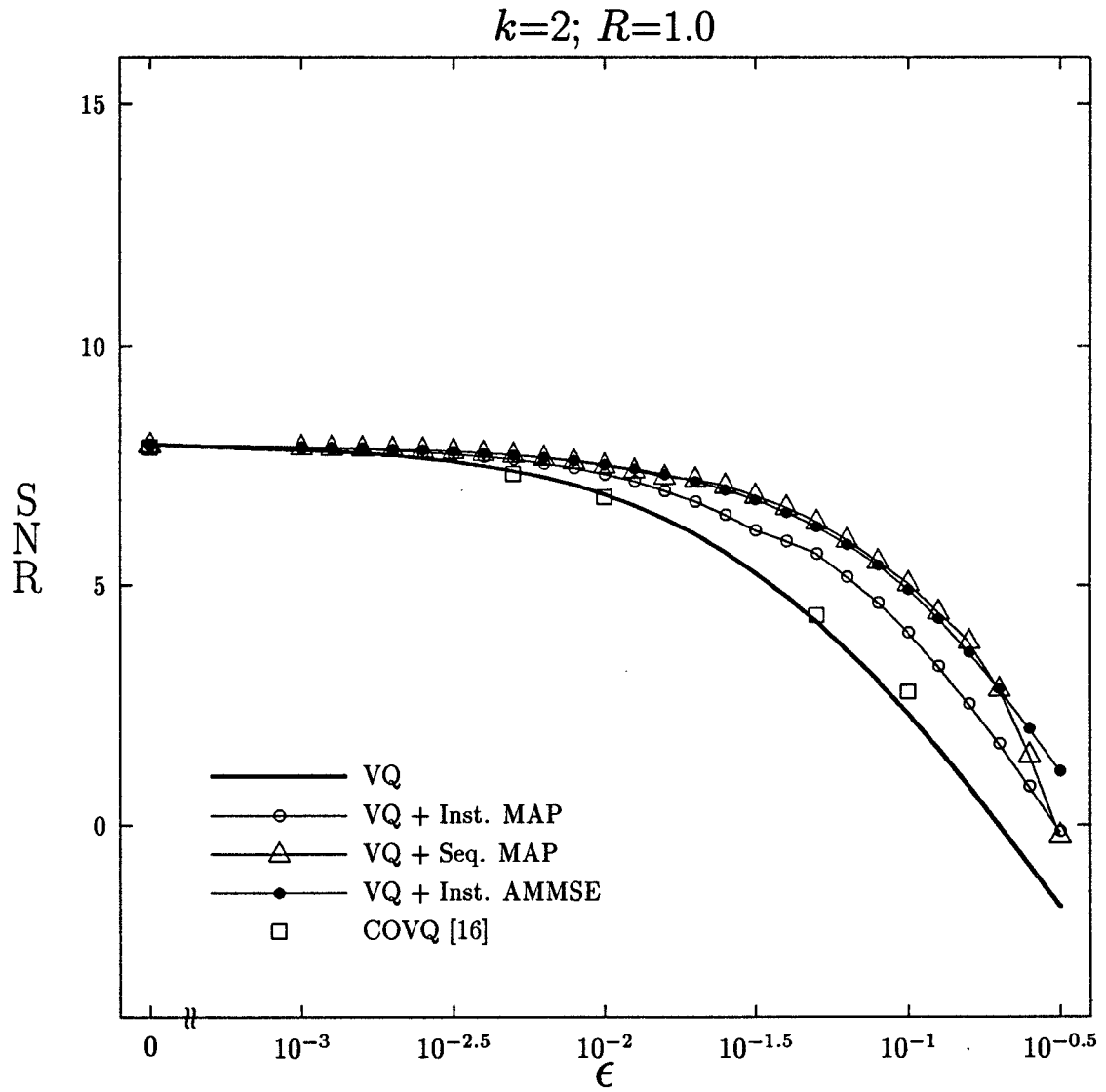


Figure 3.8: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; $k =$ Dimension; $R =$ Rate (Bits/Sample); $\epsilon =$ Channel Bit Error Rate.

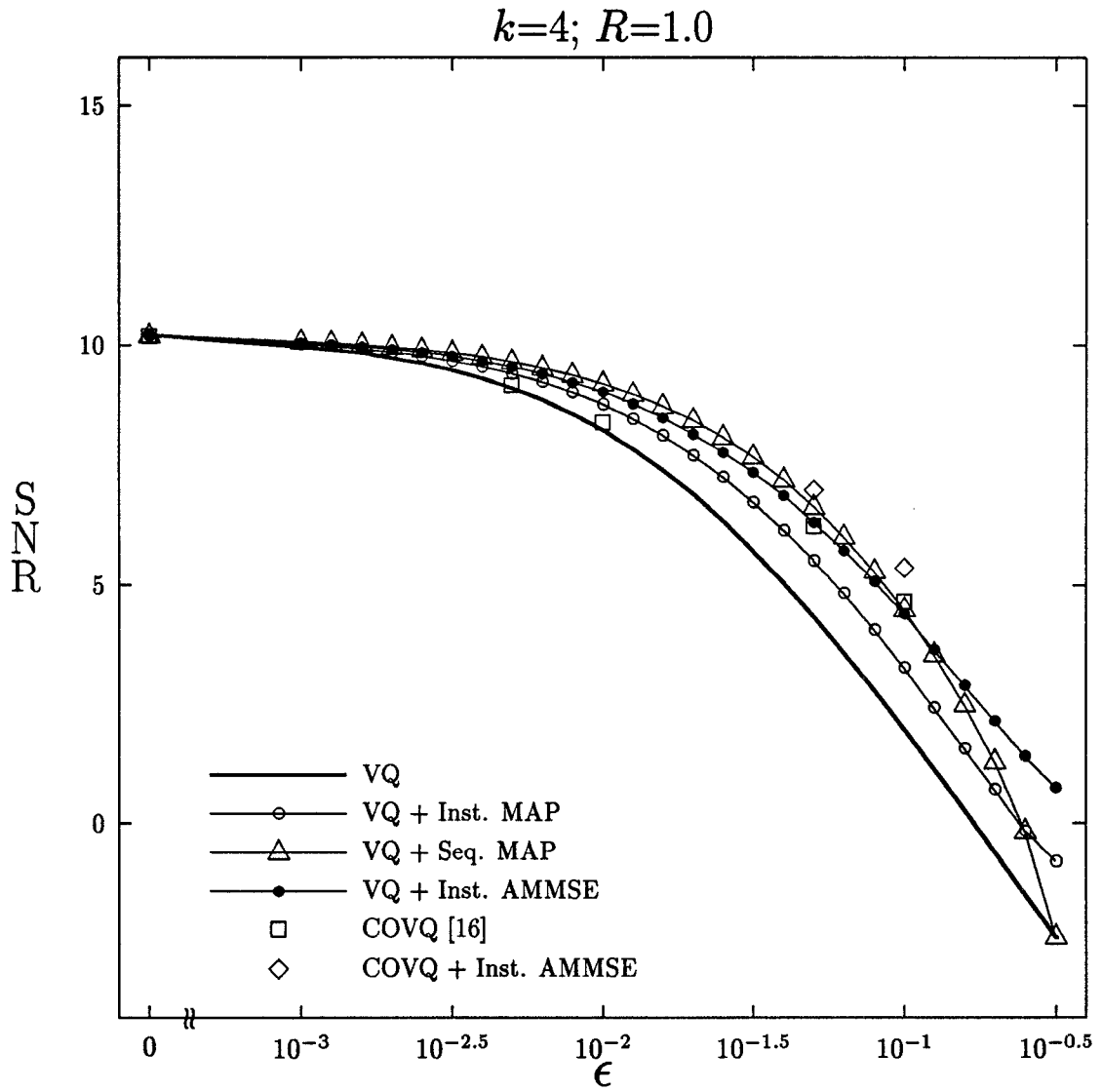


Figure 3.9: SNR (in dB) Performances of Joint Source-Channel Coding Schemes Using MAP Detection and AMMSE Decoding for AR(1) Source with $\rho = 0.9$; k = Dimension; R = Rate (Bits/Sample); ϵ = Channel Bit Error Rate.

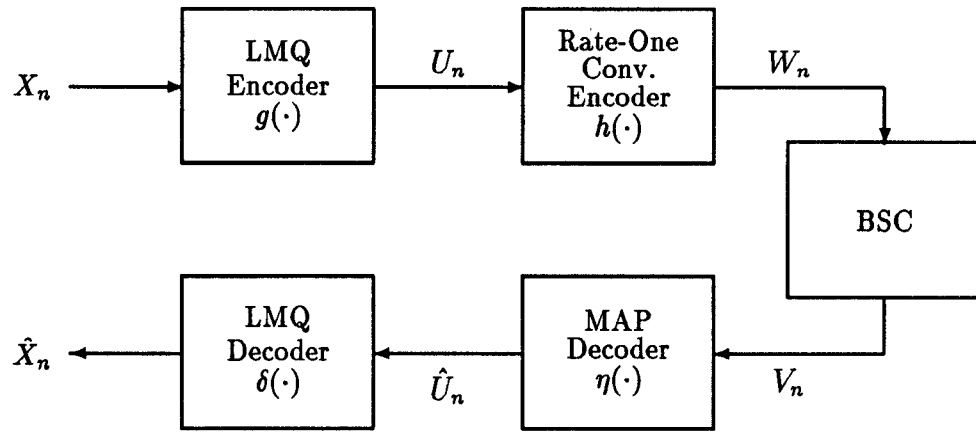


Figure 3.10: Block Diagram of Convolutionally Encoded System.

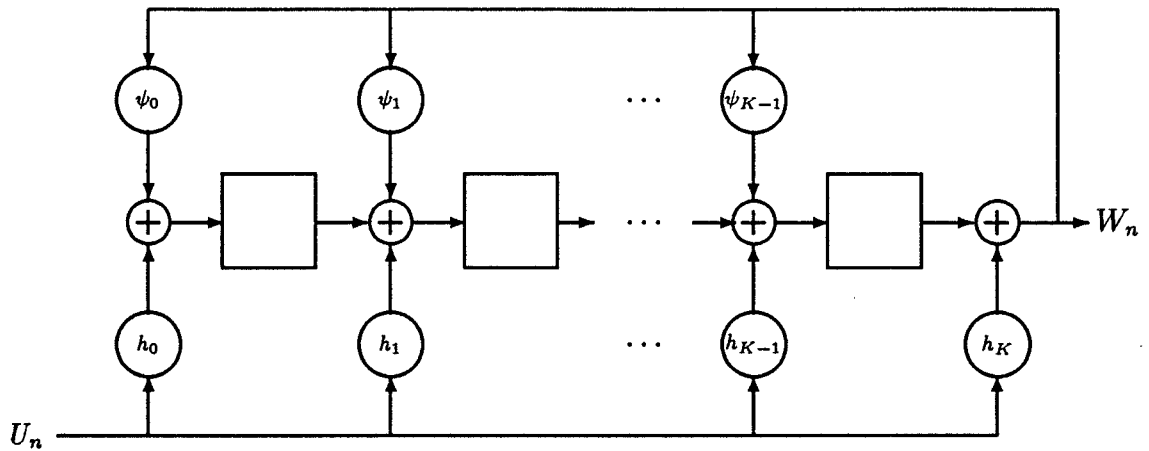


Figure 3.11: Convolutional Encoder with Feedback; K = Constraint Length.

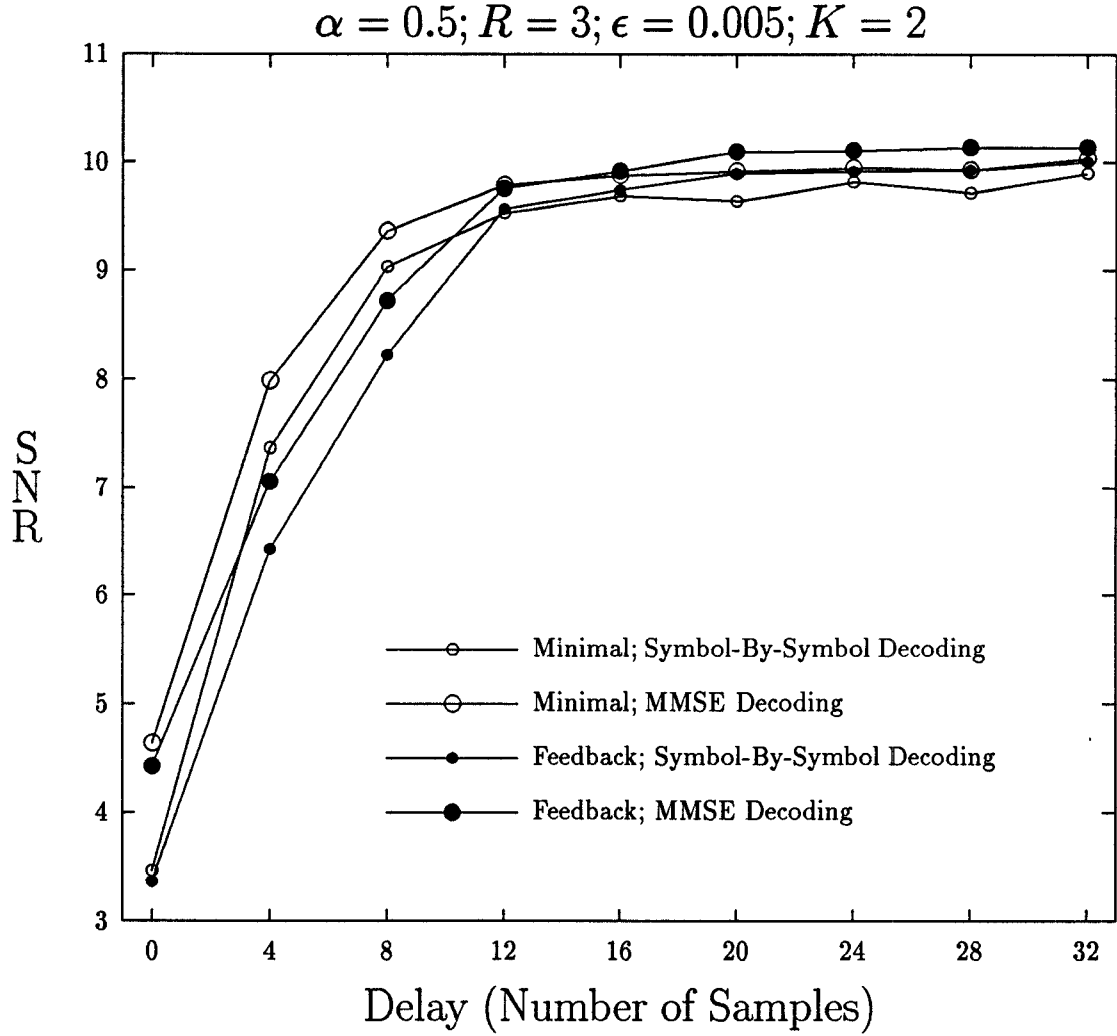


Figure 3.12: SNR (in dB) Performances of Convolutionally Encoded Systems with Symbol-By-Symbol and MMSE Decoding Vs. Delay; Generalized Gaussian Source with $\alpha = 0.5$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

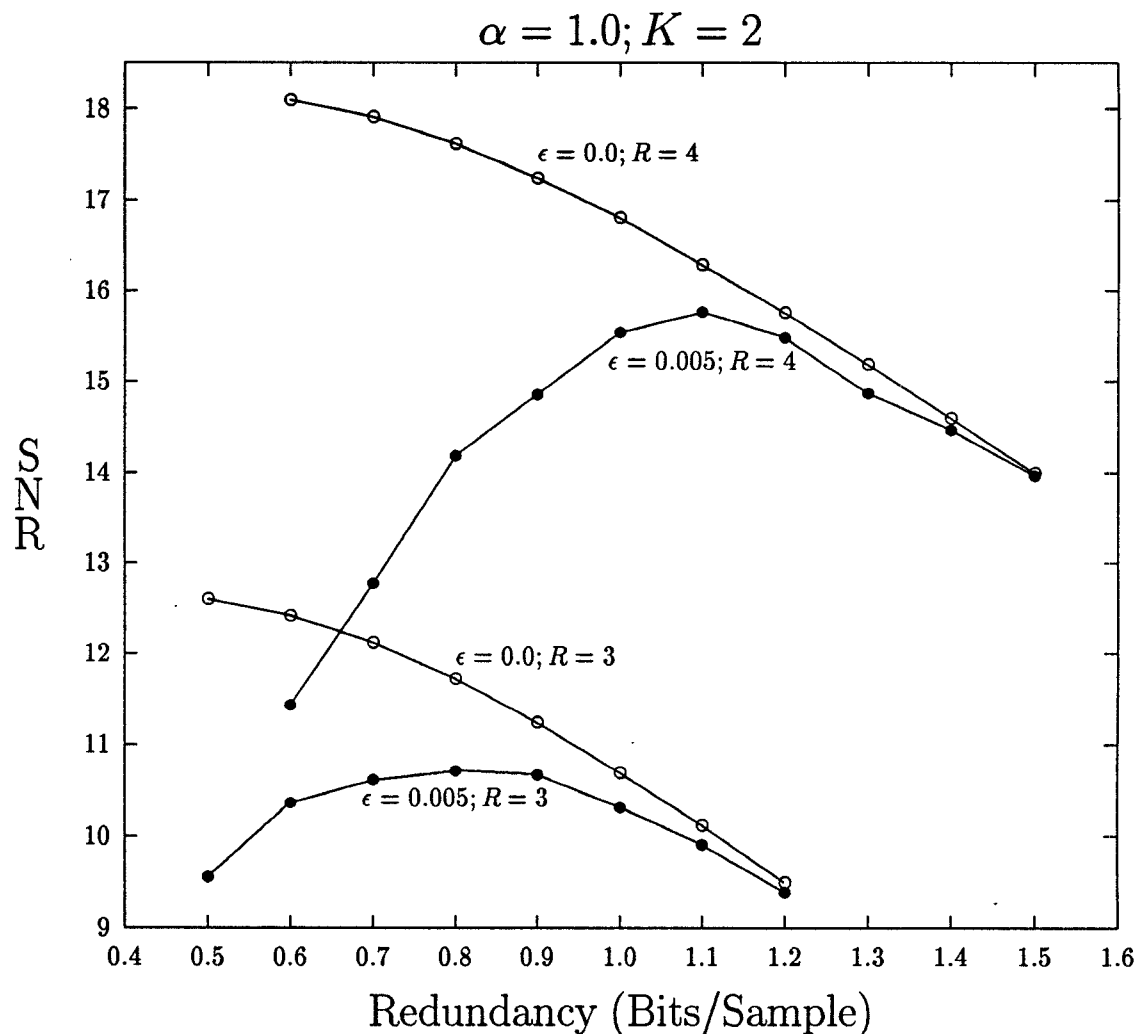


Figure 3.13: SNR (in dB) Performances of Convolutionally Encoded Systems with MAP Sequence Decoding Vs. Redundancy of Entropy-Constrained Scalar Quantizer; Generalized Gaussian Source with $\alpha = 1.0$; R = Rate of Quantizer (Bits/Sample); ϵ = Channel Bit Error Rate; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

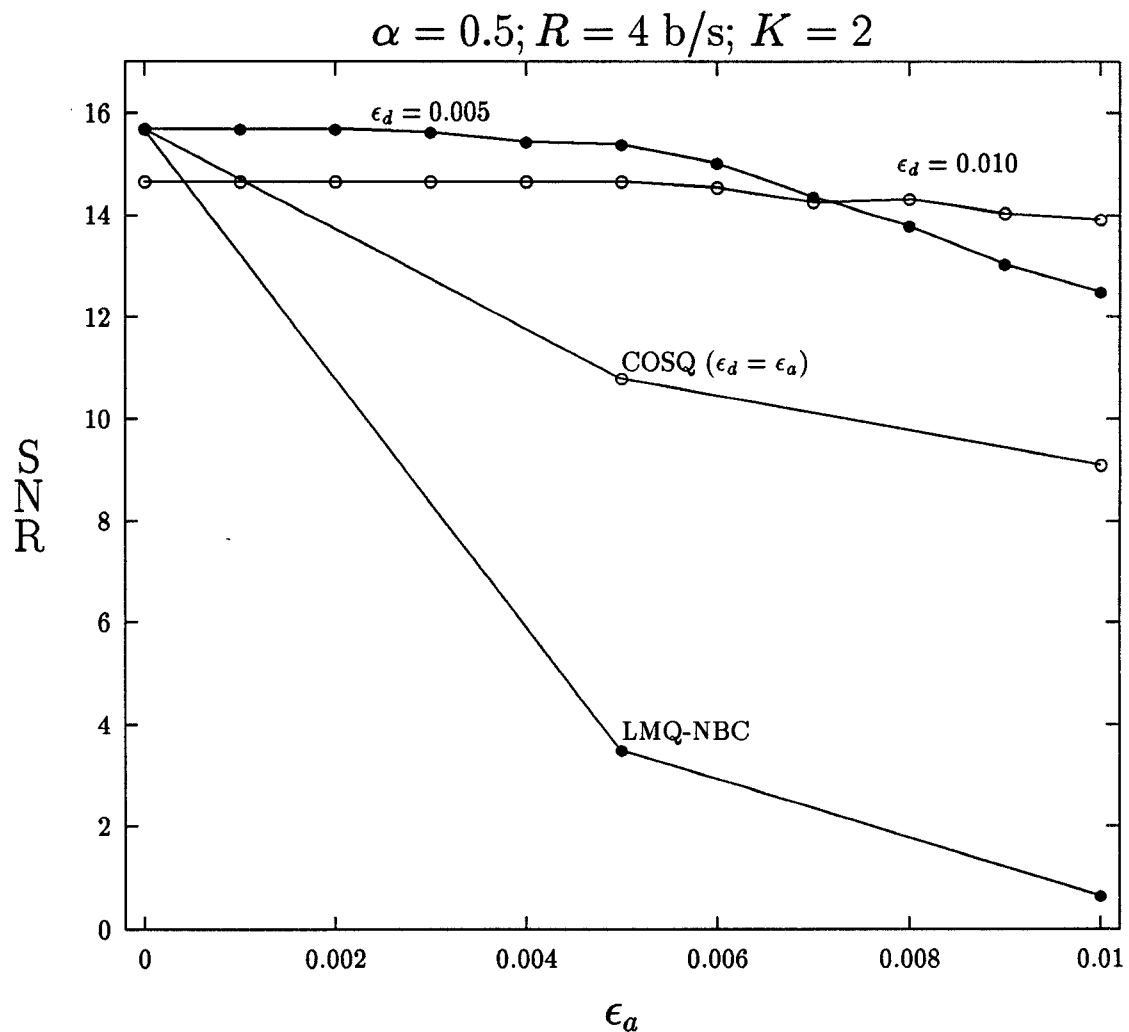


Figure 3.14: SNR (in dB) Performances of Convolutionally Encoded Systems with MAP Sequence Decoding Under Channel Mismatch; Generalized Gaussian Source with $\alpha = 0.5$; R = Rate of Quantizer (Bits/Sample); ϵ_d = Design BER; ϵ_a = Actual Channel BER; K = Constraint Length of Convolutional Encoder; 2^{KR} = Number of States.

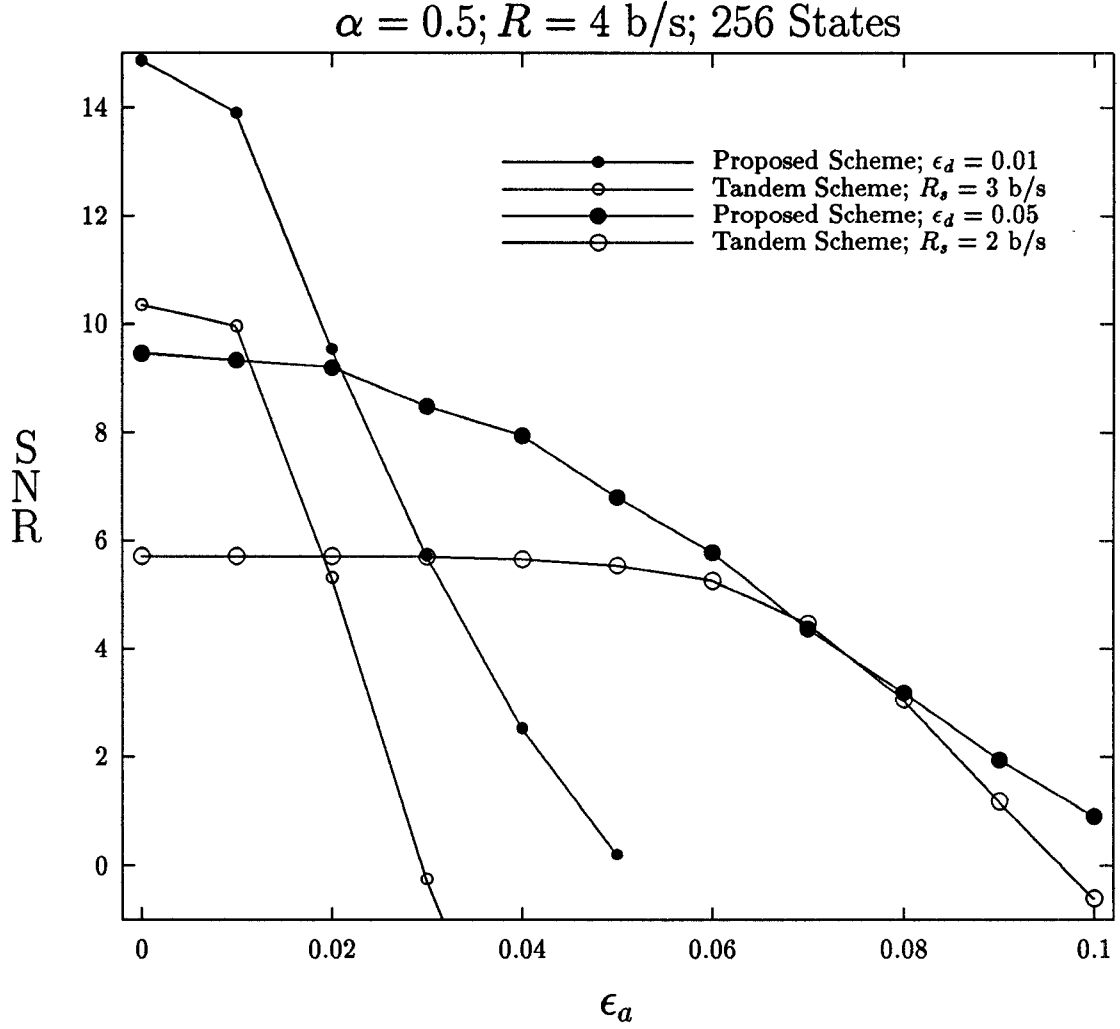


Figure 3.15: Comparisons of Proposed System Versus Tandem Source-Channel Coding System; Generalized Gaussian Source with $\alpha = 0.5$; R_s = Rate of Quantizer in Tandem System; R = Overall Rate; ϵ_d = Design BER; ϵ_a = Actual Channel BER.

Chapter 4

Summary and Future Work

In this dissertation, we considered the quantization of a discrete-time source for transmission over a discrete memoryless channel when complexity is limited. We proposed two approaches.

First, we considered the analysis and design of two low-complexity VQ's — namely, the TSVQ and MSVQ. We showed that these two VQ's can be treated in a unified framework. We presented an algorithm for designing TSVQ and MSVQ for a given noisy channel. The proposed schemes, dubbed CM-TSVQ and CM-MSVQ, show substantial improvements over the ordinary TSVQ and MSVQ when the channel is very noisy. For the two-stage TSVQ and MSVQ, designing the first stage for a channel noisier than the actual channel, in general, leads to performance gains. Additional gains can be obtained using the M -algorithm. When compared with a tandem source-channel coding scheme consisting of an LBG-VQ and a quasi-perfect linear block code, the CM-MSVQ shows superior performance in all cases considered. Furthermore, the CM-MSVQ was shown to be robust against channel mismatch.

In the second approach, we assumed that a low-complexity quantizer is given. We exploited the fact that the quantizer is sub-optimum and therefore it does not remove all of the source redundancy. The residual redundancy (which remains at

the output of the quantizer) is used to combat channel noise. We first considered the case where the residual redundancy is in the form of memory. Here, the quantizer's output was modeled by a discrete Markov chain and two MAP detection schemes were developed. The sequence MAP detector, involving a large delay, is implemented by the Viterbi algorithm while the instantaneous MAP detector, with no delay, is implemented recursively. In the special case of binary Markov chain and binary symmetric channel, necessary and sufficient conditions for the usefulness of the MAP detectors were given. Simulation results show that, in a joint source-channel coding scenario, the MAP detectors can indeed lessen the effect of channel noise. We also considered the case where the residual redundancy is in the form of a non-uniform distribution. In this case, we proposed the use of a rate-one convolutional encoder to convert the residual redundancy (in the form of a non-uniform distribution) into redundancy in the form of memory. An exact performance analysis of this system, in general, is difficult to obtain. Instead, we derived an approximate upper bound on the average distortion. This bound is then used to find "good" convolutional encoders. Through simulation results, we demonstrated that, when there is a sufficient amount of redundancy, the effect of channel errors can be reduced using this technique. A comparison with a tandem source-channel coding system again indicates superior performance of the proposed joint source-channel coding system.

In the first approach (CM-TSVQ and CM-MSVQ), most of the computational burden lies in the encoder (the decoder is easily implemented using table lookups and vector additions). On the other hand, most of the computational complexity of the second approach is in the MAP detector. Therefore, the first approach would be useful in applications where the decoder complexity is constrained (e.g., broadcasting) while the second approach would be useful in applications where the encoder complexity is constrained (e.g., remote sensing).

There are several directions in which this work can proceed. Throughout the dissertation, we had assumed that the channel is memoryless. However, most “real-world” channels have memory. With this in mind, we have begun working on extending some of these results to discrete channels with memory. For now, we are considering the channel modeled by the Polya contagion urn [2]. In the case where the memory is finite, it can be shown that this channel is an additive channel with stationary, ergodic, M -th order Markov noise [2]. Our current study focuses on the MAP detection of a memoryless non-uniform source transmitted over this channel. Some results on the usefulness of the MAP detector (similar to Theorems 1 and 2) have been obtained. In a parallel effort, we are attempting to fit this model to the “real-world” digital cellular channel. From the preliminary results, it appears that for large M the Markov channel model provides a “good” fit. We plan to make some comparisons between this model and the Gilbert channel model [21].

Another direction is to apply these methods to “real-world” data, such as speech or image. In [42], the CM-MSVQ was used to encode speech line spectrum pair parameters. Since the line spectrum pair parameters are highly correlated in time, the techniques described in Section 3.1 can also be used (in conjunction with CM-MSVQ) for this application. Applying these techniques to transform and subband image coding is also possible. In transform image coding, a large number of bits are allocated to the transformed coefficients corresponding to the low frequencies. Similarly, in subband coding, a large number of bits are allocated to the coefficients of the low frequency subbands. Furthermore, all of these coefficients, except the DC coefficient and the coefficients of the lowest frequency subband, have sharp-peaked distributions and are considered to be memoryless [44], [51]. Therefore, the techniques described in Section 3.2 will be particularly useful in the coding of these coefficients.

Finally, a challenging problem is to find methods of reducing the complexities of the proposed methods. For the first approach, reducing the computational complexity of the encoder is important — particularly with the multiple candidate CM-MSVQ. We have attempted to do this using the ideas of k - d tree and neighborhood graph [3] for the case of a noiseless channel. The results indicate that a significant amount of reduction in computational complexity is possible [3]. Of course, the next step is to investigate how these ideas can be used in the case of a noisy channel.

Appendix A

Proof of Theorem 3

Here, we provide the proof of Theorem 3.

Theorem 3 $D_j^u(\tilde{\mathbf{x}})$ is constant on $\mathcal{T}_0$, i.e.,

$$D_j^u(\tilde{\mathbf{x}}_1) = D_j^u(\tilde{\mathbf{x}}_2) \quad \forall \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \in \mathcal{T}_0, \forall j = 0, 1, \dots, RN. \quad (\text{A.1})$$

Proof: (Here, we need to use the assumption that the mapping γ is the NBC.) Fix $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \in \mathcal{T}_0$. Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be the binary representation of $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$, respectively. Also, let $\mathbf{w}_1 = h(\mathbf{u}_1)$ and $\mathbf{w}_2 = h(\mathbf{u}_2)$ be the convolutional encoder outputs corresponding to $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. Fix j , let $\mathbf{v}_1$ be that $\mathbf{v}$ which achieves the maximum in (3.67) when $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_1$ and let $\hat{\mathbf{u}}_1 = \eta(\mathbf{v}_1)$. Thus, $d_H(\mathbf{w}_1, \mathbf{v}_1) = j$, $\hat{\mathbf{u}}_1 = \arg \max APP(\mathbf{u}|\mathbf{v}_1)$ and $\frac{1}{N} \|\tilde{\mathbf{x}}_1 - \delta(\hat{\mathbf{u}}_1)\|^2 = D_j^u(\tilde{\mathbf{x}}_1)$. Let $\mathbf{z}_1$ be the input to the convolutional encoder corresponding to output $\mathbf{v}_1$, i.e., $h(\mathbf{z}_1) = \mathbf{v}_1$. We will show that $\exists \mathbf{v}_2, \hat{\mathbf{u}}_2$ s.t. $d_H(\mathbf{w}_2, \mathbf{v}_2) = j$, $\hat{\mathbf{u}}_2 = \arg \max APP(\mathbf{u}|\mathbf{v}_2)$ and $\|\tilde{\mathbf{x}}_2 - \delta(\hat{\mathbf{u}}_2)\|^2 = \|\tilde{\mathbf{x}}_1 - \delta(\hat{\mathbf{u}}_1)\|^2$.

Let $\mathbf{z}_2 = \mathbf{z}_1 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2$ and $\mathbf{v}_2 = h(\mathbf{z}_2)$. Note that, since $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ are in $\mathcal{T}_0$, each element of $\mathbf{u}_1$ and $\mathbf{u}_2$ is either $(0, 1, 1, \dots, 1)$ or $(1, 0, 0, \dots, 0)$. Hence, each element of $\mathbf{u}_1 \oplus \mathbf{u}_2$ is either all zeros or all ones. This implies that each component of $\mathbf{z}_2$ either is equal to the corresponding component of $\mathbf{z}_1$ or is the one's complement of it and each component of $\delta(\mathbf{z}_2)$ either is equal to the corresponding component of $\delta(\mathbf{z}_1)$ or is the negative of it. Thus $\delta(\mathbf{z}_2)$ and $\delta(\mathbf{z}_1)$ are of the same type.

Now,

$$d_H(\mathbf{w}_2, \mathbf{v}_2) = d_H(h(\mathbf{u}_2), h(\mathbf{z}_2)) = W_H(h(\mathbf{u}_2) \oplus h(\mathbf{z}_2)), \quad (\text{A.2})$$

where $W_H(\cdot)$ denotes the Hamming weight. Because of the linearity of $h(\cdot)$, the above is equal to

$$W_H(h(\mathbf{u}_2 \oplus \mathbf{z}_2)) = W_H(h(\mathbf{u}_1 \oplus \mathbf{z}_1)) = d_H(\mathbf{w}_1, \mathbf{v}_1) = j. \quad (\text{A.3})$$

This shows that $d_H(\mathbf{w}_2, \mathbf{v}_2) = j$. Let $\hat{\mathbf{u}}_2 = \hat{\mathbf{u}}_1 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2$. Suppose that $\hat{\mathbf{u}}_2$ is not equal to $\arg \max APP(\mathbf{u}|\mathbf{v}_2)$. Then $\exists \tilde{\mathbf{u}}_2 \neq \hat{\mathbf{u}}_2$ s.t. $APP(\tilde{\mathbf{u}}_2|\mathbf{v}_2) > APP(\hat{\mathbf{u}}_2|\mathbf{v}_2)$. Let $\tilde{\mathbf{u}}_1 = \tilde{\mathbf{u}}_2 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2$. Similar to the previous argument, $\delta(\tilde{\mathbf{u}}_1)$ and $\delta(\tilde{\mathbf{u}}_2)$ are of the same type. Hence, $\Pr\{\mathbf{U} = \tilde{\mathbf{u}}_1\} = \Pr\{\mathbf{U} = \tilde{\mathbf{u}}_2\}$. Note that $\Pr\{\mathbf{V} = \mathbf{v}_1|\mathbf{U} = \tilde{\mathbf{u}}_1\} = \epsilon^d(1 - \epsilon)^{RN-d}$, where

$$\begin{aligned} d &= d_H(\mathbf{v}_1, h(\tilde{\mathbf{u}}_1)) = d_H(h(\mathbf{z}_1), h(\tilde{\mathbf{u}}_1)) = W_H(h(\mathbf{z}_1 \oplus \tilde{\mathbf{u}}_1)), \\ &= W_H(h(\mathbf{z}_2 \oplus \tilde{\mathbf{u}}_2)) = d_H(\mathbf{v}_2, h(\tilde{\mathbf{u}}_2)). \end{aligned} \quad (\text{A.4})$$

Thus, $\Pr\{\mathbf{V} = \mathbf{v}_1|\mathbf{U} = \tilde{\mathbf{u}}_1\} = \Pr\{\mathbf{V} = \mathbf{v}_2|\mathbf{U} = \tilde{\mathbf{u}}_2\}$ and hence $APP(\tilde{\mathbf{u}}_1|\mathbf{v}_1) = APP(\tilde{\mathbf{u}}_2|\mathbf{v}_2)$. Similarly, $APP(\hat{\mathbf{u}}_1|\mathbf{v}_1) = APP(\hat{\mathbf{u}}_2|\mathbf{v}_2)$. This implies that $APP(\tilde{\mathbf{u}}_1|\mathbf{v}_1) > APP(\hat{\mathbf{u}}_1|\mathbf{v}_1)$ which contradicts the assumption on $\hat{\mathbf{u}}_1$. Thus $\hat{\mathbf{u}}_2 = \arg \max APP(\mathbf{u}|\mathbf{v}_2)$.

Note that in the components where $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ agree, $\delta(\hat{\mathbf{u}}_1)$ and $\delta(\hat{\mathbf{u}}_2)$ also agree. In the other components where they disagree, they are negative of each other (by symmetry) and, hence, $\delta(\hat{\mathbf{u}}_1)$ and $\delta(\hat{\mathbf{u}}_2)$ are also negative of each other. Thus, $\|\tilde{\mathbf{x}}_2 - \delta(\hat{\mathbf{u}}_2)\|^2 = \|\tilde{\mathbf{x}}_1 - \delta(\hat{\mathbf{u}}_1)\|^2$. Therefore, $D_j^u(\tilde{\mathbf{x}}_1) \leq D_j^u(\tilde{\mathbf{x}}_2)$. Similarly, $D_j^u(\tilde{\mathbf{x}}_2) \leq D_j^u(\tilde{\mathbf{x}}_1)$. Therefore $D_j^u(\tilde{\mathbf{x}}_1) = D_j^u(\tilde{\mathbf{x}}_2)$. $\square$

Appendix B

Proof of Theorem 4

To prove Theorem 4, we will need the following Lemma.

Lemma 1 *Let $\mathbf{u}, \hat{\mathbf{u}}, \mathbf{z} \in \{0, 1\}^{RN}$ be s.t.*

$$APP(\hat{\mathbf{u}}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z})), \quad (\text{B.1})$$

then

$$APP(\hat{\mathbf{u}}|h(\hat{\mathbf{u}})) \geq APP(\mathbf{u}|h(\hat{\mathbf{u}})). \quad (\text{B.2})$$

Proof: Let $d_1 = d_H(h(\mathbf{u}), h(\hat{\mathbf{u}}))$, $d_2 = d_H(h(\mathbf{u}), h(\mathbf{z}))$ and $d_3 = d_H(h(\mathbf{z}), h(\hat{\mathbf{u}}))$.

We have, by hypothesis,

$$\Pr\{\mathbf{U} = \hat{\mathbf{u}}\} \epsilon^{d_3} (1 - \epsilon)^{RN-d_3} \geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^{d_2} (1 - \epsilon)^{RN-d_2}. \quad (\text{B.3})$$

Multiplying by $((1 - \epsilon)/\epsilon)^{d_3}$, we get

$$\begin{aligned} \Pr\{\mathbf{U} = \hat{\mathbf{u}}\} (1 - \epsilon)^{RN} &\geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^{d_2-d_3} (1 - \epsilon)^{RN-(d_2-d_3)} \\ &\geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^{d_1} (1 - \epsilon)^{RN-d_1}. \end{aligned} \quad (\text{B.4})$$

The last inequality follows from the triangular inequality ($d_2 - d_3 \leq d_1$). Thus

$$APP(\hat{\mathbf{u}}|h(\hat{\mathbf{u}})) \geq APP(\mathbf{u}|h(\hat{\mathbf{u}})). \square \quad (\text{B.5})$$

Theorem 4 $D_j^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$ for $j = 0, 1, \dots, RN$.

Proof: Fix j and let $\mathbf{v}$ be that which achieves the maximum in (3.67) and let $\hat{\mathbf{u}} = \eta(\mathbf{v})$. Thus, $d_H(\mathbf{w}, \mathbf{v}) = j$,

$$APP(\hat{\mathbf{u}}|\mathbf{v}) \geq APP(\mathbf{u}|\mathbf{v}) \quad \forall \mathbf{u} \in \{0, 1\}^{RN}, \quad (\text{B.6})$$

and $\frac{1}{N} \|\tilde{\mathbf{x}} - \delta(\hat{\mathbf{u}})\|^2 = D_j^u(\tilde{\mathbf{x}})$. Let $\mathbf{z}$ be the input to the convolutional encoder corresponding to output $\mathbf{v}$, i.e., $h(\mathbf{z}) = \mathbf{v}$. Let $d = d_H(h(\mathbf{u}), h(\hat{\mathbf{u}}))$.

Case 1: $d \leq j$: From (B.6), we have $APP(\hat{\mathbf{u}}|h(\mathbf{z})) \geq APP(\mathbf{u}|h(\mathbf{z}))$. From Lemma 1, we have $APP(\hat{\mathbf{u}}|h(\hat{\mathbf{u}})) \geq APP(\mathbf{u}|h(\hat{\mathbf{u}}))$, implying that $\hat{\mathbf{u}} \in \mathcal{Z}^d$. Thus, $D_d^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$. By monotonicity of $D_l^*(\tilde{\mathbf{x}})$ (see step (6)), $D_j^*(\tilde{\mathbf{x}}) \geq D_d^*(\tilde{\mathbf{x}})$. Hence, $D_j^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$.

Case 2: $d > j$: Again, from (B.6) and from Lemma 1, we have $APP(\hat{\mathbf{u}}|h(\hat{\mathbf{u}})) \geq APP(\mathbf{u}|h(\hat{\mathbf{u}}))$, implying that $\hat{\mathbf{u}} \in \mathcal{Z}^d$. Let $d_3 = d_H(h(\mathbf{z}), h(\hat{\mathbf{u}}))$. Note that from (B.6),

$$\Pr\{\mathbf{U} = \hat{\mathbf{u}}\} \epsilon^{d_3} (1 - \epsilon)^{RN - d_3} \geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^j (1 - \epsilon)^{RN - j}. \quad (\text{B.7})$$

From the triangular inequality, $d_3 \geq d - j$, we have

$$\Pr\{\mathbf{U} = \hat{\mathbf{u}}\} \epsilon^{d-j} (1 - \epsilon)^{RN - (d-j)} \geq \Pr\{\mathbf{U} = \mathbf{u}\} \epsilon^j (1 - \epsilon)^{RN - j}. \quad (\text{B.8})$$

Note that in step (4) when $l = d$ and $\mathbf{z} = \hat{\mathbf{u}}$, the above satisfies condition (3.71), implying that $j \geq j^*$. Since $\hat{\mathbf{u}} \in \mathcal{Z}^d$ and because of (3.72), $D_{j^*}^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$. By monotonicity, $D_j^*(\tilde{\mathbf{x}}) \geq D_{j^*}^*(\tilde{\mathbf{x}}) \geq D_j^u(\tilde{\mathbf{x}})$. $\square$

Appendix C

Implementation of MAP Symbol-By-Symbol Decoder

Here, we describe the implementation aspect of the MAP symbol-by-symbol decoder in (3.74). The interested reader is referred to [24].

Denote the state, input and output of the convolutional encoder at time n by s_n , u_n and w_n , respectively. Let a and b be the next-state and output functions. Hence $s_n = a(u_n, s_{n-1})$, $w_n = b(u_n, s_{n-1})$. If the input u_n drives the state from s_{n-1} to s_n and produces an output w_n , we write $u_n = A(s_n, s_{n-1})$ and $w_n = B(s_n, s_{n-1})$. If transition from s_{n-1} to s_n is not possible, then $A(s_n, s_{n-1}) = B(s_n, s_{n-1}) = \emptyset$.

Let $P(u_n) = \Pr\{U_n = u_n\}$ and $Q(v_n|w_n) = \Pr\{V_n = v_n|W_n = w_n\}$ ($P(\emptyset) = Q(v_n|\emptyset) = 0$). Let $g^{(n)}(s_n) \triangleq \Pr\{S_n = s_n, \mathbf{V}_1^n = \mathbf{v}_1^n\}$ (for notational simplicity, we do not explicitly express the dependency on $\mathbf{v}_1^n$), where $\mathbf{v}_1^n = (v_1, v_2, \dots, v_n)$.

Then

$$\begin{aligned}
 g^{(n)}(s_n) &= \sum_{\mathbf{s}_1^{n-1}} \Pr\{S_1^n = \mathbf{s}_1^n, \mathbf{V}_1^n = \mathbf{v}_1^n\} \\
 &= \sum_{\mathbf{s}_1^{n-1}} \Pr\{\mathbf{V}_1^n = \mathbf{v}_1^n | S_1^n = \mathbf{s}_1^n\} \Pr\{S_1^n = \mathbf{s}_1^n\} \\
 &= \sum_{\mathbf{s}_1^{n-1}} \prod_{k=2}^n [Q(v_k | B(s_k, s_{k-1})) P(A(s_k, s_{k-1}))] \\
 &\quad \times [Q(v_1 | B(s_1, s_0)) P(A(s_1, s_0))], \tag{C.1}
 \end{aligned}$$

assuming that the initial state is s_0 . Then $g^{(n)}(s_n)$ can be computed recursively

according to (see [40])

$$g^{(n)}(s_n) = \sum_{s_{n-1}} Q(v_n|B(s_n, s_{n-1}))P(A(s_n, s_{n-1}))g^{(n-1)}(s_{n-1}). \quad (C.2)$$

Now define

$$\begin{aligned} f^{(n+\tau)}(u_n) &\triangleq \Pr\{U_n = u_n, \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}\} \\ &= \sum_{\mathbf{s}_{n+1}^{n+\tau}} \sum_{\mathbf{s}_1^{n-1}} \Pr\{\mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau} | U_n = u_n, \mathbf{S}_1^{n-1} = \mathbf{s}_1^{n-1}, \mathbf{S}_{n+1}^{n+\tau} = \mathbf{s}_{n+1}^{n+\tau}\} \\ &\quad \times \Pr\{U_n = u_n, \mathbf{S}_1^{n-1} = \mathbf{s}_1^{n-1}, \mathbf{S}_{n+1}^{n+\tau} = \mathbf{s}_{n+1}^{n+\tau}\} \end{aligned} \quad (C.3)$$

The first factor in the above is

$$\begin{aligned} &[\prod_{k=n+2}^{n+\tau} Q(v_k|B(s_k, s_{k-1}))][Q(v_{n+1}|B(s_{n+1}, a(u_n, s_{n-1})))Q(v_n|b(u_n, s_{n-1}))] \\ &\quad \times [\prod_{k=2}^{n-1} Q(v_k|B(s_k, s_{k-1}))Q(v_1|B(s_1, s_0))], \end{aligned} \quad (C.4)$$

while the second factor is

$$\begin{aligned} &[\prod_{k=n+2}^{n+\tau} P(A(s_k, s_{k-1}))][P(A(s_{n+1}, a(u_n, s_{n-1})))P(u_n)] \\ &\quad \times [\prod_{k=2}^{n-1} P(A(s_k, s_{k-1}))P(A(s_1, s_0))]. \end{aligned} \quad (C.5)$$

Thus

$$\begin{aligned} f^{(n+\tau)}(u_n) &= \sum_{\mathbf{s}_{n+1}^{n+\tau}} \prod_{k=n+2}^{n+\tau} [Q(v_k|B(s_k, s_{k-1}))P(A(s_k, s_{k-1}))] \\ &\quad \times \sum_{\mathbf{s}_1^{n-1}} Q(v_{n+1}|B(s_{n+1}, a(u_n, s_{n-1})))P(A(s_{n+1}, a(u_n, s_{n-1}))) \\ &\quad \times Q(v_n|b(u_n, s_{n-1}))P(u_n) \\ &\quad \times [\prod_{k=2}^{n-1} Q(v_k|B(s_k, s_{k-1}))P(A(s_k, s_{k-1}))] \\ &\quad \times [Q(v_1|B(s_1, s_0))P(A(s_1, s_0))]. \end{aligned} \quad (C.6)$$

The second summation in (C.6) is defined as $\psi^{(n+1)}(u_n, s_{n+1})$, which can be computed by

$$\begin{aligned} \psi^{(n+1)}(u_n, s_{n+1}) &= \sum_{s_{n-1}} Q(v_{n+1}|B(s_{n+1}, a(u_n, s_{n-1})))P(A(s_{n+1}, a(u_n, s_{n-1}))) \\ &\quad \times Q(v_n|b(u_n, s_{n-1}))P(u_n)g^{(n-1)}(s_{n-1}). \end{aligned} \quad (C.7)$$

For $m = 2, 3, \dots, \tau$, $\psi^{(n+m)}(u_n, s_{n+m})$ is defined recursively as

$$\begin{aligned} \psi^{(n+m)}(u_n, s_{n+m}) &= \sum_{s_{n+m-1}} Q(v_{n+m}|B(s_{n+m}, s_{n+m-1}))P(A(s_{n+m}, s_{n+m-1})) \\ &\quad \times \psi^{(n+m-1)}(u_n, s_{n+m-1}). \end{aligned} \quad (\text{C.8})$$

From (C.6), (C.7) and (C.8) we have

$$f^{(n+\tau)}(u_n) = \sum_{s_{n+\tau}} \psi^{(n+\tau)}(u_n, s_{n+\tau}). \quad (\text{C.9})$$

Finally, note that

$$\Pr\{U_n = u_n | \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}\} = \frac{f^{(n+\tau)}(u_n)}{\sum_{u_n} f^{(n+\tau)}(u_n)}. \quad (\text{C.10})$$

In the implementation of the symbol-by-symbol MAP decoder, at time instant n , $g^{(n-1)}(\cdot)$ is determined from $g^{(n-2)}(\cdot)$ using (C.2). Then $\psi^{(n+1)}(\cdot, \cdot)$ is determined using (C.7) and for $m = 2, 3, \dots, \tau$, $\psi^{(n+m)}(\cdot, \cdot)$ is determined using (C.8). Finally, we obtain $\Pr\{U_n = u_n | \mathbf{V}_1^{n+\tau} = \mathbf{v}_1^{n+\tau}\}$ from (C.9) and (C.10).

Bibliography

- [1] K. Abend and B. C. Fritchman, "Statistical Detection for Communication Channels with Intersymbol Interference," *Proc. IEEE*, Vol. 58, pp. 779-785, May 1970.
- [2] F. Alajaji and T. Fuja, "A Communication Channel Modeled on Contagion," to be submitted to *IEEE Trans. Inform. Theory*.
- [3] S. Arya, N. Phamdo, N. Farvardin and D. Mount, "Fast Search Algorithms with Applications to Split and Multi-Stage Vector Quantization of Speech LSP Parameters," to be presented in *IEEE Workshop on Speech Coding for Telecommunications*, St. Jovite, Quebec, Oct. 1993.
- [4] E. Ayanoglu and R. M. Gray, "The Design of Joint Source and Channel Trellis Waveform Coders," *IEEE Trans. Inform. Theory*, Vol. 33, pp. 855-865, Nov. 1987.
- [5] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, Vol. 20, pp. 284-287, Mar. 1974.
- [6] L. R. Bahl, C. D. Cullum, W. D. Frazer and F. Jelinek, "An Efficient Algorithm for Computing Free Distance," *IEEE Trans. Inform. Theory*, Vol. 18, pp. 437-439, May 1972.

- [7] C. F. Barnes and R. L. Frost, "Residual Vector Quantizers with Jointly Optimized Codebooks," in *Advances in Electronics and Electron Physics: Image Mathematics and Image Processing*. New York: Academic Press, 1992, Vol. 84, pp. 1-59.
- [8] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [9] R. E. Blahut, *Principles and Practice of Information Theory*. Reading, MA: Addison-Wesley, 1987.
- [10] J. R. B. De Marca and N. S. Jayant, "An Algorithm for Assigning Binary Indices to the Codevectors of a Multidimensional Quantizer," *Proc. IEEE Int. Conf. Comm.*, Seattle, WA, pp. 1128-1132, June 1987.
- [11] J. L. Devore, "A Note on the Observation of a Markov Source Through a Noisy Channel," *IEEE Trans. Inform. Theory*, Vol. 20, pp. 762-764, Nov. 1974.
- [12] A. W. Drake, "Observation of a Markov Source Through a Noisy Channel," *Sc. D. Thesis*, M.I.T., June 1962.
- [13] N. Farvardin, "A Study of Vector Quantization for Noisy Channels," *IEEE Trans. Inform. Theory*, Vol. 36, pp. 799-809, July 1990.
- [14] N. Farvardin and J. Modestino, "Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources," *IEEE Trans. Inform. Theory*, Vol. 30, pp. 485-497, Jan. 1984.
- [15] N. Farvardin and V. Vaishampayan, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," *IEEE Trans. Inform. Theory*, Vol. 33, pp. 827-838, Nov. 1987.

- [16] N. Farvardin and V. Vaishampayan, "On the Performance and Complexity of Channel-Optimized Vector Quantizers," *IEEE Trans. Inform. Theory*, Vol. 37, pp. 155-160, Jan. 1991.
- [17] T. Fine, "Properties of an Optimum Digital System and Applications," *IEEE Trans. Inform. Theory*, Vol. IT-10, pp. 287-296, Oct. 1964.
- [18] G. D. Forney, Jr., "Convolutional Codes I: Algebraic Structure," *IEEE Trans. Inform. Theory*, Vol. 16, pp. 720-738, Nov. 1970.
- [19] G. D. Forney, Jr., "The Viterbi Algorithm," *Proc. IEEE*, Vol. 61, pp. 268-278, Mar. 1973.
- [20] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.
- [21] E. N. Gilbert, "Capacity of a Burst-Noise Channel," *Bell Syst. Tech. J.*, Vol. 39, pp. 1253-1265, 1960.
- [22] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, Vol. 1, pp. 4-29, Apr. 1984.
- [23] R. M. Gray and Y. Linde, "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. Commun.*, Vol. COM-30, pp. 381-389, Feb. 1982.
- [24] J. F. Hayes, T. M. Cover and J. B. Riera, "Optimal Sequence Detection and Optimal Symbol-by-Symbol Detection: Similar Algorithms," *IEEE Trans. Commun.*, Vol. 30, pp. 152-157, Jan. 1982.
- [25] M. E. Hellman, "Convolutional Source Coding," *IEEE Trans. Inform. Theory*, Vol. 21, pp. 651-656, Nov. 1975.

- [26] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [27] B. H. Juang and A. H. Gray, Jr., "Multiple Stage Vector Quantization for Speech Coding," *Proc. ICASSP-82*, pp. 597-600, Apr. 1982.
- [28] V. N. Koshelev, "Direct Sequential Encoding and Decoding for Discrete Sources," *IEEE Trans. Inform. Theory*, Vol. 19, pp. 340-343, May 1973.
- [29] H. Kumazawa, M. Kasahara and T. Namekawa, "A Construction of Vector Quantizers for Noisy Channels," *Electronics and Engineering in Japan*, Vol. 67-B, pp. 39-47, Jan. 1984.
- [30] A. Kurtenbach and P. Wintz, "Quantizing for Noisy Channels," *IEEE Trans. Commun. Technology*, Vol. 17, pp. 291-302, Apr. 1969.
- [31] K. J. Larsen, "Comments on 'An Efficient Algorithm for Computing Free Distance'," *IEEE Trans. Inform. Theory*, Vol. 19, pp. 577-579, July 1973.
- [32] S. Lin and D. J. Costello, Jr., *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [33] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, Vol. 28, pp. 84-95, Dec. 1980.
- [34] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Inform. Theory*, Vol. 28, pp. 129-137, Mar. 1982.
- [35] J. Max, "Quantizing for Minimum Distortion," *IRE Trans. Inform. Theory*, Vol. IT-6, pp. 7-12, Mar. 1960.
- [36] J. W. Modestino and D. G. Daut, "Combined Source-Channel Coding of Images," *IEEE Trans. Commun.*, Vol. COM-27, pp. 1644-1659, Nov. 1979.

- [37] J. W. Modestino, D. G. Daut and A. L. Vickers, "Combined Source-Channel Coding of Images Using the Block Cosine Transform," *IEEE Trans. Commun.*, Vol. COM-29, pp. 1261-1274, Sept. 1981.
- [38] T. Moriya, "Two-Channel Conjugate Vector Quantizer for Noisy Channel Speech Coding," *IEEE J. Select. Areas in Commun.*, Vol. 10, pp. 866-874, June 1992.
- [39] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA: The MIT Press, 1972.
- [40] N. Phamdo and N. Farvardin, "Optimal Detection of Discrete Markov Sources Over Discrete Memoryless Channels — Applications to Combined Source-Channel Coding," *IEEE Trans. Inform. Theory*, to appear.
- [41] N. Phamdo, N. Farvardin and T. Moriya, "A Unified Approach to Tree-Structured and Multi-Stage Vector Quantization for Noisy Channels," *IEEE Trans. Inform. Theory*, Vol. 39, pp. 835-850, May 1993.
- [42] N. Phamdo, N. Farvardin and T. Moriya, "Combined Source-Channel Coding of Speech LSP Parameters Using Multi-Stage Vector Quantization," in *Speech and Audio Coding for Wireless and Network Applications*, B. S. Atal, V. Cuperman and A. Gersho, Editors. Norwell, MA: Kluwer, 1993, pp. 181-190.
- [43] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1988.
- [44] R. Reiningger and J. D. Gibson, "Distribution of the Two-Dimensional DCT Coefficients for Images," *IEEE Trans. Commun.*, Vol. 31, pp. 835-839, June 1983.

- [45] J. Rosebrock and P. W. Besslich, "Shape-Gain Vector Quantization for Noisy Channels with Applications to Image Coding," *IEEE J. Select. Areas in Commun.*, Vol. 10, pp. 918-925, June 1992.
- [46] N. Rydbeck and C. W. Sundberg, "Analysis of Digital Errors in Nonlinear PCM System," *IEEE Trans. Commun.*, Vol. 24, pp. 59-65, Jan. 1976.
- [47] K. Sayood and J. C. Borkenhagen, "Use of Residual Redundancy in the Design of Joint Source/Channel Coders," *IEEE Trans. Commun.*, Vol. 39, pp. 838-846, June 1991.
- [48] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, Vol. 27, pp. 379-423 and 623-656, 1948.
- [49] C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," *IRE Nat. Conv. Rec.*, pp. 142-163, Mar. 1959.
- [50] H. Suda, T. Moriya and T. Miki, "An Error Protected Transform Coder for Cellular Mobile Radio," in *Advances in Speech Coding*, B. S. Atal, V. Cuperman and A. Gersho, Editors. Norwell, MA: Kluwer, 1991, pp. 81-86.
- [51] N. Tanabe and N. Farvardin, "Subband Image Coding Using Entropy-Coded Quantization over Noisy Channels," *IEEE J. Select. Areas in Commun.*, Vol. 10, pp. 926-943, June 1992.
- [52] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding*. New York: McGraw-Hill, 1979.
- [53] K. A. Zeger and A. Gersho, "Zero Redundancy Channel Coding in Vector Quantization," *IEE Electronics Letters*, pp. 654-655, June 1987.